UNIVERSITÉ DE STRASBOURG

A thesis presented to

Observatoire astronomique de Strasbourg

by:

Anthony Rhys CONN

defended on : **7 February 2013**

for the degree of : **Docteur de l'université de Strasbourg**

Discipline: Astrophysics

Structure of the M31 Satellite System: Bayesian Distances from the Tip of the Red Giant Branch

Thesis Supervisors :

Dr. Rodrigo A. Ibata	Université de Strasbourg
Prof. Quentin A. Parker	Macquarie University
Prof. Geraint F. Lewis	University of Sydney
A/Prof. Daniel B. Zucker	Macquarie University

Members of the Jury :

Prof. Christian M. Boily	Université de Strasbourg
Dr. Michele Bellazzini	INAF—Osservatorio Astronomico di Bologna
Dr. Annie C. Robin	Observatoire de Besançon

External Examiner :

Dr. Andreas Koch	University of Heidelberg

Completed under a cotutelle agreement with Macquarie University in Sydney Australia

Typeset in LaTeX 2_ε.

Except where acknowledged in the customary manner, the material presented in this thesis is, to the best of my knowledge, original and has not been submitted in whole or part for a degree in any university.

Anthony Rhys Conn

Acknowledgements

I have been very fortunate throughout my PhD candidature to work with some very dedicated and talented astronomers and astrophysicists who have had a huge impact on the course my research has taken. I have worked most closely with my supervisors Dr Rodrigo Ibata (Université de Strasbourg) and Prof. Geraint Lewis (University of Sydney) who I would like to thank for the large amount of their time and resources they have invested in my project. Rodrigo is a renowned figure in the field of Galactic Archaeology and I have greatly benefited from his expertise. He has also been a great host on my many visits to Strasbourg and always available to discuss the problem at hand. Likewise Geraint has had a prolific impact on the field and is a veritable source of programming knowhow. He devotes an enormous amount of energy to the various projects of his many students and his input has been invaluable to my research. In addition, I would also like to thank my supervisors Prof. Quentin Parker and A/Prof. Dan Zucker (both from Macquarie University) for their constant support and encouragement. In particular I am indebted to Quentin for the great lengths he went to in securing a funded position for me at Macquarie and for his efforts in establishing and maintaining the cotutelle agreement with the Université de Strasbourg. Whilst my PhD project was somewhat external to the fields of expertise of both Quentin and Dan, they have always made themselves available to discuss my work at length and advise on all matters of administration.

In addition to the contributions of my 4 supervisors, there are a number of other people who have played an important role in my progress toward completion. I would especially

like to thank all the members of the Pan Andromeda Archaeological Survey (PAndAS) collaboration for providing a wonderful repository of data to work with and particularly those members included as coauthors on my papers for their detailed reviews of my drafts. I would also like to add that I had many meetings with Dr Nicolas Martin (Université de Strasbourg) whilst in Strasbourg which had a large impact on the work presented in Paper II. In addition I must thank Neil Ibata for his French translation of my thesis summary (Résumé de Thèse). I also received much support from my fellow PhD student Dr Anjali Varghese on my many trips to Strasbourg and relied on her superior French more than once.

There are also a number of institutions that I must acknowledge for the support they have provided. Firstly, I would like to thank Macquarie University for their financial support via the Macquarie University Research Excellence Scholarship (MQRES) and also for paying all my airfares and providing assistance with other expenses incurred during my candidature. I would also like to thank the Observatoire Astronomique of the Université de Strasbourg for providing me with accommodation on each visit to Strasbourg and I would like to thank both the aforementioned universities as well as the University of Sydney for use of computational and all other facilities.

Finally I would like to thank all of my family for their ceaseless support and encouragement. Without my family it is inconceivable that I could ever have made it this far. I would therefore like to dedicate this work to all of them: Mum, Dad, Nanna, Pa, Mardi, Pop, Andrew, Crysta, Sophie, Kristin, Remo, Alex, Chris, Matthew, Trixie and all my extended family.

A Note on Paper Co-Author Contributions

As a thesis by publication, much of the work contained in the main body of this thesis has been (or will be) published in the form of three separate papers submitted to the Astrophysical Journal. Each of these publications acknowledges contributions from a relatively large group of co-authors. This is an inevitable consequence of the use of data that is the property of a large collaboration (i.e. the Pan-Andromeda Archaeological Survey - PAndAS), and it has in no way diminished the amount of work that I have put into each of these papers. In fact, rather the opposite is true, due to the enormous amount of code adjustments and paper editing that has had to take place to accommodate the suggestions of a large group of people. The work contained in these papers has been carried out principally with the direction of the second and third co-authors, my supervisors Prof. Geraint Lewis and Dr Rodrigo Ibata, with all other contributions being in the form of written critiques of the paper drafts. This does not detract from the aid I have received from my other two supervisors Prof. Quentin Parker and A/Prof. Dan Zucker, who as well as providing similar critiques of my drafts, have been a great help with all administrational matters that I have contended with during the course of my PhD candidature. I should also note that I had many in-person discussions with Dr Nicolas Martin whilst in Strasbourg, which further influenced the way I carried out the analysis contained in the second paper.

All of this said, all three papers have been written entirely by me, and all figures they contain have been generated by me using the PGPLOT plotting package (Pearson, 2011), with the sole exception of the pole-density plots in Paper III; Ch. 5 (Figs. 2, 4, 8, 10 (right-hand column of plots), 11 and 14), which were produced using a Gaussian-smoothing

program by Geraint Lewis from the pole count analysis data that I sent him. All of the analysis contained in the papers has been undertaken using Fortran code that I have written from scratch, with examples of most of the principal versions of the code provided in the appendix. These programs do make occasional use of subroutines written by others, as now stated:

I The "*PolySelect*" subroutine is built around code written by Rodrigo Ibata, which uses the functions "*in_poly*" and "*fimag*" written by him. Given a series of points defining the corners of a polygon, it determines whether or not a given point is inside this polygon. I use this routine to make colour-cuts on the Colour-Magnitude Diagram (see calls of this subroutine in *MF_TRGB.f*95; Appendix C for example) and also to reject random satellites drawn outside of the PAndAS survey area in my "RandomPoints" subroutine, used in the analysis of Paper III.

II Part of the SVDFitter subroutine called in *MF_TRGB.f*95 (and earlier versions of this program) has been modified by Geraint Lewis.

III The "*func_i*" function called in *MF_TRGB.f*95 (and earlier versions of this program) gives the photometric error as a function of magnitude in CFHT i-band for the PAndAS data and was provided by Rodrigo Ibata.

IV The "*k_verse_alpha*" subroutine called in *SatDensity_SampCont.f*95 (Appendix C) makes use of data generated by Geraint Lewis (*k_vs_alpha.dat*), which accounts for the volume of space covered by the PAndAS survey area as a function of distance from M31.

V Several subroutines from Numerical Recipes (Press et al., 1992) have been used to perform standard functions by various programs I have written, in particular *MF_TRGB.f*95. In the programs listed in the appendices, use of these algorithms are generally noted at the end of the program with commented double lines and the words 'Libpress Algorithms' embedded. The Numerical Recipes subroutines themselves are not shown for the sake of brevity, but any subroutine or function that is called from the code as printed

in the appendices and yet is apparently absent from this printed code can be assumed to be from Numerical Recipes.

VI The SLALIB Positional Astronomy Library (Wallace, 1994) has been used to convert object RA and Dec into tangent plane coordinates in $MF\_TRGB.f95$ and earlier TRGB programs so that the user can access various parts of the PAndAS Survey directly by entering the objects Celestial coordinates. This library is also made use of in many of the programs written for Paper III to convert back and forth between tangent plane angles and real angles, and also to measure the angle between two given objects on the sky.

I believe this to be a complete disclosure of all coding aspects that I make use of that I have not written personally. They are few in number compared to the number I have written and in general, their function has been to perform tasks secondary to the principal operation of the programs that use them. Their inclusion is however, nevertheless vital to the correct functioning of the programs. In summary, whilst many have contributed to the work presented in these three papers, my contribution to each of them is exactly the same as any dedicated student would make to the work contained in a major chapter of their thesis were it not a journal publication.

Other General Comments

This thesis has been typeset in LaTeX using a template developed by Paul Cochrane, Alexei Gilchrist and Johann-Heinrich Schönfeldt. The template has been modified slightly to better suit the particular form of this thesis. Note that the digital version contains clickable hypertext providing links to relevant figures, sections and references. As a thesis by publication, the included papers have been generated independently and hence adhere to a different format. Whilst they are integrated in terms of the page numbering and access from the table of contents, their internal hyperlinks are inactive in the digital version of the thesis. Note also that whilst the references contained in the papers appear again in the the thesis 'References' section, the pages on which they are cited will only be available for citations external to the papers.

List of Publications

- Conn, A. R.; Lewis, G. F.; Ibata, R. A.; Parker, Q. A.; Zucker, D. B.; McConnachie, A. W.; Martin, N. F.; Irwin, M. J.; Tanvir, N.; Fardal, M. A.; Ferguson, A. M. N.

  A Bayesian Approach to Locating the Red Giant Branch Tip Magnitude. I.

  The **Astrophysical Journal**, Volume **740**, Issue 2, article id. **69** (**2011**)

- Conn, A. R.; Ibata, R. A.; Lewis, G. F.; Parker, Q. A.; Zucker, D. B.; Martin, N. F.; McConnachie, A. W.; Irwin, M. J.; Tanvir, N.; Fardal, M. A.; Ferguson, A. M. N.; Chapman, S. C.; Valls-Gabaud, D.

  A Bayesian Approach to Locating the Red Giant Branch Tip Magnitude. II. Distances to the Satellites of M31

  The **Astrophysical Journal**, Volume **758**, Issue 1, article id. **11** (**2012**)

- Conn, A. R.; Lewis, G. F.; Ibata, R. A.; Parker, Q. A.; Zucker, D. B.; McConnachie, A. W.; Martin, N. F.; Valls-Gabaud, D.; Tanvir, N.; Irwin, M. J.; Ferguson, A. M. N.; Chapman, S. C.

  The Three-Dimensional Structure of the M31 Satellite System; Strong Evidence for an Inhomogeneous Distribution of Satellites

  Submitted to the **Astrophysical Journal** on 9 November 2012

Abstract

The satellite system of a large galaxy represents the ideal laboratory for the study of galactic evolution. Whether that evolution has been dominated by past mergers or in situ formation, clues abound within the structure of the satellite system. This study utilizes recent photometric data obtained for the halo of M31 via the Pan-Andromeda Archaeological Survey (PAndAS), to undertake an analysis of the spatial distribution of the M31 satellite system. To do this, a new Bayesian algorithm is developed for measuring the distances to the satellites from the tip of their Red Giant Branch. The distances are obtained in the form of posterior probability distributions, which give the probability of the satellite lying at any given distance after accounting for the various spatial and photometric characteristics of the component stars. Thus robust distances are obtained for M31 and 27 of its satellite galaxies which are then transformed into three-dimensional, M31-centric positions yielding a homogenous sample of unprecedented size in any galaxy halo. A rigorous analysis of the resulting distribution is then undertaken, with the homogeneity of the sample fully exploited in characterizing the effects of data incompleteness. This analysis reveals a satellite distribution which as a whole, is roughly isothermal and no more planar than one would expect from a random distribution of equal size. A subset of 15 satellites is however found to be remarkably planar, with a root-mean-square thickness of just $12.34^{+0.75}_{-0.43}$ kpc. Of these satellites, 13 have subsequently been identified as co-rotating. This highly significant plane is all the more striking for its orientation. From the Earth we view it perfectly edge on and it is almost perpendicular to the Milky Way's disk. Furthermore, it is roughly orthogonal to the disk-like structure commonly reported for the Milky Way's satellite galaxies. The distribution is also found to be highly asymmetric, with the majority of satellites lying on the near side of M31. These findings point to a complex evolutionary history with possible links to that of our own galaxy.

Résumé de Thèse

Étude de la structure tridimensionnelle du système de satellites de M31 au moyen d'une méthode Bayesienne de localisation de la pointe de la branche des Géantes Rouges

Les étoiles de basse masse pauvres en métaux qui ont consommé tout l'hydrogène présent dans leur noyau et dont celui-ci n'a plus une densité suffisante pour fusionner de l'hélium, entrent dans la phase de la branche des géantes rouges (RGB). Après un certain temps, l'étoile devient plus lumineuse et les cendres dhélium produites par cette réaction retombent sur le noyau, accroissant sa densité jusqu'à celle-ci soit suffisante pour remettre en marche la fusion de l'hélium. L'étoile, qui n'appartiendra bientôt plus à la branche d'étoiles RGB est dite du tip of the Red Giant Branch (TRGB) . Du fait des propriétés similaires du noyau de toutes les étoiles qui arrivent à ce state de leur évolution dans une gamme spécifique de masse et de métallicité (voir Iben and Renzini 1983), leur radiation énergétique et donc leur luminosité est constante. Le TRGB pour de telles populations stellaires donne donc une mesure de la distance à cette population.

Avant le développement de la méthode de la détection d'un bord de Lee et al. (1993), la TRGB était déterminée par des Diagrammes Couleur-Magnitude (CMD) à l'oeil nu et les distances dérivées manquaient donc de précision et d'uniformité requis pour une utilisation fiable pour de nombreux objets. On a développé de nombreuses méthodes depuis celle-ci mais elles se basent toutes sur l'idée de convoluer la fonction de luminosité (LF) du RGB avec un kernel de détection de bord, afin de créer un maximum à la magnitude correspondant à la plus grande discontinuité dans la LF, qui devrait correspondre à la magnitude du TRGB. Malheureusement, de telles méthodes donnent de mauvais résultats dans la présence de bruits – notamment lorsque le RGB est noyé par des étoiles contaminantes. Pour cette raison, plusieurs alternatives d'ajustement de modèles qui utilisent toute la LF ont été proposées

(par exemple Méndez et al. 2002). Malgré cela, pour ces méthodes, les incertitudes de mesures sont souvent très grandes et mal définies et n'ont pas la possibilité d'incorporer nos informations à priori sur le système étudié. C'est pour cela que la première grande partie de cette thèse aura pour but de créer un algorithme robuste et versatile pour mesurer des distances en utilisant la magnitude du TRGB.

Les premiers chapitres décrivent le développement d'un algorithme Bayesien qui utilise une approche de maximum de vraisemblance. Les paramètres du modèle (magnitude du TRGB, pente de la LF, propriétés de contamination) sont ajustés par l'algorithme suivant une simulation Markov Chain Monte Carlo (MCMC). Cela donne accès aussi aux incertitudes sur ces paramètres. Malgré sa simplicité, cette méthode est robuste, et donne des sorties intuitives et visuelles des probabilités de paramètres et il reste facile d'ajouter de l'information à priori. La première version de cet algorithme a été publiée dans le Astrophysical Journal (Paper I), et est à la base du chapitre 3. Cette publication présente également des tests qui caractérisent la performance de cette méthode pour des LFs de différentes qualités, ainsi que son application à trois galaxies naines sphéroïdales, satellites de M31, et donne les meilleures incertitudes de toutes les méthodes basées sur le TRGB publiées jusqu'à ce jour.

Les données physiques analysées dans cette thèse viennent du Pan-Andromeda Archaeological Survey (PAndAS – McConnachie et al. 2009), un relevé ambitieux qui couvre plus de 300 degrés carrés autour de la galaxie d'Andromèdre, la galaxie géante la plus proche de la Voie Lactée. Ce relevé donne accès à la photométrie profonde en bande g' (centré sur 487 nm) et la bande i' (centré sur 770 nm), et qui couvre plus de 25 satellites galactiques qui sont idéaux pour des mesures de distance par la méthode TRGB. L'algorithme présenté en chapitre 3 a été amélioré pour utiliser ces donnes spécifiques. La contamination du fond étant la plus grande source du détriment de la qualité des distances TRGB, j'ai mis au point une routine << matched filter >> (voir Rockosi et al. 2002) pour donner des poids à chaque étoile en fonction de sa position spatiale dans le profil de densité du satellite. L'effet de l'application de cet algorithme sur la LF est de réduire la contamination du fond et ainsi d'augmenter le contraste de la troncature du RGB au TRGB. Visuellement, le changement du LF est souvent suffisant pour révéler de façon très claire la position du TRGB qui était avant à peine plus que du bruit Poissonnien.

Cette méthode améliorée, appliquée à tous les satellites (27 en total) détectés dans le relevé PAndAS est présentée dans un deuxième article soumis à l'Astrophysical Journal et constitue l'essentiel du chapitre 4 (Paper II). Cet article apporte les premières mesures de distances pour une grande partie de ces satellites et se révèle être l'analyse la plus compréhensive des distances du système de satellites de M31. Cette investigation contient également une analyse brève du profil de la densité du halo en utilisant ces nouvelles distances, que nous avons comparées aux valeurs trouvées avec l'aide d'autres méthodes.

Le grand nombre de satellites autour de M31 pour lesquels j'ai obtenu de bonnes mesures de distances donne ainsi une excellente occasion d'analyser le degré de planarité et d'asymétrie du système de satellites. Cela a des fortes répercussions sur la distribution de matire dans le halo de la galaxie hôte ainsi que sur l'histoire de formation des satellites mêmes. Plusieurs études du système satellitaire de la Voie Lactée (par exemple Lynden-Bell 1982; Zentner et al. 2005; Pawlowski et al. 2012b), trouvent des plans fortement significatifs, souvent inclinés par rapport au disque Galactique. Des résultats similaires ont été publiés pour le système de M31 (par exemple Koch and Grebel 2006). Les études du système de M31 ont été faits avec de petits échantillons de satellites et les mesures de distances proviennent donc de plusieurs auteurs (et méthodes) différentes. C'est ainsi que le chapitre 5 et une troisième publication donnent à voir une analyse détaillée du système de satellites de M31 en se basant sur les données du chapitre 4. La planarité du système de satellites est explorée par le biais du plan de meilleur ajustement en utilisant plusieurs méthodes (moindre rms, moindre distance, ajustement à un modele Gaussien). La vraisemblance de ces alignements est analysée à l'aide de simulations où chaque satellite est tiré au hasard a partir de sa distribution de distance. L'analyse de l'asymétrie est effectuée de façon similaire, en utilisant des statistiques d'asymétrie, notamment le nombre de satellites qui se trouvent sur un hémisphère du halo. Les positions 3-D présentées au chapitre 4 montrent que le pôle du plan d'asymétrie maximal se trouve très près du vecteur Terre-M31 ; la probabilité d'un tel alignement est étudiée dans cette thèse.

Contents

"I do not feel obliged to believe that the same God who has endowed us with sense, reason, and intellect has intended us to forgo their use."

Galileo Galilei (1564-1642)

1

An Introduction to Galactic Archaeology

1.1 Overview

Large galaxies like the Milky Way and it's neighbor the Andromeda Galaxy (M31) are complex, evolved structures when studied on any scale. They are a plethora of countless billions of stars and the condensing clouds of gas and dust from which they form, all in motion, all evolving since time immemorial. But far removed though their origins may be, their very structure preserves their past. However, even the structure of the Milky Way, our own galaxy, is not obvious from our vantage point deep within it and while the general structure of its basic components have been constrained, there is an underlying labyrinth of substructure remaining to be identified and interpreted with respect to its bearing on Galactic Evolution. Hence we must begin our study with an overview of the large scale structure of our own

galaxy a structure which, as might be expected, is shared by many of our galactic neighbours and indeed by nearly all those galaxies near and far of a similar type.

1.2 A portrait of a Galaxy

The Milky Way (henceforth 'the Galaxy') is a late-type barred spiral galaxy. It is known to consist of both a thin and a thick disk component, a central bulge and an enormous halo, encompassing the whole system (Freeman and Bland-Hawthorn, 2002). The thin disk has been determined to have a scale length of 2600 pc and a scale height of 300 pc (Jurić et al., 2008) with an overall radius of 15 ± 2 kpc (Ruphy et al., 1996). It is within the thin disk that both the solar neighbourhood and the spiral arms reside. The spiral arms have been traced by various methods, notably by Georgelin and Georgelin (1976), who used HII regions to trace their extent. They found two symmetrical pairs of arms with a pitch angle of 12°. The four arms in total were identified as the Sagittarius-Carina Arm, the Scutum-Crux Arm, the Norma Arm and the Perseus Arm, with the Sun residing in a spur between the inner Sagittarius-Carina Arm and the outer Perseus Arm. This is represented schematically in Figure 1.1. Based on their findings they suggest a morphological type for the Galaxy closest to Sc.

Enveloping the thin disk is a somewhat more diffuse, ancient haze of stars termed the Galactic 'thick disk' (Gilmore and Reid, 1983). It has been calculated from the Sloan Digital Sky Survey I (SDSS I) to have a scale length of 3600 pc and a scale height of 900 pc (Jurić et al., 2008). Freeman and Bland-Hawthorn (2002) describe it as a 'snap frozen relic of the heated early disk' and allocate some 10% of the Galaxy's baryonic matter to its confines. A metallicity of -2.2 < [Fe/ H] < -0.5 is quoted for the thick disk stars in contrast to the -0.5 < [Fe/ H] < 0.3 determined for the younger thin disk, and its luminosity is specified as 10% that of the thin disk.

In the inner regions of the Galaxy is a denser conglomeration of what are generally considered to be older, metal poor stars termed 'the bulge.' Freeman and Bland-Hawthorn (2002) caution however that a study of bulge red giant stars (McWilliam and Rich, 1994) suggests

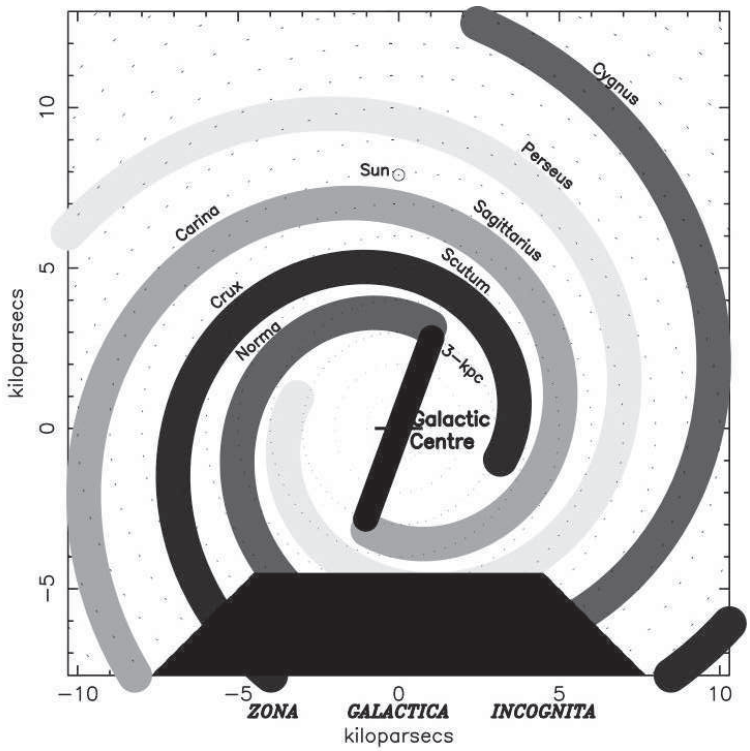


FIGURE 1.1: A Schematic of the observable portion of the Milky Way's spiral arms. (Vallée, 2005)

a metallicity much closer to the older stars of the thin disk than to the truly ancient stars in the Galactic halo. They further describe the Milky Way's bulge as appearing significantly smaller than that of M31 and somewhat 'boxy,' typical of an Sb to Sc spiral. Also of particular note, the Galaxy has long been suspected of containing a bar at its centre which has, as of 2005, been proven. Benjamin et al. (2005) find the bar to have a length of 8.8 ± 1.0 kpc with orientation such that it is rotated 44 ± 10^o from a line connecting the Sun and Galactic Centre.

Finally, the halo of the Milky Way is easily its largest and arguably its oldest major constituent. It is an enormous, roughly spherical (Ibata et al., 2001b) cocoon of ancient field stars, and approximately 150 similarly ancient globular clusters (Freeman and Bland-Hawthorn, 2002). It is also known to extend out well beyond the Small Magellanic Cloud to a distance of 100 kpc from Galactic centre and it contains at least 10 known satellite galaxies (van den Bergh, 2006). Perhaps most remarkable is that it contains $1.0^{+0.3}_{-0.2} \times 10^{12} M_{\odot}$ (Xue et al., 2008) of dark matter, which amounts to at least 90 % of the total mass of the Galaxy

(Freeman and Bland-Hawthorn, 2002). The substructure within this dark matter halo is of great interest as it lies at the heart of our current understanding of galaxy formation.

Our current knowledge of the Galaxy as presented in the above paragraphs represents some of the fruits of Galactic Archaeology. This knowledge is however fairly coarse in scope and Galactic Archaeology may still be regarded as a burgeoning field. Nevertheless, it is our means to unravel the Galaxy's past and our best hope for predicting its future.

1.3 Galactic Archaeology - The Means and the Motives

The field of Galactic Archaeology is in a sense a toolkit providing the necessary tools to wind back the cosmic clock and provide us with a high resolution view of our Galaxy and its immediate neighbours in a way that might otherwise have been restricted to the poorly resolved galaxies of the high-redshift universe. It is not a single method but rather a collection of techniques making use of large sky photometric, astrometric and kinematic surveys to study the positions, motions and chemical compositions of groups of stars in an effort to link them to ancient progenitor structures and then simulate the evolution of these structures through time to the present and beyond. In other words, if stars are found to be grouped together in 6D phase space (i.e. 3 dimensions in position and 3 dimensions of velocity) they may be members of a present day cluster whereas stars grouped together only in velocity space may be termed a moving group and be members of a since-dispersed cluster. Stars grouped together in chemical space might similarly be 'tagged' to an ancient progenitor structure. Some of these possibilities are further investigated in the following paragraphs.

With the advent of Galactic Archaeology, the discovery of moving groups has become common. In an early example, Eggen and Sandage (1959) identified the nearby moving group Groombridge 1830 and associated it with the Galaxy's globular clusters, providing an early detection of nearby halo stars. In the intervening decades, numerous further examples have been discovered associated with the halo alone, but Freeman and Bland-Hawthorn (2002) caution that the validity of some of these groups is questionable.

The tagging of stars to progenitor groups based on their chemical composition is perhaps an even more powerful technique. It relies on the assumption that the progenitor cloud

be uniformly well mixed before the formation of the surviving stars (Freeman and Bland-Hawthorn, 2002) which is conceivable if McKee and Tan (2002)'s model of cluster formation is accepted whereby all stars form at a similar time. Such a method has interesting implications not only for the origins of structure formation in the Galaxy at large, but also at a more local level, as it presents the real possibility of identifying Solar siblings – those stars that formed out of the same cloud as our Sun. Indeed Reipurth (2005) lists possible evidence supporting the idea that the Sun did in fact form in a cluster and Portegies Zwart (2009) goes so far as to provide mass and radius constraints for the cluster of 500 – 3000 $M_{\odot}$ and 1 – 3 pc respectively. They further concur that with accurate chemical abundances and phase space information, the identity of the cluster members may be recovered. A direct test of the feasibility of chemical abundance tagging is seen in De Silva et al. (2007) where of the 18 supposed members of the commoving group HR1614, 14 were found to have very little scatter in chemical abundances across a wide range of elements with the non-conforming stars conceivably 'pollution' from the non-cluster background. Thus it seems that, at least in some cases, this powerful technique proposed for Galactic Archaeology should be applicable.

So far we have encountered the means to re-construct ancient Galactic components but the question remains – how ancient? A time frame is needed to accurately model the Galaxy's evolution, as evolution is after all time dependent. There are various methods proposed to fulfill this function, all relating to the determination of stellar age, of which Freeman and Bland-Hawthorn (2002) gives a concise summary. Since we are generally concerned with stars long since removed from their parent clusters, determining age from the main sequence turnoff is obviously not an option. Instead, such methods as nucleo-cosmochronology, astero-seismology and age-metallicity relations are suggested. Nucleo-cosmochronology is concerned with ageing the elements in a star based on their remaining radioactive isotope strengths, given a certain radioactive decay rate. Since the original elemental abundances are not known, the method compares the radioactive isotope strengths to stable r-process elements. Some studies based on this technique have already been highly successful. Astero-seismology takes advantage of the evolving mean molecular weight in the cores of stars to ascertain age and has been used to provide an age for the Sun of 4.57 Gyr ± 0.12 Gyr (Gough,

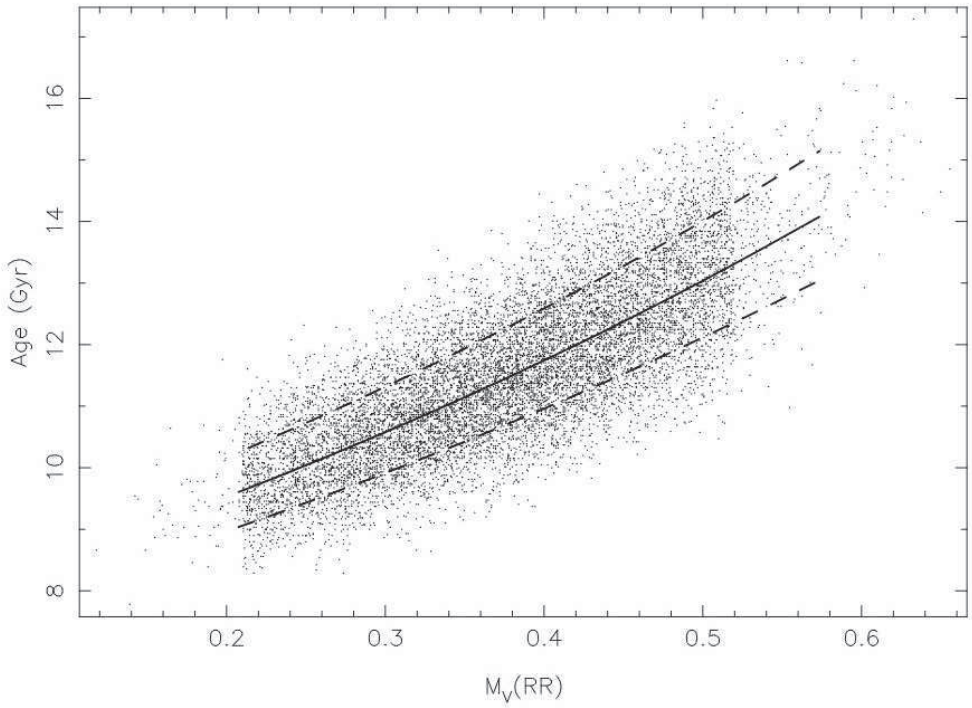


FIGURE 1.2: An example of the Age-Luminosity Relation. Here, the ages of the oldest globular clusters have been plotted as a function of the absolute visual magnitude of component RR Lyrae stars. The best fit median is represented by the solid line while the dashed lines represent 1σ limits. (Chaboyer et al., 1998)

2001), which matches well with the ages determined for the oldest meteorites by more direct means. An age-metallicity relationship would provide a more direct measure of stellar age, if indeed one could be established but alas, such a relationship only applies to a small subset of stars. Freeman and Bland-Hawthorn (2002) find such a relationship to exist only for a small range of young, hot, metal-rich stars. More useful however is the age-luminosity relationship (Figure 1.2) found to apply to the much older RR Lyrae stars, provided their distances may be accurately determined. This principle has been applied to constrain the ages of the Galaxy's globular clusters (Chaboyer et al., 1996). Still, there is a large age interval over which the latter two methods are not applicable, thus emphasizing the importance of the former two methods.

Having discussed the tools of Galactic Archaeology, what are its goals and to what extent have these goals already been met? The ultimate goal of Galactic Archaeology is to be able to

trace the current structures of the Galaxy back to their progenitor structures in the protocloud from which it formed. In so doing, the histories of the various components of the Galaxy are uncovered, spanning from the epoch of formation to the present day. As outlined in section 1, the basic structure of the Galaxy has already been established and based on the stellar ages and metallicities/ elemental abundances across the various components an hypothesis for galaxy formation has been formulated, again summarized in Freeman and Bland-Hawthorn (2002). It is suggested that the Galactic Protocloud began to form at a similar time to the epoch of reionization. At this time the Galaxy, like those around it, appeared in the form of a dark matter halo, with its central black hole and possibly its stellar bulge forming first. The prominent disk structure where most of the baryons reside did not develop until the beginning of the main epoch of baryon dissipation at a redshift of $z \sim 1 - 5$. This also coincides with the ages of the thick disk and the globular clusters. The populating of the halo with globulars and field stars is thought to have also begun very early in the formation process, the result of tidal interactions with small neighbouring dwarf galaxies. The thin disk comprises the youngest stars of the Galaxy while the thick disk is likely the dynamically heated remnant of an ancient thin disk – in fact, Galactic Archaeology may provide some clue as to the particular interaction responsible. One popular theory is that the globular cluster ω Cen is the remnant core of a small galaxy, stripped of its outer stars in an interaction precipitating the heating of the original thin disk (Bekki and Freeman, 2003). It is also believed that the current galactic bulge is not of the ancient origin of more pronounced bulges such as that found in M31, but rather a later formation in the established inner disk. This is consistent with the relatively high metallicities in the galactic core, although it must be stressed that metallicity is a better measure of the number of supernova events rather than of actual age and the density of the galactic core is bound to influence this number profoundly.

The formation sequence presented above owes little to observations of high-redshift galaxies or even to computer simulations based on Cold Dark Matter (CDM) Cosmology, but rather it is a construction based on observations of our own galaxy and those nearby. Our focus has so far been centered on the Milky Way, but it must be stressed that any galaxies close enough to have their individual stars mapped into phase space or chemical space are

within the reach of Galactic Archaeology. It should also be stressed that the methods associated with Galactic Archaeology described above form the basis for such study but such methods provide for mere data acquisition – the possible applications for the data are enormous, and hence so to is the scope of Galactic Archaeology. These points should be kept in mind as some of the various sky surveys available to the 'Galactic Archaeologist' are discussed in the next section.

1.4 Completed and Future Surveys - What can they tell us?

Modern Galactic Archaeology draws heavily on a small number of ambitious, wide field surveys focused, at least in part, on the acquisition of either photometric, astrometric or kinematic data for large numbers of stars. While there are many smaller data sets such as Hubble Space Telescope (HST) pointings and those from major ground telescopes which are also utilized, our focus here shall be limited to these major surveys.

1.4.1 Photometric

Among those surveys with the broadest scope are the photometric surveys, although the data they include is often more restrictive for Galactic Archaeology than that from the astrometric and kinematic surveys. Photometry is of particular usefulness in determining the distance to *l*arge numbers of objects. The two most recent major photometric catalogues are those from the Sloan Digital Sky Survey (SDSS) and the Two Micron All Sky Survey (2MASS). SDSS is an ongoing survey, begun in 2000, using the dedicated 2.5 m wide-field, modified Ritchey-Chrétien telescope at Apache Point Observatory and an array of 30×4 megapixel CCDs. The survey provides photometry in the u, g, r, i and z bands (see York et al. 2000 for a technical summary) as well as spectroscopy of select targets. As of the ninth data release (SDSS-III Collaboration et al., 2012), the survey had covered some 14555 square degrees of sky or more than $\frac{1}{3}$ of the entire celestial sphere, with spectra obtained for 668054 stars. The survey also features stellar positions accurate to within 150 mas for each coordinate and metallicity as well as phase space information are determinable for the observed stars. The stellar coverage is however, relatively small owing to the survey's greater emphasis on

obtaining photometry for galaxies.

The 2MASS survey (Skrutskie et al., 2006) in contrast covers an enormous quantity of stars, with some 471 million point sources extracted from the data. The survey covers the entire sky and includes photometry in the J, H and Ks near-infrared bandpasses. For entire sky coverage, two ground based telescopes were required, one in each hemisphere, and hence two 1.3 m telescopes were constructed for the task, one at Mount Hopkins, Arizona and the other at Cerro Tololo in Chile. The 7.8 second exposure for each field results in limiting magnitudes of 15.8, 15.1 and 14.3 in the J, H and K_s bands respectively. A 1σ error of < 0.03 magnitudes is determined for the photometry with an estimated error of 100 mas in the source positions. With stellar positions as well as metallicity being determinable from the data, stellar tagging is a possibility from this data set. Indeed, this survey is a useful archive of data for isolating ancient structures, especially since such structures may be expected to be delineated by luminous red giant stars which would remain visible out to great distances due to their strong emission in the near infra-red. This merit of the survey has in fact been exploited by previous studies, as exemplified by Ibata et al. (2002a) where M giants were used to trace substructure in the outer Galactic halo. Still, the lack of kinematic data obtainable from the survey does present some limitations for reconstructing ancient structures that have since dispersed.

The Skymapper Telescope (see Keller et al. 2007) is currently working to improve on the 2MASS data set, at least for the southern celestial hemisphere. Skymapper is a 1.33 m telescope operated by the Australian National University (ANU) at Siding Spring mountain. It features an array of 32×8 megapixel CCDs mounted at the Cassegrain focus of the telescope to provide a 5.7 square degree field of view. Six coloured glass filters allow photometry in the u, v, g, r, i and z bands with peak throughput in the r band at around 650 nm. A proposed 'Five-Second Survey' consisting of at least 3 images of every field per filter is capable of providing photometry for stars of magnitude 8.5 through to 15.5 with a minimum accuracy in the g and r bands of $\sigma = 0.1$ mag, thus providing comparable sensitivity and accuracy to the 2MASS survey but with a wider wavelength coverage. With 36 observation epochs over a five year period, astrometry will also be possible from the Skymapper data, with proper motions as small as 4 mas year^{-1} detectable and position information accurate to within 50

mas. Hence in using the Skymapper data, Galactic Archaeologists have at their disposal 5 dimensions of phase space data as well as basic metallicity information for each surveyed star. It might therefore be argued that Skymapper represents one of the greatest leaps forward in the field of Galactic Archaeology to date and indeed the probing of the evolution and structure of the Galaxy ranks highly as one of the projects chief science goals.

1.4.2 Astrometric

Astrometry is essentially concerned with the determination of the 5 dimensions of phase space excluding radial velocity. Two data sets stand out as major contributions to the bulk of astrometry information currently available – that from the HIPPARCOS mission (ESA, 1997) and the data contained in the United States Naval Observatory (USNO) catalogues.

HIPPARCOS is actually an acronym for HIgh Precision PARallax COllecting Satellite, chosen in honour of the Greek astronomer Hipparchus whose main contribution to astronomy was astrometry, albeit in only two dimensions of phase space! The satellite operated from 1989 to 1993 providing high precision positional and proper motion data for more than 100000 stars. The final HIPPARCOS Catalogue consists of 118218 stars within a limiting magnitude of 12.4. The stars' positions on the celestial sphere, parallaxes and proper motions were determined to within median precisions of 0.77 mas, 0.97 mas and 0.88 mas yr^{-1} respectively. Additionally, photometry was determined for each star using an HIPPARCOS-specific visible pass band. The measurements were based on ~ 110 independent observations and are accurate to a mean value of 0.0015 mag. Based on these parameters, it is clear that the HIPPARCOS Catalogue represents an extraordinarily high precision source for phase space information and some photometry applications. The fundamental drawback to the data for Galactic Archaeology however is the small number of surveyed stars. This is remedied to some extent by the addition of the Tycho Catalogue (named in honour of Tycho Brahe's significant contributions to astrometry) wherein phase space data and photometry are presented for 1 058 332 stars with a median astrometric precision of 25 mas for all stars and photometry accurate to within 0.07 mag for B band photometry and 0.06 mag for V band photometry for all stars. It should also be noted that a new catalogue, Tycho 2 (Høg et al., 2000) has been

released, based on the same raw data as the original Tycho Catalogue but with astrometry available for 2.5 million stars and slightly higher parameter accuracy owing to a different reduction technique, yielding proper motions as small as 2.5 mas yr^{-1} detectable. In summary, the quality of the proper motion data from these three surveys distinguish them from other surveys, yet still, astrometric parallax – the particular specialty of these surveys – is inevitably limited by distance, so this dimension of phase space is not going to be available for far-flung structures of the Galaxy or extragalactic targets.

One of the largest astrometric catalogues available to date is the United States Naval Observatory A2.0 (USNO-A2.0) Catalogue (Monet, 1998). The catalogue is based on the same raw data as the USNO-A1.0 Catalogue (Monet et al., 1998) which was compiled using measurements of the Palomar Observatory Sky Survey I (POSS I) O and E plates for declinations north of -35^o and the UK Science Research Council (SRC-J) and European Southern Observatory (ESO-R) survey plates for declinations south of -35^o. The plates were scanned using the Precision Measuring Machine (PMM) at the U. S. Naval Observatory Flagstaff Station with precisions of 150 mas in positional information and 0.15 mag in the b and r band photometry afforded by using the ACT Catalogue over the Guide Star Catalogue (GSC) – as was used for USNO-A1.0 – for astrometric calibration. The ACT catalogue is based on the combination of the Astrographic Catalogue and the Tycho Catalogue and provides proper motion information about an order of magnitude more accurate than that contained in the original Tycho Catalogue (Urban et al., 1998). The final product is a catalogue of some 526 280 881 stars with RA, DEC and b and r band photometry to the accuracies already specified. The data is hence limited to the 3 positional coordinates of phase space (assuming distances are obtained from the photometry) and minimal photometric information but nevertheless, the sheer bulk of stars covered warrants the inclusion of the USNO-A2.0 Catalogue as a major source of raw data for Galactic Archaeology.

In addition to these surveys, there have been some noteworthy astrometry surveys in the intervening years, such as that utilized for the Second US Naval Observatory CCD Astrograph Catalogue or UCAC2 (Zacharias et al., 2004) wherein are presented position and proper motion data for 48 330 571 sources – mostly stars – with declination between -90^o and +40^o. The precision in position is estimated between 15 and 70 mas, depending on

source magnitude, and proper motions are determined to within 1 – 3 mas yr^{-1} for stars brighter than 12th magnitude and 4 – 7 mas yr^{-1} for those between 12th and 16th magnitude. Another, more restrictive survey, is the Southern Proper Motion Program III (Girard et al., 2004) which has catalogued ∼ 10.7 million objects in an area 3700^o or 1/11 of the entire sky, with proper motions determined in some cases accurate to 4 mas yr^{-1}.

The future for the collection of astrometric data is potentially an exciting one, but alas there are many setbacks faced by would-be missions. Already, two particularly promising, ∼ 40 million star surveys – the Full-sky Astrometric Mapping Explorer (FAME) and the German Interferometer for Multichannel Photometry and Astrometry (DIVA) – have been cancelled due to escalating costs and logistic difficulties. Disappointingly, this leaves some time until a new major astrometric survey is released. Nevertheless, two even more ambitious missions are scheduled for the next decade, one – JASMINE (the Japanese Astrometry Satellite Mission for INfrared Exploration) – is purely astrometric with regard to the dimensions of phase space it is intended to explore, the other, Gaia, will provide a measure of radial velocity as well and so is discussed amongst the 'kinematic' surveys in the next sub-section. The JASMINE mission (see Gouda et al. 2005), due for launch around 2014, is a 1.5 m space-based telescope under preparation by JAXA (the Japanese Aerospace Exploration Agency), designed to peer through the gas and dust of the galactic disk at a wavelength of 0.9 microns. The telescope will be sent into a Lissajous orbit around the Sun-Earth Lagrange point L2 from where it shall undertake astrometry of some 100 million Galactic disk and bulge stars (or such stars brighter than magnitude 14 in the z band) in the Galactic Latitude range $|b| \leq$ 4.0. As such it is not an all sky survey and it is of limited use for studying any other Galactic structures but nevertheless, with an accuracy of 10 μas for position and parallax data and 10 μas yr^{-1} for proper motions, the mission has the potential to produce a substantial catalogue of data, so far unequaled in depth, for the appropriate Galactic Archaeology work.

1.4.3 Kinematic

Kinematic surveys are perhaps the most useful survey type to the Galactic Archaeologist as they provide a complete description of each star's location in phase space and provide the

best chance for the identification of those structures sharing a similar evolutionary history. When this information is coupled with elemental abundance data, which is sometimes available from the same survey, the Galactic Archaeologist is endowed with the astronomical equivalent of the Rosetta Stone – the key to piece together the ancient lives of the Galactic populace. The only concern then is that the 'Galactic census' is far enough reaching to register enough substructure to give a representative view of the Galaxy in its entirety. Kinematic surveys are a relatively recent addition to the available data but, as we shall see, plans are afoot to see the kinematic dataset explode by the end of the next decade. Bland-Hawthorn and Freeman (2006) identify the Geneva-Copenhagen Survey of the Solar Neighbourhood (Nordstrom et al., 2004) as the first major kinematic survey – a study featuring kinematic data for 16682 nearby K and G dwarfs, with full 6D phase space data available for 14139 stars after combination with HIPPARCOS parallax data and Tycho 2 proper motions. Combined with photometry and metallicity data, the survey represents the means to study the precise structure of the local stellar neighbourhood and perhaps even identify any solar siblings that have migrated along similar paths to the Sun. Still, if enough data is to be had for the Galaxy on the broadest scales, the surveyed stars are going to have to be much more numerous and include those much less luminous!

Several such projects have either been completed or are in their final or preparatory stages. The most important completed to date is SEGUE – the Sloan Extension for Galactic Understanding and Exploration (Yanny et al., 2009). It is a moderate-resolution (R = 1800) spectroscopic survey of 240000 stars, spanning the spectral range from 390 nm to 900 nm, with the principal aim of aiding the study of the kinematics and populations of the Galaxy. The survey concentrates on fainter Milky Way stars of various spectral and luminosity classes with g band magnitudes between 14.0 and 20.3. The spectra it contains are from 212 regions of sky covering a total of 3500 square degrees, scattered over three quarters of the celestial sphere, though with an emphasis on low galactic latitudes. From the spectra, radial velocities have been obtained accurate to 4 kms^{-1} for stars brighter than g = 18 and better than 15 kms^{-1} for those brighter than g = 20. Photometries are also provided for u, g, r, i and z bands, as are astrometry data (accurate to 100 mas), and determinations of metallicity and other stellar atmosphere parameters where an SNR exceeding 10 per resolution element is

available. All things considered, the SEGUE data represent an excellent resource for Galactic Archaeology in all of the major Galactic substructures and may be used as a stand-alone resource with 6 dimensions of phase space as well as metallicity data all available from the one dataset. Still the number of stars included and the region of sky surveyed are still quite restrictive, and particularly in the Galactic halo or in cases where rare spectral types are used as tracers, there may simply not be enough coverage to properly identify and characterize substructure.

The RAdial Velocity Experiment (RAVE – see Steinmetz et al. 2006) should provide a substantial compliment to the SEGUE data at least for Southern Hemisphere stars. The project aims to obtain mid-resolution ($R = 7500$) spectra of up to one million stars using the Six Degree Field Multi-Object Spectrograph on the 1.2 m UK Schmidt Telescope at Siding Spring. The spectra are concentrated on the Ca-triplet region (841.0 nm – 879.5 nm) in an effort to determine metallicity as well as temperature and surface gravity for the surveyed stars, which will be chosen to have a magnitude in I band in the range from 9 to 12. Radial velocities will be determined to better than 3.4 kms^{-1}, marking a small improvement over the SEGUE data, while proper motions are included from external sources such as Tycho-2. As of the third data release (Siebert et al., 2011), 77461 individual stars had been surveyed, so the quantity of data is still considerably smaller than that available from SEGUE.

As the 'crescendo' to this review of stellar surveys, one particular project in the preparatory stages is set to supersede all the others – the ambitious Gaia space mission. A review of the Gaia mission is found in de Bruijne (2012), wherein the basic capabilities of the Gaia satellite are discussed. Gaia is set to measure the parallaxes, positions and proper motions of the one billion brightest stars in the sky – a truly astronomic endeavor! The stellar parallax measurements obtained by the satellite are expected to be accurate to within 25 μas for stars brighter than 15th magnitude. Accompanying this astrometric data will be low-resolution spectroscopic ($R \approx 11500$) and photometric data covering the range from 330 nm to 1000 nm allowing the radial velocity to be measured to within 1 – 15 kms^{-1} and metallicity and other parameters of the stellar atmospheres to be determined. The mission is planned to launch in 2013 with final results expected by 2021. By comparison to the other surveys already discussed, this mission represents a new generation for Galactic Archaeology. Not

only will cooler main sequence stars comparable to the Sun be visible out to beyond 10 kpc but more luminous stars will be visible throughout the Local Group and even in external galaxy clusters, taking Galactic archaeology to new places quite literally. This data will provide the representative survey of the Galaxy really needed to unravel its past and to study galactic evolution in a more general sense. In closing, it should however be cautioned that 2021 is almost a decade away and budget restraints may yet curtail the ambitious scale of Gaia, and even if they do not, the next ten years should be ones of productivity in Galactic Archaeology. Hence the more immediate, albeit less ambitious surveys will be the raw material utilized to push forward the boundary of knowledge in the mean time.

1.5 Dark Matter and the Predictions of ΛCDM Cosmology

The requirement for the existence of dark matter was first identified observationally by Fritz Zwicky (Zwicky, 1933). Upon studying the high velocities of member galaxies of the Coma Cluster, he realized that their orbits must enclose substantially more matter than could be attributed to visible galaxies alone in order for them to remain bound, hence implying the existence of some unseen, yet significant component of matter (Sahni, 2004). Rotation curves for individual galaxies were also subsequently shown to imply significant amounts of matter not associated with the luminous component of the galaxies (see Figure 1.3). Studies of the Cosmic Microwave Background (CMB) and the Universe's abundance of deuterium have indicated that ordinary baryonic matter – matter made up of baryons (i.e. protons and neutrons) – constitute a mere 4% of the total mass/energy content of the Universe and that non-baryonic matter must contribute a much larger fraction, $\sim 30\%$ (Sahni, 2004). Various properties and forms have been suggested for the elusive dark matter, of which the Λ Cold Dark Matter (ΛCDM – Λ being the cosmological constant) model has been the most successful at explaining the primordial 'power spectrum of density fluctuations' and its evolution to its present state.

In ΛCDM Cosmology, the dark matter's constituent particles exhibit a small, non-relativistic velocity dispersion (hence they are termed cold), having decoupled from baryonic matter and

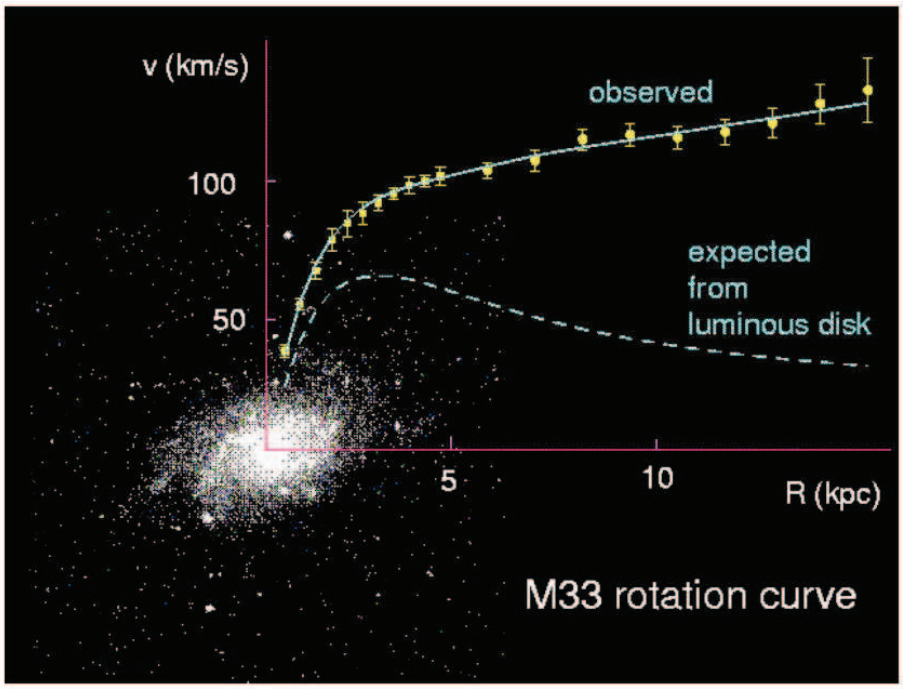


FIGURE 1.3: A schematic conveying the disparity between observed and expected galaxy rotation curves (Sahni, 2004)

energy after they had slowed to non-relativistic speeds (Sahni, 2004). Associated with the particles is a 'free-streaming distance' λ_{fs} that relates the mean distance traveled by the particles while still relativistic, before they slow to non-relativistic velocities. Since CDM cosmology already assumes 'cool' particles, this distance is not very long and so free streaming can only disrupt the primordial density distribution on small scales – hence giving rise to small-scale structure soon after the big bang. The opposite to this scenario is borne out by the Hot Dark Matter model, in which density inhomogeneities first appear on larger scales before fragmenting into the building blocks of individual galaxies – i.e. a top-down cosmology. CDM Cosmology in contrast is a bottom-up or hierarchical cosmology in which smaller structures appear first in the Universe and over time undergo gravitational clustering into larger structures such as clusters and eventually into the super-cluster-filament/ void frothy structure observed today. ΛCDM cosmology differs from the earlier standard CDM cosmology in that the mass density Ω_m is chosen to be 0.3 of the total mass-energy density (as opposed to 1) with Hubble constant (at z = 0) h ~ 70 $\text{kms}^{-1}\text{Mpc}^{-1}$, thus providing for a better fit to the shape of the current observed power spectrum.

With regard to the actual form of the dark matter, several possibilities have been proposed which can generally be summarized into two fundamental categories – the non-baryonic WIMPs (Weakly Interacting Massive Particles) and the baryonic MACHOs (MAssive Compact Halo Objects). In particular, the neutralino particle has been put forward as a strong contender for the CDM particle. The proposed neutralino is a WIMP with energy in the 100 – 1000 GeV range and is both stable and neutral so that it does not scatter light. Jungman et al. (1996) describes the neutralino as "the best motivated and most theoretically developed" of the WIMP particles and goes on to outline how it might be detected and how its abundance might be determined. Indeed, schemes are underway aimed at the direct detection of neutralinos on the Earth via their gamma-ray emitting interaction with nuclei in a detector – similar to the generation of x-rays in an x-ray tube. At least some of the missing matter however, is going to exist in the form of MACHOs such as distant white dwarfs, brown dwarfs and other low-luminosity bodies in the halo but there are theoretical and observational constraints on the percentage of dark matter made up of such baryonic matter. Theoretically, baryonic matter is not particularly successful at 'growing substructure' from the small primordial density fluctuations in the universe due to its strong coupling with radiation. On the other hand, if most of the dark matter is non-baryonic and thus not coupled to the radiation, this matter can clump together much earlier so that the comparatively small percentage of baryons simply fall into these ready made over-densities shortly afterward (Sahni, 2004). An example of observational constraints on the size of the baryonic component of dark matter is found in Alcock et al. (2000) where the low count rate of micro-lensing events in the direction of the Large Magellanic Cloud over a 5.7 year period is used to constrain the halo mass tied up in MACHOs to $\sim 20\%$. Whether MACHO or WIMP, the fact remains that the matter is dark and will not be directly observable to the astronomer – with the exception of the odd MACHO as more sensitive telescopes become available. Hence it would appear that, at least for the time being, the study of the Galactic dark matter will be restricted to the astronomical indirect measurement of the halo mass distribution (with several methods described in the next section) and the independent detection of WIMPs by particle physicists in the laboratory.

Before leaving this discussion of the ΛCDM cosmology, it must be noted that this model

is a 'best-fit' model only and is not without its own shortcomings. Two principal examples are outlined in Sahni (2004). Firstly, the model predicts an over abundance of halo substructure or subhalos which, if assumed to be accompanied by a luminous baryonic component, are not currently observed. Secondly, CDM predicts a so-called 'cuspy core,' with N-body simulations producing a halo density dropping off more steeply in the central regions than is observed such that ρ is proportional to r^{-1}. With regard to the first problem, Diemand et al. (2007) describes the results of "Via Lactea," the highest resolution simulation of the Galaxy to date, which predicts that the Milky Way halo should possess 124 subhaloes with masses comparible to the Galaxy's dwarf satellite galaxies, yet according to van den Bergh (2006), only ~ 10 such galaxies have been observed. This begs the question: where are the missing satellites? Diemand et al. (2007) goes on however to identify two studies which may hold the answer to this. A local group model by Kravtsov et al. (2004) suggests that galaxy formation will only initiate in the most massive ($> 10^9 M_\odot$) subhalos while Moore et al. (2006) find that only those subhalos forming very early on in the galaxy assembly process (at redshifts $z > 12 \pm 2$ i.e. before the epoch of reionization) with masses above the atomic cooling mass [1] Diemand et al. (2007) subsequently found that when the "Via Lactea" simulation was run backward through time only two subhalos were found to comply with each of Krastov and Moore's requirements – the same two in each case – which is a much better match to the number of satellites found to date in the Milky Way halo. Furthermore, Sahni (2004) highlights the fact that powerful winds from star formation and early supernovae may be responsible for clearing potential low mass satellites of what baryonic matter they might of had initially. With regard to the 'cuspy core' problem, Sahni (2004) goes on to draw attention to the fact that complex processes in galactic cores such as bar formation and baryon-dark-matter interactions are not treated adequately in the simulations to date.

[1] The atomic cooling mass M_H is the critical mass above which gas can cool efficiently allowing for condensation and subsequent fragmentation via excitation of the Lyman α transition of hydrogen. Assuming a virial temperature above $10^4 K$, $M_H \approx 10^8[(1+z)/10]^{-3/2} M_\odot$ which gives $M > 0.067 \times 10^9 M_\odot$ at z = 12 in order for a luminous component to develop (ref: Madau and Silk (2005))

1.6 Resolving the Matter - Methods for Measuring the Dark Matter Distribution

There are a variety of methods available to ascertain a broad picture of the dark matter distribution in galactic halos, of which three principal techniques are now discussed. The first method is that of gravitational lensing. In the last section, discussion was made of the use of microlensing to determine the percentage of dark matter attributable to MACHOs. Here we are concerned with strong lensing, where for instance a quasi-stellar object (QSO) is lensed by a foreground galaxy, and the contribution of substructure in the lense galaxy to the resulting flux distribution. Lense galaxy substructure in the form of dark subhalos will manifest itself as flux anomalies and milliarcsecond distortions in the image of the source object (Metcalf and Madau, 2001). A study into the feasibility of using such phenomena to map the subhalo distribution in the halos of lens galaxies is made in Riehm et al. (2008). Here, a test is proposed where a QSO is already known to be lensed on the arc second scale so as to ensure a suitably well-aligned, massive halo as the lensing object. Conditions are then favourable for the detection of subhalos in the $10^6 - 10^{10} M_\odot$ range based on the milliarcsecond distortions to the imaged QSO. Still, the study finds that the most realistic models currently available for the density distribution within typical subhalos do not bode well for the likelihood of their detection. Their density drops off with distance from the core at a more gradual rate than earlier models, yielding separations in the source image too small to resolve with the current generation of telescopes. Even if some subhalos are detectable, this method is not strictly in the realm of local cosmology, with inferences having to be drawn from the distant lensing galaxies as to how the halos of more local galaxies should be structured.

A much more direct method is proposed in the detection of gamma rays from annihilation of WIMPs such as from the chief contender – the neutralino. Diemand et al. (2007) goes so far as to produce an all sky map of the possible annihilation flux based on the "Via Lactea" simulation. They find that halo substructure should provide an overall boost to the annihilation signal from a galaxy when compared to a smooth halo distribution. Since the annihilation rate is proportional to the square of the density, a map of halo substructure may

soon be possible with the upcoming Gamma Ray Large Area Telescope (GLAST) which has a field of view covering approximately one sixth of the sky and sub-degree resolution at energies greater than 1 GeV. Based on the "Via Lactea" run, subhalo luminosity is predicted to be directly proportional to the mass of the subhalo, with even comparatively small examples visible to such a telescope when they are close to the Sun. The background noise from the Galactic centre is expected to hinder observations toward Sagittarius and in the Galactic Plane in general so observations may be best made looking away from these regions. Whatever the simulations may show, however, studies such as this are based heavily on assumptions and so, until such a time as observational evidence is available to support such ideas, it is important to focus on those methods that are independent of the precise nature of dark matter, relying only on its gravitational effects.

Such a method is found in the kinematic study of currently detectable halo structures such as the stellar streams found in the Andromeda halo and that of our own galaxy. Studies have been made into the feasibility of such methods for constraining the distribution of massive subhalos, notably by Ibata et al. (2002b) and Johnston et al. (2002) with some success predicted upon the availability of deeper 6D-phasespace surveys such as will be undertaken by Gaia. Ibata et al. (2002b) presents the results of N-body simulations and their implications for the possibility of inferring the presence of dark matter clumps from their heating effects on stellar streams. Specifically, a $10^6 M_\odot$ globular cluster is modeled with 10^4 particles and placed in a variety of smooth and lumpy galactic potentials both spherical and oblate. It is found that, assuming a spherical potential, the tidal stream from the cluster after a 10 Gyr period remains dynamically cold if the potential is smooth, with a width at its narrowest similar to the tidal radius of the initial cluster model. If the smooth halo is populated with subhalos so that a mere 1 % of the halo mass is tied up in this substructure, the emergent stream from the model cluster over the same time interval becomes significantly dynamically heated and hence physically wider and more diffuse. If, contrary to an earlier study (Ibata et al., 2001b) that will be discussed shortly, the Galactic halo is not spherical, but rather significantly oblate, the effect of the resulting precession of the cluster orbit can be distinguished from that of heating from subhalo disruption when the integrals of motion of the stream – the total energy and angular momentum (particularly the z-component) per unit

mass – are plotted with respect to each other. As a result, this particular method is indeed a possibility whatever the structure of the halo, but as is statistically determined using the 2MASS survey (Ibata et al., 2002a), too few stream members per disrupted globular cluster are available in the presently available data – and with incomplete phase space information – to make such streams detectable, with the stream from the disrupted Sgr Dwarf being the only one discernable from the data. Alas, the Sgr Dwarf is too large, with stellar velocities too dispersed for the subtle effects of heating from Galactic subhalos to be easily distinguishable within its stream. Hence, it is concluded that this method must wait for the Gaia data before the level of halo substructure can reasonably be determined. Johnston et al. (2002) concur with this conclusion but they do find that data for the Sgr Stream is sufficient to isolate some dynamical heating due to 'lumpiness' in the halo, although they point out that the observed scattering may be accounted for by the effects of the Large Magellanic Cloud (LMC) alone. Further, they predict that even an improved data sample for the stream is unlikely to improve on the deductive possibilities of the technique due to the alignments of the orbits of the two progenitor satellites. It is pointed out however, that future deep halo surveys may allow detection of colder extended streams from other Milky Way satellites that are relatively unaffected by the LMC and ideal for probing the halo substructure.

Whilst a study of the subhalo distribution in the Galactic halo may not yet be practical, initial investigations regarding the overall shape of the Milky Way halo and mass of the M31 halo have already taken place. Ibata et al. (2001b) determines with a high level of confidence that the Milky Way halo cannot be significantly oblate. The study used the Automatic Plate Measuring Facility halo carbon star (APM) survey (Totten and Irwin, 1998), which utilized Palomar Sky Survey plates and those from the UK Schmidt Telescope, to examine the distribution of carbon stars and their possible association with known halo structures. Carbon stars were chosen as the structure tracers of choice owing to their high intrinsic luminosity, rarity, distinct photometry and intermediate age, all of which act to make such stars easily identifiable and useful markers of recent Galactic accretion. Of the 75 carbon stars identified, 38 were found to lie within 10° of the great circle on the celestial sphere corresponding to the predicted Sgr Dwarf orbit and a further 28 within a similar proximity to the projected orbit of the Magellanic Clouds as represented in Figure 1.4 using a pole-count analysis. These

represent 6σ and 4σ overdensities respectively with regard to the statistically expected value of ~10 counts. To further illustrate the significance of these results, simulations were run which factored in the sky coverage of the survey plates employed and randomly positioned stars accordingly, and despite 1000 runs, no such overdensities were produced. Because the Sagittarius Stream delineated by these stars is observed as an approximate great circle, Ibata et al. (2001b) suggest that the orbit traced by the Sgr Dwarf must occupy a region of spherically symmetric gravitational potential since orbital precession must otherwise take place with orbital angular momentum no longer conserved. To better understand the evolution of the Sgr Dwarf responsible for the presently observed stream, the team represented the progenitor galaxy first as compact and then as a more loosely bound structure, evolving it each time within a Galactic potential with mass distribution:

$$\rho(R,z) = \rho_0(\frac{s}{r_0})^{-\gamma}(1+\frac{s}{r_0})^{\gamma-\beta}e^{\frac{s^2}{r_t^2}}$$

where r_0 is the core radius, r_t is the truncation radius and γ and β are the power law indices for in and outside of the core respectively. Two particular halo models were investigated based on observational constraints, each with slightly different parameters input into the mass distribution equation. Further to this, each of these halos was simulated for 3 different circular velocities (v_c – determined at 50 kpc) and 11 different values of the halo density flattening (q_m). In short, both progenitor models were evolved in 66 different versions of the Galactic potential in order to find the combination best fitting observations. It was found that those models with low flattening (e.g. $q_m \geq 0.9$) are a much better match to the observed carbon star distribution whilst those halos with $q_m \leq 0.7$ are refuted with high confidence. Hence Ibata et al. (2001b) conclude that the galaxy cannot be significantly oblate throughout the Galactocentric radii occupied by the orbit.

As a further example of the utility afforded by the study of halo stream kinematics, Ibata et al. (2004) uses the detection of 'giant stellar stream' stars in a kinematic survey using the DEep Imaging Multi-Object Spectrograph (DEIMOS) on Keck2 to obtain a mass estimate for the dark matter halo of the Andromeda Galaxy (M31). The measurement is made using a realistic galaxy model (Klypin et al., 2002) incorporating the disc and bulge components

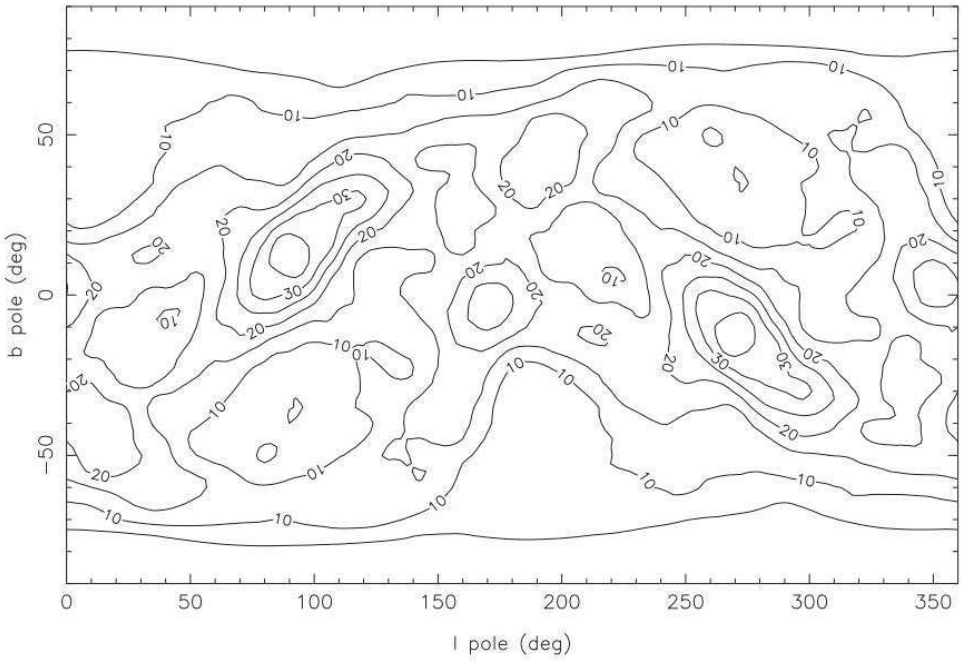


Figure 1.4: A pole-count analysis of the APM survey carbon stars where the number of carbon stars lying within 10^o of a given great circle are represented at the pole of the respective great circle using contour lines. The poles of the Sgr Dwarf stream are at l = 90^o/ 270^o, b = 15^o/-15^o and those for the Magellanic stream are at l = 170^o/ 350^o, b = -5^o/5^o. (Ibata et al., 2001b)

of the galaxy in addition to the dark halo. From the radial velocity gradient of stream stars in the 9 surveyed fields with confirmed stream components, a mass of $7.5^{+2.5}_{-1.3} \times 10^{11} M_{\odot}$ is obtained for the halo component located within 125 kpc of galactic centre.

1.7 The Pan-Andromeda Archaeological Survey

Up to this point we have concentrated our discussion of Galactic Archaeology on the Milky Way Galaxy. Due to our position within it, it has long been the only galaxy for which deep, comprehensive survey data has been available. But this is no longer the case, with the completion in 2011 of the Pan-Andromeda Archaeological Survey (PAndAS).

The origins of the PAndAS survey lie in the 25-square-degree survey of the disk and inner halo of M31 undertaken with the 2.5 m Isaac Newton Telescope (INT). The survey sought to identify the transition between the disk and inner halo, but identified extensive substructure

and culminated in the discovery of the Giant Stellar Stream (Ibata et al., 2001a). A comprehensive study of the stellar density and metallicity was undertaken by Ferguson et al. (2002) on a field-by-field basis using the INT data. In an effort to map the full spatial extent of the Giant Stellar Stream, the whole southern quadrant of the M31 halo out to 150 kpc was mapped using the 3.6m Canada-France-Hawaii Telescope (CFHT) on Mauna Kea, with an extension out to M33 more than 200 kpc from M31 (Ibata et al., 2007). With the wealth of substructure discovered, and given the large window of the M31 halo already covered, it was then decided to map the the remaining three quadrants out to 150 kpc with CFHT. This major undertaking marked the official birth of the PAndAS survey, with the initial results published in Nature in 2009 (McConnachie et al., 2009). In total, the survey incorporates some 400 square-degrees of sky covering most of the constellation of Andromeda, with extensions into Cassiopeia and Triangulum. It covers the entire halo of M31 out to 150 kpc as well as that of M33 out to 50 kpc. A map of the survey showing the extent of its coverage just prior to completion is presented in Fig. 1.5.

PAndAS is a deep photometric survey which has been undertaken in two bands, g and i using CFHT with the MegaCam instrument. MegaCam is an array of 36, 2048×4612 pixel CCD chips, covering approximately one square degree on the sky with a resolution perfectly matched to the 0.7" median seeing atop Mauna Kea. The MegaCam g and i band filters have a very similar throughput to the SDSS filters, with g spanning from approximately 4000Å to 5700Å and i from 6700Å to 8500Å see Gwyn (2010). A comparison of the two filter sets with the corresponding SDSS filters is illustrated in Fig 1.6. Each of the PAndAS fields reaches a depth of approximately magnitude 25.5 in g band and 24.5 in i band, though data incompletion is noticeable at these magnitudes.

Since the first PAndAS data has become available, a great many studies have been undertaken across a diverse range of topics concerning the structure of the M31 halo system. Possible tidal interactions have been investigated and numerous satellite galaxies, globular clusters and streams have been detected. McConnachie et al. (2009) details the discovery of "stars and coherent structures" that are very likely the remains of ancient dwarf galaxies long since cannibalized. They also identify the remnants of a recent encounter between M31 and

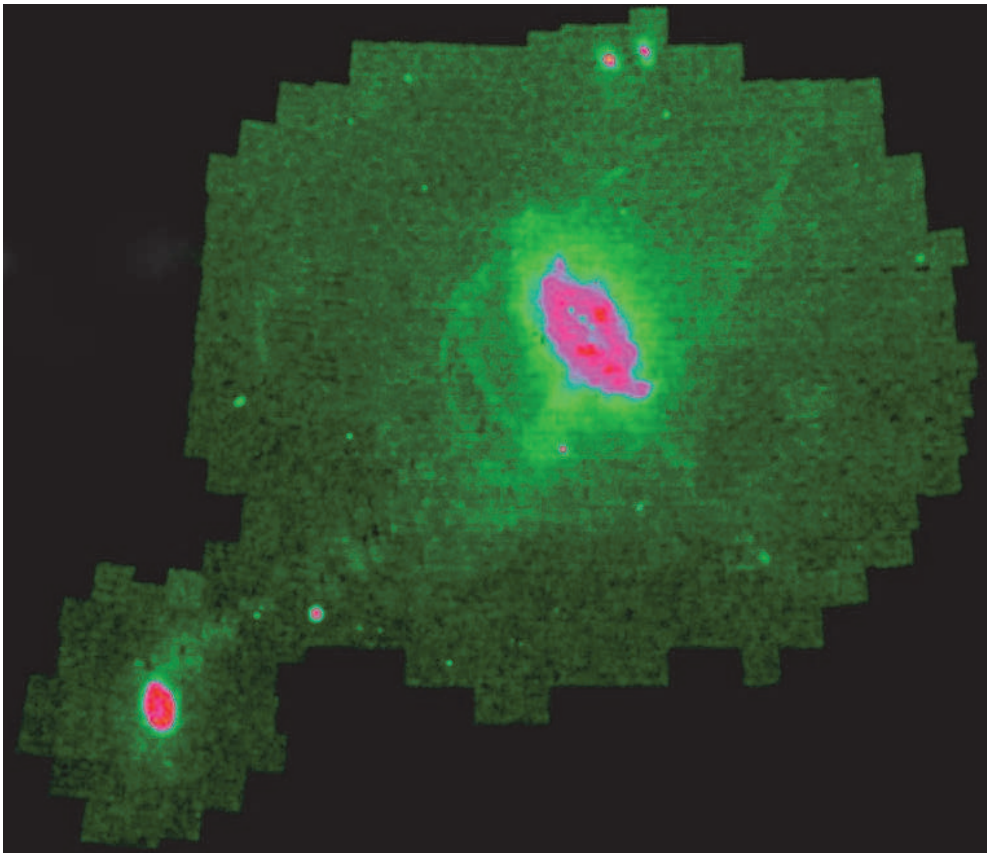


Figure 1.5: The Pan-Andromeda Archaeological Survey. This map of the PAndAS survey was generated just prior to its completion. It is generated from the most metal poor stars only and thus highlights the location of the various satellite dwarf galaxies. Also visible is the complex network of tidal streams marking the trails of past galaxy interactions. (McConnachie, 2010)

M33, and conclude that the wealth of halo structure present in the survey provides excellent evidence for the validity of hierarchical galaxy formation. The globular cluster system of the outer halos of both M31(Mackey et al., 2010) and M33 (Cockcroft et al., 2011) have been investigated, with a strong correlation identified between prominent streams and the locations of the known globular clusters. The presence of a large number of dark matter haloes has also been suggested by Carlberg et al. (2011), after a study of the 120 kpc long North West Stream found density fluctuations that should not arise in a smooth galactic potential. The locations and masses of known dwarf galaxies are also insufficient to explain the density variations. Many new satellites have also been discovered from the PAndAS survey, including Andromedas XVIII, XIX and XX (McConnachie et al., 2008), XXI and XXII (Martin et al., 2009) XXIII-XXVII (Richardson et al., 2011) and XXX (Irwin, 2012).

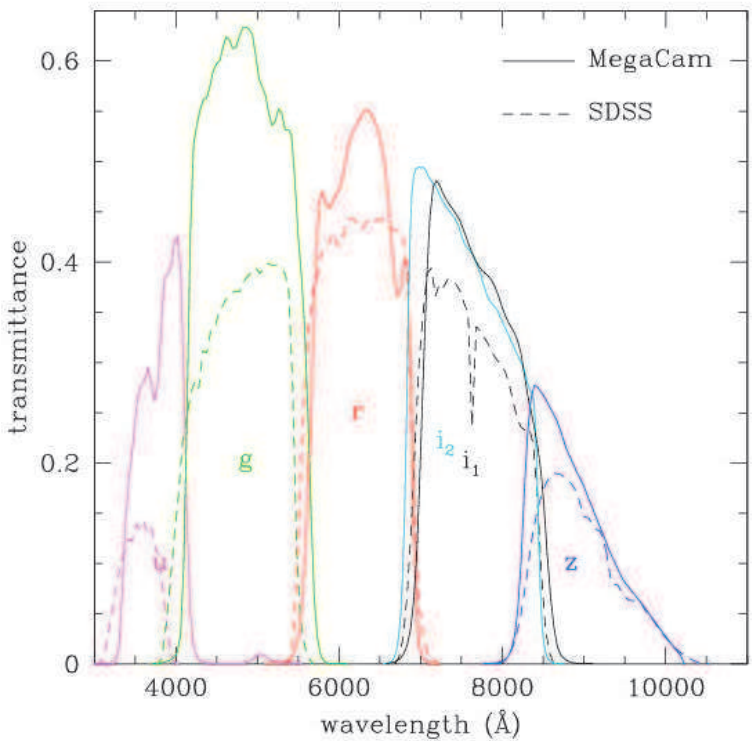


FIGURE 1.6: MegaCam Filter Set response compared with corresponding SDSS filter responses. (Gwyn, 2010)

In summary, PAndAS represents our first opportunity to study an entire galaxy halo system from an unobstructed view point outside the galaxy. Though the survey is now complete, its legacy has just begun as the many studies underway continue to unravel the secrets of Galaxy formation.

1.8 The Importance of Position

Due to the enormous distances separating us from all astronomical objects, with out considerable effort, the Universe remains a purely two-dimensional realm. For the local universe, we can use the Earth's orbit as a base line to measure the angular parallax of an object, and derive a distance accordingly. Further afield at the distance of M31 however, even the 300 million km diameter of the Earth's orbit is of little use in gaging distances, and hence we must turn to indirect means. Nevertheless, the prospect of PAndAS in three-dimensions is an exciting one which would allow us to constrain orbits much more accurately and fully

explore the matter distribution of the M31 halo.

At the distance of M31, there are several Standard Candles that could potentially be used as our distance gage. Two that are commonly invoked are Cepheid Variable and RR Lyrae stars. Indeed it was the Cepheid Variable that provided the first measure of the distance to the "Spiral Nebulae" thus establishing them as "Island Universes" external to our own Milky Way. The "Spiral Nebulae" targeted were of course M31 and M33 (Hubble, 1925). Nevertheless, Cepheid Variable stars are rare and require multiple epochs of observation to determine their light curves and hence use the Period-Luminosity relation to derive a distance. RR Lyrae stars are much more common than Cepheids but also much fainter and still require multiple observation epochs for distance measurements. Hence we turn our attention to the Tip of the Red Giant Branch (TRGB) standard candle.

The Red Giant Branch forms the backbone of the average metal poor galaxy and at the distance of M31, given the photometric depth of the PAndAS survey, it accounts for almost all of the stars observed to form any given structure. The TRGB standard candle is therefore applicable to even the most sparsely populated object and can even be used to gage distances at multiple points along streams. It also requires only one epoch of observation and hence is readily applicable to a large scale survey such as PAndAS. A study by Salaris and Cassisi (1997) has shown that Cepheid and RR Lyrae determined distances are consistant with those obtained using the TRGB to within 5%.

The TRGB standard candle arises due to the properties common to all Red Giant Branch stars in a particular mass and luminosity range as they approach the onset of core helium fusion. Such stars first enter the Red Giant Branch toward the end of their life when their source of core hydrogen is depleted. To fuse the helium ash left over in their core requires an immense pressure which the core density is as yet insufficient to produce, and so hydrostatic equilibrium is instead maintained by hydrogen fusion in a shell around the core. This process continues for the duration of the star's life on the Red Giant Branch, with the star gradually becoming more luminous as more and more energy is produced in the hydrogen fusion shell. Due to a relationship between the core helium mass and the luminosity of the star, the rate at which the luminosity increases grows as the star continues it's evolution (Salaris et al. 2002; Zoccali and Piotto 2000). This means that more stars will be observed at the fainter end of

the Red Giant Branch than at the brighter end, with the result that the luminosity function of a particular object is observed to follow a power law trend (Méndez et al., 2002).

As the star continues its evolution toward the bright end of the Red Giant Branch, the increasing buildup of helium ash in the core steadily increases the core density. In the particular mass range applicable to the TRGB standard candle, the stellar core succumbs to electron degeneracy before helium fusion can ignite and so all such stars have very similar core properties which in turn yields very similar energy outputs in the surrounding hydrogen fusion shell (see Iben and Renzini 1983, particularly Fig. 7). At the very instant of core helium fusion, the stars are at their most luminous and hence lie at the bright *tip* of the Red Giant Branch before undergoing the Helium Flash as the core pressure becomes sufficient for helium fusion to ignite. At this point, the stars contract and their luminosity diminishes as they enter life on the horizontal branch, resulting in a sudden truncation at the bright end of the luminosity function - i.e. the TRGB. One of the earliest detailed studies of the evolution of Population II stars toward the TRGB can be found in Hoyle and Schwarzschild (1955). A schematic summarizing this evolution is presented in Fig. 1.7.

In order to make use of the near-constant luminosity of the TRGB as a distance gage, it is usual to take measurements in the near infra-red region of the spectrum where dependence on metallicity is minimal. Indeed, Lee et al. (1993) show that for metallicities in the range $-2.2 < [Fe/H] < -0.7$, the absolute magnitude of the TRGB in Johnson-Cousins I band is constant to within 0.1 magnitudes, where in V band it varies by 1.3 magnitudes. This small variation is a consequence of the near-constant absorption in the stellar atmosphere in near-infrared wavelengths. Using very accurate I band photometry for the globular cluster ωCen, Bellazzini et al. (2001) derived the absolute magnitude of the TRGB as M_I as -4.04 ± 0.12. The MegaCam i bandpass is however significantly different to Johnson-Cousins I band and so for our PAndAS photometry, it is more suitable to use $M_i = -3.44 \pm 0.12$ as derived in Bellazzini (2008) for SDSS i band. This is justified given the similar throughputs of the MegaCam and SDSS i band filters as illustrated in Fig. 1.6.

At this point, having now introduced the burgeoning field of Galactic Archaeology, the PAndAS survey and the unique opportunity it has provided to explore galaxy evolution in

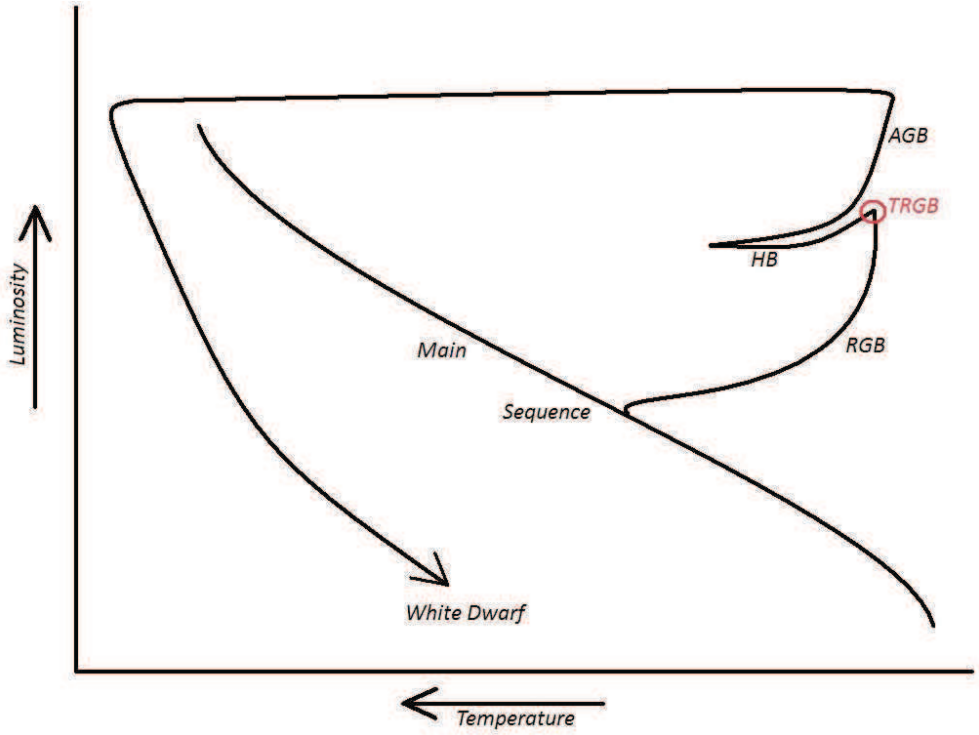


Figure 1.7: Schematic showing the evolution in temperature and luminosity of an intermediate mass, metal poor star. The star 'turns off' the Main Sequence onto the Red Giant Branch (RGB) after exhausting its core supply of hydrogen. The star expands and cools as it fuses hydrogen in a shell surrounding the core. At the onset of core helium fusion, the star has reached the Tip of the Red Giant Branch (TRGB) from which point it cools and contracts and enters life as a Horizontal Branch (HB) star. When it exhausts its core supply of helium it continues to fuse helium in a shell around the core once again becoming more luminous and following a path approaching the RGB asymptoticly. At this stage in its evolution the star is hence known as an Asymptotic Giant Branch (AGB) star. Note that stars spend only a tiny fraction of their life time as an AGB star in comparison to the time they spend as RGB stars and hence AGB stars are much rarer and so do little to diminish the contrast of the TRGB in an object's luminosity function.

action, we come to the specific aims of the research contained in this thesis. The highest ambition any research thesis can aspire to, is to make an original and significant contribution to the field furnished with clear and accurate results. This is indeed a major underlying motivation for this thesis, though of course, the contribution must inevitably be a specialized one in a field with such enormous scope. To this end, the focus is concentrated on the satellite system of M31. With the known satellite population of the M31 halo so greatly increased in the last 5 years, largely thanks to the PAndAS survey, the time is ripe for a renewed study of the three-dimensional spatial structure of the system. Such a study has the potential to shed

light on the past evolution of the satellite system as well as the distribution of matter within the M31 halo. It will also provide for a much needed comparison with the satellite system of the Milky Way and, when combined with velocity information, will facilitate a new, more accurate determination of the M31 halo mass. But before such a study can be undertaken, accurate satellite positions derived consistently via a single method are paramount. Hence, we turn our attention toward the development of a brand new algorithm for locating the TRGB – in particular, one that takes into full account all prior information available about the object's luminosity function.

“Remember that all models are wrong; the practical question is how wrong do they have to be to not be useful.”

George E. P. Box (1987)

2

Building the Framework for a new TRGB Algorithm

2.1 The RGB Tip Finding Problem

Given the broad applicability of the Red Giant Branch tip magnitude as a standard candle, it is not surprising to find that it is invoked frequently for distance measurements within the Local Group. Identifying the magnitude of the TRGB accurately however is not without its challenges, and hence many have resorted to simple “eyeball” measurements, read off from the Luminosity Function (LF) of the object in question. Such an approach is acceptable perhaps for distance measurements to a single, well populated object, but it falls short of the task when a consistent measurement is desired for numerous objects within the same group, or when the LF is poorly populated and the bright edge of the Red Giant Branch (RGB) is

not clear-cut. There is also the problem of ascribing an accurate measurement uncertainty to such an approach. It is therefore desirable to have an automated routine which, given the object LF, returns the most likely position of the RGB truncation with a measure of the uncertainty in this position.

The development of such a TRGB-finding algorithm is not without its difficulties however. Binned luminosity functions by their very nature suffer from Poisson noise, and thus star counts in two neighboring bins may differ by a significant factor. This is a serious problem when the primary task of our algorithm is to locate a sudden jump in star counts that might signal the bright edge of the RGB. It is however, less problematic for those objects exhibiting well populated RGBs. There is also the question of how the LF is effected by the stellar "background" contribution. After all, if the background LF contribution can be isolated perfectly and subtracted from the net LF for the object field, the RGB tip magnitude is simply the brightest non-zero bin remaining.

These issues have been approached in various ways over the years, and a more detailed literature review is provided in Chapter 3 (see Paper I Introduction), but relevant developmental landmarks are discussed below.

The first attempt at an automated tip-finding routine was introduced by Lee et al. (1993), who employed what is essentially an edge-finding technique, similar to what one might encounter in image processing. Instead of a 2D matrix however, the 'image' is the one-dimensional, binned luminosity function and the edge finding kernel is a one-dimensional Sobel kernel. The LF is convolved with this kernel, and peaks are produced at the locations where the discontinuity in star counts is greatest. With this method, they find that they can regularly recover the location of the tip, accurate to within 0.2 of a magnitude. Whilst this approach represents the first automated, repeatable TRGB finding method, the size of the uncertainties limits its usefulness. At the distance of M31 for instance, an uncertainty of 0.2 magnitudes corresponds to an uncertainty of approximately ±70 kpc in the distance. The edge-finding method of Sakai et al. (1996) improves on this technique significantly by addressing the luminosity function binning issue via Gaussian smoothing, so that stars no longer fall in one single bin but rather contribute to all bins.

As shall be seen in the next section, some similar techniques to these were experimented

with in the earliest days of the work contained in this thesis. Nevertheless, even with the inclusion of Gaussian smoothing of the LF, the issue of the Poisson noise is still a major concern with any pure edge-finding algorithm. Such techniques also ignore the distinction between object and background contributions to the LF and in so doing, throw away valuable information that might be used to constrain the location of the tip. Hence it is arguable that a model-fitting approach is superior to simple edge-fitting, in that it is less susceptible to the effects of Poisson noise, and more versatile with respect to the incorporation of prior knowledge.

The base method introduced in Chapter 3; Paper I makes use of these advantages by modeling both the background LF and signal or RGB LF separately. The RGB component of the model, whereby the RGB is approximated by a truncated power law, is inspired by Méndez et al. (2002). As in the base method of Chapter 3, they employ a maximum likelihood approach where the model parameters are updated at each iteration, and the likelihood of the model being correct given the data is evaluated. They assume a fixed functional form for the background bright-ward of the tip however, as well as a fixed value for the RGB slope. A more sophisticated approach is to fit the functional form of the background on a case-by-case basis using a suitable (and separate) background field. Likewise, the RGB slope can be set as a separate free parameter.

2.2 Early Trials of TRGB Finding Algorithms

During the preliminary, "pathfinding" phase of the development of the base algorithm of Chapter 3, various edge-finding algorithms were experimented with, some of which are now discussed. The relevant code can be found in Appendix A ('EdgeFinder7.f95' and 'RGB-PeakFinder6.f95').

The very first algorithms tested made use of artificial luminosity functions, where a deliberate 'kink' could be placed in the LF, and various algorithms used to recover the location of that kink. The kink was generated by summing two luminosity functions together, one displaced toward fainter magnitudes relative to the first such that the brightest non-zero bin in the second LF would mark the beginning of the RGB. In effect, the first LF simulated

the 'background' contamination whilst the second represented the luminosity function of the actual RGB. The prominence of the discontinuity at the beginning of the RGB could be controlled by adding a constant value to the RGB component of the LF. With the model LF set up in this way, it can then be populated with the desired number of stars.

The first edge detecting algorithms implemented on this artificial data were comparable to the kernel convolution method of Lee et al. (1993). Starting from the bright edge of the luminosity function, the gradient between each consecutive pair of magnitude bins was measured and stored. Once the whole LF had been scanned, the magnitude of the induced kink (i.e. the TRGB) was taken to be the fainter of the two consecutive bins for which the maximum gradient was recorded. An equivalent method replaced the measure of gradient with that of the angle subtended by each consecutive set of three magnitude bins. The central bin of the set producing the smallest angle was then taken as the magnitude of the TRGB. Both of these approaches are of course susceptible to confusing a noise spike in the LF for the 'TRGB' if the RGB truncation is not suitably prominent.

In an effort to lessen the sensitivity of the algorithm to Poisson noise, the possibility of fitting either a single polynomial or polynomial splines to the LF was investigated. If a suitable polynomial interpolation of the LF magnitude bins could be found, one would effectively have a smoothed LF, hopefully devoid of problematic noise spikes. The location of the tip could then be determined from the turning points in the second derivative of the fitted polynomial. This approach is fraught with difficulties however, as the degree of the polynomial or number of splines required depends on how smooth the LF is to begin with. If the TRGB truncation lies amidst noise spikes of comparable prominence, it will be very difficult to choose a polynomial which preserves the discontinuity in star counts at the TRGB whilst simultaneously smoothing out the surrounding noise spikes. Furthermore, such an approach inevitably requires a case-specific setup by the user and thus introduces significant biases into the measurement process.

In addition to the above tests carried out on artificial data, further experimentation was carried out on the luminosity functions of real objects. One fairly successful test incorporated a Gaussian smoothing, similar to that employed by Sakai et al. (1996). The dependence of the LF on binning was removed by replacing each star with a normalized Gaussian of some

user-specified width and summing all Gaussians together. The width of the Gaussian was chosen so as to produce the desired level of smoothing in the resulting LF, but in practice, this value should be dependent on the photometric error at the magnitude in question (as is the case in the base method of Chapter 3). Having produced this smoothed version of the LF, the magnitudes at which significant star-count discontinuities occurred could be identified from the function's second derivative. The results of applying this process to the colour-magnitude diagram of the M31 satellite galaxy Andromeda I (illustrated in Fig. 2.1) is presented in Fig. 2.2. It shows the smoothed Andromeda I luminosity function, created by replacing each star with a Gaussian of width[1] 0.2 magnitudes. The superimposed function in red is the second derivative of the LF, weighted by the star counts at that magnitude. It denotes inflection points in the LF gradient. It is clear to the eye that the inflection point corresponding to the TRGB is that identified at $i_0 = 20.77$, which would correspond to a distance to Andromeda I of 695 kpc. This is roughly consistent with the distances in the literature, though as shall be clear from later measurements, the degree of smoothing applied to the LF has shifted the tip brightward by (predictably) ~ 0.1 magnitudes.

As is clear from Fig. 2.2, this approach does provide a useful compliment to a simple "eyeball" measurement, providing the user with a computationally based readout of the most likely tip locations. Precisely *which* location is the correct location is left to the discretion of the user, and hence the method stumbles when the onset of the RGB is not clear to the naked eye. As was the case for the polynomial fitting, there also remains the problem that the degree of smoothing required will vary from object to object, and the measurement is thus biased by the user's choices and lacks consistency.

The results of all of these trials culminated in the realization that any method that is to perform consistently across all luminosity functions, will need to abandon the hope that the effects of Poisson noise can somehow be eliminated from the luminosity function. If the method is to perform consistently across all luminosity functions that might be encountered, the data contained therein should be accepted for what it is in its raw form, and a suitable model developed that best explains it. In this way, we can incorporate our expectations of

[1] Specifically the width between the two inflection points of the Gaussian

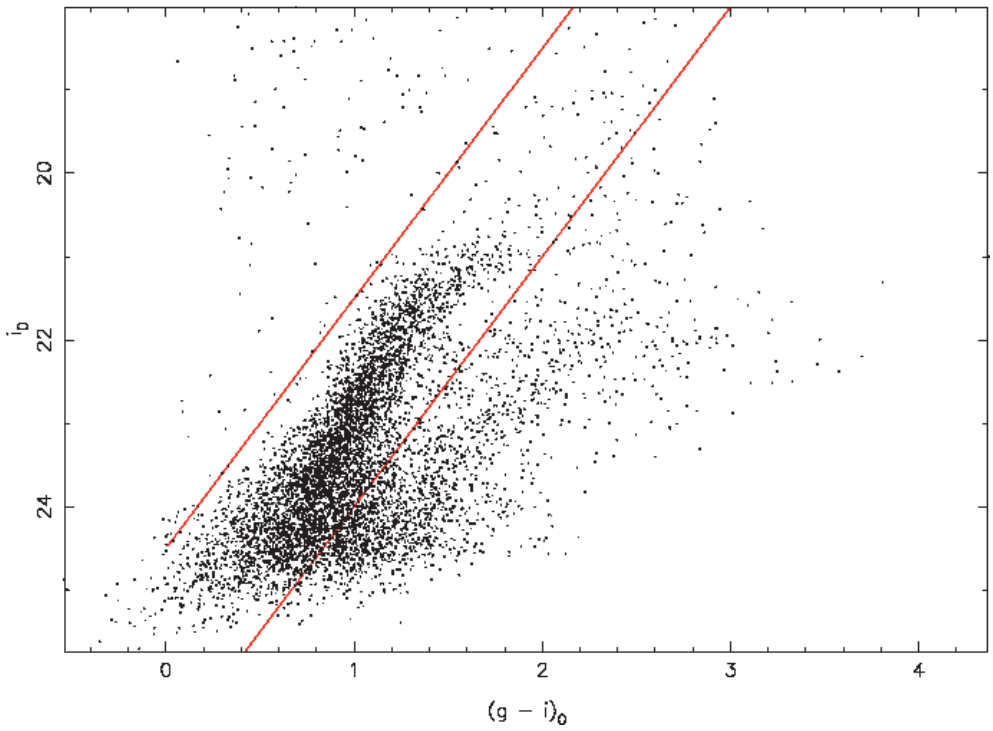


FIGURE 2.1: Colour-Magnitude Diagram for the dwarf spheroidal galaxy Andromeda I. It contains all stars in the PAndAS survey with i-band magnitude i_0 in the range $18 < i_0 < 26$, located within a circular field of radius 0.1° centered on Andromeda I. The red lines indicate the colour-cut imposed on the data in order to improve the contrast between the RGB and background stars. The Red Giant Branch of the M31 Giant Stellar Stream is visible as a second, faint RGB on the red side of the Andromeda I RGB. It can be seen much more prominently in Paper I, Fig. 2 where a larger field size is plotted.

what form the LF might take were it so well populated that Poisson noise was no longer an issue, and then let the data decide which version of the model approximates it best.

2.3 A Simple Maximum Likelihood Test

In order to gain a thorough understanding of how the RGB tip magnitude might be ascertained via model fitting, it is essential to "start simple." As the primary objective of any such algorithm is to locate a sharp discontinuity in star counts, we can begin by approximating the luminosity function with a simple step function, normalized so as to contain unit area. We can then set the 'step' at a particular location, populate the resulting LF with the desired number of star magnitudes and then attempt to recover the position of the step from those

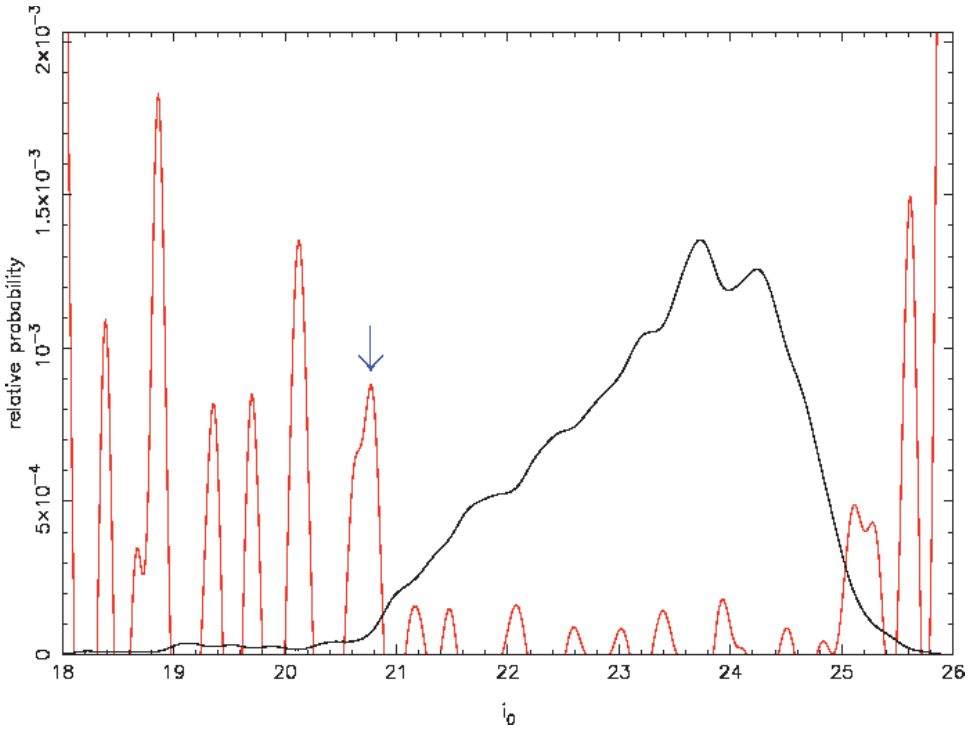


Figure 2.2: A smoothed version of the Andromeda I luminosity function, built using the stars plotted in Fig. 2.1 after rejecting those stars outside of the colour-cut indicated. The superimposed function in red is the second derivative of the smoothed LF, divided by the LF height, and indicates the location of inflection points in the LF gradient. A blue arrow points out the inflection point at $i_0 = 20.77$ corresponding to the RGB tip. Note that the falloff in star counts faint-ward of $i_0 = 23.5$ is due to data incompleteness.

star magnitudes. The code pertaining to this section can be found in the program 'spikes.f95' in Appendix A.

In using a maximum likelihood approach, we note that our model LF - i.e. our normalized step function - can actually be considered a probability distribution. It tells us the probability of finding a star at any given magnitude. Bright-ward of the step where we have only background stars, the probability of finding a star is lower but faint-ward of the step we have both background stars and stars from the object's RGB and so we expect more counts at a given magnitude. We can make use of this probability distribution both to generate our random sample of stellar magnitudes, and to recover the original state of the model that produced them.

To produce our random magnitude sample, we make use of a random number generator - but weight the likelihood of drawing a particular magnitude by our step function. This can be

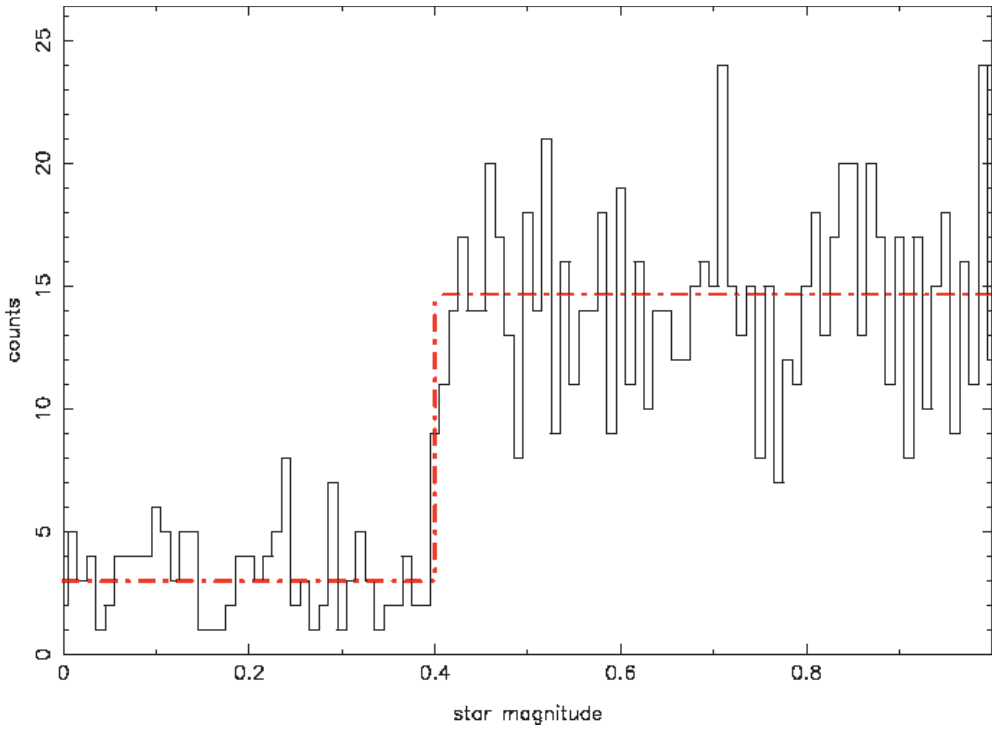


FIGURE 2.3: Random realization of a model luminosity function. It contains 1000 stars whose magnitudes were drawn at random from a step function probability distribution (shown as a red dashed line). The step is located at a magnitude of 0.4 and the signal and background components are in the ratio 0.7 : 0.3 respectively.

done by taking the integral of the step function, which gives us the cumulative area under the step function. This is equivalent to the probability of finding a star bright-ward of a particular magnitude. We then multiply the total area under the step function by a number between 0 and 1 generated by the random number generator and then find the magnitude corresponding to this value of the integral. In so doing we generate a 'random realization' of the step function in question. In practice, any number of random realizations can be generated from a single step function but the larger the sample, the better it will resemble the model that was used to produced it. Fig. 2.3 shows one such random realization. It contains 1000 stars and was produced from a step function where the fraction of stars contained in the RGB and background components were 0.7 and 0.3 respectively. The step was located at a magnitude of 0.4.

To recover the state of the model luminosity function which produced our artificial data, we need to test the likelihood of the model reproducing that data for a range of different

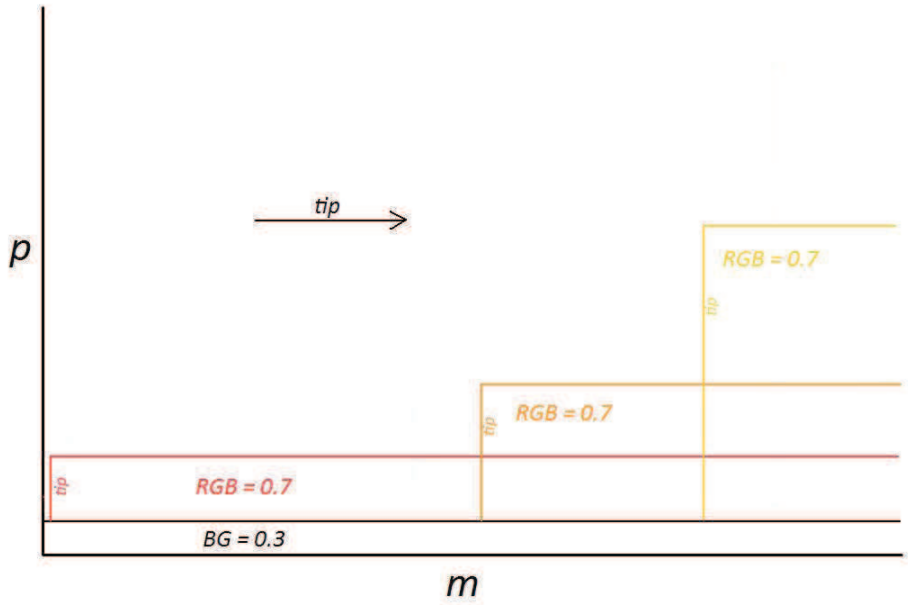


Figure 2.4: Schematic showing the scaling of the *normalized* step function as the step location is moved to fainter magnitudes.

step positions. Since our model LF is a probability distribution, the likelihood of the model producing a star at a particular magnitude is simply the value of the model at that magnitude. Thus if we are given a single star at a particular magnitude, we can find the version of the model that is most likely to have produced it by sliding the step location from the bright to the faint end of the LF and then noting which step location produced a maximum likelihood at the magnitude of the star. This process is illustrated in Fig. 2.4.

As can be seen in the figure, it is critical to the procedure that an equal area is preserved under the model for all step positions tried. The model represents all possible magnitudes at which a star can be observed and so the area underneath it should be unity. Likewise, the ratio of the background and RGB contributions to the model should remain constant. Given these requirements, the form of the probability distribution for the location of the step, as determined from a luminosity function containing a single star can be understood. As the step slides to fainter magnitudes, the RGB height rises to preserve equal area under the model. Hence the likelihood of the model producing the star at the observed magnitude grows at an increasing rate. Eventually however, the step slides past the magnitude at which the star is

observed and the likelihood for any subsequent step positions being correct therefore drops immediately to the background height.

In reality of course, one star does not constitute a useful luminosity function. We must therefore understand how probability distributions for the step location are produced when more than one star is present. To determine the likelihood of a particular state of the model producing two or more stars at their specifically observed magnitudes, we multiply together the likelihoods of the model separately producing each star. Equivalently, the final probability distribution for the step location is simply the product of the individual distributions generated for each star. This result is illustrated in Fig. 2.5 for a 10 star luminosity function.

Whilst the probability distribution of Fig. 2.5 (b) reveals a large uncertainty in the location of the step, little else can be expected from such a poorly populated luminosity function. More important is the fact that we have a reliable measure of the uncertainty in the most likely step position identified, assuming of course that our chosen model is a good approximation for the mechanisms responsible for producing the LF, as is the case here. Nevertheless, for well populated luminosity functions, the step location is generally locatable with considerable precision, as can be seen in Fig. 2.6 which has been generated from the luminosity function of Fig. 2.3.

From the simple tests presented in this section, the power, as well as the relative simplicity of the maximum likelihood model-fitting approach begin to emerge. Such an approach is particularly robust, as every single data point is taken into account every time a possible RGB tip location is considered. This greatly desensitizes any algorithm that implements it against the localized effects of Poisson noise. The approach is also versatile, being applicable to any model no matter how simple or complex. If we are to advance from a simple 'step' luminosity function however, we shall also need other tools at our disposal.

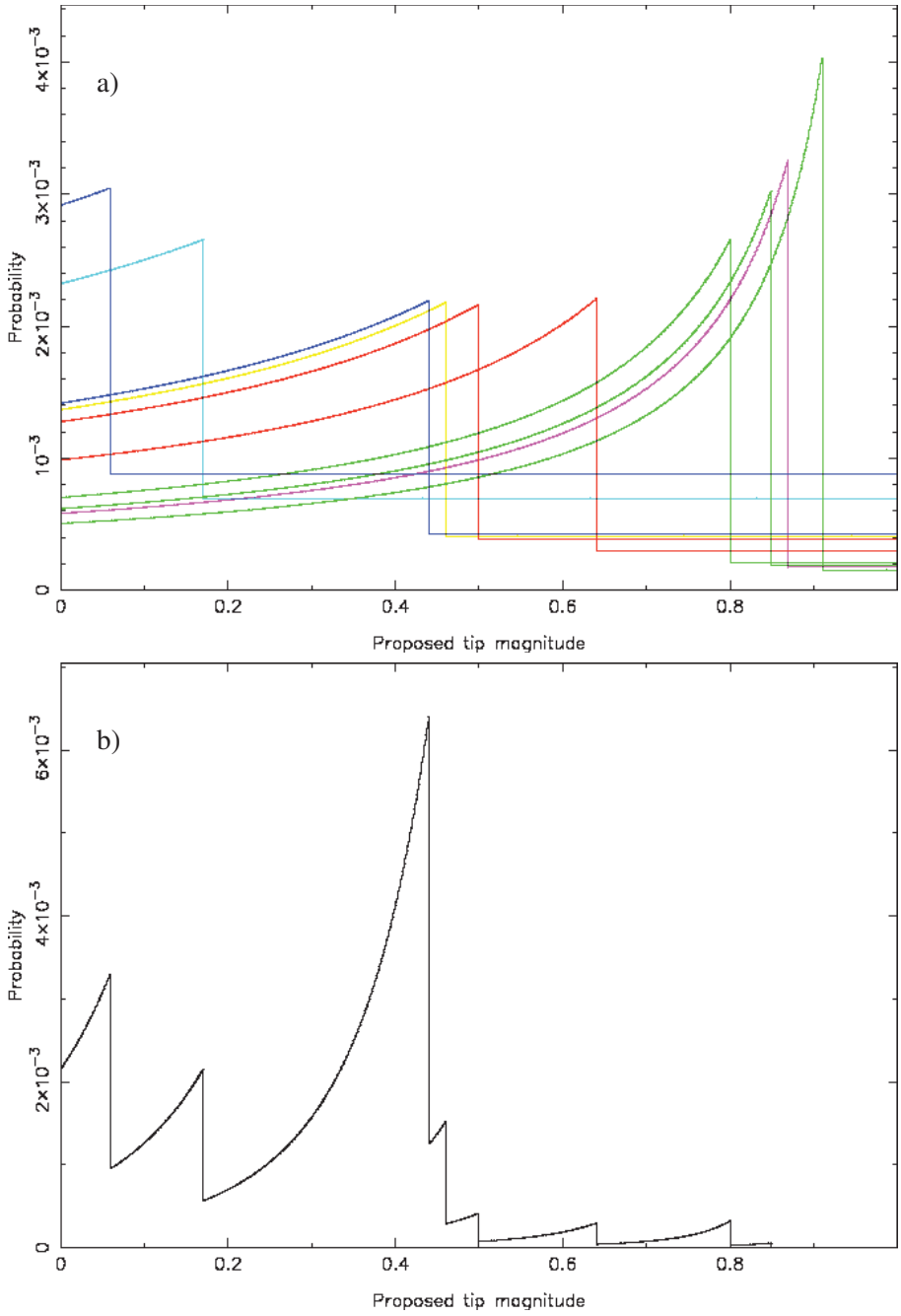


Figure 2.5: Probability distributions for the location of the step (i.e. RGB tip) in a 10 star, 'step' luminosity function. Fig. (a) shows the individual probability distributions resulting from each star whilst Fig. (b) shows the product of these individual distributions which forms the probability distribution given the whole luminosity function. As in Figures 2.3 and 2.4, the model which produced the stars consisted of RGB and background contributions in the ratio of 0.7 : 0.3 and the step was located at a magnitude of 0.4. Note that the total area under all distributions in both (a) and (b) is unity, with all possible locations of the step position represented. This follows from the normalization of the model for all step positions.

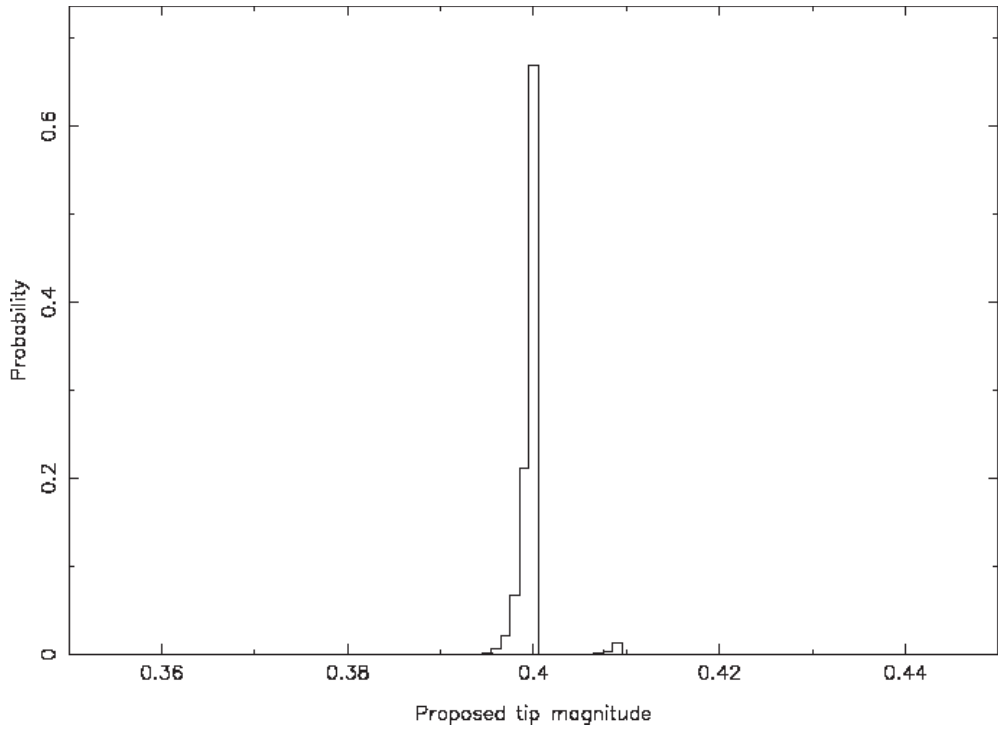


FIGURE 2.6: Probability distribution for the location of the step in the 1000 star 'step' luminosity function of Fig. 2.3.

2.4 The Markov Chain Monte Carlo Method

The tests of §2.3 approximated an object's luminosity function with a single-parameter step function. All possible states of such a model are obtainable by changing only the step location. As such, it was not computationally intensive to calculate the likelihood for the model at every possible step location, even for very small increments in the step location and for very densely populated luminosity functions. A better model however, would take into consideration other aspects of the LF, such as the background to RGB contribution ratio and the slope of the RGB component. But the number of possible states of the model increases exponentially with the number of free parameters, and hence so too does the computation time. Hence the deterministic approach used above quickly becomes impractical as we add increased complexity to our model. For this reason it is advantageous to adopt a Monte Carlo method, which builds up a picture of the likelihood of the state space of the model by taking samples of the model likelihood at randomly chosen parameter values. Our method of choice is the Markov Chain Monte Carlo (MCMC) Method.

The MCMC works via the construction of a Markov Chain. Named for the Russian Mathematician Andrey Markov, it is essentially a statistically representative sample of all possible states of a model, given a specific set of data. The sample is a *chain*, in the sense that each newly chosen parameter set is affected by the previous. The creation of the chain is however a 'memoryless' procedure, with each newly chosen state having no dependence on past states in the chain, with the exception of the state that immediately preceded it. To properly represent the differing likelihoods of various states, the chain should be created in such a way as not to prohibit the possible recurrence of certain states. The extent to which the chain explores the full state space of the model is of course dependent on the length of the chain. One must therefore be careful to insure that the chain is indeed long enough to be a true representation of the model states and their likelihoods. A detailed overview of Markovian models with examples can be found in the online reference work Meyn and Tweedie (1993).

There are various ways that a Markov Chain can be constructed, but the one employed for the analysis contained in this thesis makes use of the Metropolis-Hastings algorithm (Metropolis et al. 1953, Hastings 1970). An excellent introduction to this algorithm, with examples can be found in Gregory (2005). The algorithm essentially provides an 'intelligent' random walk through the state space of the model by preferentially choosing steps toward model parameter sets that have a higher likelihood. To initiate the algorithm, one must first choose a 'jumping distribution' as well as a starting value, for each of the model parameters. The jumping distribution is a probability distribution that defines how likely a jump to any given parameter value is, given the present value of the parameter. For our implementation of the MCMC, a Gaussian jumping distribution is chosen due to its symmetry and preference for jumps to nearby parameter values. The appropriate width of the distribution for each parameter is chosen through experimentation.

With the jumping distributions and initial parameter values defined, new parameter values are proposed and the model likelihood for those values is calculated. If the likelihood is greater with these new parameter values than the current model likelihood, the proposed parameter values are automatically accepted and thus define the next model state in the chain.

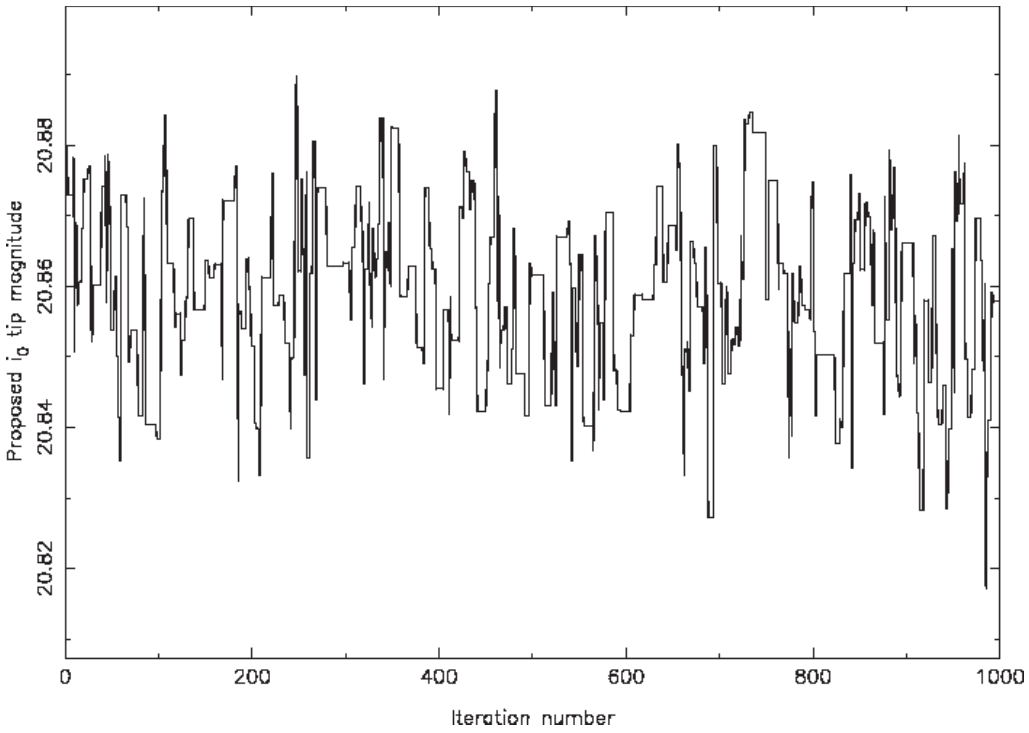


FIGURE 2.7: The random walk in the RGB tip magnitude over 1000 iterations, resulting from the application of the MCMC to the Andromeda I luminosity function. The model LF being sampled consisted of two free parameters, the RGB tip magnitude and the slope a of the power law chosen to approximate the RGB component of the LF. (Generated using '*BayesianTRGB_ANDI.f*95' - see Appendix B)

If the model likelihood is smaller with these new parameter values, the probability of accepting them is weighted by the ratio r of the two likelihoods, with $r = \frac{\mathcal{L}_{proposed}}{\mathcal{L}_{current}}$. Specifically, a random number d is drawn between 0 and 1 and the proposed step is taken only if $d \leq r$. An example of the random walk that results from many iterations of this process is provided in Fig. 2.7, which shows the sampled values of the RGB tip parameter in the Andromeda I LF in a Markov Chain of 1000 iterations. Note that this figure has been generated with prior knowledge of the approximate magnitude of the tip, and hence there is no obvious lead-in period. In practice, it is advantageous to excise the first thousand or so iterations (depending on the step size and initial starting value) from the sample, to remove the initial walk to the general location of the best-fit parameter value.

The single-parameter random walk exemplified by Fig. 2.7 is of course, only a partial description of the Markov Chain. It ignores what any other parameters are doing at each

iteration. It is therefore useful to plot each pair of parameters against each other to observe any correlation between them. This said, whilst many parameters might be coupled, our ultimate goal is to obtain the absolute probability of particular values of each parameter without regard for what any other parameters are doing, thus creating a probability distribution for the parameter. This is achieved by marginalizing over the other parameters. If our model were to have n free parameters, the probability of one of the parameters having some particular value is equal to the integral under the $n-1$ dimensional surface relating the probability distributions of all the other parameters at that value. In practice, it is actually very straightforward to create the probability distributions of individual parameters of a Markov Chain, by simply representing the number of occurrences of each parameter value as a histogram. This process is illustrated in Fig. 2.8.

It is clear from Fig. 2.8 (b) that 1000 iterations of our MCMC algorithm is insufficient to fully explore the likelihoods of the various model states. Indeed, for the probability distributions presented in Papers I and II, hundreds of thousands of iterations were found necessary before the distributions were as smooth as one would expect to achieve via a deterministic approach. With the underlying model luminosity function used for the analysis in each of these papers being defined by only two *free* parameters, the MCMC approach might be described as 'overkill' for the model used. It must be stressed however that this approach has been built into the algorithm from the outset in order to facilitate added model complexity with only minimal coding changes.

2.5 The Bayesian and the Frequentist

Statisticians are of two minds with regard to the nature of probability and how it should be calculated. The traditional 'Frequentist' view, as the name suggests, holds that the probability of an event is related to the frequency with which it occurs over a large number of samples. Strictly speaking, it is the limit in that frequency as the number of samples goes to infinity and therefore it can never be calculated exactly. Frequentists hence speak of confidence intervals, an interval over which they have some degree of confidence that an event

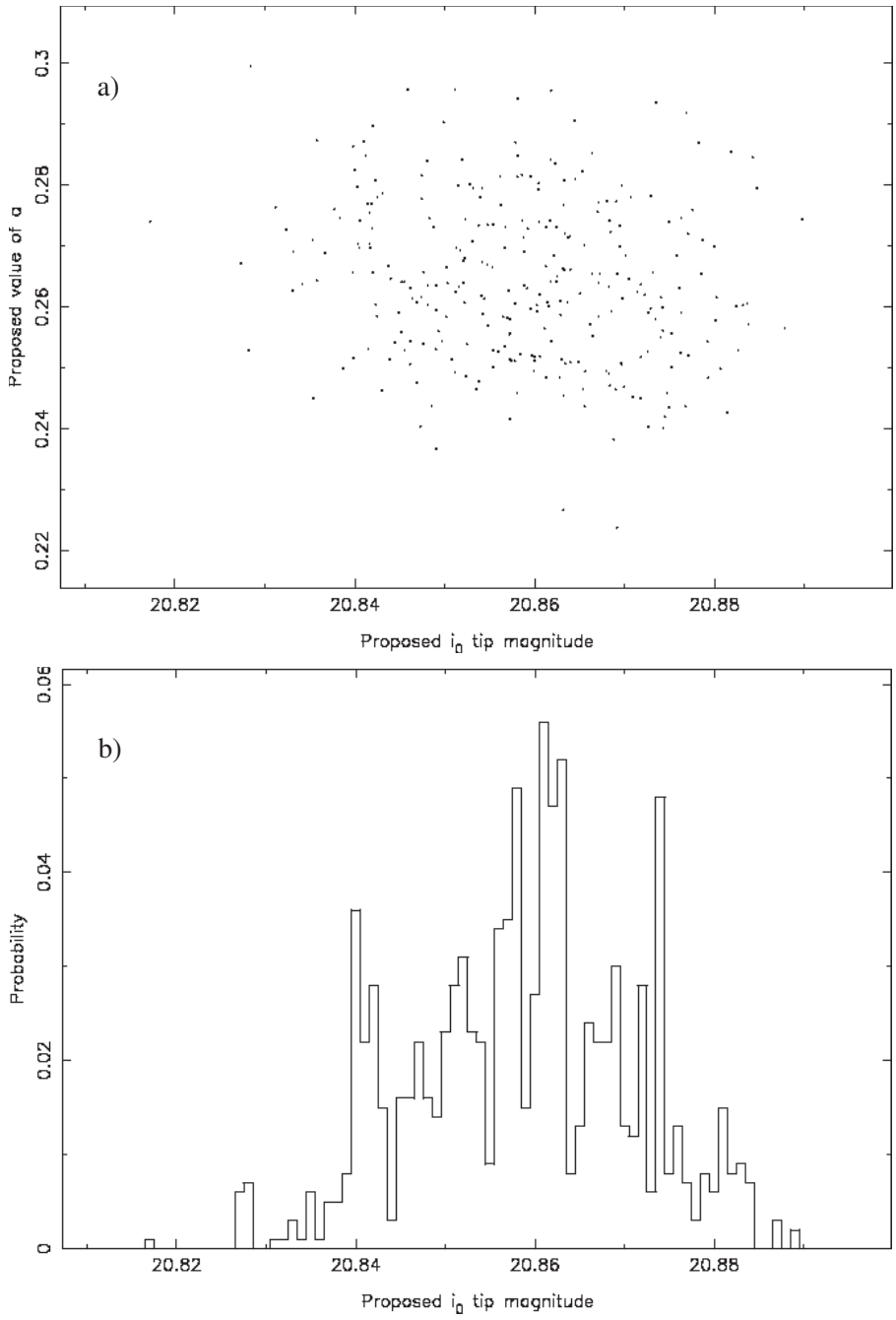


FIGURE 2.8: Generating the probability distribution for a single parameter via marginalization. Both Figures (a) and (b) are generated from the same Markov Chain as Fig. 2.7. Fig. (a) shows the value of the RGB slope *a* corresponding to each value of the RGB tip. As the fitted model LF is defined solely by these two parameters, the figure portrays all information contained in the Markov Chain. Fig. (b) portrays the binned probability distribution in the RGB tip position. The height of each bin reflects the number of data points in Fig. (a) recorded in the magnitude range represented by each bin. It is clear that a longer chain is warranted if it is to form a truly representative sample of the state space of the model. (Generated using '*BayesianTRGB_ANDI.f*95' - see Appendix B)

will be observed. Beginning with the work of the mathematician Thomas Bayes (Bayes and Price, 1763), a new perspective on the nature of probability began to emerge however. It essentially regarded probability as a subjective construct, the distribution of which depended on the prior knowledge available to the investigator. Herein lies the power of the Bayesian technique. The probability of an event can be weighted by our knowledge of the laws that govern it, thus producing a probability distribution which characterizes the credibility of the measurement.

Central to Bayesian Inference is Bayes theorem. The modern form of Bayes Theorem has it's origin in the work of Pierre-Simon Laplace (Laplace, 1812). It is commonly written as follows:

$$P(A \mid B) = \frac{P(B \mid A) \times P(A)}{P(B)} \tag{2.1}$$

where $P(A|B)$ is the probability of A being true, given observation B; $P(B|A)$ is the probability of B being true, given A; and $P(A)$ and $P(B)$ are the absolute probabilities of A and B respectively. As an example, suppose someone comes across a set of 10 old coins in the attic, apparently all identical. Upon closer inspection however, they note that one of them has a particular mintmark. After some investigation they determine this mintmark to be quite rare, being found on only 1% of coins of that type. They also learn however, that 30% of all coins of the type are fakes, and that 90% of the fakes bear the mintmark. So what is the probability that they have found an authentic example? We have

$$\begin{aligned} P(real \mid mintmark) &= \frac{P(mintmark \mid real) \times P(real)}{P(mintmark)} \\ &= \frac{0.01 \times 0.7}{0.01 \times 0.7 + 0.9 \times 0.3} \\ &= 2.5\%. \end{aligned} \tag{2.2}$$

Note however that if the finder of the coin had not done their research, they would have had to assume that approximately 1 in 10 coins of the type bear the mintmark, and that the authenticity of the coin was not in question. Prior knowledge has thus completely changed their perspective.

We can equally well apply Bayes Theorem to our model-fitting RGB tip finding algorithm

thus:

$$f(M_p \mid D) = \frac{f(D \mid M_p) \times f(M_p)}{f(D)}, \tag{2.3}$$

where $f(M_p|D)$ is the 'posterior' distribution in model parameter p after taking into account the data D. The function $f(D|M_p)$ is the distribution of likelihoods obtained from the model for the various values of p, $f(M_p)$ is our 'prior' or initial assumption as to the distribution in p before accounting for the data, and $f(D)$ is the probability distribution for the data points - i.e. the luminosity function. Since $f(D)$ is constant regardless of the model parameters, it scales all probabilities by the same amount and hence is dealt with the by the normalization of the posterior distribution. In the examples of §2.3 and §2.4, a 'uniform' prior has always been assumed, such that $f(M_p) = c$, a constant. If we are to better constrain our posterior distributions however, we should incorporate all prior information we have on the object studied.

2.6 Prior Information

As shall be seen in the next two chapters, prior information has been incorporated into our TRGB algorithm in a variety of ways. Some of our prior knowledge of the objects under study has been applied in the form of an independent 'prior' distribution which is multiplied by the likelihood distribution as per Eq. 2.3, while other prior information has been built into the model luminosity function directly. The model LF by it's very nature reflects our assumptions as to the form it would take were it populated with an infinite number of stars, thereby eradicating any Poisson noise. We assume a truncated power law for the RGB and fit the background component directly with a polynomial. This process is discussed in detail in Paper I, but it is important to realize that the model LF applied to each object is 'custom built' for that object. The likelihood distribution generated for each parameter thus already incorporates prior information we have for the object.

To expand on this, it shall be seen in the next two chapters that the areas under the RGB and background components of the model LF are set based on the average stellar density in the object field as compared with that of a suitable 'background' field. The background field is chosen so that it lies very close to the object field, and is usually in the form of a large

rectangle centered on the object, but with the object field subtracted so that the object RGB contributes negligibly to the LF of the background field [2]. In so doing, we are effectively assigning a prior constraint on how many stars we expect to find at a given magnitude on either side of the RGB tip. If the average stellar density in the background field is very low compared with that of the object field, than we do not expect to find very many stars that are not true members of the object RGB.

With the base method presented in Paper I, the ratio of the RGB to background model contributions is held constant for all stars. When the density matched filter is applied in Paper II however, individual stars are assigned a weight that reflects our prior expectation as to the probability of their being true object members. Using this information, we can tailor our model LF not only to the object in question, but to the specific star in question. The ratio of the RGB to background model contributions is calculated for each star individually, based on its position within the object's density profile.

In addition to the prior information that has been incorporated directly into the model LF, additional prior information has been incorporated in the conventional sense, i.e. as a prior probability distribution. The default prior distribution is a uniform prior, an assumption of equal probability for all parameter values. Various prior distributions were experimented with in the initial development of the base algorithm, each one devised so as to put some constraint on the distance at which the object can be found. A Gaussian distribution could be chosen for example, with the center of the distribution corresponding to the distance of M31 and the width reflecting our assumptions as to the extent of the halo. Alternatively, a flattened Gaussian could be chosen so as to yield equal probability over some desired distance range whilst cutting off sharply outside of that range. The priors on the tip magnitude m_{TRGB} and RGB slope a for both papers have been simple top hat functions, such that $19.5 \leq m_{TRGB} \leq 23.5$ and $0 < a < 2$ are predicted with equal probability whilst values outside of those ranges are assumed impossible. A more subjective prior is assumed in Paper II for the object

[2]Note that in Paper I, the background fields used were long stripes running along the Galactic Latitude of the object, as can be seen in Paper I, Fig. 3. It was later determined however that smaller fields provided a better approximation to the localized background contained in the object field. Hence, the background fields used for Paper II were rectangles of approximately $2^\circ \times 1^\circ$ oriented as before with the longer axis parallel to lines of equal Galactic Latitide.

distance however, namely the density profile of a simple spherical halo along the line of sight passing through the object (see Paper II, Fig. 6).

The net result of the inclusion of the prior information discussed above, is that we transform our simple maximum likelihood technique of §2.3 into an 'educated' tip finding algorithm. We effectively combine the best of both worlds by automating the tip finding process, but at the same time imparting some of the intuition to the tip finding algorithm that we would use if estimating the tip magnitude from the LF by eye. Such is the power of Bayesian Inference, that we can combine the information obtainable from one lone data sample with all other knowledge we can possibly infer about the circumstances which produced it.

“Every statistician would be a Bayesian if he took the trouble to read the literature thoroughly...”

D. V. Lindley (1986)

3

Paper I: A Bayesian Approach to Locating the Red Giant Branch Tip Magnitude. I.

Paper I Preface

This chapter presents the first of three papers which, together, represent the very heart of the thesis. They are perhaps best thought of as a 'trilogy,' as each one flows naturally into the next and, though written to stand as independent contributions in their own right they are best understood in the light of their companion papers. Unlike papers II and III however, Paper I is primarily a 'methods' paper. It lays the foundations for a new approach to the long standing Tip of the Red Giant Branch problem which is further developed in Paper II where it is applied to the majority of the M31 satellites. A preliminary analysis of the satellite distribution is provided in that paper with a study of the halo density profile but the real 'fruits of the labour' follow in Paper III which contains a thorough analysis of the satellite spatial distribution which has led to some very interesting results. Whilst papers II and III shall likewise be introduced with their own preface, chapters 1 and 2 arguably form the real 'preface' for this paper and hence it seemed apt that this paper should be introduced with a discussion of its place within a broader picture. Note also that the principal programing code pertinent to the material presented in this paper can be found in Appendix B.

The Astrophysical Journal, 740:69 (11pp), 2011 October 20 doi:10.1088/0004-637X/740/2/69

A BAYESIAN APPROACH TO LOCATING THE RED GIANT BRANCH TIP MAGNITUDE. I.

A. R. Conn[1,2], G. F. Lewis[3], R. A. Ibata[2], Q. A. Parker[1,4], D. B. Zucker[1,4], A. W. McConnachie[5], N. F. Martin[6], M. J. Irwin[7], N. Tanvir[8], M. A. Fardal[9], and A. M. N. Ferguson[10]
[1] Department of Physics & Astronomy, Macquarie University, Sydney 2109, Australia
[2] Observatoire Astronomique, Université de Strasbourg, CNRS, 67000 Strasbourg, France
[3] Institute of Astronomy, School of Physics, A29, University of Sydney, Sydney, NSW 2006, Australia
[4] Australian Astronomical Observatory, Epping, NSW 2121, Australia
[5] NRC Herzberg Institute of Astrophysics, Victoria, British Columbia V9E 2E7, Canada
[6] Max-Planck-Institut für Astronomie, D-69117 Heidelberg, Germany
[7] Institute of Astronomy, University of Cambridge, Cambridge CB3 0HA, UK
[8] Department of Physics and Astronomy, University of Leicester, Leicester LE1 7RH, UK
[9] Department of Astronomy, University of Massachusetts, LGRT 619-E, Amherst, MA 01003-9305, USA
[10] Institute for Astronomy, University of Edinburgh, Royal Observatory, Edinburgh EH9 3HJ, UK
Received 2011 March 1; accepted 2011 July 15; published 2011 September 30

ABSTRACT

We present a new approach for identifying the tip of the red giant branch (TRGB) which, as we show, works robustly even on sparsely populated targets. Moreover, the approach is highly adaptable to the available data for the stellar population under study, with prior information readily incorporable into the algorithm. The uncertainty in the derived distances is also made tangible and easily calculable from posterior probability distributions. We provide an outline of the development of the algorithm and present the results of tests designed to characterize its capabilities and limitations. We then apply the new algorithm to three M31 satellites: Andromeda I, Andromeda II, and the fainter Andromeda XXIII, using data from the Pan-Andromeda Archaeological Survey (PAndAS), and derive their distances as $731^{(+5)+18}_{(-4)-17}$ kpc, $634^{(+2)+15}_{(-2)-14}$ kpc, and $733^{(+13)+23}_{(-11)-22}$ kpc, respectively, where the errors appearing in parentheses are the components intrinsic to the method, while the larger values give the errors after accounting for additional sources of error. These results agree well with the best distance determinations in the literature and provide the smallest uncertainties to date. This paper is an introduction to the workings and capabilities of our new approach in its basic form, while a follow-up paper shall make full use of the method's ability to incorporate priors and use the resulting algorithm to systematically obtain distances to all of M31's satellites identifiable in the PAndAS survey area.

Key words: galaxies: general – galaxies: stellar content – Local Group

Online-only material: color figures

1. INTRODUCTION

The tip of the red giant branch (TRGB) is a very useful standard candle for gauging distances to extended, metal-poor structures. The tip corresponds to the very brightest members of the first ascent red giant branch (RGB), at which point stars are on the brink of fusing helium into carbon in their cores and hence contracting and dimming to become horizontal branch stars. The result is a truncation to the RGB when the color–magnitude diagram (CMD) for an old stellar population is generated, beyond which lie only the comparatively rare asymptotic giant branch (AGB) stars and sources external to the system of interest. The (highly variable) contamination from such objects provides the principal obstacle to simply "reading off" the tip position from the RGB's luminosity function (LF) and the truncation of the AGB can even masquerade as the TRGB in certain instances. The *I* band is the traditionally favored region of the spectrum for TRGB measurements, minimizing the interstellar reddening that plagues shorter wavelengths, while keeping dependence on metallicity lower than it would be at longer IR wavelengths. It should also be remembered that stars approaching the TRGB generally exhibit peak emission in this regime. Iben & Renzini (1983) determined that low-mass ($<1.6\ M_\odot$ for Population I, $<1\ M_\odot$ for Population II), metal-poor ([Fe/H] < -0.7 dex) stars older than 2 Gyr produce a TRGB magnitude that varies by only 0.1 mag. More recently, Bellazzini et al. (2001) determined the tip magnitude to lie at an *I*-band magnitude of $M_{\rm TRGB} = -4.04 \pm 0.12$. This low variation can be attributed to the fact that all such stars have a degenerate core at the onset of helium ignition and so their cores have similar properties regardless of the global properties of the stars. The result is a standard candle that is widely applicable to the old, metal-poor structures that occupy the halos of major galaxies. Distances derived from the TRGB, unlike those from a Cepheid variable or RR Lyrae star, for example, can be determined from a single epoch of observation, making it very useful for wide-area survey data. Furthermore, Salaris & Cassisi (1997) confirmed agreement between Cepheid and RR Lyrae distances and TRGB distances to within ∼5%.

Until Lee et al. (1993) published their edge-finding algorithm, the tip had always been found by eye, but clearly if the wide-reaching applications of the TRGB standard candle were to be realized, a more consistent, repeatable approach was in order. The aforementioned paper shows that, if a binned LF for the desired field is convolved with a zero sum Sobel kernel [−2, 0, +2], a maximum is produced at the magnitude bin corresponding to the greatest discontinuity in star counts, which they attribute to the tip. Using this method, they were able to obtain accuracies of better than 0.2 mag. Sakai et al. (1996) set out to improve on this approach by replacing the binned LF and kernel with their smoothed equivalents. To do this, they equate each star with a Gaussian probability distribution whose FWHM

is determined by the photometric error at the magnitude actually recorded for the star. Then, rather than each star falling within a particular bin, it contributes to all bins via a normalized Gaussian centered on the magnitude recorded for it. This is illustrated in Equation (1):

$$\Phi(m) = \sum_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left[-\frac{(m_i - m)^2}{2\sigma_i^2}\right], \quad (1)$$

where m is the magnitude of the bin in question and m_i and σ_i^2 are the central magnitude and variance, respectively, of the Gaussian probability distribution for the ith star. This method halved the error associated with the non-smoothed version of the algorithm and an identical smoothing is hence just incorporated into the model LF for our Bayesian approach.

In a more recent variation on the edge detection methods, Madore et al. (2009) once again applied a Sobel kernel, but fit to an LF built from composite stellar magnitudes $T \equiv I - \beta[(V - I)_0 - 1.50]$ where β is the slope of the TRGB as a function of color. This, they argued, results in a sharper output response from the filter, and allows all stars, regardless of color, to contribute equally to the derived tip position. Rizzi et al. (2007) derived a value of 0.22 ± 0.02 for β after a study of five nearby galaxies, and showed that it is quite consistent from one galaxy to another.

Méndez et al. (2002) made a departure from the simple "edge-finding" algorithms above by adapting a maximum likelihood model fitting procedure into their technique. They pointed out that the LF faintward of the tip is well modeled as a power law:

$$L(m \geqslant m_{\rm TRGB}) = 10^{a(m - m_{\rm TRGB})}, \quad (2)$$

where $m \geqslant m_{\rm TRGB}$ and a is fixed at 0.3. They then ascribed the location of the tip to the magnitude at which this power law truncates, i.e., $m = m_{\rm TRGB}$. Brightward of the tip they assumed a functional form

$$L(m < m_{\rm TRGB}) = 10^{b(m - m_{\rm TRGB}) - c}, \quad (3)$$

where b is the slope of the power law brightward of the tip and c is the magnitude of the step at the RGB tip.

Such a model, though simplistic, is robust against the strong Poisson noise that is inevitable in more sparsely populated LFs, making it a significant improvement over the previous, purely "edge-finding" methods.

Makarov et al. (2006) followed in a similar vein, demonstrating the proven advantages of a maximum likelihood approach over simple edge detection techniques, despite a model dependence. Unlike Méndez et al. (2002) however, they allowed a as a free parameter, arguing its notable variance from 0.3, and importantly, they smoothed their model LF using a photometric error function deduced from artificial star experiments. One shortcoming of both of these methods, however, is that the most likely parameter values alone are obtained, without their respective distributions or representation of their dependence on the other parameters. Also, with regard to the background contamination, the RGB LF in fact sits on top of non-system stars in the field and so rather than model the background exclusively brightward of the TRGB, the truncated power law of Equation (2) can be added onto some predefined function of the contamination.

Arguably the most successful method developed so far has been that devised by McConnachie et al. (2004). It has been used to ascertain accurate distances to 17 members of the Local Group (McConnachie et al. 2005). It combines aspects of both "edge-finding" and model fitting to zero in more accurately on the tip. They argued that as the precise shape of the LF at the location of the tip is not known, a simple Sobel Kernel approach that assumes a sharp edge to the RGB does not necessarily produce a maximum at the right location. They instead used a least-squares model-fitting technique that fits to the LF in small windows searching for the portion best modeled by a simple slope function. This, they reasoned, marks the location of the steepest decline in star counts which is attributable to the tip location. This method is capable of finding the tip location accurate to better than 0.05 mag, although is still susceptible to being thrown off by noise spikes in a poorly populated LF.

Despite the merits of previous methods such as these, none of them work particularly well when confronted with the high levels of Poisson noise that abound in the more poorly populated structures of galaxy halos. Furthermore, in such conditions as these where the offset between detected and true tip position will likely be at its greatest, it is of great use to have a full picture of likelihood space, as opposed to merely the determined, most probable value. This has led us to develop a new, Bayesian approach to locating the TRGB, specifically, one that incorporates a Markov Chain Monte Carlo (MCMC) algorithm. As shall become apparent in the next section, such a method is very robust against noise spikes in the LF and allows all prior knowledge about the system to be incorporated into the tip-finding process—something lacking in the previous approaches. Further to this, the MCMC provides for a remarkably simple, yet highly accurate error analysis. It also makes it possible to marginalize over parameters to provide posterior probability distributions (PPDs) of each parameter, or to obtain plots of the dependence of each parameter on every other. In Section 2, a detailed explanation of our approach and its limitations is given. Section 2.1 introduces the method by applying the algorithm to one of M31's brightest dwarf spheroidals, Andromeda I. Section 2.2 discusses the nature of systematic errors that apply to the method. Section 2.3 investigates the accuracy that the basic method (before addition of priors) is capable of given the number of stars populating the LF for the field and the strength of the non-RGB background while Section 2.4 deals with its performance when faced with a composite LF. Section 3 then applies our new approach to two additional M31 dwarf satellite galaxies and Section 4 summarizes the advantages of the method and outlines the expected applicability of the method in the immediate future.

2. METHOD

2.1. The MCMC Method

The MCMC method is an iterative technique that, given some model and its associated parameters, rebuilds the model again and again with different values assigned to each parameter, in order that a model be found that is the best fit to the data at hand. It does this by comparing the likelihood of one model, built from newly proposed parameter values, being correct for the data, as opposed to the likelihood for the model built from the previously accepted set of model parameters. The MCMC then accepts or rejects the newly proposed parameter values weighted by the relative likelihoods of the current and proposed model parameter values. At every iteration of the MCMC, the currently accepted value of each parameter is stored so that the number of instances of each value occurring can be used to build

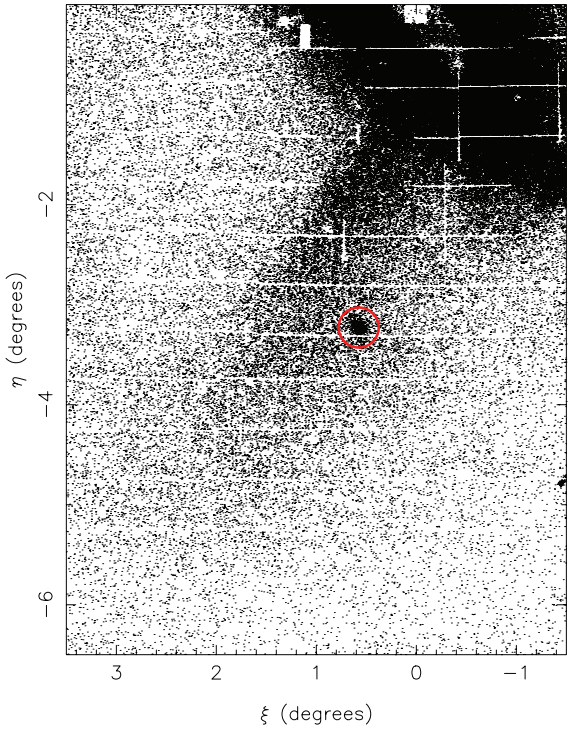


Figure 1. Position of Andromeda I relative to the M31 disk. The saturated disk dominates the northwest corner of the field while Andromeda I itself appears as an overdensity within the Giant Stellar Stream (GSS). The GSS in actuality lies well behind Andromeda I, as is evidenced by the CMD in Figure 2. A strict color-cut was imposed on the data to highlight the location of the satellite and the extent of the stream with greatest contrast.

(A color version of this figure is available in the online journal.)

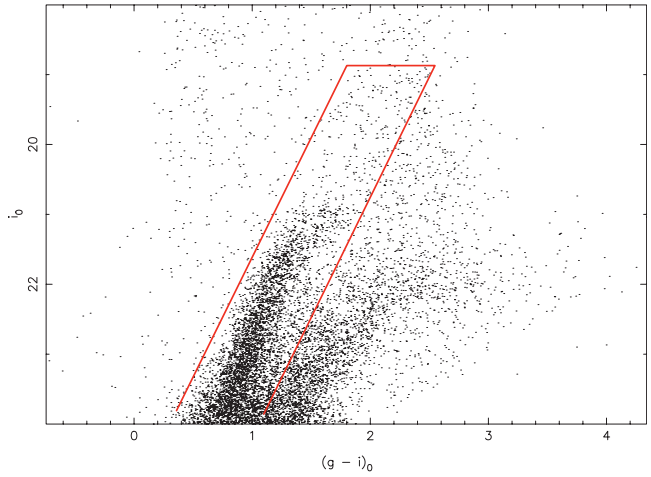


Figure 2. Color–magnitude diagram for a circular field of radius 0.°2 centered on Andromeda I. Two red giant branches are clearly visible, that of Andromeda I (within the red rectangle color-cut) and that of the Giant Stellar Stream which lies behind Andromeda I in the same line of sight.

(A color version of this figure is available in the online journal.)

a likelihood distribution histogram—which can be interpreted as a PPD—for each model parameter. Hence, the MCMC is a way of exploring the likelihood space of complicated models with many free parameters or possible priors imposed, where a pure maximum likelihood method would be quickly overwhelmed. With the PPD generated, the parameter values that produce the best-fit model to the data can simply be read off from the peak of the PPD for each parameter. Similarly, the associated error can be ascertained from the specific shape of the distribution. A detailed description of the MCMC with worked examples can be found in Gregory (2005, Chap. 12).

To illustrate the precise workings of our MCMC tip-finding algorithm, its application to a well-populated dwarf galaxy in the M31 halo is described. Andromeda I was discovered by van den Bergh (1971) and at a projected distance of ∼45 kpc from M31 (Da Costa et al. 1996), it is one of its closest satellites. Da Costa et al. (1996) ascribed to it an age of ∼10 Gyr and a relatively low metallicity of $\langle \mathrm{Fe/H} \rangle = -1.45 \pm 0.2$ dex which is clearly exemplified in the CMD for Andromeda I presented in Figure 2. Here the RGB of Andromeda I lies well to the blue side of that of the Giant Stellar Stream (GSS) which lies behind Andromeda I but in the same field of view. Mould & Kristian (1990) provide the first TRGB-based distance measurement to Andromeda I, which they deduce as 790 ± 60 kpc, based solely on a visual study of the RGB. McConnachie et al. (2004) improve on this significantly, producing a distance determination of 735 ± 23 kpc, based on a tip magnitude of $20.40^{+0.03}_{-0.02}$ in the *I* band.

Andromeda I's position with respect to M31 and the GSS is presented in Figure 1, where the red circle indicates the precise field area fed to our MCMC algorithm. An object-to-background ratio (OBR) of 11.0 was recorded for this field with the color-cut applied, based on comparisons of the signal field stellar density with that of an appropriate background field. The data presented in this figure, as with all other data discussed in this paper, were obtained as part of the Pan-Andromeda Archaeological Survey (PAndAS; McConnachie 2009), undertaken by the 3.6 m Canada–France–Hawaii Telescope (CFHT) on Mauna Kea equipped with the MegaCam imager. CFHT utilizes its own unique photometric bandpasses *i* and *g* based on the AB system. We work directly with the extinction-corrected CFHT *i* and *g* magnitudes and it is these that appear in all relevant subsequent figures. The extinction-correction data applied to each star have been interpolated using the data from Schlegel et al. (1998).

At the heart of our tip-finding algorithm is the model LF that the MCMC builds from the newly chosen parameters at every iteration. The LF is a continuous function which we subsequently convolve with a Gaussian kernel to account for the photometric error at each magnitude. This is achieved by discretizing both functions on a scale of 0.01 mag. Like Méndez et al. (2002), we assume the LF faintward of the tip to follow a simple power law, of the form given in Equation (2); however, we set *a* as a free parameter. The bin height at each magnitude is then calculated by integrating along this function setting the bin edges as the limits of integration. The value for the bin which is set to contain the RGB tip for the current iteration is calculated by integrating along the function from the precise tip location to the faint edge of the bin. All other bins are then set at 0. A bin width of 0.01 mag for our model was found to provide a good balance between magnitude resolution, which is limited by the photometric error in the MegaCam data (∼0.01 mag at m = 20.5), and the computational cost for a higher number of bins. We stress here, however, that each star's likelihood is calculated from the model independently, so that the actual data LF is "fed" to the MCMC in an unbinned state. A faint edge to the model LF was imposed at $m = 23.5$ to remove any significant effects from data incompletion and increasing photometric error.

Further to this, we add a background function to this truncated power law. While the scaling of the background strength relative to the RGB signal strength could be set as another free parameter, and indeed was initially, it makes better use

of our prior information to instead determine the fraction of background stars or "background height" (f) manually. This is achieved simply by calculating the average density of stars in the background field $D_{\rm BG}$ and in the "signal" field $D_{\rm SIG}$ with f then being the ratio of the two, i.e., $f = D_{\rm BG}/D_{\rm SIG}$. Note that this is not directly the inverse of the object-to-background star ratio, OBR $= (D_{\rm SIG} - D_{\rm BG})/D_{\rm BG}$, as f represents the percentage of *all* stars lying inside the signal field that can be expected to be external to the object of interest. Hence, when we normalize the area under the model LF so that it may be used by the MCMC as a probability distribution, the background component will have area f while the RGB component will have area $1 - f$. Now, with f known, what we then have is a simplified two-parameter model, allowing for faster convergence of the MCMC algorithm.

We have thus devised our model so that the MCMC is tasked with the problem of finding just two parameters, namely the slope of the RGB LF (a) and of course the location of the RGB tip magnitude ($m_{\rm TRGB}$). For simplicity in this first paper, we impose uniform priors on each of these parameters, where $19.5 \leqslant m_{\rm TRGB} \leqslant 23.5$ and $0 \leqslant a \leqslant 2$. We also do not account for the color dependence of the tip magnitude which is only slight in the I band (see Rizzi et al. 2007) and for the metal-poor targets examined here, but these effects will be dealt with in future publications. While it is true that two parameters are tractable analytically, we apply the numerical MCMC in order to set the framework for computationally more challenging models with non-uniform priors that will become necessary for the more sparsely populated structures presented in future contributions. There are, however, several more complexities to the model that have yet to be discussed. First, the choice of background function is not arbitrary. It has been found that the best way to model the background is to fit it directly by taking the LF of an appropriate "background" field. The best choice of background field is arguably one that is at similar galactic latitude to the structure of interest, as field contamination is often largely Galactic in origin, and hence closely dependent on angular distance from the Galactic plane. Furthermore, the field should be chosen so that the presence of any substructure is minimal, so as to prevent the signature of another halo object interfering with the LF for the structure of interest.

In addition to these constraints, owing to the low stellar density of the uncontaminated halo, it is preferable that the background field be as large as possible to keep down the Poisson noise and hence it will of necessity be much larger than that of the field of interest. As a result, the main error in the background fit will arise from background mismatching and is not random. In addition, the large background field size may inevitably contain some substructure, requiring removal. This may be done by physically subtracting contaminated portions of the background area, but this is often unnecessary as the CMD color-cut imposed on the signal field must also be applied to the background field, usually ridding the sample of any substantial substructure that may be present. In the case of our Andromeda I background field, however, we have removed a large $2\overset{\circ}{.}4$ portion crossing numerous streams (as shown in Figure 3) as these streams do trespass into the chosen Andromeda I color-cut. Nevertheless, this is just a precaution, because for well-populated systems such as Andromeda I and Andromeda II, the algorithm is impervious to small discrepancies in the functional form of the background.

Once an appropriate background field has been selected, its LF can be fitted by a high-order polynomial. This polynomial then becomes the function added to our model and scaled by f as described earlier. Our choice of background field for Andromeda I (along with Andromeda II and Andromeda XXIII) and the polynomial fit to its LF are presented in Figures 3 and 4, respectively.

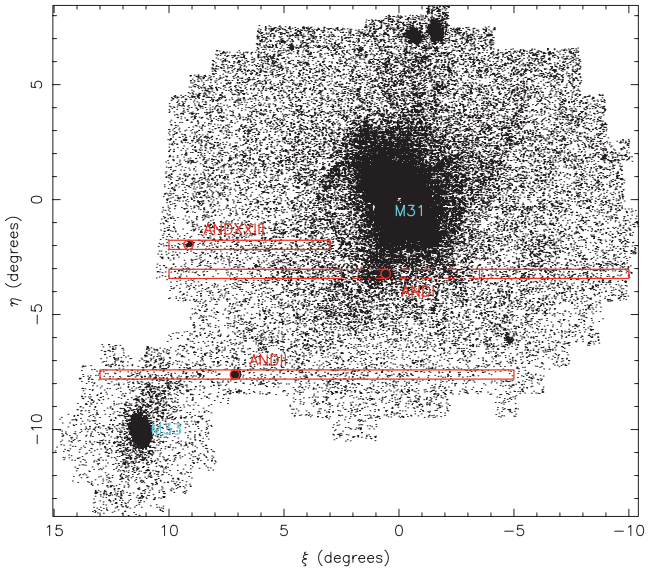


Figure 3. Map of the entire PAndAS survey area, with color-cut chosen to favor the low metallicities exhibited by many of M31's satellite galaxies. The three dwarf spheroidal companions of M31 studied in this paper are labeled, along with the signal fields (small circles of radius $0\overset{\circ}{.}2$) and their respective background fields fed to our algorithm. Note that the background fields are chosen to be as narrow as possible in Galactic latitude while retaining as large an area as possible. In each case, the signal field areas are subtracted from their respective background fields to prevent contamination.

(A color version of this figure is available in the online journal.)

The other major consideration that has yet to be addressed is the effect of photometric error on the LF. This is dealt with by convolving the initial binned model with a normalized Gaussian whose width is adjusted as a function of magnitude in accordance with the error analysis conducted on the PAndAS data. This is equivalent to the method of Sakai et al. (1996) described in Equation (1). As described earlier, this procedure has the added advantage of making the model independent of binning. It is also important in this stage, as it is at every stage, that the model and all constituent parts are normalized so that the model can be used as a probability distribution.

With these issues addressed, the MCMC algorithm can be set in motion. The i-band magnitudes and $(g - i)_0$ data for the desired field is read into data arrays, spurious sources are rejected, and a color-cut is imposed to remove as many non-members of the structure's RGB as possible. The same constraints are of course applied to the background field as well. The MCMC then applies preset starting values of a and $m_{\rm TRGB}$ and builds the corresponding model for the first iteration. Within this iteration, the MCMC proposes new values for each parameter, displaced by some random Gaussian deviate from the currently set values and re-constructs the appropriate model. The step size, or width of the Gaussian deviate is chosen so as to be large enough for the MCMC to explore the entire span of probability space, while small enough to provide a high-resolution coverage of whatever features are present. The ratio of the likelihoods of the two models is then calculated (the Metropolis Ratio r) and a swap of accepted parameter values made if a new, uniform random deviate drawn from the

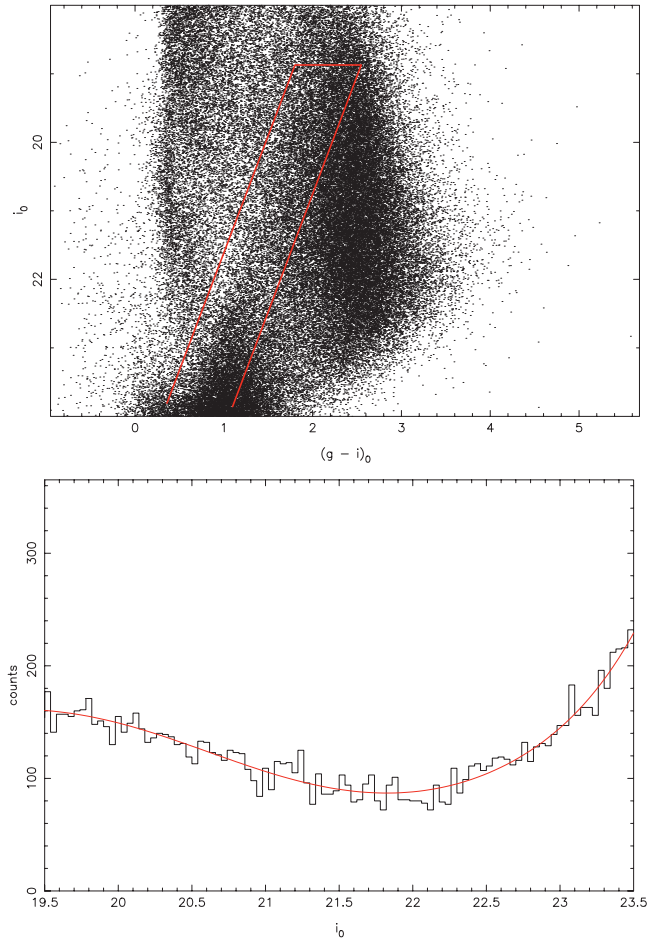


Figure 4. Top: CMD for the Andromeda I background field (see Figure 3). The same color-cut is applied as in the CMD for the signal field (Figure 2). Bottom: The binned luminosity function for the background with the fitted polynomial superimposed. A polynomial of degree seven was found adequate to represent the luminosity function.

(A color version of this figure is available in the online journal.)

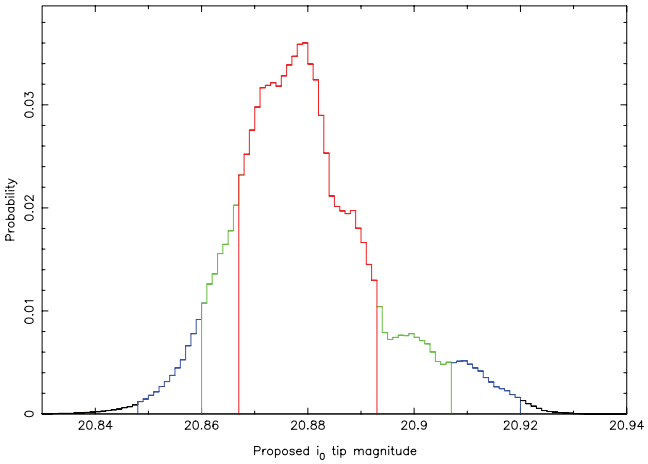


Figure 5. Posterior probability distribution for three million iterations of the MCMC on the Andromeda I CMD color-cut presented in Figure 2. The peak probability is located at $i_0 = 20.88$. The distribution is color coded, with red indicating tip magnitudes within 68.2% (Gaussian 1σ) on either side of the distribution mode, green those within 90%, and blue those within 99%.

(A color version of this figure is available in the online journal.)

interval [0,1], is less than or equal to r. The calculation of the Metropolis Ratio for our model is exemplified in Equations (4) and (5):

$$r = \frac{\mathcal{L}_{\rm proposed}}{\mathcal{L}_{\rm current}} \tag{4}$$

with the value for each of the likelihoods $\mathcal{L}$ being calculated thus

$$\mathcal{L} = \prod_{n=1}^{ndata} M(m_{\rm TRGB}, a, m_n) \tag{5}$$

with

$$M(m_n \geqslant m_{\rm TRGB}) = {\rm RGB}(m_n) + {\rm BG}(m_n)$$
$$M(m_n < m_{\rm TRGB}) = {\rm BG}(m_n)$$

where $${\rm RGB}(m_n) = 10^{a(m_n - m_{\rm TRGB})} \tag{6}$$

and $$\int_{m=m_{\rm TRGB}}^{m=23.5} {\rm RGB}\, dm = 1 - f$$

and $$\int_{m=19.5}^{m=23.5} {\rm BG}\, dm = f,$$

where $m_{\rm TRGB}$ and a are the parameters currently chosen for the model by the MCMC, *ndata* is the number of stars and m_n is the i-band magnitude of the nth star. BG represents the fitted background function (see Figure 4). The MCMC then stores the new choice for the current parameter values and cycles to the next iteration. In order to ascertain a reasonable number of iterations, the chains for each parameter were inspected to insure that they were well mixed, resulting in posterior distributions that appeared smooth (by eye).

When the MCMC has finished running, the PPD for each parameter is generated. By binning up the number of occurrences of each parameter value over the course of the MCMC's iterations, the probability of each value is directly determined and the most probable value can be adopted as the correct model value for the data. If one assumes a Gaussian probability distribution, then the 1σ errors associated with each parameter value can be obtained simply by finding the value range centered on the best-fit value that contains 68.2% of the data points. As our PPDs are not always Gaussian, our quoted 1σ errors in the tip magnitude represent more strictly a 68.2% credibility interval. We do not fit a Gaussian to our PPDs to obtain 1σ errors. Our 1σ errors in tip magnitude are obtained by finding the magnitude range spanning 68.2% of the PPD data points, on one side of the distribution mode and then the other. It must be stressed that these quoted errors are merely an indicator of the span of the parameter likelihood distribution and are no substitute for examining the PPDs themselves. Figures 5 and 6 present the PPD for the RGB tip magnitude based on the Andromeda I CMD (Figure 2) and the best-fit model to the LF for the field, respectively. The PPD for the LF slope a is presented in Figure 7 and a contour map of the distribution of the tip magnitude versus a is presented in Figure 8.

Upon the completion of the algorithm, the RGB tip for Andromeda I was identified at $m = 20.879^{+0.014}_{-0.012}$. This corresponds to an extinction-corrected distance of $731^{(+5)+18}_{(-4)-17}$ kpc, where the final errors include contributions from the extinction and the uncertainty in the absolute magnitude of the TRGB (see Section 2.2). The i-band extinction in the direction of Andromeda I is taken as $A_\lambda = 0.105$ mag (Schlegel et al.

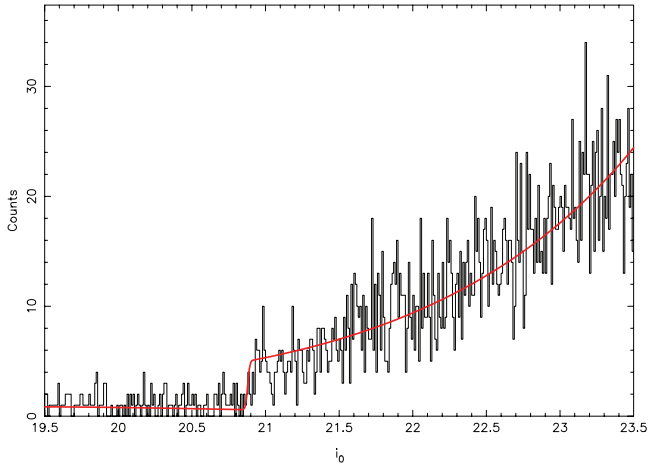


Figure 6. Four-magnitude segment of the Andromeda I luminosity function fitted by our MCMC algorithm. It is built from 3355 stars. The best-fit model is overlaid in red. The bin width for the LF is 0.01 mag.

(A color version of this figure is available in the online journal.)

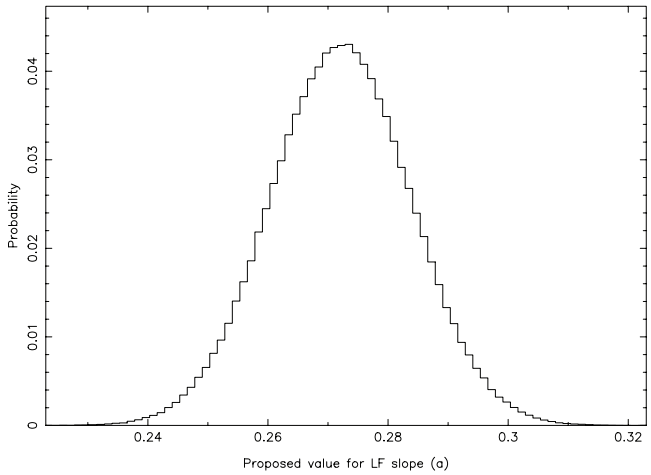


Figure 7. Posterior probability distribution obtained for the slope a of the Andromeda I luminosity function. The distribution is a clean Gaussian with the distribution mode at 0.273.

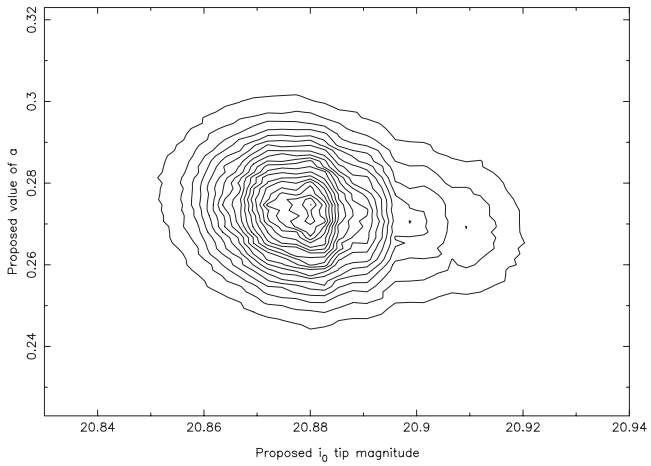


Figure 8. Contour map of the distribution of the tip magnitude vs. the LF slope a. It is noteworthy that there is little correlation between the two parameters, with the peak of the distribution of a more or less independent of tip magnitude. Regardless of any correlation, the respective PPDs of each parameter are the result of marginalizing over the other parameter, and thus take into account any covariance between parameters.

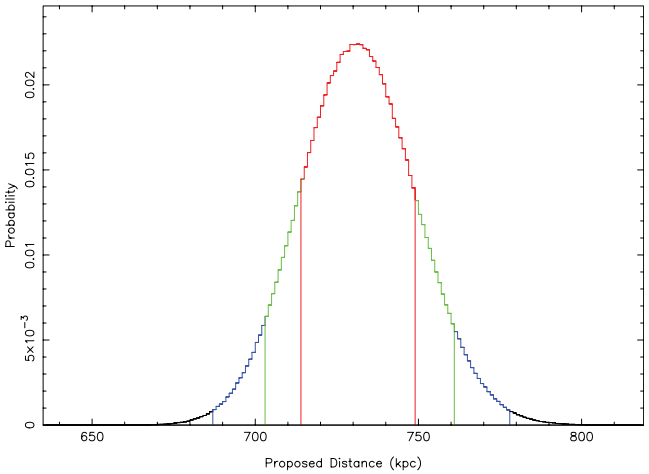


Figure 9. Plot of the distribution of possible distances to Andromeda I obtained through the application of our method. Once again, the colors red, green, and blue denote distances within 68.2%, 90%, and 99% credibility intervals, respectively.

(A color version of this figure is available in the online journal.)

1998). The parameters a and f were derived as 0.273 ± 0.011 and 0.083, respectively. This distance measurement is in excellent agreement with the distance determined by McConnachie et al. (2004). It is noteworthy, however, that our method searches for the TRGB itself as distinct from the RGB star closest to the TRGB as sort out by the method of McConnachie et al. (2004), which would contribute to our slightly smaller distance measurement. A similar discrepancy arises in the case of Andromeda II (see Section 3).

2.2. A Note on Distance Errors

Despite the small errors in the tip magnitude afforded by our approach, there are a number of factors that contribute to produce a somewhat larger error in the absolute distance. These arise due to uncertainties both in the extinction corrections applied and in the absolute magnitude of the TRGB in the i band. Both of these contributions are assumed to be Gaussian, where the 1σ error in the extinction correction, $\Delta\{A_\lambda\}$, is taken as 10% of the correction applied, and the error in the absolute magnitude of the tip is expressed in Equation (7) below

$$\begin{aligned}\Delta\left\{M_i^{\rm TRGB}\right\} &= \sqrt{\Delta^2\left\{m_i^{\rm TRGB}\right\}_{\omega{\rm Cen}} + \Delta^2\{A_\lambda\}_{\omega{\rm Cen}} + \Delta^2\{m-M\}_{\omega{\rm Cen}}} \\ &= \sqrt{\{0.04\}^2 + \{0.03\}^2 + \{0.11\}^2} \\ &= \pm 0.12. \end{aligned} \quad (7)$$

As we are working in the native CFHT i and g bands, we adopt this magnitude as $M_i^{\rm TRGB} = -3.44 \pm 0.12$, where the conversion from $M_I^{\rm TRGB}$ is based on the absolute magnitude for the TRGB identified for the Sloan Digital Sky Survey (SDSS) i band (Bellazzini 2008). This is justified by the color equations applying to the new MegaCam i-band filter (Gwyn 2010). Noting that the largest contribution to this error is that from the distance modulus to ωCen, $(m-M)_{\omega{\rm Cen}}$, derived from the eclipsing binary OGLEGC 17, we consider only the contributions from the extinction $\{A_\lambda\}_{\omega{\rm Cen}}$, which is taken as 10% of the Schlegel et al. (1998) values, and the apparent tip magnitude determination $\{m_i^{\rm TRGB}\}_{\omega{\rm Cen}}$ and note that our

derived distance modulus may be systematically displaced by up to 0.1 of a magnitude. This then gives us $M_i^{\rm TRGB} = -3.44 \pm \sqrt{0.04^2 + 0.03^2} = -3.44 \pm 0.05$. Since our principal motive is to obtain relative distances between structures within the M31 halo rather than the absolute distances to the structures, this offset is not important. Furthermore, as measurements for the ωCen distance modulus improve, our distances are instantly updatable by applying the necessary distance shift.

While these external contributions to our distance uncertainties may be taken as Gaussian, the often non-Gaussian profile of our TRGB ($m_i^{\rm TRGB}$) posterior distributions necessitates a more robust treatment then simply adding the separate error components in quadrature. Hence to obtain final distance uncertainties, we produce a distance distribution obtained by sampling combinations of $m_i^{\rm TRGB}$, A_λ and $M_i^{\rm TRGB}$ from their respective likelihood distributions, thus giving us a true picture of the likelihood space for the distance. The result of this process for Andromeda I is illustrated in Figure 9. From this distribution, we determine not only the quoted 1σ errors but also that Andromeda I lies at a distance between 703 and 761 kpc with 90% credibility and between 687 and 778 kpc with 99% credibility.

2.3. Initial Tests

In order to gain a better understanding of the capabilities of our method when faced with varying levels of LF quality, a series of tests were conducted on artificial "random realization" data, as well as on sub-samples of the Andromeda I field utilized above. There are two major factors that affect the quality of LF available to work with, namely, the number of stars from which it is built and the strength of the background component relative to the RGB component. Hence to simulate the varying degrees of LF quality that are likely to be encountered in the M31 halo, artificial LFs were built for 99 combinations of background height versus number of stars. Specifically, background heights of $f = 0.1, 0.2, \ldots, 0.9$ were tested against each of $ndata = 10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10{,}000,$ and 20,000 stars populating the LF.

To achieve this, a model was built as discussed in Section 2.1, with a constant tip magnitude and RGB slope of $m_{\rm TRGB} = 20.5$ and $a = 0.3$, respectively, and a background height f set to one of the nine levels given above. The functional form of the background was kept as a horizontal line for the sake of the tests. An LF was then built from the model, using one of the 11 possible values for the number of stars listed above. This was achieved by assigning to each of the *ndata* stars a magnitude chosen at random, but weighted by the model LF probability distribution—a "random realization" of the model. The MCMC algorithm was then run on this artificial data set as described in the previous subsection with $m_{\rm TRGB}$ and a as free parameters to be recovered. The tests also assume the photometric errors of the PAndAS survey and further assume that incompleteness is not an issue in the magnitude range utilized. The error in the recovered tip position and the offset of this position from the known tip position in the artificial data ($I = 20.5$) were then recorded. The results are presented in Figures 10 and 11 below. Each pixel represents the average result of ten 200,000 iteration MCMC runs for the given background height versus number of stars combination. Note that the kpc distances given correlate to an object distance of 809 kpc—i.e., $m_I^{\rm TRGB} = 20.5$—which is in keeping with distances to the central regions of the M31 halo. Furthermore, all stars of the random realization were generated within a 1 mag range centered on this tip value.

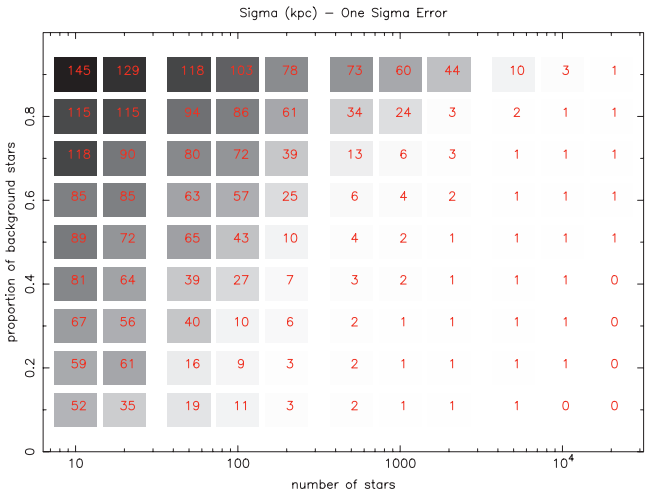


Figure 10. Gray-scale map of the 1σ error in tip magnitude obtained for different combinations of background height and number of sources. The actual value recorded for the error (in kpc) is overlaid on each pixel in red. For these tests, we approximate the 1σ error as the half-width of the central 68.2% of the PPD span.

(A color version of this figure is available in the online journal.)

Figures 10 and 11 are intended to serve as a reference for future use of the basic method, with regard to the number of stars required to obtain the distance to within the desired uncertainty for the available signal-to-noise ratio. The results follow the inevitable trend of greater performance when the background height is small and there are many stars populating the LF. There are some minor deviations from this trend but these result from single outlying values whose effects would diminish if a higher number of samples were averaged. It is also noteworthy that the offsets recorded clearly correlate with the 1σ errors and are consistently less than their associated errors.

The results of these random realization tests are borne out by similar tests conducted on subsamples of the Andromeda I field. Random samples were drawn containing 335 (10% of the total sample), 200, 100, and 50 stars. These correspond approximately to 10, 20, 50, and 100 stars in the 1 mag range centered on the tip. In no case was the derived tip location more than 80 kpc from that identified from the full sample, and the offset grew steadily less as the number of stars in the sample was increased. Furthermore, the offsets were almost always less than the 1σ errors.

2.4. Algorithm Behavior for Composite Luminosity Functions

When a field is fed to any RGB tip finding algorithm, it must be remembered that field is in fact three dimensions of space projected onto two, and therefore it is possible that two structures at very different distances may be present within it. Such a scenario becomes especially likely when dealing with the busy hive of activity that the PAndAS Survey has come to reveal around M31. The result of such an alignment along the line of sight is an LF built from two superimposed RGBs with two different—possibly widely separated—tip magnitudes. Hence it is important to understand how the TRGB algorithm applied to such a field will respond.

Unlike other algorithms that have been developed, our Bayesian approach provides us with a measure for the probability of the tip being at any given magnitude (the PPD). But this also leads to an important caveat—the selection criteria imposed

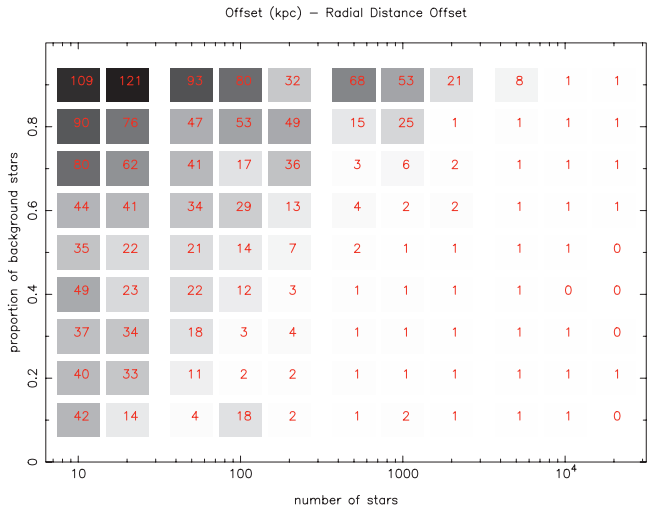


Figure 11. Gray-scale map of the offset from the true tip magnitude obtained for different combinations of background height and number of sources. The actual (absolute) value recorded for the offset (in kpc) is overlaid on each pixel in red. These values convey the discrepancy between the true object distance and that recovered by the MCMC. It was necessary to remove the direction of the individual offsets before averaging as the values would otherwise largely cancel out. Examination of the individual offsets shows no significant bias toward either direction however.

(A color version of this figure is available in the online journal.)

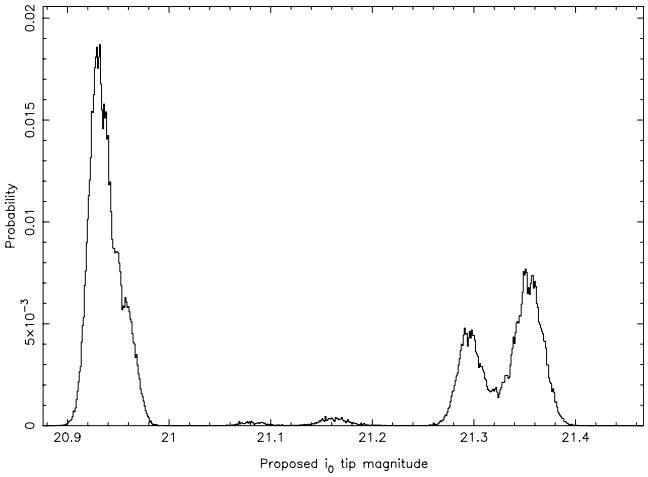


Figure 12. Posterior probability distribution for the cold sampler chain of a four-chain parallel-tempering regime. The MCMC was run for 1.5 million iterations. The strong peak at $m = 20.93$ results from the tip of the Andromeda I RGB, but it has been shifted faintward by the presence of the Giant Stellar Stream, responsible for the peaks at $m = 21.29$ and $m = 21.35$. Without the addition of parallel tempering, the MCMC is liable to spend an inordinate amount of time stuck in the first major probability peak it encounters.

on the data that is fed to the algorithm biases it strongly toward the structure whose distance we are trying to measure. Taking the Andromeda I measurement of Section 2.1 for example, this satellite sits on top of the GSS which contributes prominently to the field CMD, yet its contribution to the LF fed to the MCMC is almost eradicated by our choice of color-cut. Yet if this stringent color-cut is removed, the algorithm remains surprisingly insensitive to the GSS tip. This is because of another prior constraint we impose on the routine—the background height. With this fixed background imposed on our fitted model, the MCMC looks for the first consistent break of the data from the background—i.e., the tip of the Andromeda I RGB. It is therefore necessary to reinstate the background height as a free parameter of the MCMC to give it any chance of finding the tip of the GSS's RGB. By this stage, enough of our prior constraints have been removed to give the method freedom to choose the best fit of the unrestricted model to the entire data set from the field. Nevertheless, the more (correct) prior information we can feed the algorithm, the better the result we can expect to receive.

Still, while the method has not been tailored toward composite LFs, it is worth noting that it can be used successfully to identify more than one object in the line of sight—a useful ability when the two structures are poorly separated in color–magnitude space. The model used assumes only one RGB and thus one tip; to do otherwise would increase computation times. If two distinct structures are identified by this method and cannot be separated using an appropriate color-cut or altered field boundaries, an appropriate double RGB model should be built to accurately locate the tip for each structure. But even with the basic single-RGB model (which will suffice for the vast majority of cases), at least the presence of a second structure is indicated. If we take the example of Andromeda I again, the ideal way to obtain a distance measurement to the portion of the GSS that sits behind it would be to make a color-cut that favors it and removes Andromeda I, but we can force the algorithm to consider both structures to demonstrate the extreme case of what might be encountered in a general halo field. The result is two broad bumps in the PPD well separated in magnitude. The nature of the MCMC however is to converge straight onto the nearest major probability peak, seldom venturing far from that peak. This is remedied by the addition to the algorithm of Parallel Tempering.

While an infinite number of iterations of the MCMC would accurately map probability space in its entirety, Parallel Tempering is a way of achieving this goal much more quickly. Parallel Tempering involves a simple modification to the MCMC algorithm, whereby multiple chains are run in parallel. One chain, the "cold sampler" runs exactly as before, but additional chains have their likelihoods weighted down producing a flatter PPD that is more readily traversed by the MCMC. The further the chain is from the cold sampler chain, the heavier the weight that is applied. Every so many iterations, a swap of parameters is proposed between two random but adjacent chains so that even the "hottest" chains eventually affect the cold sampler chain and allow it to escape any local maximum it may be stuck in. The result is a cold sampler chain PPD that is more representative of the full extent of the LF (see Gregory 2005, Chap. 12 for a more detailed discussion). The result of applying a four-chain MCMC to the region of Andromeda I is summarized in the PPD of Figure 12.

While the Andromeda I TRGB is found much less accurately by this method as a result of the removal of our prior constraints for illustrative purposes, it is nevertheless clear that the addition of Parallel Tempering adds to our algorithm the facility to identify other structures in the field that may require separate analysis. Even given a properly constrained model and data set, the safeguard it provides against a poorly explored probability space arguably warrants its inclusion.

3. DISTANCES TO TWO MORE SATELLITES

To further illustrate the capabilities of our basic method as outlined in Section 2, we have applied it to two more of M31's brighter satellites, whose distances have been determined in

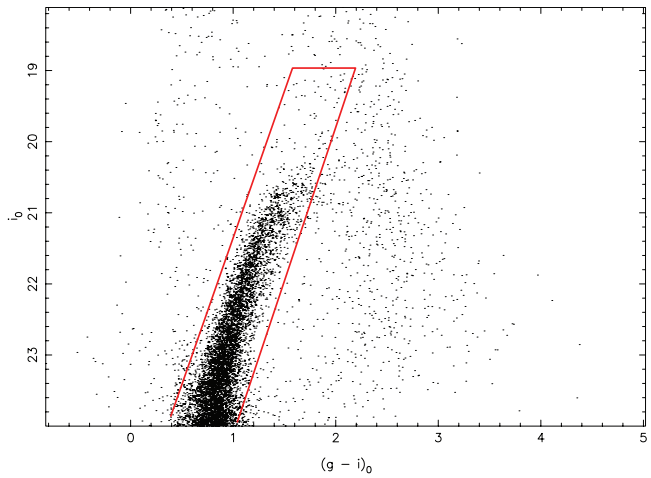


Figure 13. CMD for a circular field of radius 0.°2 centered on Andromeda II. It is more densely populated than the Andromeda I CMD (Figure 2) and is very well defined against the stellar background. The RGB tip is clearly visible at $i_0 \sim 20.6$.

(A color version of this figure is available in the online journal.)

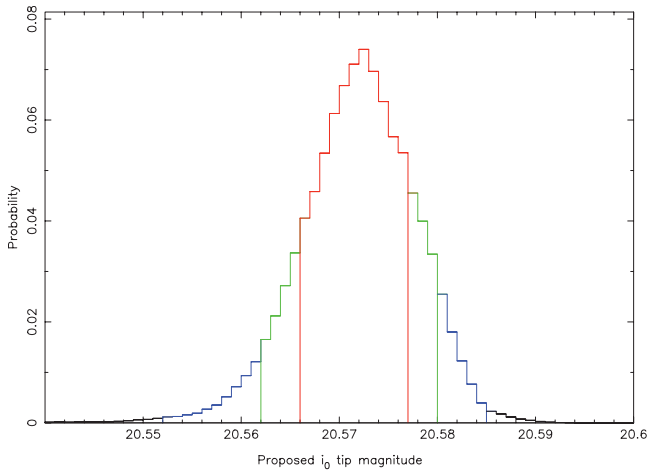


Figure 14. Posterior probability distribution for three million iterations of the MCMC on a 4 mag interval (see Figure 15) of the Andromeda II CMD selection presented in Figure 13. The peak probability of the distribution is well defined at $i_0 \approx 20.57$. The distribution is again color coded as in Figure 5, with red, green, and blue corresponding to 68.2%, 90%, and 99% credibility intervals, respectively.

(A color version of this figure is available in the online journal.)

past measurements using a range of methods, including TRGB-finding algorithms. The additional satellites chosen for this study are the relatively luminous dwarf spheroidal Andromeda II and the somewhat fainter, newly discovered Andromeda XXIII dwarf. The location of both satellites within the M31 halo can be seen in Figure 3.

3.1. Andromeda II

Andromeda II was discovered as a result of the same survey as Andromeda I using the 1.2 m Palomar Schmidt telescope (van den Bergh 1971). Da Costa et al. (2000) deduce a similar age for Andromeda II as for Andromeda I but with a wider spread of metallicities centered on $\langle \mathrm{Fe/H} \rangle = -1.49 \pm 0.11$ dex. Our Andromeda II LF was built from a circular field of radius 0.°2 centered on the dwarf spheroidal with an OBR of 34.0 recorded.

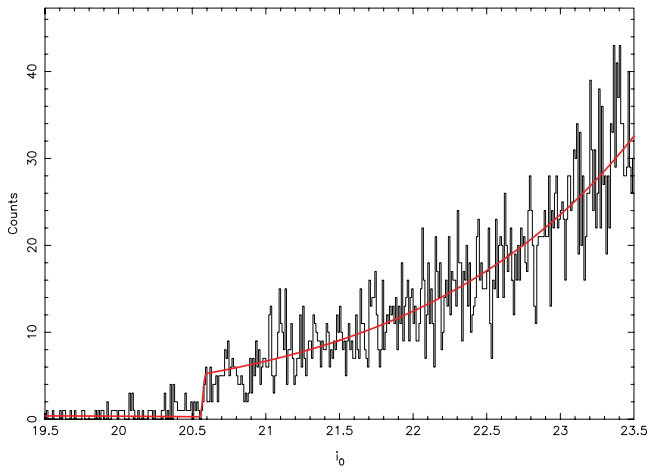


Figure 15. Four-magnitude segment of the Andromeda II luminosity function fitted by our MCMC algorithm. It is built from 4409 stars. The best-fit model is overlaid in red. The bin width for the LF is again 0.01 mag.

(A color version of this figure is available in the online journal.)

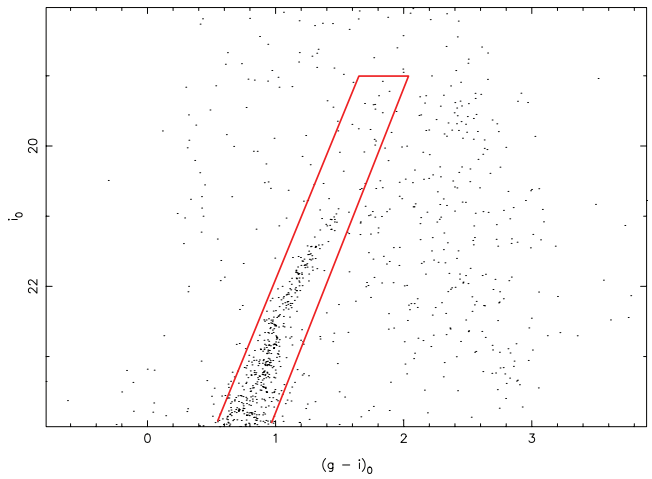


Figure 16. Color–magnitude diagram for a circular field of radius 0.°1 centered on Andromeda XXIII. It is much more sparsely populated than those of Andromeda I and Andromeda II. The RGB tip appears to lie just brightward of $i_0 = 21$.

(A color version of this figure is available in the online journal.)

This high OBR is not unexpected with Andromeda II arguably the best populated of M31's dwarf spheroidal satellites. The CMD for this field is presented in Figure 13.

Application of our algorithm to Andromeda II yields a tip magnitude of $i_0 = 20.572^{+0.005}_{-0.006}$ for the RGB which corresponds to an extinction-corrected distance to Andromeda II of $634^{(+2)+15}_{(-2)-14}$ kpc, where the i-band extinction is taken as $A_\lambda = 0.121$ mag (Schlegel et al. 1998). This is in good agreement with McConnachie et al.'s (2004) derived distance of 645 ± 19 kpc. Values for a and f were recovered as 0.276 ± 0.009 and 0.028, respectively. The m_i^{TRGB} PPD and best-fit model found by our method are illustrated in Figures 14 and 15, respectively.

3.2. Andromeda XXIII

Despite its relative brightness among the other satellites of the M31 system, Andromeda XXIII was only discovered with the undertaking of the outer portion of the PAndAS survey in 2009/2010, being too faint at $M_V = -10.2 \pm 0.5$ to identify from

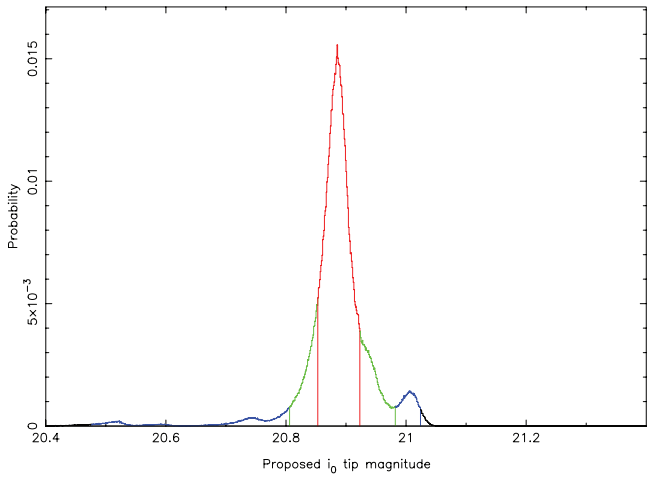


Figure 17. Posterior probability distribution for three million iterations of the MCMC on a 4 magnitude interval (see Figure 18) of the Andromeda XXIII CMD selection presented in Figure 16. There are several probability peaks in this instance but the preferred peak lies at 20.885. The distribution is again color coded as in Figure 5, with red, green, and blue corresponding to 68.2%, 90%, and 99% credibility intervals, respectively.

(A color version of this figure is available in the online journal.)

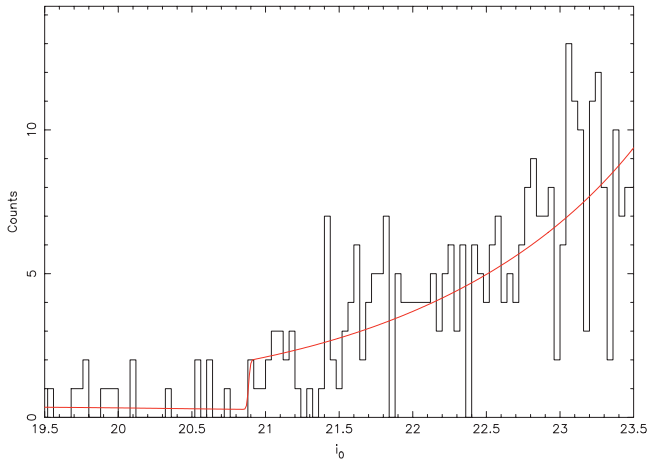


Figure 18. Four-magnitude segment of the Andromeda XXIII luminosity function fitted by our MCMC algorithm. It is built from 328 stars. The best-fit model is overlaid in red. While the model LF tested by the MCMC retained the resolution of 100 bins per magnitude described in Section 2.1, the data LF is re-produced here at the lower resolution of 0.04 mag per bin to better reveal its structure to the eye.

(A color version of this figure is available in the online journal.)

the SDSS (Richardson et al. 2011). The said paper presents its vital statistics along with those for the other newly discovered satellites Andromeda XXIV–XXVII. It is a dwarf spheroidal galaxy and has the lowest recorded metallicity of the satellites we present with $\langle \mathrm{Fe/H} \rangle = -1.8 \pm 0.2$. Making use of the deeper coverage of PAndAS in the g band, Richardson et al. (2011) obtain a distance measurement of 767 ± 44 kpc from the horizontal branch of the CMD.

Andromeda XXIII is a more challenging target for our algorithm in its current form, with less than ∼50 stars lying within the 1 mag range centered on the tip and an OBR of 8.4 for the field and color-cut employed. The CMD for this circular field of radius 0.°1 is presented in Figure 16. We find the RGB tip at an i-band magnitude of $20.885^{+0.038}_{-0.032}$, which, given an i-band extinction of 0.112 mag in the direction of Andromeda XXIII (Schlegel et al. 1998), corresponds to a distance of $733^{(+13)+23}_{(-11)-22}$ kpc. We derive the values of a and f as 0.270 ± 0.039 and 0.105, respectively. Curiously, the MCMC finds several peaks very close to the major peak in the PPD (see Figure 17), but these are attributable to the lower star counts available in the LF around the tip. This has the effect of creating large magnitude gaps between the stars that are just brightward of the tip so that each individual star can mimic the sudden increase in star counts associated with the beginning of the RGB. As a result, there is a range of likely locations for the tip, but the PPD shows that the object cannot be more distant than 802 kpc nor closer than 601 kpc with 99% confidence. The best fit model determined by the MCMC is overlaid on the LF in red in Figure 18.

4. CONCLUSIONS

The versatility and robustness of our new method can be appreciated from Section 2 and its high level of accuracy is evident from the measurement errors which are consistently smaller than those in the literature to date. In addition, it is our hope that with the correct priors imposed, this new approach carries with it the ability to gauge distances to even the most poorly populated substructures, bringing a whole new range of objects with in reach of the TRGB standard candle. In the case of the M31 halo alone, it will be possible to obtain distances to all of the new satellites discovered by the PAndAS survey—a feat previously impractical using the TRGB. Furthermore, PAndAS has revealed a complicated network of tidal streams that contain valuable information as to the distribution of dark matter within the M31 halo. With our new method, it will be possible to systematically obtain distances at multiple points along these streams, thus providing vital information for constraining their orbits.

The great advantage of our new Bayesian method over a pure maximum likelihood method is the ease with which prior information may be built into the algorithm, making it more sensitive to the tip. Herein lies the great power of the Bayesian approach, whereby the addition of a few carefully chosen priors can reduce the measurement errors 10 fold. The result is an algorithm that is not only very accurate but highly adaptable and readily applicable to a wide range of structures within the distance (and metallicity) limitations of the TRGB standard candle. With instruments such as the 6.5 m infrared *James Webb Space Telescope* and the 42 m European Extremely Large Telescope expected to be operational within the decade, these distance limitations will soon be greatly reduced. This will bring an enormous volume of space within reach of the TRGB method, including the region of the Virgo Cluster. A tool with which it is possible to apply the TRGB standard candle to small, sparsely populated structures and small subsections of large structures alike is hence, needless to say, invaluable.

A.R.C. thanks Sydney University for allowing him the use of their computational and other resources. In addition, A.R.C. thanks fellow student Anjali Varghese for her assistance with and practical insights with regard to the implementation of parallel tempering. A.R.C. also thanks Neil Conn for assistance in proofreading the document. G.F.L. thanks the Australian Research Council for support through his Future Fellowship (FT100100268) and Discovery Project (DP110100678). N.F.M. acknowledges funding by Sonderforschungsbereich SFB 881 "The Milky Way System" (subproject A3) of the German Research Foundation (DFG).

REFERENCES

Bellazzini, M. 2008, Mem. Soc. Astron. Ital., 79, 440
Bellazzini, M., Ferraro, F. R., & Pancino, E. 2001, ApJ, 556, 635
Da Costa, G. S., Armandroff, T. E., Caldwell, N., & Seitzer, P. 1996, AJ, 112, 2576
Da Costa, G. S., Armandroff, T. E., Caldwell, N., & Seitzer, P. 2000, AJ, 119, 705
Gregory, P. C. 2005, Bayesian Logical Data Analysis for the Physical Sciences (Cambridge: Cambridge Univ. Press)
Gwyn, S. 2010, http://cadcwww.dao.nrc.ca/megapipe/docs/filters.html
Ibata, R., Martin, N. F., Irwin, M., et al. 2007, ApJ, 671, 1591
Iben, I., Jr., & Renzini, A. 1983, ARA&A, 21, 271
Lee, M. G., Freedman, W. L., & Madore, B. F. 1993, ApJ, 417, 553
Letarte, B., Chapman, S. C., Collins, M., et al. 2009, MNRAS, 400, 1472
Madore, B. F., Mager, V., & Freedman, W. L. 2009, ApJ, 690, 389
Makarov, D., Makarova, L., Rizzi, L., et al. 2006, AJ, 132, 2729
McConnachie, A. W. 2009, BAAS, 41, 278
McConnachie, A. W., Irwin, M. J., Ferguson, A. M. N., et al. 2004, MNRAS, 350, 243
McConnachie, A. W., Irwin, M. J., Ferguson, A. M. N., et al. 2005, MNRAS, 356, 979
Méndez, B., Davis, M., Moustakas, J., et al. 2002, AJ, 124, 213
Mould, J., & Kristian, J. 1990, ApJ, 354, 438
Richardson, J. C., Irwin, M. J., McConnachie, A. W., et al. 2011, ApJ, 732, 76
Rizzi, L., Tully, R. B., Makarov, D., et al. 2007, ApJ, 661, 815
Sakai, S., Madore, B. F., & Freedman, W. L. 1996, ApJ, 461, 713
Salaris, M., & Cassisi, S. 1997, MNRAS, 289, 406
Schlegel, D. J., Finkbeiner, D. P., & Davis, M. 1998, ApJ, 500, 525
van den Bergh, S. 1971, ApJ, 171, L31

“Theories crumble, but good observations never fade.”

Harlow Shapey (1885 - 1972)

4

Paper II: A Bayesian Approach to Locating the Red Giant Branch Tip Magnitude. II. Distances to the Satellites of M31

Paper II Preface

The first tip of the red giant branch paper (Paper I) was written with the intention that a second paper would soon follow which would further develop the method and apply it to the entire satellite sample of M31[1]. As it came to pass, Paper II would not be accepted as an Astrophysical Journal publication until one year and three days after the acceptance of the first paper, despite being begun well before the first paper was accepted. This paper therefore represents a significant portion of my PhD candidature.

The method employed to gain the satellite distance distributions presented in Paper II, differs from that introduced in Paper I, chiefly in the way that prior information is taken into account. Most notably, 'matched filtering' is introduced to weight stars in accordance with their likelihood of being true object members. The object's density profile (as a function of radius) is treated as a probability distribution of object membership such that stars found in the densest central regions of the object are given more weight when fitting the object's luminosity function. In many cases, the contrast between the luminosity function with and without the matched filtering switched on is profound, with the RGB tip becoming clearly visible to the eye where before it was lost in a mass of masquerading background stars. In addition to the matched filtering, a prior is also imposed on the expected object distance in the form of a halo density prior. A cross-section through the (expected) M31 halo density profile along the line of sight to the object is used to weight the probability of finding the object at any distance along its distance probability distribution.

The reason for the rather lengthy time interval between the publication of the two papers was not due to any major issues with the method in this new paper, but rather the amount of feedback I received from those interested in the satellite distances. It became clear very early on that a lot of people had a vested interest in having access to accurate distances accompanied by accurate uncertainty distributions in those distances. It was also clear that many held clear-cut views as to how the distances should be obtained and presented. As a result I had to incorporate a particularly large amount of changes into the method and in turn

[1] Due to the advantages of using a single data set for all measurements, only those satellites contained within the PAndAS survey were actually included in the paper. An inner cutoff ellipse around the M31 disk was also necessary due to its obscuring effects (see Fig. 10 (c)).

the draft of the paper which inevitably meant a large number of complete re-runs on all of the data and analysis it contained. After making the necessary changes to the method and re-writting various parts of the paper, it was finally ready for submission to Astrophysical Journal. All of this said, there is no doubt that the method is more robust as a result of this lengthy process.

One of the most important changes that arose from this scrutiny concerned the way the density profiles of the target objects were generated. Originally, the density profiles were being produced simply be drawing a series of evenly spaced concentric circles (or bands) about the object center and determining the density of stars in each band. The resulting binned profile was then fit in log space by a straight line (i.e. approximating the profiles as exponential). This of course assumes spherical symmetry which is not always a fair assumption, with some of M31's satellites being strongly elliptical. It was therefore decided to take this ellipticity fully into account which required a substantial re-write of the code for the density matched filter. These changes also warranted a second look at the luminosity function of each object and extra care was taken to insure that the CMD colour-cuts and the inner and outer cutoff radii for each object combined to produce luminosity functions with the greatest tip contrast possible. Other shortcomings in the algorithm code (see 'MF_TRGB.f95' in Appendix C) were also subsequently identified as the need arose for faster processing times and so provisions were made for feeding in the necessary object parameters in the command line and other portions of the code were altered to run more efficiently. Improvements to the PAndAS photometry calibration at the beginning of 2012 also required another re-run on the M31 satellites which further improved the quality of the distance measurements.

In many respects, the real climax of Paper II is the application of the distances to produce a new 3D view of the M31 system, as is presented in Fig. 10. This represents the true beginning of our study of the three dimensional structure of the satellite system, at the heart of which is the trigonometry necessary to convert the earth distances into an M31-centric coordinate system. Fig. 4.1 was created to aid in the determination of the necessary conversions. The coordinate system used here is that which arises most naturally from an Earth based perspective, with z pointing along the line of site to the center of M31, and x and y pointing along lines of constant Declination and Right Ascension respectively. A more typical

orientation of the coordinate system is later adapted in Paper III by implimenting rotations about the x and z axes so that z points toward the M31 Galactic north pole, with the Earth at a longitude of 0°.

Now, with M31 and its satellites represented by a series of points in three dimensions, we are in a position to begin an analysis of the distribution. This analysis is begun in Paper II with a study of the satellite density profile within the M31 halo. Of particular note, this study takes into full account the uneven coverage of the PAndAS survey, whereby certain radii from the center of M31 receive better coverage than others. The study also gives full account to the distance uncertainty distributions for each satellite by sampling possible positions from each distribution over many iterations. A more thorough study of the satellite distribution then follows in Paper III. Note that all of the principal code used throughout the analysis in Paper II can be found in Appendix C, along with a brief summary of what each program does.

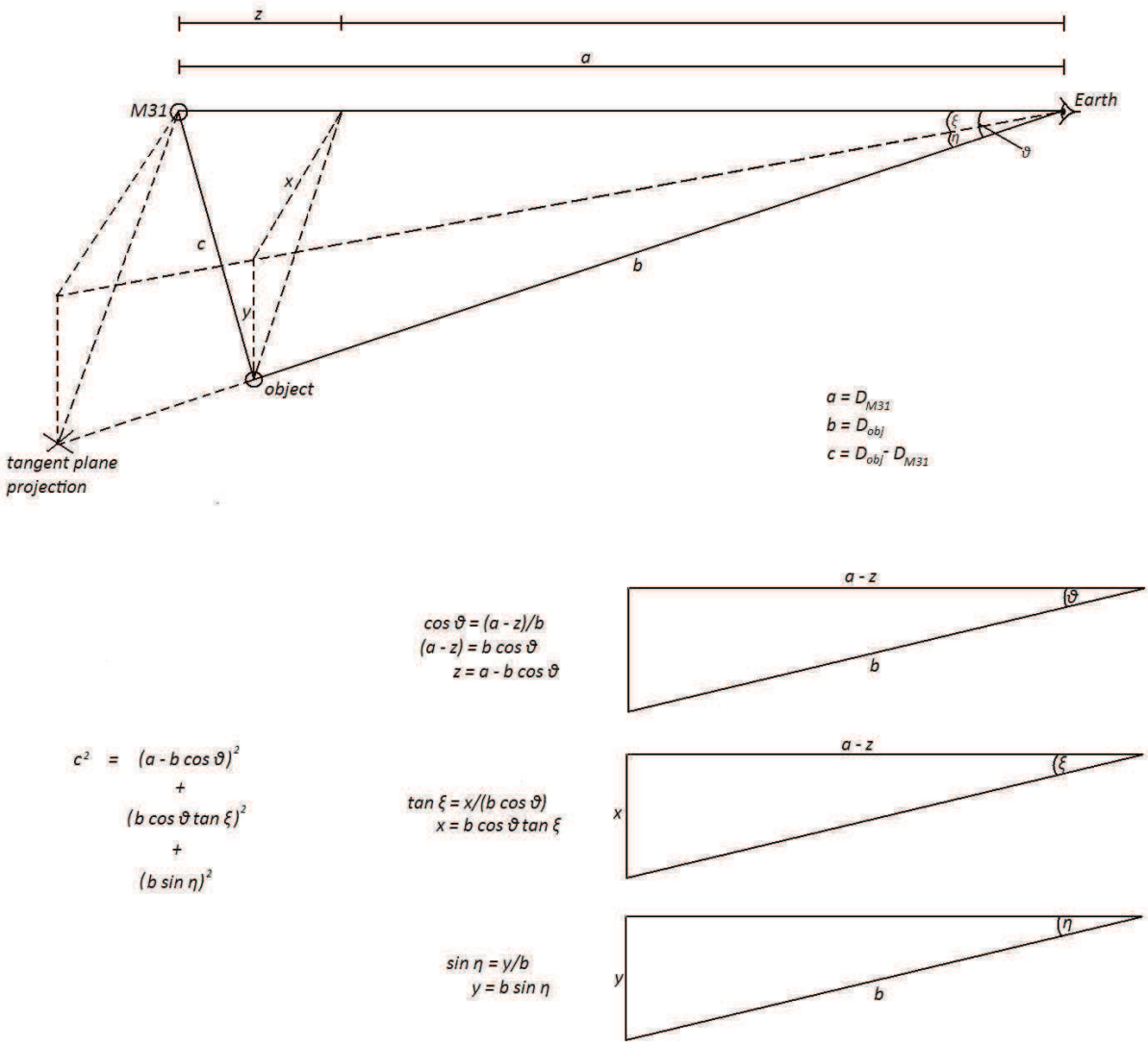


FIGURE 4.1: Conversion of Earth-to-object distances into an M31-centric cartesian coordinate system. This figure was created to help visualize the geometry of Earth-M31-object alignments. The top part of the diagram shows the projection of the target object (satellite) on to the M31 tangent plane and the x,y,z of the coordinate system used. Positive x points East (toward increasing ξ), positive y points North (toward increasing η) and positive z points along the line of sight (to M31) away from Earth. The three triangles in the lower half of the figure show how each coordinate can be determined from the Earth-to-M31 (a) and Earth-to-object (b) distances.

The Astrophysical Journal, 758:11 (19pp), 2012 October 10 doi:10.1088/0004-637X/758/1/11

A BAYESIAN APPROACH TO LOCATING THE RED GIANT BRANCH TIP MAGNITUDE. II. DISTANCES TO THE SATELLITES OF M31

A. R. Conn[1,2,3], R. A. Ibata[3], G. F. Lewis[4], Q. A. Parker[1,2,5], D. B. Zucker[1,2,5], N. F. Martin[3], A. W. McConnachie[6], M. J. Irwin[7], N. Tanvir[8], M. A. Fardal[9], A. M. N. Ferguson[10], S. C. Chapman[7], and D. Valls-Gabaud[11]
[1] Department of Physics & Astronomy, Macquarie University, NSW 2109, Australia
[2] Research Centre in Astronomy, Astrophysics, and Astrophotonics (MQAASTRO), Macquarie University, NSW 2109, Australia
[3] Observatoire Astronomique, Universite de Strasbourg, CNRS, F-67000 Strasbourg, France
[4] Sydney Institute for Astronomy, School of Physics, A28, University of Sydney, Sydney, NSW 2006, Australia
[5] Australian Astronomical Observatory, P.O. Box 296, Epping, NSW 2121, Australia
[6] NRC Herzberg Institute of Astrophysics, 5071 West Saanich Road, Victoria, British Columbia V9E 2E7, Canada
[7] Institute of Astronomy, University of Cambridge, Madingley Road, Cambridge CB3 0HA, UK
[8] Department of Physics and Astronomy, University of Leicester, Leicester LE1 7RH, UK
[9] University of Massachusetts, Department of Astronomy, LGRT 619-E, 710 N. Pleasant Street, Amherst, MA 01003-9305, USA
[10] Institute for Astronomy, University of Edinburgh, Royal Observatory, Blackford Hill, Edinburgh EH9 3HJ, UK
[11] Observatoire de Paris, LERMA, 61 Avenue de l'Observatoire, F-75014 Paris, France
Received 2012 March 1; accepted 2012 July 18; published 2012 September 20

ABSTRACT

In "A Bayesian Approach to Locating the Red Giant Branch Tip Magnitude (Part I)," a new technique was introduced for obtaining distances using the tip of the red giant branch (TRGB) standard candle. Here we describe a useful complement to the technique with the potential to further reduce the uncertainty in our distance measurements by incorporating a matched-filter weighting scheme into the model likelihood calculations. In this scheme, stars are weighted according to their probability of being true object members. We then re-test our modified algorithm using random-realization artificial data to verify the validity of the generated posterior probability distributions (PPDs) and proceed to apply the algorithm to the satellite system of M31, culminating in a three-dimensional view of the system. Further to the distributions thus obtained, we apply a satellite-specific prior on the satellite distances to weight the resulting distance posterior distributions, based on the halo density profile. Thus in a single publication, using a single method, a comprehensive coverage of the distances to the companion galaxies of M31 is presented, encompassing the dwarf spheroidals Andromedas I–III, V, IX–XXVII, and XXX along with NGC 147, NGC 185, M33, and M31 itself. Of these, the distances to Andromedas XXIV–XXVII and Andromeda XXX have never before been derived using the TRGB. Object distances are determined from high-resolution tip magnitude posterior distributions generated using the Markov Chain Monte Carlo technique and associated sampling of these distributions to take into account uncertainties in foreground extinction and the absolute magnitude of the TRGB as well as photometric errors. The distance PPDs obtained for each object both with and without the aforementioned prior are made available to the reader in tabular form. The large object coverage takes advantage of the unprecedented size and photometric depth of the Pan-Andromeda Archaeological Survey. Finally, a preliminary investigation into the satellite density distribution within the halo is made using the obtained distance distributions. For simplicity, this investigation assumes a single power law for the density as a function of radius, with the slope of this power law examined for several subsets of the entire satellite sample.

Key words: galaxies: general – galaxies: stellar content – Local Group

Online-only material: color figures, machine-readable table

1. INTRODUCTION

The tip of the red giant branch (TRGB) is a well-established standard candle for ascertaining distances to extended, metal-poor structures containing a sufficient red giant population. Its near constant luminosity across applicable stellar mass and metallicity ranges (see Iben & Renzini 1983) arises due to the prevailing core conditions of these medium-mass stars as core helium fusion ensues. Their cores lack the necessary pressure to ignite immediate helium fusion on the depletion of their hydrogen fuel and so they continue to fuse hydrogen in a shell around an inert, helium ash core. This core is supported by electron degeneracy and grows in mass as more helium ash is deposited by the surrounding layer of hydrogen fusion. On reaching a critical mass, core helium fusion ignites, and the star undergoes the helium flash before fading from its position at the TRGB, to begin life as a horizontal branch star. Due to the very similar core properties of the stars at this point, their energy output is almost independent of their total mass, resulting in a distinct edge to the RGB in the color–magnitude diagram (CMD) of any significant red giant population.

With the TRGB standard candle applicable wherever there is an RGB population, it is an obvious choice for obtaining distances to the more sparsely populated objects in the Local Group and other nearby groups where Cepheid variables seldom reside. Even when Cepheids are available, the TRGB often remains a more desirable alternative, requiring only one epoch of observation, and facilitating multiple distance measurements across an extended structure. Good agreement between TRGB-obtained distances and those obtained using Cepheid variables as well as the much fainter RR Lyrae variables have been confirmed by Salaris & Cassisi (1997), with discrepancies of no more than ∼5% (see also Tammann et al. 2008 for an extensive list of distance comparisons utilizing the three standard candles). Of the satellites of M31, many are very faint and poorly populated and thus have poorly constrained distances which propagate on into related measurements concerning the structure of the halo system. Hence, a technique for refining

the distances that can be applied universally to all halo objects, while *accurately* conveying the associated distance errors has been a long sought goal.

In "A Bayesian Approach to Locating the Red Giant Branch Tip Magnitude (Part I)"—Conn et al. (2011), hereafter Paper I, we reviewed the challenges of identifying the TRGB given the contamination to the pure RGB luminosity function (LF) typically encountered. We also outlined some of the methods that have been devised to meet these challenges since the earliest approach, put forward by Lee et al. (1993). We then introduced our own unique Bayesian approach, incorporating Markov Chain Monte Carlo (MCMC) fitting of the LFs. This approach was essentially the base algorithm, designed to easily incorporate priors to suit the task at hand. Here we present the results of an adaptation of that algorithm, intended for use on small, compact objects—specifically the dwarf spheroidal companions of M31. Once again, we utilize the data of the Pan-Andromeda Archaeological Survey (PAndAS; McConnachie et al. 2009), a two-color ($i' = 770$ nm, $g' = 487$ nm) panoramic survey of the entire region around M31 and M33 undertaken using the Canada–France–Hawaii Telescope (CFHT). The tip is measured in the i' band where dependence on metallicity is minimal. Following a recap of the base method in Section 2, we introduce the aforementioned new adaptations to the method in Section 3.1 and in Section 3.2 we describe the results of tests intended to characterize the modified algorithms performance as well as check the accuracy of its outputs. In addition, Section 3.3 outlines the application of a further prior on the satellite distances. Section 4.1 presents the results of applying the modified algorithm to the companions of M31, while Section 4.2 details the method by which the object-to-M31 distances are obtained and Section 4.3 uses the obtained distances to analyze the density profile of these objects within the halo. Conclusions follow in Section 5.

2. A RECAP OF THE BASE METHOD

In Paper I, we introduced our "base" method, whereby the LF of a target field was modeled by a single, truncated power law (the RGB of the object of interest) added to a representative background polynomial. The location of the truncation (the TRGB) and the slope of the power law were set as free parameters of the model, with the best fit derived using an MCMC algorithm. The functional form of the background component was modeled by directly fitting a polynomial to the LF of an appropriate background field, and then scaling the polynomial to reflect the expected number of background stars in the target field. The resulting model was then convolved with a Gaussian of width increasing in proportion to the photometric error as a function of magnitude. The posterior distribution in the tip magnitude returned by the MCMC, which thus already incorporates the photometric error, is then sampled together with Gaussian distributions representing the distribution in the absolute magnitude of the tip ($M_i^{\rm TRGB} = -3.44 \pm 0.05$) and the distribution in the extinction ($A_\lambda \pm 0.1A_\lambda$) to give a final posterior distribution in the distance. The mode of this distribution is then adopted as the distance to the object, with the $\pm 1\sigma$ error calculated from the portion of the distribution lying on the far and near side of the mode, respectively.

A more detailed discussion of the assumptions and rationale behind the base method is provided in paper I, but the reader should again be made aware of the most fundamental assumptions it entails. At the heart of the calculations of course is the choice of the absolute magnitude of the tip and its associated uncertainty. We adopt the values of this parameter stated above based on the value derived for the SDSS *i* band in Bellazzini (2008), noting the near-identical bandpass characteristics of the MegaCam *i*-band filter as detailed by Gwyn (2010). We adopt somewhat smaller uncertainties than those derived by Bellazzini (2008) following the same argument as McConnachie et al. (2004) that the quoted uncertainty in the absolute magnitude of the tip is conservative and it is a systematic error effecting all distance measurements in an identical way. As almost all applications of the distances to the satellites are concerned with their relative positions to one another and M31, this component of the error is of minimal importance. Nevertheless, it often forms the major component of the quoted errors in our distances.

Mention should also be made as to the effects of metallicity and internal reddening within the objects under study as well as the zero-point uncertainty in the PAndAS photometry. While there is a metallicity dependence of $M_i^{\rm TRGB}$ (though minimal when compared with other bands), it is only really an issue for more metal-rich targets (e.g., [Fe/H] > -1; see Bellazzini 2008, Figure 6) and thus will primarily affect measurements to the large, diverse systems such as M31 itself and M33. But the TRGB for more metal-rich populations is fainter than that for their metal-poor counterparts and thus it is this metal-poor population component which dominates the measurement. A similar situation is encountered with the internal reddening present in the objects under study, where the vast majority of objects, chiefly the dwarf spheroidal galaxies, are almost completely devoid of such effects. Those objects most strongly affected are the large, well-populated systems which will provide ample signal from the least affected stars on the near side of the system, for a good distance determination. The uncertainty in the zero point of the photometry is consistent throughout the survey at approximately 0.02 mag. (R. A. Ibata et al. 2012, in preparation).

Lastly, a brief discussion of the distance posterior distributions themselves is warranted. As noted above, they are produced by the sampling of the distribution of possible tip positions (as generated by the MCMC and with photometric errors incorporated) along with sampling of the Gaussian distributions representing the uncertainties in the foreground extinction (A_λ) and in the absolute magnitude of the tip ($M_i^{\rm TRGB}$). Specifically, 500,000 possible distances are drawn to form the distance PPD, where for each draw κ, the distance modulus μ is

$$\mu(\kappa) = m_i^{\rm TRGB}(\kappa) - A_\lambda(\kappa) - M_i^{\rm TRGB}(\kappa), \qquad (1)$$

where each of $m_i^{\rm TRGB}(\kappa)$, $A_\lambda(\kappa)$, and $M_i^{\rm TRGB}(\kappa)$ is the values drawn from the uncertainty distributions in the tip position, foreground extinction, and absolute magnitude of the tip, respectively. The foreground extinction and its uncertainty vary from object to object but the error in the absolute magnitude of the tip is a systematic error as already discussed. In using this method, there are two situations that can be encountered. The first is that the object is very well populated and the tip position is thus well constrained with a narrow PPD. In such instances, the uncertainty in $M_i^{\rm TRGB}$ far outweighs any other contributions to the error budget and is almost solely responsible for the width of the distance PPD. In the second situation, the object is poorly populated and the tip position PPD is very wide and typically asymmetric. If the LF population is not extremely low, the uncertainty in $M_i^{\rm TRGB}$ will contribute noticeably to the distance PPD, otherwise the distance PPD will essentially depend solely on the uncertainty in the determined tip positions. Hence while

some of the smaller contributions to the distance uncertainties are omitted from the calculations, their overall effects will be washed out by the contributions from these two principal sources of error.

3. ADDITION OF A MATCHED FILTER

3.1. Matched Filtering using Radial Density Profiles

With the introduction of our method in Paper I, it was stressed that one of its greatest attributes was its adaptability to the prior knowledge available for the object of interest. When applying the method to compact satellites, there is one very conspicuous attribute that can be incorporated into the prior information constraining the model fit—namely, the object's density as a function of radius. The simplest way to achieve this is with the addition to the algorithm of a matched-filter weighting scheme, wherein the weighting is *matched* to the specific data by accounting for the data within the filter itself.

The successes of Rockosi et al. (2002) using a matched filter in color–magnitude space to identify member stars of globular cluster Palomar 5 amidst the stellar background provide the inspiration for our technique. They make use of the characteristic RGB of the globular cluster to weight stars as to their likelihood of being cluster members. To achieve such a goal, a matched filter can be created by binning the CMD of the field in which the cluster lies into a two-dimensional matrix and then dividing that matrix by a similarly created background matrix. Stars found in the densest regions of the resulting matched filter CMD are then assigned the highest weight, being the most likely cluster members. In this way, they can greatly improve the signal-to-noise ratio (S/N) with respect to that of their original, unmodified data and are able to trace tidal streams from the globular cluster well into the surrounding background. Hence we have applied a similar approach to weight field stars fed to the MCMC in terms of their probability of being object members. In our case, however, the stars proximity to the object's center provides the basis for the weighting scheme, with the innermost stars being the most likely to be actual object members as opposed to background stars, and so a one-dimensional matched filter is sufficient.

The first step in implementing our weighting scheme is to ascertain a model of stellar density as a function of radius specific to the object of interest. For this purpose, we employ the best fits presented in N. F. Martin et al. (2012a, in preparation) for the dwarf spheroidal satellites, wherein the optimal ellipticity ϵ, position angle (P.A.), half-light radii (r_h), and object centers are given for exponential density profiles fitted to each satellite. For the two dwarf ellipticals, in the case of NGC 147 we assume $\epsilon = 0.44$ and P.A. $= 28^\circ$ as specified by Geha et al. (2010) and we derive the r_h manually as $10'$, which produces the best-fit profile to the data when coupled with the other two parameters. For NGC 185, we adopt $\epsilon = 0.26$ and P.A. $= 41^\circ$ based on the findings of Hodge (1963) and once again derive the r_h manually, this time as $6'$. For both NGC 147 and 185, we employ the object centers derived from the Two Micron All Sky Survey (2MASS; Skrutskie et al. 2006). With the ellipticity, P.A., half-light radius and object center know, we can proceed to produce a weighting scheme proportional to the density profile ρ of the object, where ρ is of the form

$$\rho(r_\epsilon) = e^{\frac{-r_\epsilon}{R}}, \tag{2}$$

where $R = (r_h/1.678)$ is the scale radius and r_ϵ is the elliptical radius at which the star lies, as now defined. With the P.A. and object center of the object known, a rotation of coordinates is used to define each star's position (x', y') with respect to the center of the ellipse. The projected elliptical radius r_ϵ of the ellipse on which the star lies is then

$$r_\epsilon = \left((y')^2 + \left(\frac{x'}{1-\epsilon} \right)^2 \right)^{1/2}, \tag{3}$$

where the y' axis is assumed as the major axis of the ellipse.

While Equation (2) gives us the functional form of our weighting scheme, it is further necessary to define the absolute values of the weights given to each star, so as to scale them appropriately with respect to the background density $\rho_{\rm bg}$. This is achieved by insuring that the area under the function $\rho(r_\epsilon)$ between any imposed inner and outer radius limits is set equal to the number of signal stars in the observed region. Hence, our weighting scheme is ultimately defined by

$$W(r_\epsilon) = S e^{\frac{-r_\epsilon}{R}} \tag{4}$$

with

$$S = \frac{(\rho_{\rm total} - \rho_{\rm bg}) \times A}{2\pi R(1-\epsilon)[(e^{\frac{-r_{\rm inner}}{R}})(R + r_{\rm inner}) - (e^{\frac{r_{\rm outer}}{R}})(R + r_{\rm outer})]}, \tag{5}$$

where $\rho_{\rm total}$ is the density of stars in the observed region before subtraction of the background density and A is the area of the observed region which is either an ellipse in the (usual) case that $r_{\rm inner} = 0$ or an elliptical annulus otherwise. $r_{\rm inner}$ and $r_{\rm outer}$ are the inner and outer cutoffs respectively of the range of r_ϵ values observed.

In Figure 1, the result of our fitting procedure as applied to the sparsely populated dwarf spheroidal Andromeda X is presented. In this case, stars out to $r_\epsilon = 0.^\circ15$ are fitted, with no inner cutoff radius imposed. While most of the satellites are too poorly populated for blending to be an issue, in the case of several, the stellar density counts at the innermost radii drop off in spite of the predicted counts from the fitted density profile. This is a good indicator of blending or overcrowding in those radii which can hinder the accuracy of the photometry for the affected stars and so in such cases, these inner radii are omitted. This was the case with Andromeda III ($r_{\rm inner} = 0.^\circ0175$), Andromeda V ($r_{\rm inner} = 0.^\circ011$), and Andromeda XVI ($r_{\rm inner} = 0.^\circ005$). For the dwarf ellipticals NGC 147 and NGC 185, it was found beneficial to avoid the inner regions altogether, with the presence of a wider range of metallicities in these regions degrading the contrast of the RGB tip. Similarly, an outer cutoff radius was chosen for these objects inside of 3 r_h to help sharpen the tip discontinuity, so that for NGC 147, $r_{\rm inner} = 0.^\circ28$ and $r_{\rm outer} = 0.^\circ33$ and for NGC 185, $r_{\rm inner} = 0.^\circ18$ and $r_{\rm outer} = 0.^\circ26$. M31 and M33 are treated similarly to the dwarf ellipticals but with still thinner annuli so that any weighting is unnecessary. They are discussed in more detail in Section 4.1.

With regard to the actual likelihood calculations used at each iteration of the MCMC, these are undertaken not by simple multiplication of the likelihood for each star by the respective weight, but by physically adjusting the relative proportions of the RGB and background components of the LF. Up until now, we have assumed a generic LF and calculated the likelihood contributions from each star from this single LF. But in reality, the outer regions of the field are more accurately represented by a shallow-signal/high-background LF while the innermost stars obey an LF which has almost no background component.

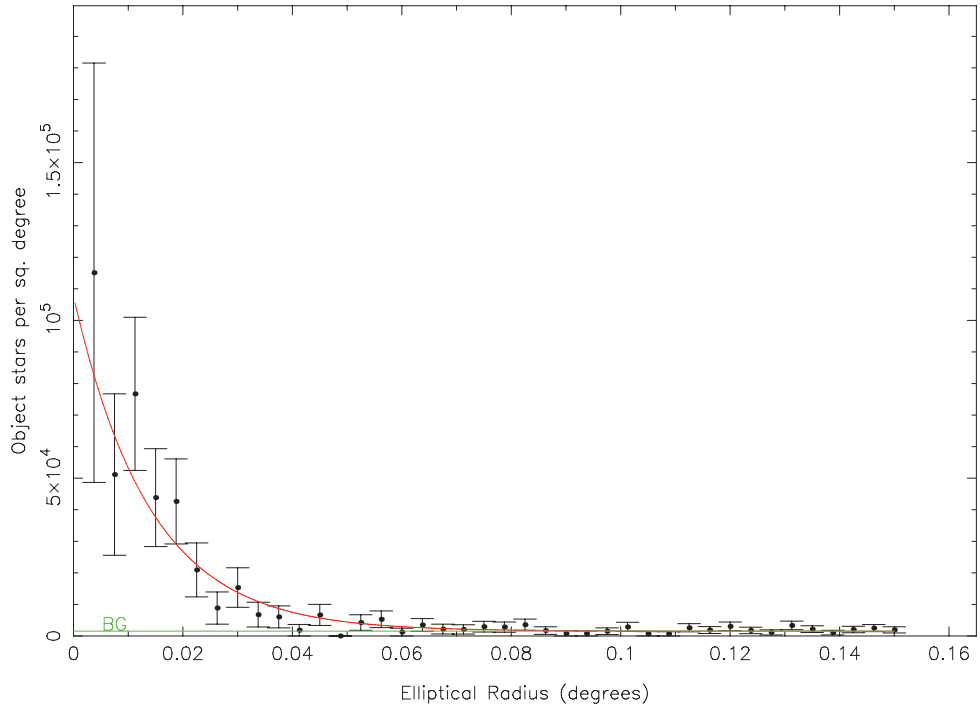


Figure 1. Radial density profile (proportional to object membership probability) for Andromeda X. The error bars represent the Poisson error in the density for each bin, with each bin representing an elliptical annulus at the stated radius. Hence the innermost annuli have the smallest areas and thus the largest error bars. Note that this binned density distribution is for comparison only and has no bearing on the fit. The background level is marked "BG."

(A color version of this figure is available in the online journal.)

Hence using the radial density profile obtained above, we can essentially build an individual LF for each star, tailored to suit its position within the object. In practice, this is achieved with almost no extra computational effort, as the background and signal can be normalized separately and only the signal component is changed by the MCMC at each iteration so that the background component need only be generated once. The two components are normalized to contain an area of unity and then the bin of each corresponding to the star's magnitude is scaled according to the ratios of the star's weight and the background level when its contribution to the model likelihood is calculated by the MCMC.

The result of the incorporation of this extra prior information is a marked improvement in the performance of the algorithm for the more sparsely populated targets. In such objects, the RGB component is typically overwhelmed by non-system stars, even with the most carefully chosen field size. This can greatly diminish the prospects of obtaining a well-constrained tip measurement. This is apparent from Figures 2 and 3 which show the LF and corresponding posterior distributions before and after the application of the matched filter to the dwarf spheroidal Andromeda X. With the matched filtering applied, the great majority of non-system stars are severely suppressed, revealing clearly the RGB component, which in turn provides much stronger constraints on the location of the tip, as evidenced by Figure 3. Herein lies an example of the power of the Bayesian approach, where a single prior can cast the available data in a completely different light.

3.2. A Test for the Refined Algorithm

In Section 2.3 of Paper I, the results of a series of tests were presented that characterized the performance of our original algorithm given a range of possible background density levels and LF populations. Here we present the results of similar tests applied to our new, matched-filter-equipped algorithm, but with some important differences. Most fundamentally, the way our artificial test data are generated is quite different. As we are now concerned with the position of each star in the field, a distance from field center must be generated for each star. To do this, we have randomly assigned a radial distance to each star, but weighted by a circularly symmetric ($\epsilon = 0$) exponential density profile. Further to this, the magnitudes of our stars are now generated directly from our convolved LF, so that photometric error as a function of stellar magnitude is incorporated.

The other important change from the previous tests concerns the way in which the artificial LFs are populated. Whereas in the former tests all of the sampled stars were drawn from the model LF within the one magnitude range $20 \leqslant m_{\rm star} \leqslant 21$, in the current tests the stars are drawn from within the much larger magnitude range actually utilized for our satellite measurements, namely $19.5 \leqslant m_{\rm star} \leqslant 23.5$. Hence a 100 star LF in these tests for example corresponds to a much smaller sample of stars than in the tests described in Section 2.3 of Paper I. Aside from these critical differences, the current tests are undertaken and presented as per the previous publication, with measurements of the average sigma and tip offset given for each combination of background level (f) versus number of stars (*ndata*) where $f = 0.1, 0.2, \ldots, 0.9$ and *ndata* = 10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10,000, 20,000. The results are presented in Figures 4 and 5, respectively.

Examination of the figures reveals the expected trend of increased 1σ error and tip offset with increasing background height and decreasing LF population levels. Once again, there is very good agreement between the derived errors and the actual offsets obtained. Most importantly, it is clear by comparing these results with those of Paper I that the matched filtering has

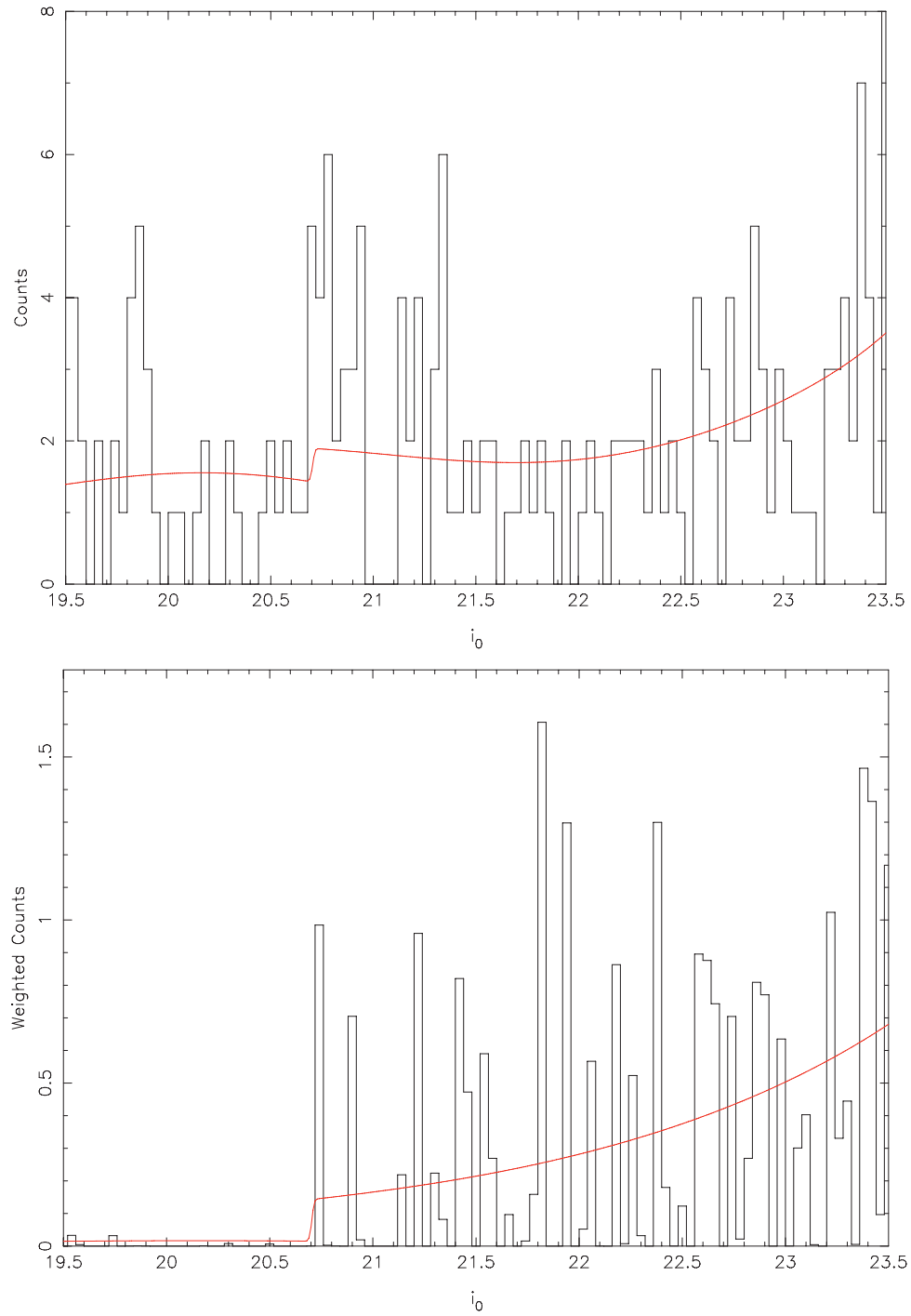


Figure 2. Best-fit model to the luminosity function of Andromeda X, obtained with the addition of matched filtering. The top figure shows the best fit overlaid on the unmodified LF (i.e., histogram created without the weighting afforded by the matched filter). The bottom figure shows the same best-fit model after applying the weighting. A field radius of $0.^{\circ}15$ was used to generate the LF histograms, wherein each star contributes between 0 and 1 "counts," depending on its proximity to the field center and the density profile of the object.

(A color version of this figure is available in the online journal.)

greatly diminished the effects of the background contamination, as exemplified by the much gentler increase in 1σ errors and offsets with increasing background star proportion.

3.3. An Additional Prior

In addition to our density matched filter, a further prior may be devised so as to constrain our distance posterior probability distributions (PPDs) in accordance with our knowledge of the M31 halo dwarf density profile. The expected falloff in density of subhalos within an M31-sized galaxy halo is not well constrained. The largest particle simulation of an M31-sized dark matter halo to date, the Aquarius Project (Springel et al. 2008), favored the density of subhalos to fall off following an Einasto profile with $r_{-2} = 200$ kpc and $\alpha = 0.678$, and furthermore identified no significant dependence of the

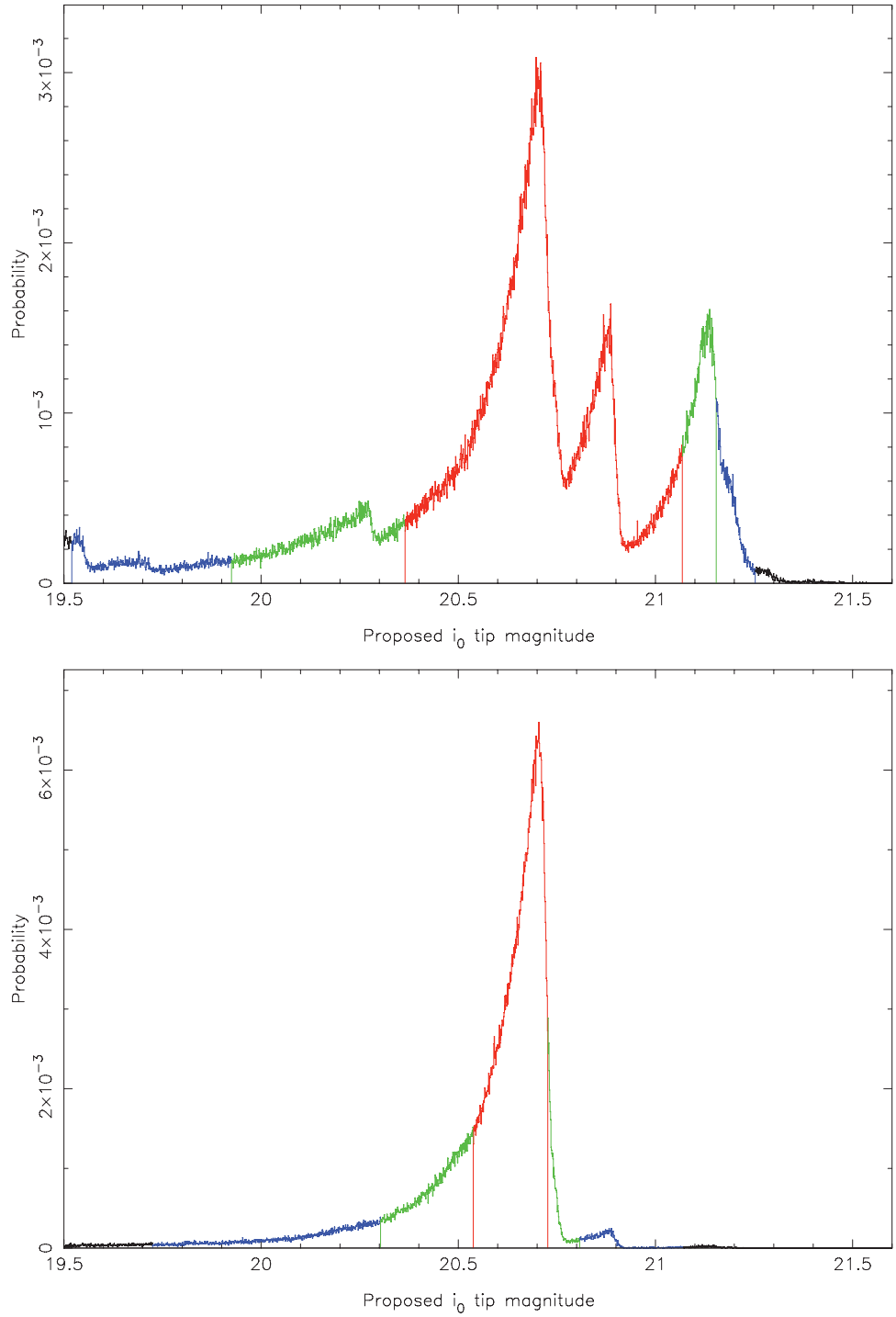


Figure 3. Posterior distributions obtained for Andromeda X before (top) and after (bottom) the application of the matched filter. For the "before" case, a circular field of radius 0.°05 (2.143 × r_h) has been chosen, specifically to provide the most possible signal with the least possible background contamination. For the "after" case, the same LF as presented in Figure 2 is used.

(A color version of this figure is available in the online journal.)

relationship on subhalo mass. For the specific case of the satellites within the M31 halo, Richardson et al. (2011) found a relation of $\rho \propto r^{-\alpha}$ where $\alpha = 1$ a better fit to the data, drawing largely from the PAndAS survey, although this does not take into account the slightly irregular distribution of the survey area. We adopt this more gentle density falloff with radius giving us a more subtle prior on the satellite density distribution and note that α may be changed significantly without great effect on our measured distances.

So in effect, we assume a spherical halo centered on M31, such that $\rho(\mathrm{sat}) \propto r^{-1}$ and integrate along a path through the halo at an angle corresponding to the angular displacement on

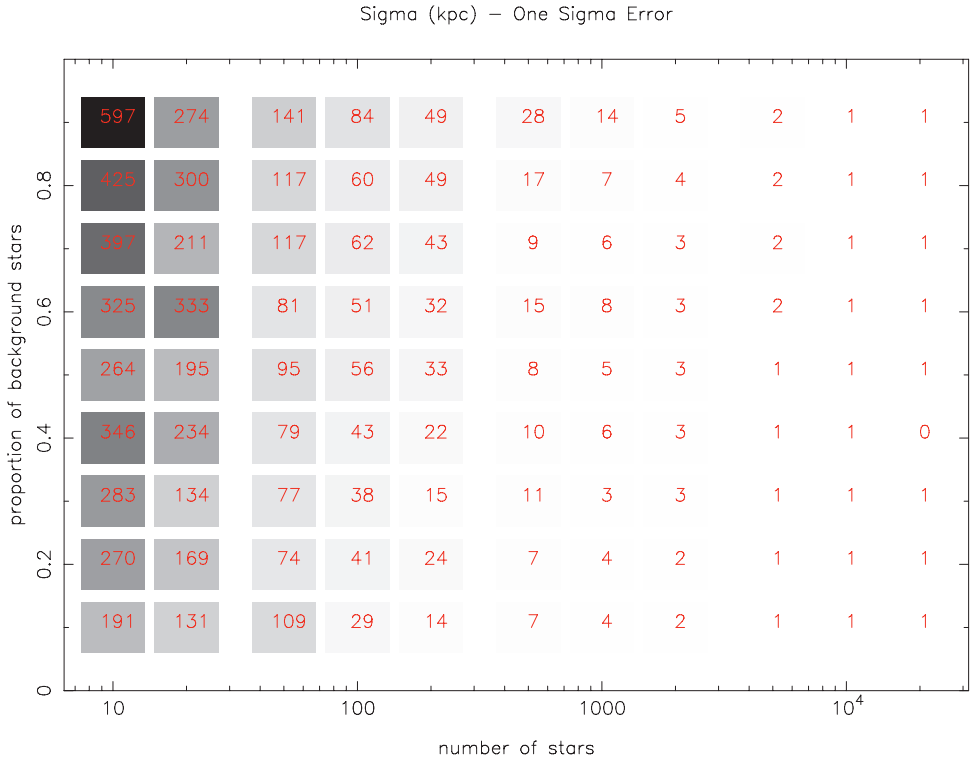


Figure 4. Gray-scale map of the 1σ error in tip magnitude obtained for different combinations of background height and number of sources. The actual value recorded for the error (in kpc) is overlaid on each pixel in red. Each value is the average of twenty 50,000 iteration runs for the given background height/LF population combination.

(A color version of this figure is available in the online journal.)

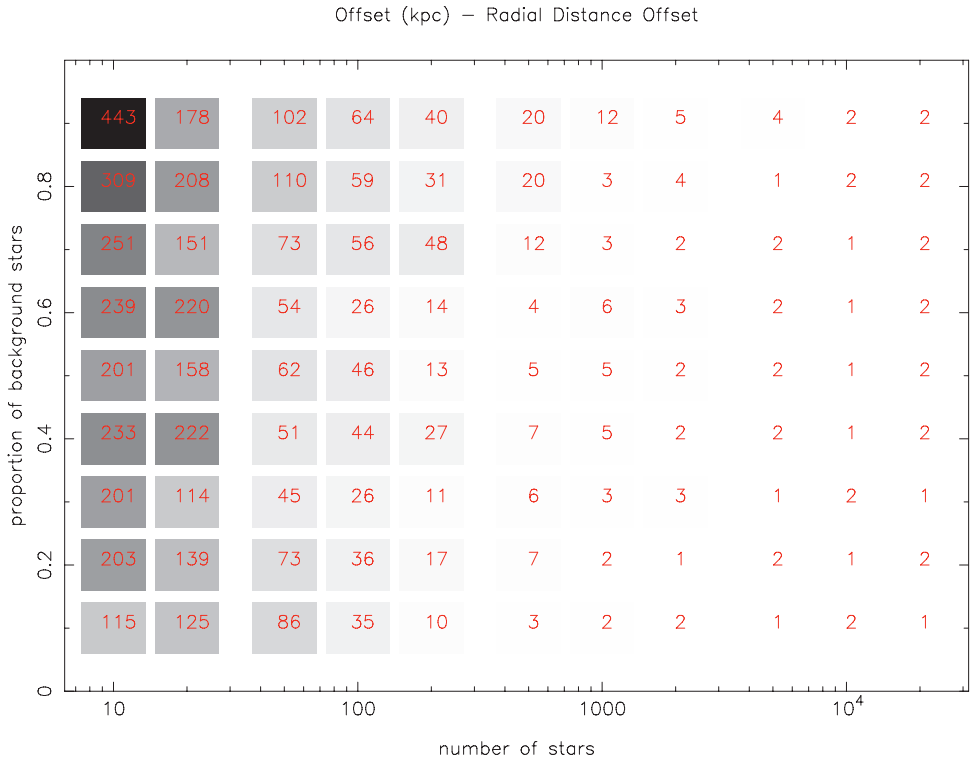


Figure 5. Gray-scale map of the offset of the measured tip value from the true tip value obtained for different combinations of background height and number of sources. The actual value recorded (in kpc) is overlaid on each pixel in red. Each value is the average of twenty 50,000 iteration runs for the given background height/LF population combination.

(A color version of this figure is available in the online journal.)

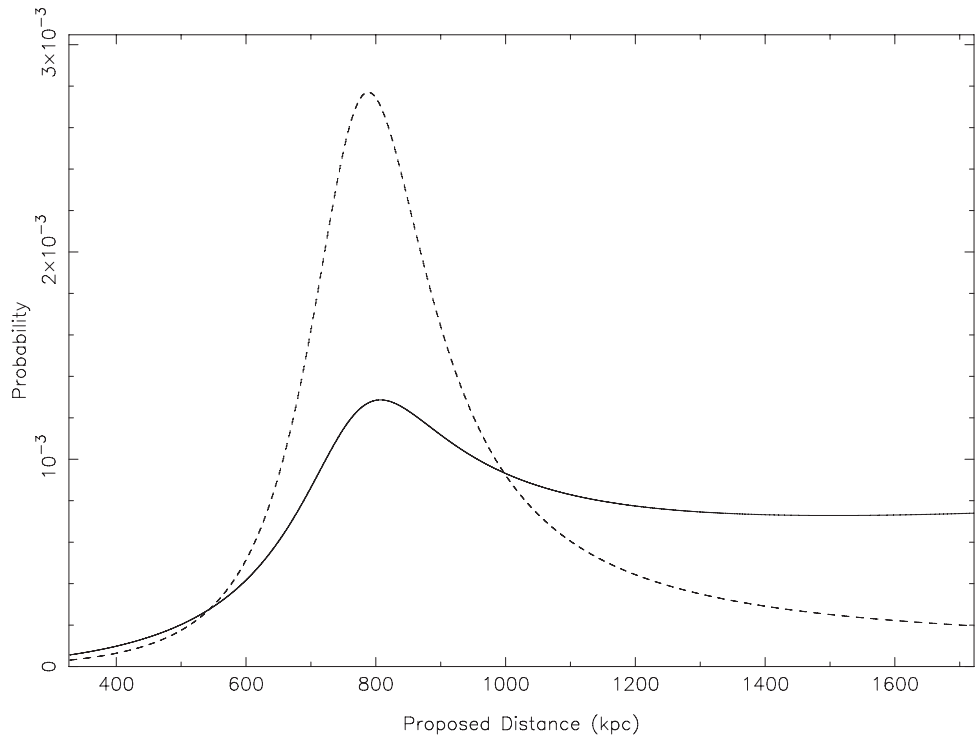


Figure 6. Distance prior applied to Andromeda XIII (solid line; $\alpha = 1$). The distribution gives the likelihood of the satellite existing at a particular distance, given an angular separation on the sky of 8.°5 from M31 (the halo center) and assuming a distance of 779 kpc for M31. The distribution peaks where the line of sight traverses the innermost region of the halo, and flattens out at large distances due to the increasing volume of the halo subtended by the unit of solid angle observed. The same prior with $\alpha = 2$ is shown as a dashed line for reference. While this value for alpha is in closer agreement with the results of Section 4.3, we deliberately adopt the less restrictive $\alpha = 1$ prior, so as not to suppress the probability of satellites in the outer halo too greatly.

the sky of the satellite from M31. This yields an equation of the form

$$P(d) \propto \frac{d^2}{((d^2 + 779^2 - 2d \times 779 \times \cos(\theta))^{\alpha})^{1/2}}, \quad (6)$$

where $\alpha = 1{,}779$ kpc is the distance to M31, and $P(d)$ is the relative probability of the satellite lying at distance d (in kpc) given an angular separation of θ degrees from M31. Note that this produces a peak where the line of sight most closely approaches M31, and that $P(d \gg 779)$ is approximately proportional to d. The equation is normalized between limits appropriate to the size of the halo.

We thus generate a separate prior for the probability as a function of distance for each satellite, tailored to its specific position with respect to M31. The effect of the prior is to suppress unlikely peaks in the multi-peaked posterior distributions obtained for certain satellites, while leaving the peak positions unaffected. As such, the prior has very little effect on single-peaked distributions, whatever the angular position and distance of the satellite it represents. The distance prior applied to the Andromeda XIII distance PPD is shown in Figure 6 for illustration.

4. A NEW PERSPECTIVE ON THE COMPANIONS OF M31

4.1. Galaxy Distances

The PAndAS survey provides us with a unique opportunity to apply a single method to a homogeneous data sample encompassing the entire M31 halo out to 150 kpc. The data encompass many dwarf spheroidals, along with the dwarf ellipticals NGC 147 and NGC 185, and of course the M31 disk itself with additional fields bridging the gap out to the companion spiral galaxy M33, some 15° distant. Of these objects, the vast majority have metallicities [Fe/H] $\leqslant -1$, so that any variation in the absolute magnitude of the tip is slight. Indeed, Bellazzini (2008) suggests that for such metallicities, the variation in the region of the spectrum admitted by the CFHT i' filter is perhaps less than in Cousins' I. Perhaps of greatest concern are the cases of M31 and M33, which will contain substructure at a variety of metallicities. In this case, however, the more metal-rich portions will exhibit a fainter TRGB than those in the regime [Fe/H] $\leqslant -1$, such that the brightest RGB stars will fall within this regime.

In this section we present distance measurements to these many halo objects, culminating in Figure 10 below, a three-dimensional map of the satellite distribution, and Table 2, which presents the satellite data pertinent to our distance measurements. Figures 11 and 12 below present the distance posterior distributions obtained for every object in this study. It has been common practice in the majority of TRGB measurements to quote simply the most likely distance and estimated 1σ uncertainties, but this throws away much of the information, except in the rare case that the distance distribution is actually a perfect Gaussian. On account of this, as well as providing the actual distance PPDs themselves for visual reference, we also provide the same information in condensed tabular form, where the object distance is given at 1% increments of the PPD, both for the prior-inclusive cases (as in Figures 11 and 12) and for the case in which no prior is invoked on the halo density. Note that for M31, no halo density prior is applied and so this column is set to zero. A sample of this information, as provided for

Table 1
Tabulated Distance Posterior Distribution

Percentage	Distance (kpc, no density prior)	Distance (kpc)
1	684	687
2	688	692
3	691	695
4	693	697
5	695	699
6	697	701
7	698	702
8	699	703
9	700	704
10	701	705
:	:	:
:	:	:
100	820	820

Notes. Distance posterior probability distributions for Andromeda I given at 1% intervals for the case of no halo density prior (Column 2) and with the angle-specific prior outlined in Section 3.3 applied (Column 3).

(This table is available in its entirety in a machine-readable form in the online journal. A portion is shown here for guidance regarding its form and content.)

Andromeda I, is presented in Table 1. The reader may then sample from these distributions directly rather than use the single quoted best-fit value, thus taking into account the true uncertainties in the measurements.

Due to the large number of objects studied, it is not practical to discuss each in detail within this paper. For this reason, Andromeda I will be discussed in further detail below as a representative example, followed by two of the more problematic cases for completeness. First, however, we describe the exceptional cases of M31 itself and M33.

M31 and M33 due to their large extent on the sky and the variety of substructure in their disks require a slightly different approach to that used for the other objects in this study. As was the case for NGC 147 and NGC 185, it was necessary to define a thin elliptical annulus so as to limit as much as possible the amount of substructure from other radii contaminating the LF. For both M31 and M33 such a thin annulus was used that any weighting with respect to the elliptical radius of the stars was trivial and so no weighting was used. For M31, an ellipticity of 0.68 was adopted, with P.A. $= 37^\circ$. The inner and outer elliptical cutoff radii were set to $2\overset{\circ}{.}45$ and $2\overset{\circ}{.}5$, respectively. To check for any inconsistencies in the TRGB location across the whole annulus, it was divided up into NE, NW, SE, and SW quarters and then the distance measured from each quarter, giving distances of 782^{+19}_{-19}, 782^{+18}_{-18}, 775^{+20}_{-18}, and 781^{+19}_{-19} kpc, respectively. It is tempting to associate the slightly lower distance to the SE quadrant with the effects on the LF of the Giant Stellar Stream, though the distance is still within close agreement with the other three quadrants, such that all four are perfectly consistent. Hence, the distance was remeasured using the whole annulus to give 779^{+19}_{-18} kpc. This is in good agreement both with the findings of McConnachie et al. (2005) (785^{+25}_{-25}) utilizing the TRGB and the more recent determination by Riess et al. (2012) using Cepheid variables (765^{+28}_{-28}).

For M33, we employ an ellipticity of 0.4 as used by McConnachie et al. (2005), but find a position angle of P.A. $=$ 17° in closest agreement with the data. Inner and outer elliptical radii of $r_{\rm inner} = 0\overset{\circ}{.}75$ and $r_{\rm outer} = 0\overset{\circ}{.}9$ were adopted to give a very sharp discontinuity at the location of the tip. After applying an appropriate color-cut, the qualifying stars were fed into our algorithm to give a distance of 820^{+20}_{-19} kpc. This distance is in good agreement with that of 809^{+24}_{-24} kpc obtained by McConnachie et al. (2005) and yields an M33-to-M31 distance of 214^{+6}_{-5} kpc. It is interesting to note that a variety of quite different M33 distances exist in the literature, with derived distance moduli ranging from 24.32 (730 kpc, water masers; Brunthaler et al. 2005) through 24.92 (964 kpc, detached eclipsing binaries; Bonanos et al. 2006). Indeed, the variety of standard candles utilized would suggest that M33 provides an ideal environment for calibrating the relative offsets between them. McConnachie (2005) suggests that the dispersion of M33 distances in the literature is tied to an inadequate understanding of the extinction in the region of M33. Most measurements, including those presented here, use the Galactic extinction values derived by Schlegel et al. (1998), although these do not account for extinction within M33 itself and are calculated via an interpolation of the extinction values for the surrounding region. Nevertheless, the elliptical annulus employed in our approach will act to smooth out the field-to-field variation that might exist between smaller regional fields.

4.1.1. Andromeda I: Example of an Ideal Luminosity Function

It would seem prudent to illustrate the performance of our new method by presenting the results for a range of the dwarf spheroidals from the most populated to the least populated. Hence Andromeda I, the first discovered and one of the two most highly populated of these objects, is the obvious place to start. The field employed for our Andromeda I distance measurement incorporated stars at elliptical radii between $0^\circ \leqslant r_\epsilon \leqslant 0\overset{\circ}{.}3$ and, after removal of stars outside of the range $19.5 \leqslant i_0 \leqslant 23.5$ and beyond our chosen color-cut, yielded a star count of 4375. The CMD for this field is presented in Figure 7(a). This figure color-codes the stars in the CMD as per the color distribution in the inset field and plots them so that those innermost within the field (and hence those accorded the highest weight) are represented by the largest dots. In the case of Andromeda I, the RGB is so dominant over the background that our density matched filter is hardly necessary and hence does little to improve the already stark contrast. It is not surprising therefore that the distance and uncertainty obtained are almost identical to those obtained by the base method as presented in Paper I. Andromeda I is thus confirmed at a distance of 727^{+18}_{-17}, which allows us to derive a similarly accurate separation distance from M31 of 68^{+22}_{-16} kpc.

4.1.2. Andromeda XV: Example of a Multi-peaked Distance PPD

As an example of a dwarf spheroidal of intermediate size, we present the comparatively compact Andromeda XV. Far from being the tidiest example of the many intermediate-sized objects covered in this study, Andromeda XV provides something of a challenge. Examination of Figure 8 reveals a gradual rise in star counts when scanning from the top of the CMD color-cut faintward toward the Andromeda XV RGB and a correspondingly broad range in the possible tip locations in the tip magnitude PPD. Indeed, two peaks are prominent in the distance PPD of Figure 8(c), with the distribution mode at 626 kpc (our adopted distance) and the 1σ credibility interval spanning from 591 kpc to 705 kpc as a consequence of the second peak. Ibata et al. (2007) determine this object to lie at a distance of 630^{+60}_{-60} kpc, which would correspond to a tip magnitude of approximately $m_i^{\rm TRGB} = 20.56$ assuming

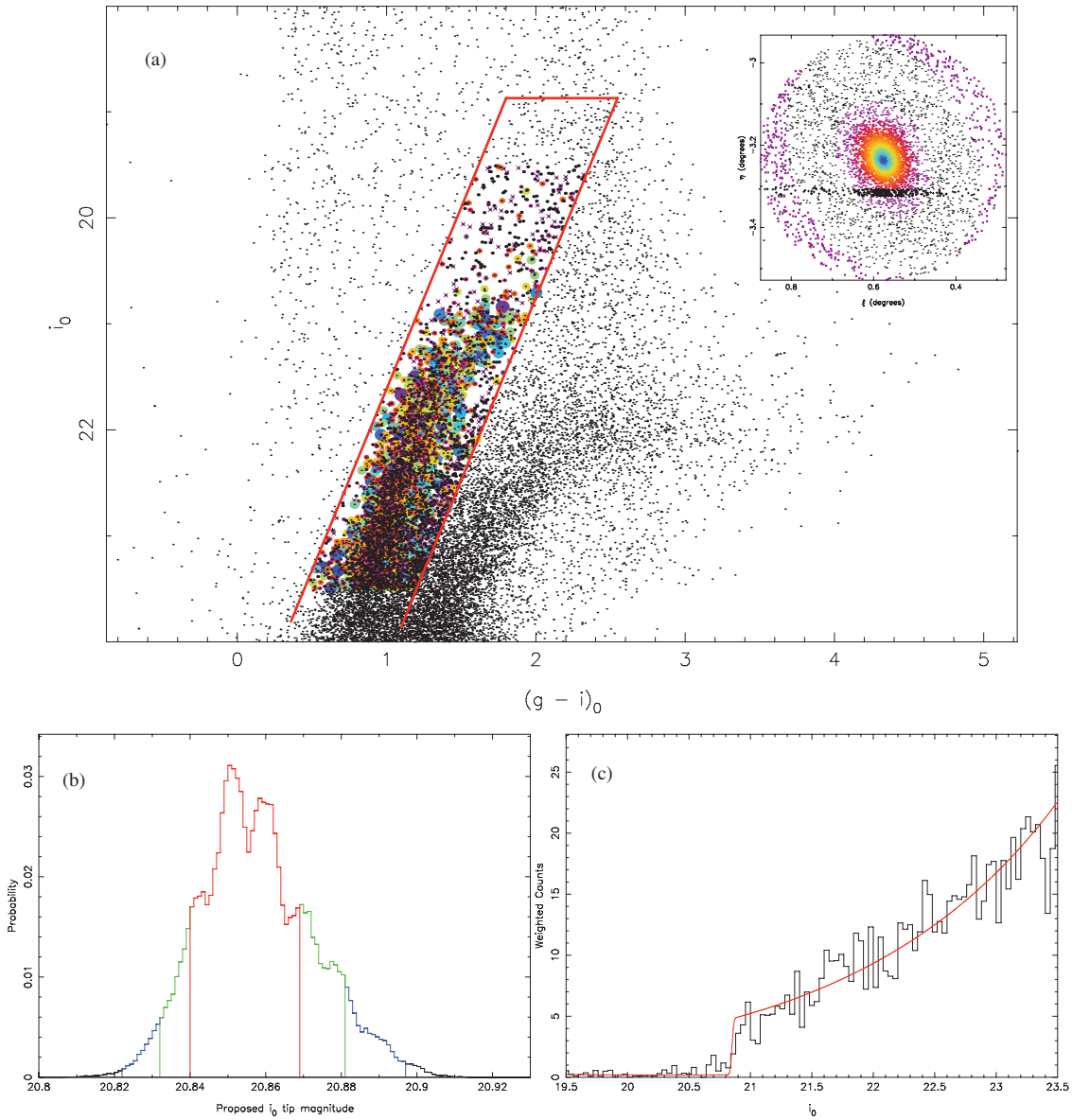


Figure 7. Andromeda I: (a) color-coded CMD representing the weight given to each star in the field. Only stars within the red selection box with magnitudes $19.5 \leqslant i_0 \leqslant 23.5$ were fitted and hence color-coded. The second, fainter RGB lying toward the redder end of the CMD is that of the giant stellar stream which passes behind our Andromeda I field. The inset at top right shows the field with the same color-coding and acts as a key. The field is divided into 20 radii bins following a linear decrease in density from the core (blue) to the field edge (purple). Stars marked as a purple "×" lie outside of the outer elliptical cutoff radius $r_{\rm outer}$. Stars marked as a black "+" are artificial stars used in the estimation of the background density and are ignored by the MCMC; (b): posterior probability distribution for the TRGB magnitude. The distribution is color-coded, with red indicating tip magnitudes within 68.2% (Gaussian 1-sigma) on either side of the distribution mode, green those within 90%, and blue those within 99%; (c) weighted LF of satellite with superimposed best-fit model in red. A star at the very center of the satellite contributes 1 count to the luminosity function while those further out are assigned some fraction of 1 count in proportion with the satellite's density profile.

(A color version of this figure is available in the online journal.)

$M_i^{\rm TRGB} = -3.44$. This is in excellent agreement with the $m_i^{\rm TRGB} = 20.57^{+0.23}_{-0.14}$ recovered by this study. Letarte et al. (2009) however derive a distance of 770^{+70}_{-70} kpc which places it toward the far edge of our 99% credibility interval on the distance (see Figure 8(c)). This measurement was derived after three stars that had been found to lie close to the Andromeda XV RGB tip in the former investigation were identified as Galactic foreground stars, following measurements of their radial velocities obtained with the Deep Imaging Multi-Object Spectrograph on Keck II. Of these stars, however, none lies within 2′ from our object center by which point the maximum possible weighting has already dropped to below 10%, meaning that even the highest weighted of these three stars will have minimal effect on the likelihood calculation. This would then suggest that each of these three stars has magnitude consistent with belonging to the Andromeda XV RGB.

4.1.3. Andromeda XIII: Example of a very Poorly Populated Luminosity Function

Andromeda XIII is among the most sparsely populated objects targeted by the current study and it is important to realize that it is impossible to obtain distances to such objects with small uncertainties using the TRGB standard candle, unless of course one of the few member stars can be positively identified as

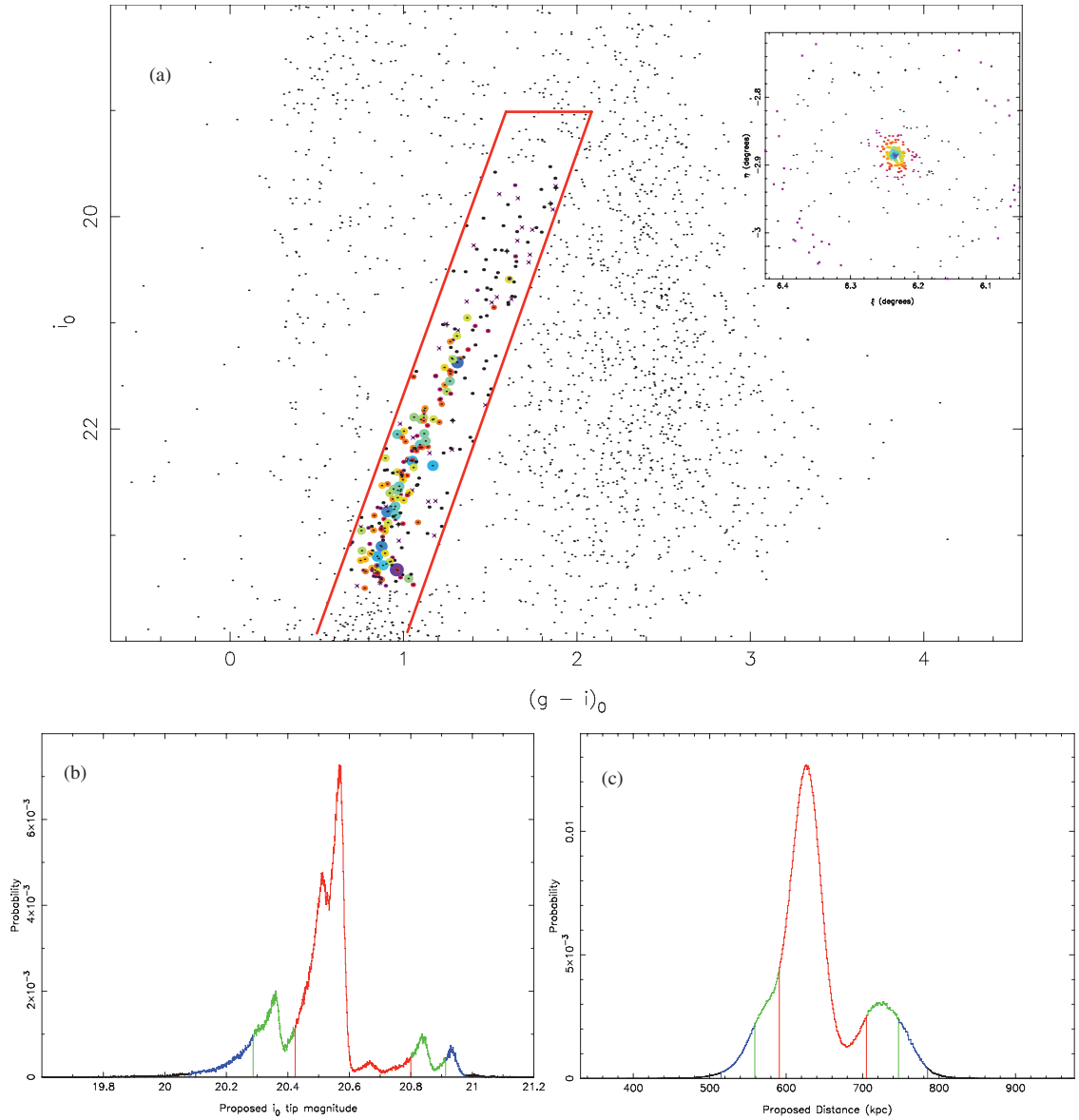


Figure 8. Andromeda XV: (a) same as Figure 7(a) but for Andromeda XV; (b) same as Figure 7(b) but for Andromeda XV; (c) sampled distance posterior probability distribution, obtained by calculating the distance 3 million times, each time randomly drawing on the tip magnitude, absolute magnitude of the tip, and extinction from their respective probability distributions. The distribution is color-coded, with red indicating possible distances within 68.2% (Gaussian 1 σ) on either side of the distribution mode, green those within 90%, and blue those within 99%. Note that the large uncertainty in the absolute magnitude of the RGB tip is primarily responsible for the much smoother appearance of the distance PPD (c) compared with the tip PPD (b).

(A color version of this figure is available in the online journal.)

being right on the brink of core helium fusion. Nevertheless, though large uncertainties are inevitable, an accurate estimation of those uncertainties is still achievable, and this is the aspiration of the method here presented. Distances to Andromeda XI and XIII have been obtained with higher accuracy using RR Lyrae stars as a standard candle with photometry from the *Hubble Space Telescope* (Yang & Sarajedini 2012). In the case of Andromeda XI, the tip magnitude identified by our method agrees well with the distance identified by that study, but in the case of Andromeda XIII, a brighter star in the central regions of the field causes some confusion. Indeed in such a sparsely populated field it is quite difficult to apply any effective density-based weighting scheme. Nevertheless, after sampling the tip magnitude PPD (Figure 9(b)), together with those for the absolute magnitude of the tip and the extinction in this region of sky to obtain a sampled distance PPD, and multiplying that distribution with the angle-specific halo density prior as is standard for all our measurements, we are able to produce a distance PPD (Figure 9(c)) in good agreement with the findings of Yang & Sarajedini (2012).

4.2. Determining the Distances from M31

Once a satellite's distance from Earth is determined, it is straightforward to determine the distance from M31 using the cosine rule:

$$r = (d^2 + (d_{M31})^2 - 2dd_{M31}\cos(\theta))^{1/2}, \quad (7)$$

where r is the satellite's distance from M31, d is the distance of the satellite from Earth, d_{M31} is the distance of M31 from Earth,

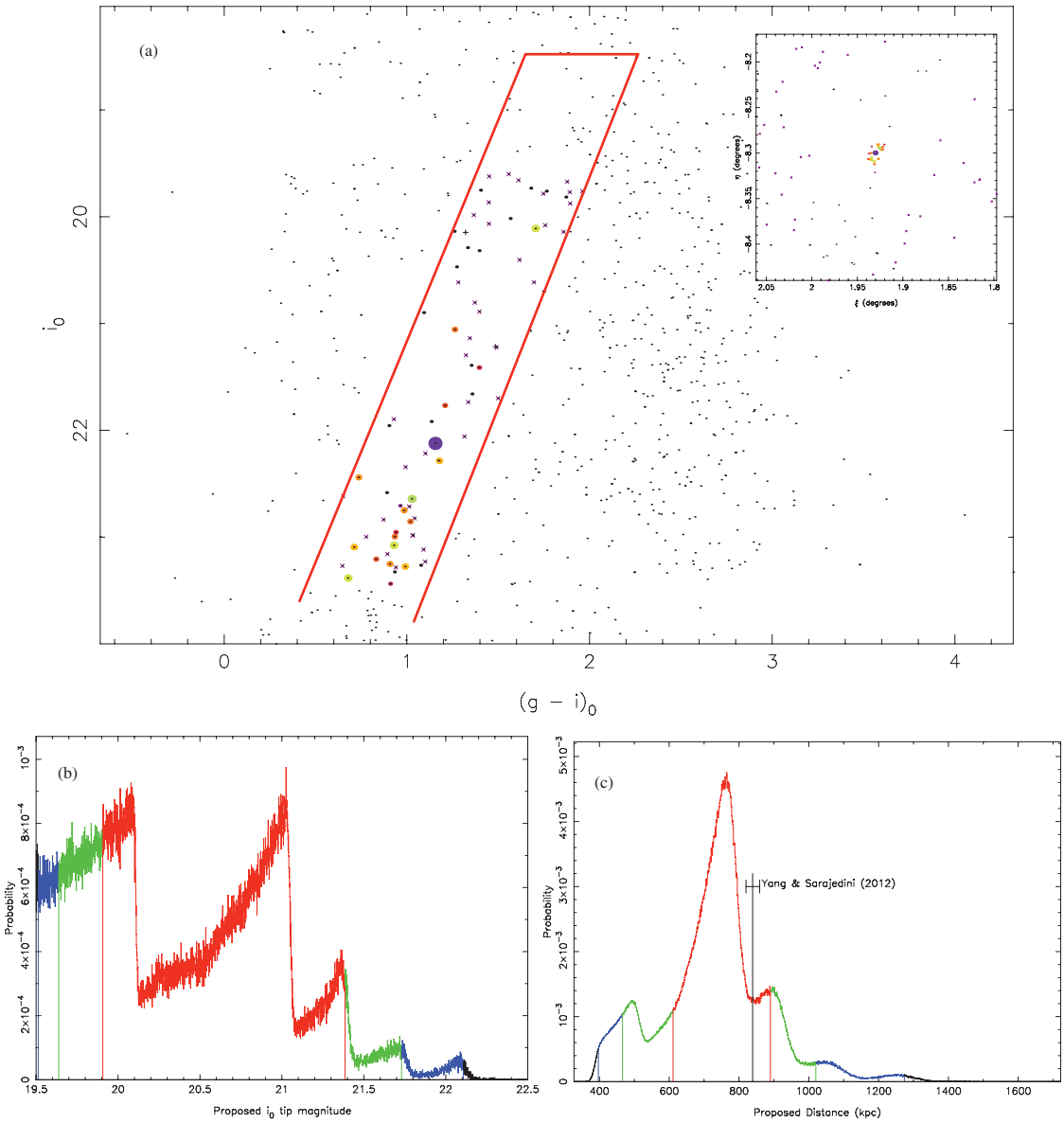


Figure 9. Andromeda XIII: All figures as per Figure 8, but for Andromeda XIII. The distance derived by Yang & Sarajedini (2012) is plotted in (c) along with error bars for comparison.

(A color version of this figure is available in the online journal.)

and θ is the angle on the sky between M31 and the satellite. For convenience, we use a small angle approximation equating θ with its M31 tangent plane projection and note that any displacement of r is insignificant due to the size of the 1σ errors. If the uncertainty in distance to both M31 and the satellite takes on a Gaussian distribution, it is straightforward to determine the error in the satellite–M31 separation by adding the individual errors in quadrature. While it is reasonable to approximate the M31 distance uncertainty distribution as a Gaussian, the same cannot be said for each of the companion satellites. Hence once again it is more appropriate to sample values from the individual distance probability distributions. Thus, a histogram of r values for the satellite is built up by sampling d and d_{M31} from their respective distributions over many iterations. This brings to the fore an important consideration: there is an integrable singularity in the resulting distribution at the closest approach distance to M31 ($r_c = d_{M31}\sin(\theta)$) as shown below.

The probability distribution for the satellite-to-Earth distance $P(d)$ is related to that of the satellite-to-M31 distance $P(r)$ as follows:

$$P(r) = \frac{\delta d}{\delta r} P(d). \tag{8}$$

From Equation (7), and further noting that the satellite-to-Earth distance corresponding to r_c is $d_c = d_{M31}\cos(\theta)$, we have

$$\frac{\delta d}{\delta r} = \frac{r}{d - d_c}, \tag{9}$$

which allows us to derive

$$P(r) = \frac{r}{\left(r^2 - r_c^2\right)^{1/2}} P(d), \tag{10}$$

thus producing the singularity at $r = r_c$. In practice, after factoring in the Gaussian distribution in d_{M31}, this results in a sharp

peak at the minimum possible satellite-to-M31 distance when dealing with the more asymmetric satellite-to-Earth distance probability distributions. Hence when considering the distribution of satellites as a function of distance from M31, one can either take the distances as determined directly from Equation (7) using solely the most likely distance from the satellite-to-Earth distance distributions or the whole distance probability distribution for a satellite can be allowed to influence the calculations, as accomplished via sampling. The final result can be quite different, depending on the choice.

4.3. A First Approximation of the Satellite Density Profile within the Halo

In the completed PAndAS survey, we have for the first time a comprehensive coverage of a galaxy halo, with a *uniform* photometric depth sufficient to identify even the comparatively faint satellite companions. In addition, in this paper we have provided distances to every one of these objects, all obtained via the *same* method. We are thus presented with an excellent opportunity to study the density of satellites as a function of radius within a Milky Way like halo.

As hinted at in the previous section, obtaining an accurate picture of the satellite density profile (SDP) is not a trivial task. The first major consideration is to devise a way of factoring in the selection function. Comprehensive though the survey coverage is, it is not symmetric and not infinite. Second, the choice of model for the SDP is not arbitrary. Whether a simple, unbroken power law is sufficient is not immediately clear. Furthermore, does it even make any sense to treat the halo as a radially symmetric, isotropic distribution? A glance at the obvious asymmetry in Figure 10(a) would suggest otherwise. Nevertheless, for a first approximation it is reasonable to consider what the best-fitting radially symmetric, unbroken power law to the SDP would be.

The PAndAS survey covers approximately 400 deg^2 of sky and is roughly symmetric about the center of the M31 disk but with a major protrusion in the southeast to encompass the M33 environs. For the purpose of obtaining an accurate measure of the survey coverage of the halo as a function of radius, as well as factoring in the actual survey borders, an inner ellipse was also subtracted where the presence of the M31 disk has made satellite detection more difficult. Both the outer survey borders and the inner cutoff ellipse are plotted in Figure 10. The inner cutoff ellipse has an eccentricity $\epsilon = \sqrt{0.84}$ and is inclined with the semi-major axis angled 51.°9 with respect to the x-axis ($\eta = 0$). The dwarf galaxies M32 and M110 lie inside this ellipse as do the somewhat dubious satellite identifications Andromeda VIII and Andromeda IV (see Ferguson et al. 2000), hence their omission from the data presented in Table 2. With the inner and outer boundaries suitably delineated, the procedure then was to determine what fraction of halo volume at a given radius $f(r)$ would fall within these boundaries once projected onto the M31 tangent plane. This was achieved by implementing the even–odd rule on the projections of uniformly populated halo shells.

Having determined $f(r)$, we can proceed to determine the required normalization for a power law of any given α, allowing us to use the power law directly as a probability distribution. Setting the problem out in terms of probabilities, we require to determine the probability of each tested M31-to-object distance (henceforth simply "radius") r given a power law with slope α:

$$P(r|\alpha) = \frac{k}{r^\alpha}, \tag{11}$$

where k is the normalization constant and $r_{\rm min} \leqslant r \leqslant r_{\rm max}$. Using the assumed spherical symmetry, we then have

$$f(r)\int_{r_{\rm min}}^{r_{\rm max}} P(r|\alpha)\int_0^{2\pi}\int_0^{\pi} r^2 \sin\theta d\theta d\phi dr = 1 \tag{12}$$

so that

$$4\pi f(r)\int_{r_{\rm min}}^{r_{\rm max}} kr^{2-\alpha} dr = 1. \tag{13}$$

Hence, for a given radius at a given α, we have

$$k(r,\alpha) = \left[4\pi f(r)\left(\frac{r^{3-\alpha}}{3-\alpha}\right)_{r_{\rm min}}^{r_{\rm max}}\right]^{-1}. \tag{14}$$

The calculation of the likelihood for a power law of a given slope α may be simplified by noting that for any given radius, $f(r)$ and hence k act to scale the probability in an identical way whatever the value of α. Thus, the dependence of k on r is effectively marginalized over when the posterior distribution for α is calculated, so long as any sampling of radii utilizes the same radii at every value of α. The likelihood for a given power law (i.e., a given α) is thus

$$\mathcal{L}(\alpha) = \prod_{n=i}^{\rm nsat} kr_i^{2-\alpha}, \tag{15}$$

where nsat is the number of satellites—i.e., the 27 companions of M31 listed in Table 2. As discussed in Section 4.2, there are essentially two ways we can determine the likelihood of a given α. The most straightforward is to use single values of r_i as determined directly from the mode in the posterior distribution for each satellite using Equation (7). The second and arguably more robust method is to use the entire radius probability distribution (RPD) for each satellite. In the case of this second approach, the likelihood for the power law determined for each satellite becomes a convolution of the power law with the satellite's RPD, so that the likelihoods of the individual samples are summed. The final likelihoods determined for each satellite can then be simply multiplied as before, giving a total likelihood as follows:

$$\begin{aligned}\mathcal{L}(\alpha) &= \left(kr_{1,1}^{2-\alpha} + kr_{2,1}^{2-\alpha} + \cdots + kr_{\rm nsam,1}^{2-\alpha}\right)\\ &\quad\times\left(kr_{1,2}^{2-\alpha} + kr_{2,2}^{2-\alpha} + \cdots + kr_{\rm nsam,2}^{2-\alpha}\right)\times\cdots\\ &\quad\times\left(kr_{1,\rm nsat}^{2-\alpha} + kr_{2,\rm nsat}^{2-\alpha} + \cdots + kr_{\rm nsam,nsat}^{2-\alpha}\right)\\ &= \prod_{n=i}^{\rm nsat}\left[\sum_{n=j}^{\rm nsam} kr_{j,i}^{2-\alpha}\right]\end{aligned} \tag{16}$$

where $r_{j,i}$ is the jth sampled radius of the ith satellite, and nsam is the total number of samples.

The resulting distribution achieved by implementing the first approach is presented in Figure 13(a) from which a value for α of $1.92^{+0.32}_{-0.30}$ is obtained. It is interesting to note that this value is consistent with an isothermal satellite distribution with uniform velocity dispersion. Replacing the individual best-fit radii with 500,000 samples from the respective RPD for each satellite as per the second approach, the result is substantially different, as demonstrated by Figure 13(b). Here a value for α of $1.52^{+0.35}_{-0.32}$ provides the best fit to the data. This discrepancy is presumably a consequence of the non-Gaussian RPD profiles for

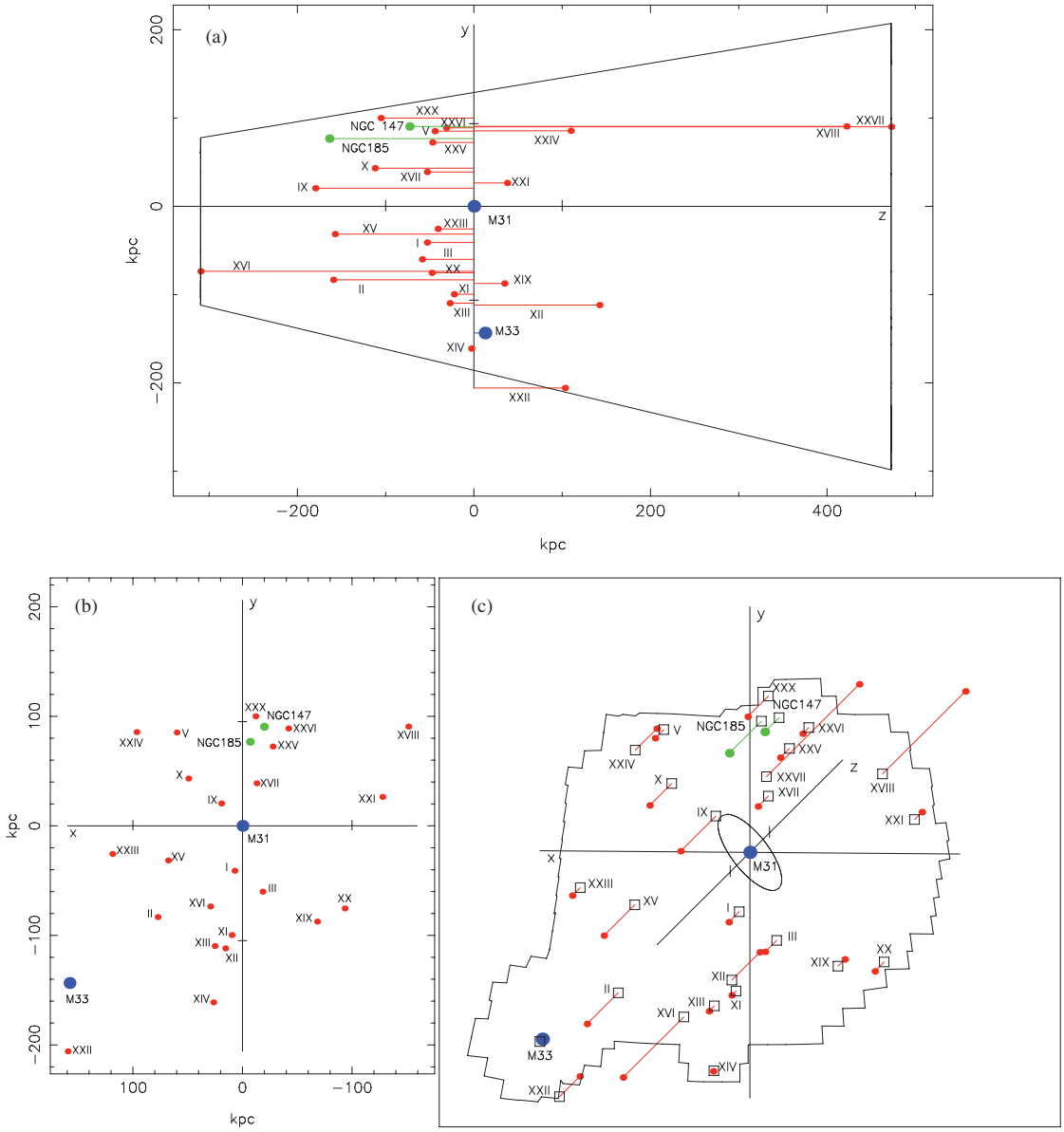


Figure 10. Three views of the M31 neighborhood: (a) a view of the satellites of M31 along the y–z plane. The conic section illustrates the extent of volume covered by the PAndAS footprint as a function of distance from Earth; (b) a view of the satellites of M31 in the x–y plane, revealing their true positions on the x–y plane after removing the effects of perspective (assuming the distances quoted in Column 4 of Table 2). Note that Andromeda XXVII lies directly behind NGC 147 in this plot and is not labeled; (c) a three-dimensional view of the satellites of M31. The satellite positions on the PAndAS footprint are indicated (i.e., with perspective conserved) along with the z-vector giving distance from the M31-centered tangent plane. The central ellipse indicates the approximate area of the survey where satellite detection is hindered by the M31 disk; note that the perpendicular bars on relevant axes indicate 100 kpc intervals.

(A color version of this figure is available in the online journal.)

the more poorly populated satellites, as noted in Section 4.2. In fact, if the 15 most Gaussian-like distributions are taken alone, namely Andromedas I, II, III, V, X, XVI, XVII, XVIII, XX, XXI, XXIII, XXIV, NGC 147, NGC 185, and M33, the results are in much closer agreement, with $\alpha = 1.87^{+0.46}_{-0.42}$ with sampling and $\alpha = 2.02^{+0.43}_{-0.41}$ without.

Given the obvious asymmetry in the satellite distribution in Figure 10, it is interesting to consider the effects of isolating various other satellites from the calculations. The stark asymmetry between the number of satellites on the near side as opposed to the far side of the M31 tangent plane for instance (as had been initially reported by McConnachie & Irwin 2006) is echoed in the respective density profiles, with an α of $2.37^{+0.42}_{-0.37}$ (*no sampling*) recorded when only the near-side satellites are considered, and that of $0.93^{+0.56}_{-0.49}$ (*no sampling*) when instead the far-side galaxies alone are included. When the individual satellite RPDs are sampled, the corresponding values are $1.87^{+0.43}_{-0.40}$ and $0.78^{+0.61}_{-0.46}$, respectively. Despite the large uncertainties, the results clearly do not support symmetry of any kind about the tangent plane. It is important to note, however, that this asymmetry may not be physical, but rather an effect of incompleteness in the data at the fainter magnitudes of the satellites on the far side of M31. McConnachie & Irwin (2006) do however observe this asymmetry even when only the more luminous satellites are considered. In time, it is hoped that the nature of the data incompleteness will be better understood and

Table 2
M31 Satellite Parameters: Distance and Associated Parameters of M31 and its Companions

Source	Distance Modulus	$E(B-V)$	Distance (kpc)	M31 Distance (kpc)	Literature Distance Values (kpc)
M31	$24.46^{+0.05}_{-0.05}$	0.062	779^{+19}_{-18}	...	785^{+25}_{-25} TRGB; McConnachie et al. (2005) 784^{+17}_{-17} RC; Stanek & Garnavich (1998) 765^{+28}_{-28} Ceph; Riess et al. (2012)
And I	$24.31^{+0.05}_{-0.05}$	0.054	727^{+18}_{-17}	68^{+21}_{-17}	731^{+18}_{-17} TRGB; Conn et al. (2011) 735^{+23}_{-23} TRGB; McConnachie et al. (2004)
And II	$24.00^{+0.05}_{-0.05}$	0.062	630^{+15}_{-15}	195^{+20}_{-17}	634^{+15}_{-14} TRGB; Conn et al. (2011) 645^{+19}_{-19} TRGB; McConnachie et al. (2004)
And III	$24.30^{+0.05}_{-0.07}$	0.057	723^{+18}_{-24}	86^{+25}_{-15}	749^{+24}_{-24} TRGB; McConnachie et al. (2005)
And V	$24.35^{+0.06}_{-0.07}$	0.125	742^{+21}_{-22}	113^{+9}_{-6}	774^{+28}_{-28} TRGB; McConnachie et al. (2005)
And IX	$23.89^{+0.31}_{-0.08}$	0.076	600^{+91}_{-23}	182^{+38}_{-66}	765^{+24}_{-24} TRGB; McConnachie et al. (2005)
And X	$24.13^{+0.08}_{-0.13}$	0.126	670^{+24}_{-39}	130^{+60}_{-17}	667 − 738 TRGB; Zucker et al. (2007)
And XI	$24.41^{+0.08}_{-0.32}$	0.080	763^{+29}_{-106}	102^{+149}_{-1}	740 − 955 TRGB; Martin et al. (2006) 735^{+17}_{-17} RR Ly; Yang & Sarajedini (2012)
And XII	$24.84^{+0.09}_{-0.34}$	0.111	928^{+40}_{-136}	181^{+19}_{-87}	825^{+85}_{-159} TRGB; (MCMC without MF) 740 − 955 TRGB; Martin et al. (2006)
And XIII	$24.40^{+0.33}_{-0.49}$	0.082	760^{+126}_{-154}	115^{+207}_{-2}	890^{+360}_{-361} TRGB; (MCMC without MF) 740 − 955 TRGB; Martin et al. (2006) 839^{+20}_{-19} RR Ly; Yang & Sarajedini (2012)
And XIV	$24.50^{+0.06}_{-0.56}$	0.060	793^{+23}_{-179}	161^{+81}_{-3}	630 − 850 TRGB; Majewski et al. (2007)
And XV	$23.98^{+0.26}_{-0.12}$	0.046	626^{+79}_{-35}	174^{+46}_{-32}	630^{+60}_{-60} TRGB; Ibata et al. (2007) 770^{+70}_{-70} TRGB; Letarte et al. (2009)
And XVI	$23.39^{+0.19}_{-0.14}$	0.066	476^{+44}_{-29}	319^{+43}_{-27}	525^{+50}_{-50} TRGB; Ibata et al. (2007) 525^{+50}_{-50} TRGB; Letarte et al. (2009)
And XVII	$24.31^{+0.11}_{-0.08}$	0.075	727^{+39}_{-25}	67^{20}_{-24}	794^{+40}_{-40} TRGB; Irwin et al. (2008)
And XVIII	$25.42^{+0.07}_{-0.08}$	0.104	1214^{+40}_{-43}	457^{+39}_{-47}	1355^{+88}_{-88} TRGB; McConnachie et al. (2008)
And XIX	$24.57^{+0.08}_{-0.43}$	0.062	821^{+32}_{-148}	115^{+96}_{-9}	933^{+61}_{-61} TRGB; McConnachie et al. (2008)
And XX	$24.35^{+0.12}_{-0.16}$	0.058	741^{+42}_{-52}	128^{+28}_{-5}	802^{+297}_{-96} TRGB; McConnachie et al. (2008)
And XXI	$24.59^{+0.06}_{-0.07}$	0.093	827^{+23}_{-25}	135^{+8}_{-10}	859^{+51}_{-51} TRGB; Martin et al. (2009)
And XXII (Tri I)	$24.82^{+0.07}_{-0.36}$	0.075	920^{+32}_{-139}	275^{+8}_{-60}	794^{+239}_{-0} TRGB; Martin et al. (2009)
And XXIII	$24.37^{+0.09}_{-0.06}$	0.066	748^{+31}_{-21}	127^{+7}_{-4}	733^{+23}_{-22} TRGB; Conn et al. (2011) 767^{+44}_{-44} HB; Richardson et al. (2011)
And XXIV	$24.77^{+0.07}_{-0.10}$	0.083	898^{+28}_{-42}	169^{+29}_{-29}	600^{+33}_{-33} HB; Richardson et al. (2011)
And XXV	$24.33^{+0.07}_{-0.21}$	0.101	736^{+23}_{-69}	90^{+57}_{-10}	812^{+46}_{-46} HB; Richardson et al. (2011)
And XXVI	$24.39^{+0.55}_{-0.53}$	0.110	754^{+218}_{-164}	103^{+234}_{-3}	762^{+42}_{-42} HB; Richardson et al. (2011)
And XXVII	$25.49^{+0.07}_{-1.03}$	0.080	1255^{+42}_{-474}	482^{+0}_{-425}	827^{+47}_{-47} HB; Richardson et al. (2011)
And XXX[a] (Cass II)	$24.17^{+0.10}_{-0.26}$	0.166	681^{+32}_{-78}	145^{+95}_{-4}	565^{+25}_{-25} TRGB g-band; M. J. Irwin (2012, in preparation)
NGC 147	$24.26^{+0.06}_{-0.06}$	0.173	712^{+21}_{-19}	118^{+15}_{-15}	675^{+27}_{-27} TRGB; McConnachie et al. (2005)
NGC 185	$23.96^{+0.07}_{-0.06}$	0.182	620^{+19}_{-18}	181^{+25}_{-20}	616^{+26}_{-26} TRGB; McConnachie et al. (2005)
M33	$24.57^{+0.05}_{-0.05}$	0.042	820^{+20}_{-19}	210^{+6}_{-5}	809^{+24}_{-24} TRGB; McConnachie et al. (2005) 964^{+54}_{-54} DEB; Bonanos et al. (2006)

Notes. All distance measurements utilize the data from the Pan-Andromeda Archaeological Survey (McConnachie et al. 2009) and have been obtained using the method presented in this paper. A value of $M_i^{\rm TRGB} = -3.44 \pm 0.05$ is assumed for the absolute magnitude of the RGB tip in CFHT MegaCam i band, based on the value identified for the SDSS i band (Bellazzini 2008) and justified for use here by the color equations applicable to the new MegaCam i-band filter (Gwyn 2010). Values for the extinction in MegaCam i band have been adopted as $A_\lambda = 2.086 \times E(B-V)$ for the same reasons, with uncertainties taken as ±10%. The extinction values quoted are for the object centers, though the actual calculations apply individual corrections to each member star according to their coordinates. Note that the uncertainties in the M31 distance are based on the sampled distributions while the quoted value is that derived directly from the Earth distance as per Equation (7). The last column gives alternative distances from the literature. TRGB-derived distances are quoted wherever possible. Distance derivation methods: TRGB, tip of the red giant branch; Ceph, Cepheid period–luminosity relation; RR Ly, RR Lyrae period–luminosity relation; RC, red clump; HB, horizontal branch; DEB, detached eclipsing binary.

[a] Andromeda XXX is a new discovery, and will also be known as Cassiopeia II, being the second dwarf spheroidal satellite of M31 to be discovered in the constellation of Cassiopeia (M. J. Irwin 2012, in preparation).

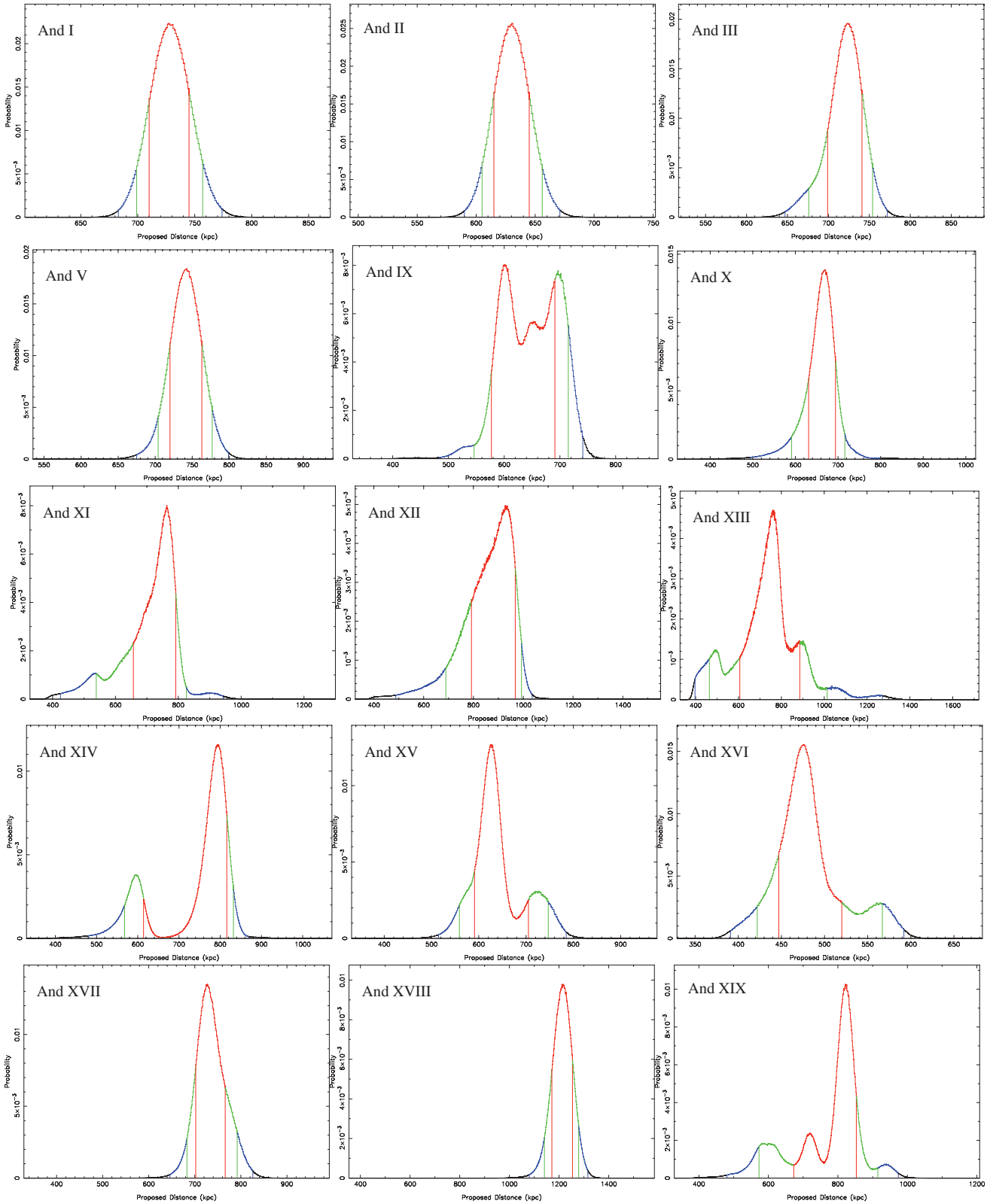


Figure 11. Distance posterior distributions for dwarf spheroidal satellites And I–III, And V and And IX–XIX. The distributions are color-coded with red, green, and blue denoting 1σ (68.2%), 90%, and 99% credibility intervals, respectively. The credibility intervals are measured from either side of the highest peak.
(A color version of this figure is available in the online journal.)

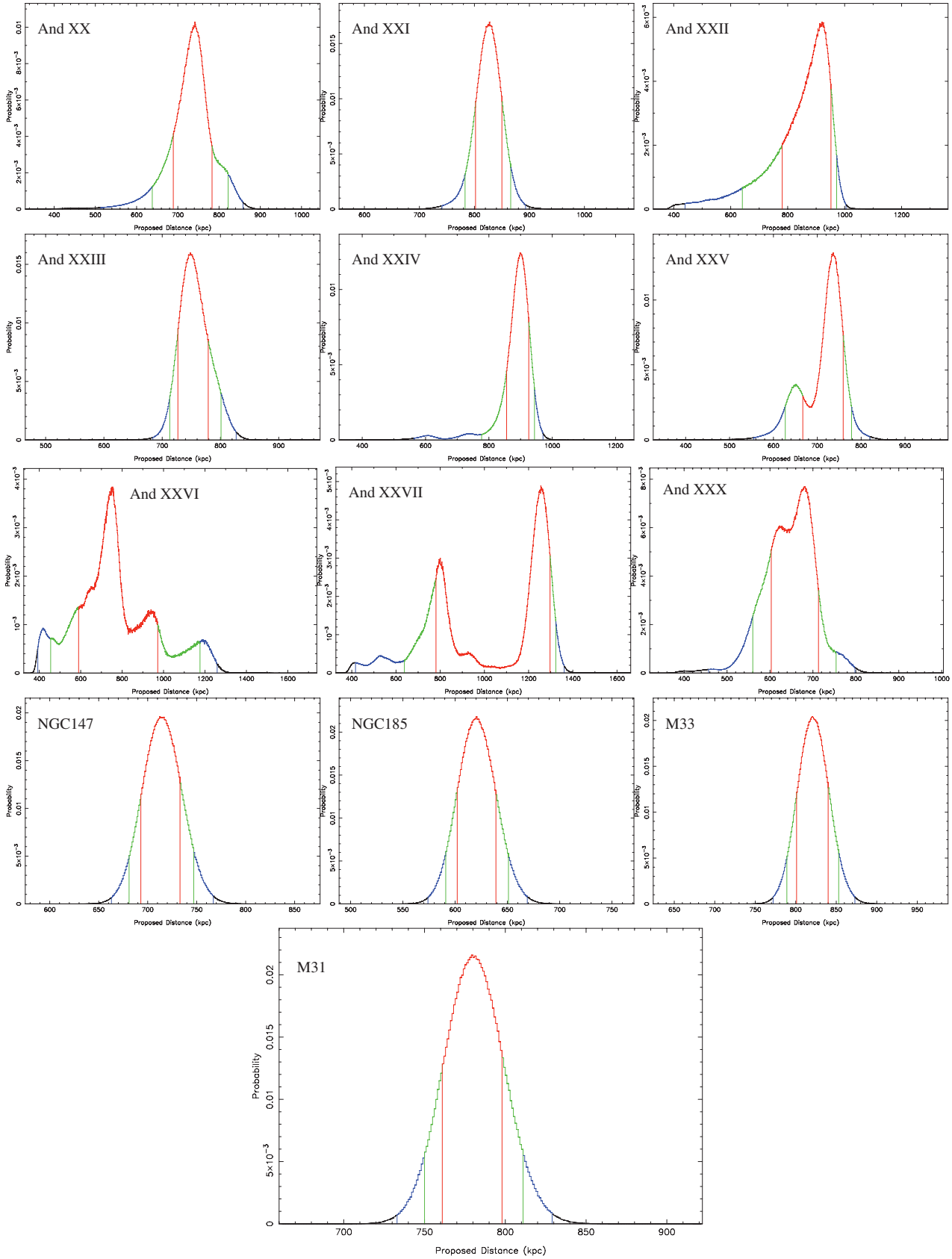


Figure 12. Distance posterior distributions for dwarf spheroidal satellites And XX–XXVII and And XXX, dwarf elliptical satellites NGC 147 and NGC 185, and major galaxies M31 and M33. The distributions are color-coded with red, green, and blue denoting 1σ (68.2%), 90%, and 99% credibility intervals, respectively.
(A color version of this figure is available in the online journal.)

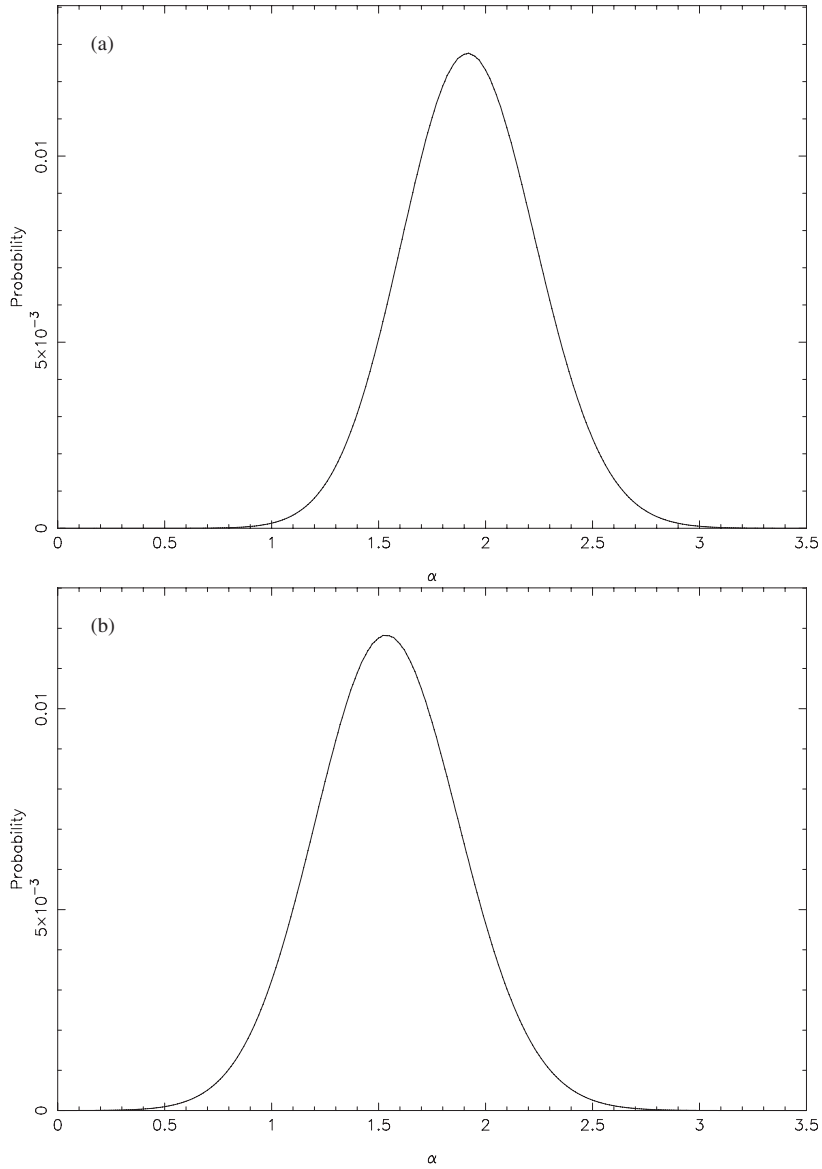


Figure 13. Probability distributions for the slope α of a single power law used to model the M31 halo satellite distribution, given the entire set of 27 M31 companions presented in Table 2. Panel (a) gives the distribution assuming a single best-fit radius for each of the satellites as determined from the mode in the satellite's distance posterior distribution (as given in Column 4 of Table 2). Panel (b) shows the same distribution when the entire radius probability distribution for each satellite is sampled 500,000 times.

effort *is* underway to determine the completeness functions for dwarf galaxy detection in the PAndAS survey (N. F. Martin et al. 2012b, in preparation). In the mean time, it would seem prudent to regard the contribution to the density profile of the far-side satellites with caution, instead taking the density profile measured from the near-side satellites alone as the best measurement.

On a final note with regard to near-side–far-side asymmetry, it is important to realize that the uncertainty in the distance to M31 has a large effect on how many satellites will lie on either side of the M31 tangent plane, and indeed on the density measurement as a whole. Where the individual PPDs are sampled, this is taken into account as the M31 PPD is sampled for each measurement. Nevertheless, it is interesting to consider the specific (non-sampled) case where M31 is measured at a closer distance, while all best-fit satellite distances remain unchanged. From the M31 PPD in Figure 12, it can be seen that there is a 5% chance that M31 lies at 750 kpc or closer. If M31 is taken to lie at 750 kpc, Andromedas XI, XIII, and XIV move onto the far side of the M31 tangent plane, going someway to even out the asymmetry. However, if the distances of all the satellites from M31 are re-measured for this new M31 position, the same stark contrast between the density profiles for the near and far sides remains and in fact grows. Using only those satellites on the near side of the new M31 tangent plane, an α of $2.87^{+0.50}_{-0.45}$ is determined whereas if only those satellites on the far side are considered, an α of $1.22^{+0.47}_{-0.47}$ is obtained. Hence it would seem unlikely that the observed near-side–far-side asymmetry is primarily a consequence of an overestimated M31 distance.

Recent research, such as that presented by Koch & Grebel (2006) and Metz et al. (2007) point toward highly significant planar alignments of various collections of satellites within the M31 halo, even though as a whole, no such distribution is prominent. Interestingly, the former investigation finds that it is

predominantly the objects morphologically similar to the dwarf spheroidals in their sample that can be constrained to a relatively thin disk, which also includes NGC 147 and M33. While our sample is considerably larger, it nevertheless consists nearly entirely of such objects, so it will be interesting to determine what degree of symmetry may be found within and on either side of the best-fit plane. We intend to investigate this in an upcoming publication, though it must still be noted that outliers from the planar trend have already been noted in this small sample, such as Andromeda II and NGC 185. Furthermore, other members are known not to conform to the norm of M31 satellite dynamics, with Andromeda XIV for instance apparently at the escape velocity for the M31 system for its determined distance (Majewski et al. 2007). Indeed, it would seem that whatever model is assumed, a few outliers are inevitable.

5. CONCLUSIONS

With the ready applicability of the TRGB standard candle to almost any of our galactic neighbors, there can be no question that its role will continue to be an important one. As the world's premier telescopes grow in size, so too will the radius of the "neighborhood" of galaxies to which the TRGB can be applied. Hence a technique which accurately characterizes the true probability space of the TRGB distances determined is a great asset. Indeed this quality comes to play an increasingly important role as more and more sparsely populated objects are found to frequent the environs of our larger nearby neighbors. The differences in the results achieved in the previous section with and without sampling of the actual distance distributions illustrate this fact.

Where in Paper I the foundations were laid for a TRGB method with such desirable qualities, its full value only becomes apparent when one actually employs its full Bayesian potential. It only requires a brief glance at Figures 2 and 3 to see how powerful a single data-specific prior can be. Similarly, the simple distance weighting prior outlined in Section 3.3 can make a poorly constrained model quite workable, as illustrated in the case of Andromeda XIII. Both tools will likely prove very useful when the method is used further afield.

It should also be remembered that the TRGB standard candle is in many ways, just the "first assault." When photometric data of sufficient depth are obtained, the horizontal branch can often pin down the distance with still greater accuracy. With a simple adjustment to the model LF, the techniques outlined in this paper and its predecessor commute quite readily to implementation on the horizontal branch.

Last, it must also be said that the distances presented herein provide an excellent opportunity to provide a new, updated analysis of the asymmetry and density of the M31 halo satellite distribution, one only touched on here. With such comprehensive and consistent coverage, there is great potential in these distances to further constrain the possible evolution and dynamical history of the M31 halo system.

A.R.C. would like to thank Macquarie University for their financial support through the Macquarie University Research Excellence Scholarship (MQRES) and both the University of Sydney and Université de Strasbourg for the use of computational and other facilities. R.A.I. and D.V.G. gratefully acknowledge support from the Agence Nationale de la Recherche though the grant POMMME (ANR 09-BLAN-0228). G.F.L. thanks the Australian Research Council for support through his Future Fellowship (FT100100268) and Discovery Project (DP110100678). M.A.F. acknowledges support from NSF grant AST-1009652. The analysis contained in this publication uses photometric data obtained with MegaPrime/MegaCam, a joint project of CFHT and CEA/DAPNIA, at the Canada-France-Hawaii Telescope (CFHT) which is operated by the National Research Council (NRC) of Canada, the Institute National des Sciences de l'Univers of the Centre National de la Recherche Scientifique of France, and the University of Hawaii. Our thanks go to the entire staff at CFHT for their great efforts and continuing support throughout the PAndAS project.

REFERENCES

Bellazzini, M. 2008, Mem. Soc. Astron. Ital., 79, 440
Bonanos, A. Z., Stanek, K. Z., Kudritzki, R. P., et al. 2006, ApJ, 652, 313
Brunthaler, A., Reid, M. J., Falcke, H., Greenhill, L. J., & Henkel, C. 2005, Science, 307, 1440
Conn, A. R., Lewis, G. F., Ibata, R. A., et al. 2011, ApJ, 740, 69
Ferguson, A. M. N., Gallagher, J. S., & Wyse, R. F. G. 2000, AJ, 120, 821
Geha, M., van der Marel, R. P., Guhathakurta, P., et al. 2010, ApJ, 711, 361
Gwyn, S. 2010, http://cadcwww.dao.nrc.ca/megapipe/docs/filters.html
Hodge, P. W. 1963, AJ, 68, 691
Ibata, R., Martin, N. F., Irwin, M., et al. 2007, ApJ, 671, 1591
Iben, I., Jr., & Renzini, A. 1983, ARA&A, 21, 271
Irwin, M. J., Ferguson, A. M. N., Huxor, A. P., et al. 2008, ApJ, 676, L17
Koch, A., & Grebel, E. K. 2006, AJ, 131, 1405
Lee, M. G., Freedman, W. L., & Madore, B. F. 1993, ApJ, 417, 553
Letarte, B., Chapman, S. C., Collins, M., et al. 2009, MNRAS, 400, 1472
Majewski, S. R., Beaton, R. L., Patterson, R. J., et al. 2007, ApJ, 670, L9
Martin, N. F., Ibata, R. A., Irwin, M. J., et al. 2006, MNRAS, 371, 1983
Martin, N. F., McConnachie, A. W., Irwin, M., et al. 2009, ApJ, 705, 758
McConnachie, A. W. 2005, PhD thesis, Univ. Cambridge
McConnachie, A. W., Huxor, A., Martin, N. F., et al. 2008, ApJ, 688, 1009
McConnachie, A. W., & Irwin, M. J. 2006, MNRAS, 365, 902
McConnachie, A. W., Irwin, M. J., Ibata, R. A., et al. 2009, Nature, 461, 66
McConnachie, A. W., Irwin, M. J., Ferguson, A. M. N., et al. 2004, MNRAS, 350, 243
McConnachie, A. W., Irwin, M. J., Ferguson, A. M. N., et al. 2005, MNRAS, 356, 979
Metz, M., Kroupa, P., & Jerjen, H. 2007, MNRAS, 374, 1125
Richardson, J. C., Irwin, M. J., McConnachie, A. W., et al. 2011, ApJ, 732, 76
Riess, A. G., Fliri, J., & Valls-Gabaud, D. 2012, ApJ, 745, 156
Rockosi, C. M., Odenkirchen, M., Grebel, E. K., et al. 2002, AJ, 124, 349
Salaris, M., & Cassisi, S. 1997, MNRAS, 289, 406
Schlegel, D. J., Finkbeiner, D. P., & Davis, M. 1998, ApJ, 500, 525
Skrutskie, M. F., Cutri, R. M., Stiening, R., et al. 2006, AJ, 131, 1163
Springel, V., Wang, J., Vogelsberger, M., et al. 2008, MNRAS, 391, 1685
Stanek, K. Z., & Garnavich, P. M. 1998, ApJ, 503, L131
Tammann, G. A., Sandage, A., & Reindl, B. 2008, A&AR, 15, 289
Yang, S.-C., & Sarajedini, A. 2012, MNRAS, 419, 1362
Zucker, D. B., Kniazev, A. Y., Martínez-Delgado, D., et al. 2007, ApJ, 659, L21

“Distance lends enchantment to the view.”

Mark Twain (1835-1910)

5

Paper III: The Three Dimensional Structure of the M31 Satellite System; Strong Evidence for an Inhomogeneous Distribution of Satellites

Paper III Preface

This paper in many respects represents the climax of the thesis and the culmination of the work presented in the three previous chapters. Specifically, it takes the satellite distance distributions presented in the previous chapter (i.e. Paper II), converts them into three-dimensional positions and then proceeds with a thorough analysis of the resulting distribution, leading to some exciting revelations as to the structure of the satellite system. Central to the work presented in Paper III are a variety of tools utilized throughout the analysis. These tools are discussed in the method section of the paper, but they are elaborated upon here in an effort to help the reader visualize the processes described and thus provide a suitable preface to the forthcoming material.

The first consideration in our analysis of the M31 satellite system must be to devise a suitable means for viewing the distribution in a clear and consistent way. Since we are concerned with a system completely external to our own Milky Way, it is intuitive to depart from our Earth-bound view and instead view the system as it would appear from the center of M31. To do this, we shift to M31-centric galactic coordinates. The convention in this regard has been to orient the north galactic pole ($b = +90°$) in the *reverse* direction to that of the net angular momentum of the disk (i.e. perpendicular to the disk of the galaxy) with the meridian of longitude $l = 0°$ aligned so as to pass through the Milky Way (or specifically Earth). Hence, we take the x, y and z coordinates derived as per Fig. 4.1 and perform a rotation of coordinates (quantified in the method section of the paper - see 'PlaneSigRMS.f95' in Appendix D for implementation) so as to bring our satellite positions into the new coordinate system. For plotting purposes, we can then simply convert from cartesian to spherical coordinates and plot each object's latitude and longitude in an Aitoff-Hammer projection (see aitoff_hammer.f95 in Appendix D), where positive x points in the direction of $l = 0°$, $b = 0°$; positive y toward $l = -90°$, $b = 0°$; and positive z toward $l = 0°$, $b = +90°$.

With a suitable method for visualizing the satellite distribution devised, the next consideration in our analysis concerns plane fitting. The whole of Paper III is essentially built around plane fitting, whether it be to identify planes of satellites with in the distribution, or the asymmetry of the distributions as a whole. Where we are interested in identifying

physically significant planes or disks of satellites, we need to seek out the plane that most closely approximates the constituent satellites as determined by any one of many possible 'goodness-of-fit' statistics. When instead we wish to find the magnitude and direction of the asymmetry of the distribution, we need simply find that plane which divides the sample most unequally. What ever the application, for ease of implementation and versatility, a simple scanning routine was chosen to accomplish the task.

The algorithm devised for the plane fitting is best understood by visualizing a plane pivoted at the center of our coordinate system (i.e. the center of M31) with its normal vector projecting out from this point. This plane is then rotated such that its normal vector or 'pole' scans a complete hemisphere of the sky. In so doing, every possible orientation of the plane is passed through exactly once. In practice, discrete pointings of the normal vector are used and at each one the goodness-of-fit statistic or the asymmetry is measured and compared with the best-fit value encountered so far. It is then either stored or discarded accordingly. Thus by the completion of the scan, the true best-fit pole has been identified and retained. In order to scan the hemisphere at a suitably high resolution whilst retaining computational efficiency, a low resolution scan is made first and then a localized high resolution search initiated about the best-fit pole. This process is illustrated in Fig. 5.1, where the poles tested in a single instance of plane fitting have been plotted using the *TOPCAT* graphics program (Taylor, 2005). Once the pole to the best-fit plane has been identified via this method, its orientation can then be converted into M31-centric latitude and longitude and plotted on an aitoff-hammer projection. This may be done once, as in the case for a particular set of satellites, or many times, in order to produce a pole distribution plot representing all possible combinations of a particular number of satellites for instance.

We are now equipped to identify the best-fit plane for a particular set of satellites, but this is only part of our analysis. We now need to ascertain the significance of the identified plane; is it likely to be a chance alignment or is it a real physical structure? To answer this question, we need a method by which the identical measurement can be performed repeatedly on a large number of 'realizations' of randomly distributed artificial satellites, each one subject to the same constraints as the real data. Our first requirement in producing

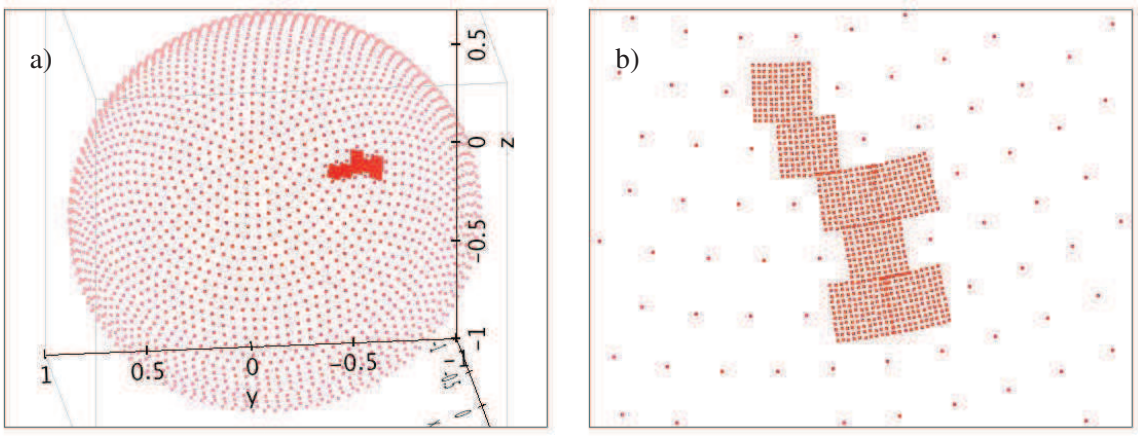


FIGURE 5.1: Plane Fitting; Poles of Tested Planes. This figure illustrates the plane fitting method utilized for most of the analysis contained in Paper III. Figure (a) shows a hemisphere of equally spaced points, each one the pole of a tested plane. This plot represents the low resolution scan undertaken for every instance of plane fitting to a particular set of satellites. Once the scan is complete, a high resolution scan is undertaken about the best-fit pole. Figure (b) is a close up of several of these high resolution scans, indicating the effective resolution of each scan. Note that several adjacent scans are shown to indicate the overlapping coverage. For a given instance of plane fitting, only one high resolution search (i.e. square) need be made. Plotted using *TOPCAT*.

such 'random realizations' is a tool by which a given satellite position vector can be spun around to any random position on the M31 sky. Development of such a tool is not a trivial task, as lines of constant latitude on a sphere are not great circles, but decrease in diameter toward the poles. Thus simply drawing a latitude and longitude at random will produce a disproportionate number of satellites at high latitude. We therefore weight the probability of drawing a particular latitude in proportion to the cosine of the latitude. This is illustrated in Fig. 5.2 (a). Fig. (b) shows 10,000 unit vectors distributed truly randomly following this procedure. Note that this same process is not only applicable to positioning satellites randomly on the sky but is also another means of generating random poles for plane fitting (see §3.4 of Paper III for instance).

With the means to spin satellite position vectors to random orientations now in place, we can proceed to build our random satellite realizations. The general procedure then is as follows. First, the desired number of satellites to be included in the random realization is chosen. This is always the total number of satellites in the real distribution i.e. usually 27, but 25 where NGC147, NGC185 and Andromeda XXX are grouped together as a single

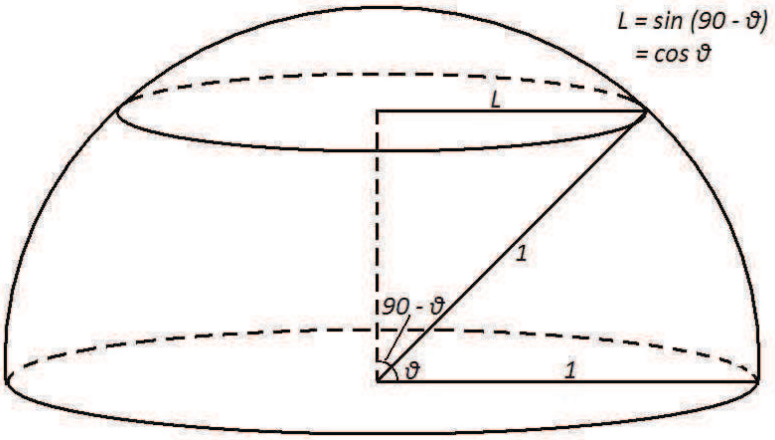
point (see §3.3 of Paper III). Then to generate a position for each artificial satellite, one of the real satellites is chosen at random and a distance is drawn from its associated distance distribution. The M31-centric position vector is then calculated and spun around to a new random orientation as described above. As we wish to subject our random sample to the same constraints as the real sample, it is very important at this point that we verify that the new orientation does not place the object outside of the utilized region of the PAndAS survey area. We therefore project the new satellite position back onto the sky and determine whether it meets this criterion. If it does, we proceed to generate a position for the next satellite following the same procedure, if it does not, we reject the new position and likewise repeat the process, until we get an acceptable position for the current satellite. The process repeats until the desired number of satellites are produced.

By this point, we have generated our random realization with *one* possible position for each satellite. If we are to mimic the data most closely, we should have in effect a full line-of-sight distance distribution for each artificial satellite. Hence once we have determined a single set of acceptable three-dimensional positions for the satellites, the positions of each on the sky as viewed from Earth are stored, as are the new Earth-to-satellite distances. The original satellite distance distributions used for each one can then be sampled and appropriately mapped to the new positions. Fig. 5.3 below illustrates the procedure for the random generation of satellites. Figure (a) illustrates the positions on the sky of 1000 accepted satellite positions as viewed from Earth. As with most of the figures in this preface, this figure was generated to verify the correct behavior of the algorithm. Hence a very large number of satellites were generated in order to insure that all accepted satellites did indeed fall inside the utilized portion of the PAndAS survey area. Figure (b) likewise was generated to insure that the final random realizations had the correct appearance. It shows 1000 possible positions drawn for each of 27 artificial satellites.

With all of the above tools in place, we are now in a position to embark on our analysis of the three dimensional structure of the M31 satellite system. As shall become apparent in the paper, the analysis reveals some very interesting results. There can, for instance, be very little doubt that the distribution is significantly inhomogeneous, with degrees of planarity and

asymmetry observed which are shown to be very unlikely to arise by chance. Most striking is a very thin plane or disk made up of 15 satellites from the total sample of 27. The orientation of this 'great plane' is also of particular note.

a)



b)

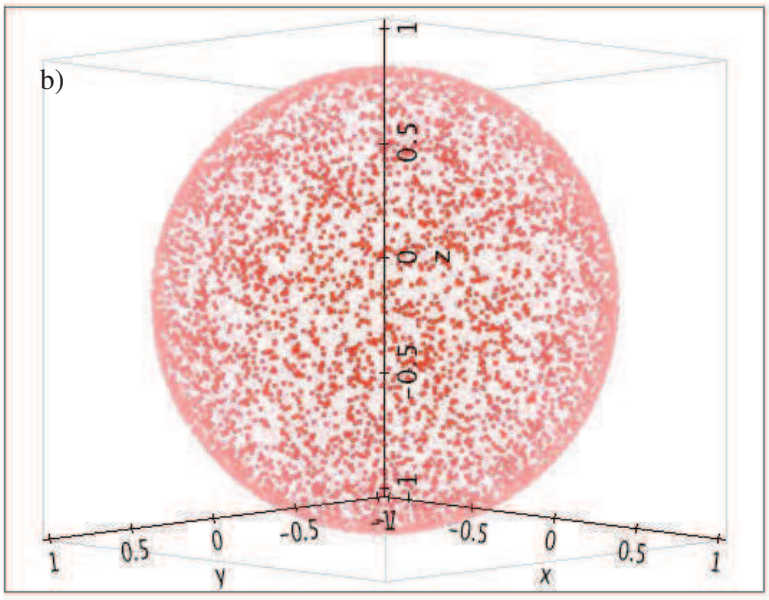


FIGURE 5.2: Method for rotating vectors to random angles. Fig. (a) illustrates the calculation of the necessary weighting factor as a function of latitude. Fig. (b) shows 10,000 unit vectors spun to random angles after incorporating this weighting factor. Note that had this weighting factor not been included, the density of poles would be greater at higher latitudes. Fig. (b) plotted using *TOPCAT*.

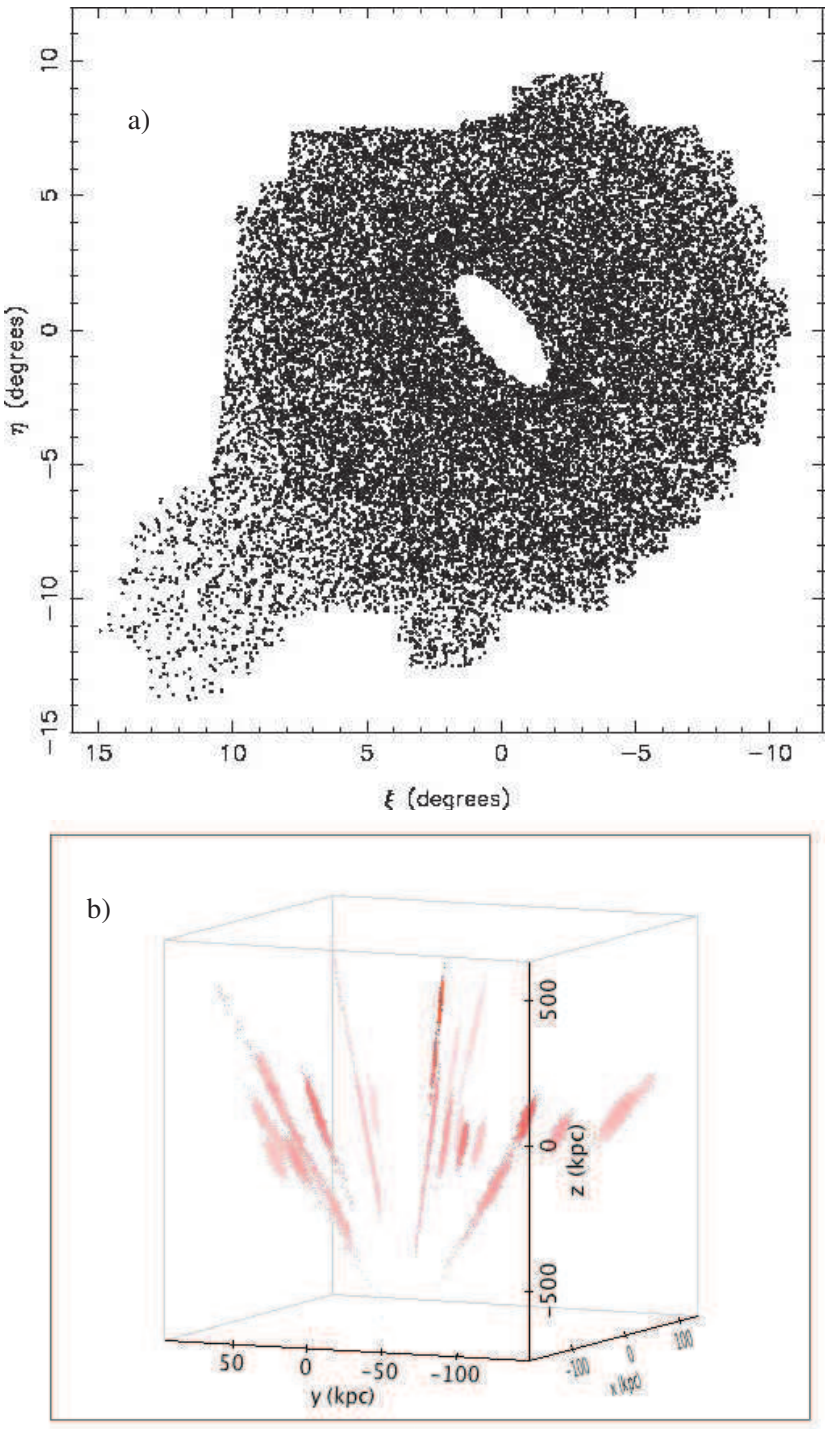


FIGURE 5.3: Generating random satellite realizations. Fig (a) illustrates the acceptable on-sky positions in which artificial satellites are allowed to appear, via the creation of a large 1000-satellite random realization. Note that satellites are less likely to be positioned at larger distances from M31 due to the small number of satellites in the real sample lying at equivalent distances. Figure (b) plots a single random realization of 27 satellites complete with sampled distance distributions for each. This is the form of the random realizations used in the actual analysis. Fig. (b) plotted using *TOPCAT*.

SUBMITTED TO *Astrophysical Journal*
Preprint typeset using LaTeX style emulateapj v. 08/13/06

THE THREE-DIMENSIONAL STRUCTURE OF THE M31 SATELLITE SYSTEM; STRONG EVIDENCE FOR AN INHOMOGENEOUS DISTRIBUTION OF SATELLITES

A. R. Conn[1,2,3], G. F. Lewis[4], R. A. Ibata[3], Q. A. Parker[1,2,5], D. B. Zucker[1,2,5], A. W. McConnachie[6], N. F. Martin[3], D. Valls-Gabaud[7], N. Tanvir[8], M. J. Irwin[9], A. M. N. Ferguson[10], and S. C. Chapman[9]

SUBMITTED TO *Astrophysical Journal*

ABSTRACT

We undertake an investigation into the spatial structure of the M31 satellite system utilizing the distance distributions presented in a previous publication. These distances make use of the unique combination of depth and spatial coverage of the Pan-Andromeda Archaeological Survey (PAndAS) to provide a large, homogeneous sample consisting of 27 of M31's satellites, as well as M31 itself. We find that the satellite distribution, when viewed as a whole, is no more planar than one would expect from a random distribution of equal size. A disk consisting of a large subset of 15 of the satellites is however found to be highly significant, and surprisingly thin, with a root-mean-square thickness of just $12.34^{+0.75}_{-0.43}$ kpc. This disk is oriented approximately edge on with respect to the Milky Way and almost perpendicular to the Milky Way disk. It is also roughly orthogonal to the disk like structure regularly reported for the Milky Way satellite system and in close alignment with M31's Giant Stellar Stream. A similar analysis of the asymmetry of the M31 satellite distribution finds that it is also significantly larger than one would expect from a random distribution. In particular, it is remarkable that 20 of the 27 satellites most likely lie on the Milky Way side of the galaxy. This lopsidedness is all the more intriguing in light of the apparent orthogonality observed between the satellite systems of the Milky Way and M31.

Subject headings: galaxies: distribution — galaxies: dwarf — galaxies: individual (M31) — galaxies: satellites

1. INTRODUCTION

The possibility that irregular distributions of satellite galaxies may be a common feature of large galaxy halos was originally bolstered by several studies of the anisotropic distribution of our own galaxy's satellites. Lynden-Bell (1976) found that the Magellanic Stream along with Sculptor and the Draco-Ursa Minor Stream and their associated dwarf spheroidal galaxies all appear to lie in the orbital plane of the Magellanic Clouds. In Lynden-Bell (1982), all the then known dwarf spheroidal companions of the Milky Way are identified as lying in one of two streams. Kroupa, Theis, & Boily (2005) examined the likelihood of producing the observed disk-like distribution of Milky Way satellites from a spherical or oblate dark matter halo. From comparisons with theoretical isotropic satellite distributions produced from such a halo, they find that the chance of producing the observed distribution from the dark-matter sub-halos of cold-dark-matter (CDM) cosmology is less than 0.5 %. They examine various combinations of the inner most satellites and find a best-fit plane that is almost perpendicular to the plane of the Milky Way with a root-mean-square height ranging from only about 10 to 30 kpc. Zentner et al. (2005), whilst finding a similar plane to Kroupa, Theis, & Boily (2005) for the satellites of M31, disagree with their assumption that such a plane is unlikely to arise from a conventional CDM dark matter halo. They argue that that the most luminous satellites cannot be taken for granted as forming randomly from the isotropic sub-halo distribution but instead, lie preferentially at smaller distances from the halo centre and co-planar with the major axis of the host halo. Coupled with the finding that galaxies preferentially align themselves with their major-axis highly-inclined or even perpendicular to that of the surrounding matter (e.g. Navarro, Abadi, & Steinmetz 2004; Hartwick 2000), this then provides a good explanation for the observed orientation of the best fit plane.

More recently, Lovell et al. (2011), using the six halo models in the Aquarius Simulations (Springel et al. 2008), find that all six halos produce a significant population of sub-halos with quasi-planar orbits aligned with the main halo spin. This, they argue, is a natural explanation for the observed satellite distribution of the Milky Way. Pawlowski et al. (2012) argue against this however. With the calculation of the angular momenta of 8 Milky Way Satellites (Metz, Kroupa, & Libeskind 2008) revealing a strong alignment between 6 of the orbital poles, Pawlowski et al. (2012) examine the likelihood of randomly drawing 6 sub-halos from each of the 6 Aquarius simulations (among other halo simulations), and finding a similar degree of alignment. More precisely, they draw 10^5 sets of 8 satellites from each of the 6 simulations, and select the 6 with the highest degree of alignment between their orbits, thus emulating the findings of Metz, Kroupa, & Libeskind (2008). They then look at the degree of clumping of the orbital poles Δ_{sph} as well as the angular distance of the average of the orbital pole inclinations from the model equator d and find that the actual degree of planarity observed for the six satellites identified by Metz, Kroupa, & Libeskind (2008)

[1] Department of Physics & Astronomy, Macquarie University, NSW 2109, Australia.
[2] Research Centre in Astronomy, Astrophysics and Astrophotonics (MQAAAstro), Macquarie University, NSW 2109, Australia.
[3] Observatoire Astronomique, Universite de Strasbourg, CNRS, 67000 Strasbourg, France.
[4] Sydney Institute for Astronomy, School of Physics, A28, University of Sydney, Sydney NSW 2006, Australia.
[5] Australian Astronomical Observatory, PO Box 296, Epping, NSW 2121, Australia.
[6] NRC Herzberg Institute of Astrophysics, 5071 West Saanich Road, Victoria, British Columbia, Canada V9E 2E7.
[7] Observatoire de Paris, LERMA, 61 Avenue de l'Observatoire FR 75014 Paris France.
[8] Department of Physics and Astronomy, University of Leicester, Leicester LE1 7RH, UK.
[9] Institute of Astronomy, University of Cambridge, Madingley Road, Cambridge CB3 0HA, UK.
[10] Institute for Astronomy, University of Edinburgh, Royal Observatory, Blackford Hill, Edinburgh EH9 3HJ, UK.

($\Delta_{sph}^{MW} = 35.4°$ and $d_{MW} = 9.4°$) are equalled or exceeded in the random draws in less than 10% of cases when Δ_{sph} is considered and less than 15% of cases for d. Starkenburg et al. (2012) also find that the degree of planarity observed for the Milky Way satellites is uncommon in all six of the Aquarius halos (see Fig. 7 of that study).

In addition to the revelation that the Milky Way's satellites appear to inhabit highly-inclined great planes, they also appear to corroborate the finding of Holmberg (1969), namely that the companions of Spiral Galaxies preferentially congregate at high galactic latitudes (the Holmberg Effect), as observed in his study of 174 galaxy groups. It is not clear why this should be the case, or even if it truly is the case, although if the apparent adherence of satellite systems to polar great planes is typical of galaxies in general, then the Holmberg Effect seems to be an extension of this. Quinn & Goodman (1986) proposed that dynamical friction may be responsible for the observed polar great planes, with those orbits spending the most time in close proximity to the galactic disk, experiencing the fastest decay, while those that take the most direct route through the disk environs, namely the polar orbits, experiencing the slowest orbital decay.

Besides the conjecture that satellite great planes trace the major-axis of the dark-matter halo in which the parent galaxy resides, there are other proposed mechanisms for their creation. One hypothesis is that these planes trace the orbits of ancient galaxies that have been cannibalized by the host galaxy. Palma, Majewski, & Johnston (2002) have investigated this hypothesis by looking for planes among groups of satellite galaxies and globular clusters in the Milky Way's outer halo and find various members to be co-planar with either the Magellanic or Sagittarius streams. The findings of Lynden-Bell & Lynden-Bell (1995) are also consistent with such a hypothesis. Indeed, it is this hypothesis which is most strongly supported by Pawlowski et al. (2012), wherein the Δ_{sph} and d of satellites drawn from various tidal models equal or exceed Δ_{sph}^{MW} and d_{MW} in over 80% of draws in some cases. A similar hypothesis, which in some regards links the galaxy-cannibalization and dark-matter hypotheses, proposes that the observed planes result from the orientation of the large-scale filamentary structure of galaxy clusters (e.g. Knebe et al. 2004), an orientation traced out by those minor galaxies which fall into the halo of a major galaxy. Metz et al. (2009) argue however that extra-galactic associations of dwarf galaxies are too extended to account for the high degree of planarity observed for the Milky Way satellites.

The great obstacle to a conclusive resolution of these issues is the lack of systems for which reliable spatial (and kinematic) data exists. While some such data does exist for large galaxy clusters such as Virgo and Coma, accurate 3D distributions of galaxies within their halo have for a long time been known only for our own galaxy's halo, ascertainable due to our central position within it. It has only been in recent times that a second system has opened up to us - that of our counterpart in the Local Group, M31. Whilst various databases of photometry and other data have been available for M31 and some of its brighter companions for over a decade, it is the Pan-Andromeda Archaeological Survey (PAndAS - McConnachie et al. 2009) - a deep photometric, 2-colour survey providing a uniform coverage of the M31 halo out to approximately 150 kpc - that has provided a new level of detail for this system. It is from this survey that we obtained our distances to M31 and 27 of its companions, following the method developed in Conn et al. (2011) (henceforth CLI11) and further adapted for this purpose in Conn et al. (2012) (henceforth CIL12). The distances themselves and their associated uncertainty distributions are presented in CIL12 and it is these distributions that are utilized for all analysis contained in this paper.

With regard to previous studies of the anisotropy in the M31 satellite distribution, two investigations warrant consideration at this point. McConnachie & Irwin (2006), making use of Wide Field Camera (WFC) photometry from the Isaac Newton Telescope (INT) in what was essentially the forerunner to the PAndAS Survey, focus on "Ghostly Streams" of satellite galaxies following a similar approach as Lynden-Bell & Lynden-Bell (1995) used for the Milky Way. In addition, they characterize the large degree of asymmetry in the satellite distribution, a feature also noted in CIL12, and examine the radial distribution of the satellites, noting a (statistically insignificant) larger average distance from M31 than that observed between the Milky Way and its satellites. They find a large number of candidate satellite streams, with some favoring the dwarf spheroidal members. Koch & Grebel (2006) utilize distance measurements from a variety of sources and focus particularly on planes of satellites and, whilst they do not find a particularly significant best fit plane when their whole satellite sample is considered, it is rather interesting that they find a 99.7 % statistical significance to their best fit plane when the then-known dwarf spheroidal galaxies dominate their sample. Furthermore, this plane is near-polar - as has been observed for the Milky Way, although they find little support for the Holmberg Effect. Koch & Grebel (2006) utilize a particularly robust method in their search for high-significance planar fits to subsets of galaxies by considering every possible combination of a given number of satellites from their sample.

In the current study we employ a similar approach, but with the great advantage of having a considerably extended sample of galaxies in our sample, with all distances derived by the same method and from the same data as described in CLI11 and CIL12. As a result, we are able to give full consideration to the effects of selection bias on the observed satellite distribution. This then presents an excellent opportunity to greatly improve our knowledge of the three-dimensional structure of the M31 satellite distribution, with important implications regarding the recent evolution of the system.

A breakdown of the structure of the paper is as follows. In Section §2, we outline our method for plane fitting (§2.1) and locating significant planes of satellites as well as the orientation, magnitude and significance of the asymmetry of the distribution. A method for generating random realizations of satellites subject to the same selection biases as the real data is also discussed in this section (§2.2) as is the selection bias itself (§2.3). §3 then presents the results of applying these methods, first to the sample as a whole, and then to subsets of galaxies. Specifically, §3.1 presents a study of planarity within the satellite system when all satellites contribute to the determination of the best fit plane; §3.2 examines the asymmetry in a similar way; §3.3 examines the orientations of planes of smaller subsets of satellites within the distribution; and §3.4 concludes this section with a determination of the significance of a 'Great Plane' of satellites emerging from the preceding sections. Sections 4 and 5 then follow with discussion and conclusions.

Note that this paper was written in conjunction with a shorter contribution (Ibata et al. 2012; hereafter ILC12) which

announced some of the key discoveries resulting from the analysis we present here. In particular, the process of identifying the member satellites of the 'great plane' discussed in ILC12 is described here in more detail. In this analysis however, we concern ourselves with the *spatial* structure of the satellite system only and so the reader should refer to ILC12 for the interesting insight provided by the addition of the velocity information.

2. METHOD

2.1. *Plane Fitting*

In order to find planes of satellites within the M31 satellite system, our first concern is to convert the satellite distances as presented in CIL12 into three-dimensional positions. To do this, we begin with an M31-centered, cartesian coordinate system oriented such that the x and y axes lie in the M31 tangent plane with the z-axis pointed toward the Earth. Specifically, the x-axis corresponds to $\eta_{tp} = 0$ which is the projection of M31's Declination onto the tangent plane. The y-axis then corresponds to $\xi_{tp} = 0$ - the projection of M31's Right Ascension onto the tangent plane. The z-axis then points along the Earth-to-M31 vector, with magnitude increasing with distance from Earth. This orientation can be seen in Fig. 10(c) of CIL12. Thus:

$$\begin{aligned} x &= D_{sat}\cos(\theta)\tan(\xi) \\ y &= D_{sat}\sin(\eta) \\ z &= D_{sat}\cos(\theta) - \mathrm{D_{M31}} \end{aligned} \tag{1}$$

where D_{M31} and D_{sat} are the distances from Earth to M31 and from Earth to the satellite respectively, θ is the angular separation on the sky between M31 and the satellite, and η and ξ are the real-angle equivalents of the tangent plane projection angles η_{tp} and ξ_{tp} respectively.

Next, we rotate this reference frame to the conventional M31 reference frame such that the positive z-axis points toward M31's north galactic pole[1] (i.e. $b_{M31} = +90°$) and the $l_{M31} = 0°$ meridian passes through the Earth. So as to be consistent with the earlier work of McConnachie & Irwin (2006), we have adopted the same values for M31's position angle (39.8°) and inclination (77.5° - de Vaucouleurs 1958). Each object is hence rotated by 39.8° about the z-axis to counter the effect of its position angle, and then 77.5° about the x-axis to account for M31's inclination. A final rotation of 90° about the z-axis is then necessary to bring $l_{M31} = 0°$ into alignment with the direction of Earth (which hence lies at $l_{M31} = 0°, b_{M31} = -12.5°$). The resulting spherical coordinates for each object in the sample are plotted onto an Aitoff-Hammer projection in Fig. 1. This same figure also shows the uncertainties in position associated with each object, generated via sampling of the respective distance posterior probability distributions (PPDs) of each object and subsequent conversion of each drawn distance into a three-dimensional position.

With the satellites' positions determined in cartesian coordinates, it is straight forward to determine the minimum distance of each satellite from a given plane as follows:

$$D_{plane} = |ax + by + cz + d| \tag{2}$$

where D_{plane} is the distance of a satellite at a point (x, y, z) from a plane whose normal vector is (a, b, c) and is of unit length. For simplicity, we invoke the reasonable requirement that all planes must pass through the center of M31 and so in our case, $d = 0$ and the plane normal vector points out from the center of M31. Hence, in order to find the best-fit or maximum significance plane to a set of satellites, we need simply minimize D_{plane} for the satellites to be fitted. This can be done via a variety of means, some of which are compared in the following section, but perhaps the most robust and the predominant method employed in this study, is that of minimizing the root-mean-square (RMS) of the distances to the fitted satellites.

In order to measure the asymmetry of the satellite distribution about a given plane, we need only count the number of satellites on one side of the plane. To do this, we can simply remove the absolute value signs from equation 2, so that the side of the plane on which a satellite lies can be determined by whether D_{plane} is positive or negative. The plane of maximum asymmetry is then taken to be that which divides the sample such that the difference in satellite counts for opposite sides of the plane is greatest.

Whether we wish to determine the best fit plane through a sample of satellites or the plane of maximum asymmetry, we require a system by which a large number of planes can be tested on the sample so that the goodness of fit (or asymmetry) can be calculated for each. To do this, we define each tested plane by its normal vector or pole (a, b, c) so that Eq. 2 can be applied directly. We then rotate this pole to different orientations around the sky in such a way as to 'scan' the whole sphere evenly and at a suitably high resolution. In practice, we need to be able to apply this routine many thousands of times for a large number of samples and so a fast computational time is of the essence. To this end, for a given sample, our algorithm determines the desired plane following a two step procedure.

Firstly, a low resolution scan of the sphere is made to determine the approximate direction on the sky of the pole to the best-fit plane. Only half the sphere actually needs to be scanned since poles lying on the opposite hemisphere correspond to the identical planes flipped upside down. The low resolution scan tests 2233 different poles across the hemisphere. A near-uniform coverage is achieved by decreasing the number of planes tested in proportion to the cosine of the latitude of the planes' pole. This prohibits what would otherwise be an increased coverage at the higher latitudes of the coordinate system. With the pole to the best-fit plane determined in low-resolution, a high resolution search is then made around the identified coordinates at 10 times the resolution. In this way a pole can effectively be found at any of approximately 250,000 evenly spread locations on the hemisphere.

2.2. *Generating Random Satellite Samples*

Whilst we are now equipped to identify best-fit planes to our sample and subsamples thereof, it is necessary to have some means of determining the significance of these planes in an absolute sense. The most intuitive way to do this is to perform the same analysis on a randomly generated sample of equal size. In particular, when we are concerned with all possible combinations of a particular number of satellites that can be produced from the whole sample, we are often dealing with a very large number of subsamples and so it is inevitable that some of these subsets of satellites will exhibit a very high degree of planarity. Identical analysis must therefore be performed on random distributions, to see if there are similar numbers of subsets with equal degrees of planarity.

[1] Defined so as to point north in Equatorial coordinates

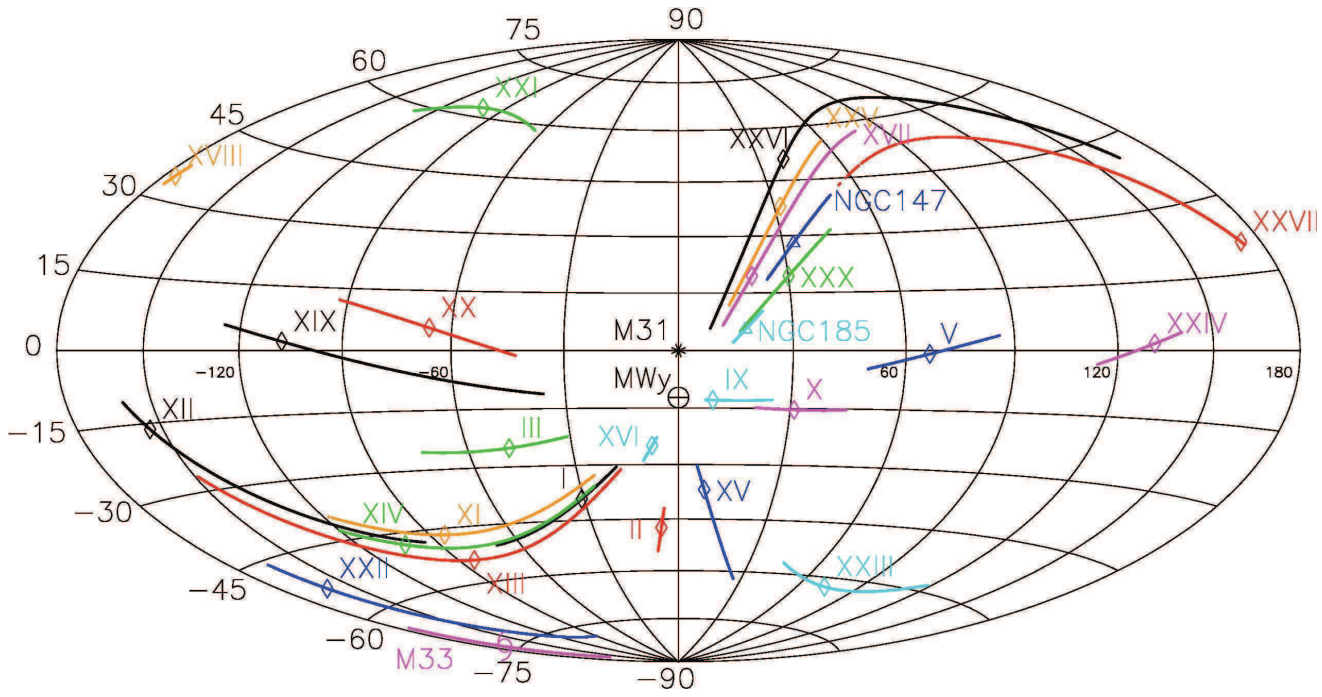


FIG. 1.— An Aitoff-Hammer Projection showing the positions of M31's satellites, along with their associated 1σ uncertainties. The positions show where each object would appear in the sky if viewed from the centre of M31, and are given in M31's inherent galactic latitude and longitude. The position of the Milky Way is also shown for reference. The position uncertainties trace single arcs across the M31 sky, rather than two-dimensional patches on account of the restriction of the uncertainty to lie solely along the line of sight to the object from Earth. For this same reason, all the lines point radially outward from the Milky Way. These uncertainties also take into account the uncertainty in M31's distance. The existence of a prominent plane, broadly consisting of Andromedas I, XI, XII, XIII, XIV, XVI, XVII, XXV, XXVI, XXVII, XXX and NGC147 and NGC185, is hinted at by the close proximity of their respective arcs.

For this reason, considerable care was taken to design an algorithm capable of providing a unique random realization of the desired number of satellites whenever it is called. The algorithm makes use of the distance PPD for each satellite, and also takes into account the irregular window function (i.e. useable portion) of the PAndAS survey. Each time a satellite is to be added to the random realization, one of the 27 actual satellites is chosen at random and a distance is drawn from its associated PPD. This distance (D_{sat}) is then converted into a three dimensional position (x, y, z) following equation set 1 and this satellite-to-M31 separation vector is then spun around to a new, random location in the M31 sky. Note that for each random realization, a new value of D_{M31} is similarly drawn from the M31 distance PPD.

Once again, care must be taken in this step to ensure that the whole sphere is given equal weight, otherwise there is a higher likelihood for the artificial satellites to be positioned at high latitude. Again, this is remedied by weighting the likelihood by the cosine of the latitude.

With the new, random location for the satellite chosen, it is then projected back onto the sky as it would appear from Earth and a check is made to ensure that it does indeed lie within the boundaries of the PAndAS survey area, and outside of the central ellipse (5° major axis, 2° minor axis - see Fig. 10 (c) of CIL12) where the disk of M31 inhibits reliable measurements. If the satellite does not meet these requirements, it is rejected and the satellite drawing process is repeated until a suitable position is generated. By repeating this process until the desired number of satellites are produced, a new, random comparison sample is generated which gives full account to the constraints on the actual data.

In order for the random satellite realizations to mimic the actual data most closely, it is necessary that each artificial satellite is represented not by just one point, but rather a string of points reflecting the uncertainty in the Earth-to-Object distance. Hence once acceptable positions for each satellite are drawn as described above, the distance distributions for each object are sampled and projected to their equivalent positions along the line of sight about the initially placed point. For sections 3.1, 3.2 and 3.4 each artificial satellite's distance distribution is represented by 1000 points such that each plane-fitting measurement is made for 1000 possible positions of the object and then the average value of the measurements is taken. The only exception to this number is where the maximum-likelihood approach is used in §3.1. Due to the inclusion of a second fitting parameter in this case, only 100 samples are taken for each satellite. For §3.3, as we are not concerned with comparisons of plane significance between the real sample and the random realizations, it is sufficient to use a single drawn position for each artificial satellite.

2.3. *A note on Satellite Detection Bias*

By employing a similar method to that described above, it is also possible to explore the effect of the PAndAS survey area boundaries on the satellite detection bias as viewed from the center of M31. It is intuitive that more satellites are likely to be detected along the line of sight to Earth, since even satellites at a large distance from M31 will still appear within the survey boundaries if they lie along this line. We can visualize this effect by generating a large number of randomly distributed satellites and plotting them on the M31 sky after first rejecting those satellites that would appear outside the survey area 'mask' if viewed from Earth. To do this, one million satellites were drawn from a spherically symmetric halo potential with density falling off as a function of the square of the distance from the halo center. Satellites were hence drawn at

distances between 0 and 700 kpc from M31 with equal probability. The satellites were then projected onto the M31 tangent plane and those satellites lying outside the survey area or inside the M31 disk obstruction area were excised from the density map. The resulting anisotropy of the satellites on the M31 sky is presented in Fig. 2.

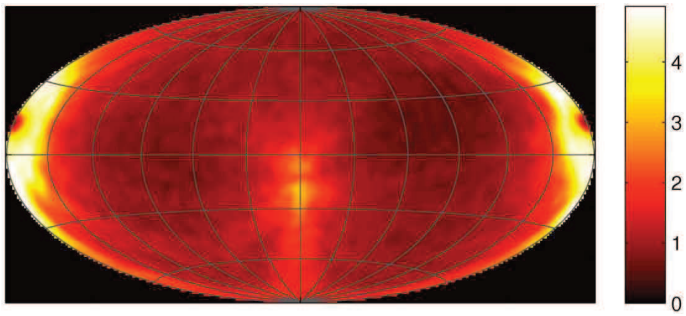


FIG. 2.— An Aitoff-Hammer projection illustrating the satellite detection bias resulting from the PAndAS survey boundaries and M31 disk obstruction. Note that this figure utilizes a Gaussian blurring of radius 5°, as do all of the subsequent pole-density plots.

As can be seen from the figure, the probability of detection is indeed higher along a great circle oriented edge-on with respect to the direction of Earth, and perpendicular to the M31 disk ($b_{M31} = 0°$). This great circle has its pole/ anti-pole at $l_{M31} = \pm 90°, b_{M31} = 0°$ and hence we would expect a predisposition toward finding planes of satellites with a pole in this vicinity. We would also expect, though to a lesser extent, to find an excess of satellite planes oriented edge-on with respect to Earth at any inclination. Such planes would have poles lying anywhere on the great circle whose normal is directed toward Earth. The drop in the satellite density at $l_{M31} = 0°, b_{M31} = -12.5°$ and $l_{M31} = \pm 180°, b_{M31} = 12.5°$ is a consequence of the hinderance to detection caused by the M31 disk. Due to the increased volume of space covered by the survey at greater distances from Earth, unhindered satellite detection is possible over a larger range of angles on the far side of M31 in comparison to the Earth-ward side.

3. RESULTS

3.1. *Best Fit Plane to the Entire Satellite Sample*

In order to find the best-fit plane to the satellite system as a whole, the procedure of §2.1 is applied to the whole sample of 27 satellites presented in CIL12. The RMS thickness of the sample is used here, as in subsequent sections, as the statistic of planarity; we find it to be a robust measure and it has the convenient property of being computationally inexpensive. Since we are dealing with only one sample in this case, two other measures are also used for comparison. The first calculates the sum of the absolute values of the distances of each of the satellites from the tested plane. The second is essentially a maximum likelihood approach and replaces the plane of zero-thickness with a 'Gaussian Plane' such that a satellite's position within the Gaussian determines the plane's goodness-of-fit to that satellite. This second approach requires that different Gaussian widths σ be tested for each plane orientation in order to find the width that best matches the satellite distribution. Values between 5kpc and 150 kpc were tested at 5 kpc intervals for each tested plane orientation. Hence an additional characteristic of the satellite distribution is obtained, but at the expense of a considerably longer computation time.

For each of the three measures of goodness-of-fit described above, the first step is to find the best-fit plane to the satellite positions with their positions determined from their best-fit distances. When either the RMS or maximum likelihood approach is used, the same best-fit plane is found as $0.153x + 0.932y + 0.329z = 0$ with pole at $(l_{M31}, b_{M31}) = (-80.7°, 19.2°)$. This plane is plotted as a great circle on the M31 sky in Fig. 3 with the poles of the plane indicated. When the absolute distance sum is used instead, the pole is found farther from the plane of the galaxy, at $(l_{M31}, b_{M31}) = (-74.9°, 24.3°)$. Nevertheless, the polar-plane described by Koch & Grebel (2006) is supported by either measurement, and is reminiscent of the satellite streams identified in the Milky Way satellite system. In light of the detection biases imposed by the PAndAS survey area as illustrated in Fig. 2, the result in this case must clearly be treated with suitable caution however. Like Koch & Grebel (2006), we find little evidence for the Holmberg Effect, with only 3 best-fit satellite positions falling within 30° of the M31 galactic poles, and only 6 of the 1σ error trails from Fig. 1 pass beyond $b_{M31} = \pm 60°$.

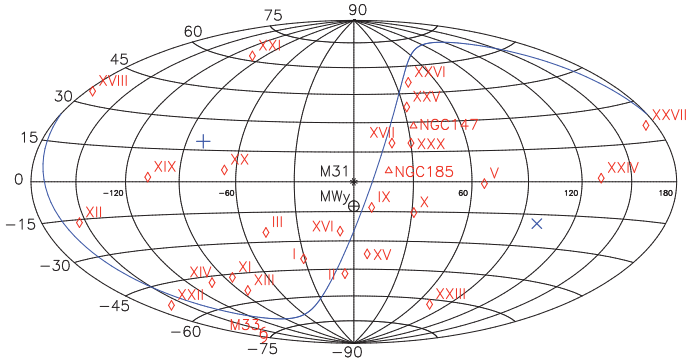


FIG. 3.— An Aitoff-Hammer Projection showing the best-fit plane to the satellite system as a whole. The pole and anti-pole of the plane are denoted by '+' and '×' symbols respectively. Only the best-fit satellite positions were incorporated into the fit for this figure. The distribution of poles obtainable from other possible realizations of the satellite distribution is presented in Fig. 4. Note that the plane is near-polar, similar to the preferred plane orientations identified for the Milky Way Satellite System.

To determine the uncertainty in the plane's goodness-of-fit, we need to repeat the procedure for a large number of realizations of the satellite sample, with the best-fit satellite distances replaced with a distance drawn at random from their respective satellite distance PPDs. A density map of the best-fit plane poles identified from 200,000 such realizations is presented in Fig. 4. This figure was generated using the distribution RMS as the goodness-of-fit statistic, and contains 71.1% of all poles within a 5° radius of the best-fit pole stated above. When the sum of absolute distances is used in place of the RMS, this fraction falls to 68.3%, or to 70.9% when the maximum likelihood approach is used. It should be noted that the distribution of poles lies in close proximity to the pole of maximum detection bias at $l_{M31} = -90°, b_{M31} = 0°$, again suggesting that the detection bias is having a strong influence on the polar orientation of the best-fit plane.

In order to determine whether the goodness-of-fit of the best-fit plane is really physically significant, similar analysis should be performed on a large number of random realizations of satellites, to see how often distributions of satellites

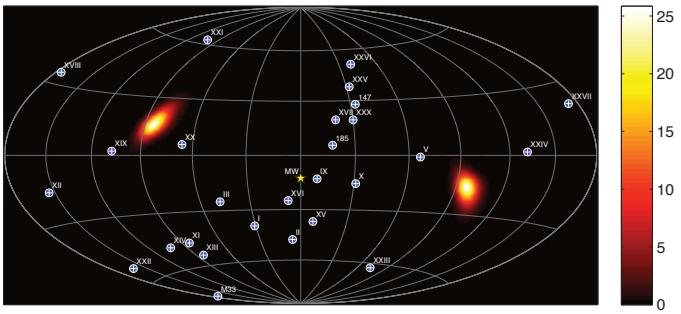


FIG. 4.— A pole-density map showing the effective uncertainty in the location of the best-fit plane to the whole satellite sample. The poles of the best-fit planes derived for 200,000 possible realizations of the data are plotted, along with their corresponding anti-poles.

arise with a comparable degree of planarity. Figure 5 presents probability distributions of the plane significance for possible realizations of the real satellite sample along with average values from random realizations of the satellites (as per §2.2), obtained using the three measures of goodness-of-fit stated above.

It is immediately clear from Fig. 5 that regardless of the choice of the measure of goodness-of-fit, the range of values obtainable from possible realizations of the real satellite positions are similar to the most likely values to be expected from completely random realizations of the satellites. Hence, whilst a prominent plane of satellites comprising roughly half of the sample is suggested in Fig. 1, it would seem that the sample as a whole is no more planar than would be expected from a strictly random distribution. Again, this is in keeping with the findings of Koch & Grebel (2006), and detracts from any physical significance that should be attributed to the plane's polar orientation.

Further to this finding, the overall width of the 'plane' is again in keeping with that expected from a purely random satellite distribution. From fitting the Gaussian Plane to the best-fit satellite positions, a 1σ width of 60 kpc is found to produce the best fit to the data. When the 200,000 PPD-sampled realizations were tested, a 1σ of 60 kpc was found preferential in 66.3% of cases, with a 1σ of 55 kpc being preferred in 32.7% of cases. Values of 50 kpc make up the remaining 1% almost entirely. The average value for the actual satellite distribution was thus determined as 58.3 kpc. This value is similar to the most likely width identified from the 10,000 random realizations, as can be seen in Fig. 6.

3.2. *The Plane of Maximum Asymmetry*

To determine the plane of maximum asymmetry and its significance, we employ an identical approach as in the preceding section, but with the goodness-of-fit statistic replaced with a count of the number of satellites on each side of the plane as per §2.1. As was suggested by the three-dimensional satellite distribution generated in CIL12, the asymmetry about the M31 tangent plane is close to a maximum, with 19 satellites on the near-side of the plane but only 8 on the other when the best-fit satellite positions are assumed. The highest asymmetry plane possible from this same distribution has 21 satellites on one side and 6 on the other, with the equation of the plane identified by the algorithm as $-0.797x - 0.315y + 0.515z = 0$. The anti-pole of this plane lies 27.2° away from the Milky Way at $(l_{M31}, b_{M31}) = (-21.6°, -31.0°)$. This plane is plotted as a great circle on the M31 sky in Fig. 7.

When 200,000 realizations of the satellite sample are generated using the satellite's respective distance probability distributions, the most likely asymmetry of the sample is actually found to be greater than this, with 23 satellites on one side and only 4 on the other. Such a scenario is more than twice as likely as the 21 : 6 scenario. In one realization, a plane was identified which could divide the sample such that all 27 satellites lay in a single hemisphere, while an asymmetry of 26 : 1 was found possible for 815 (0.4%) of the realizations. The distribution of maximum-asymmetry poles on the sky, as determined from realizations of possible satellite positions, is illustrated in Fig. 8, whilst Fig. 9 (*a*) plots the probability distribution for the greatest number of satellites that can be found in one hemisphere for a given realization of the observed satellite sample. The average value of this distribution is 22.7 (shown as a dashed line in Fig. 9 (*b*)), a value which is equalled or exceeded for 422 out of the 10,000 random realizations represented in Fig. 9 (*b*). A maximum asymmetry ratio of 21 : 6, as was observed for the best-fit satellite distribution plotted in Fig. 7, is more common however, falling inside the 1σ credibility interval.

What is particularly striking about the satellite distribution however, is the orientation of the asymmetry, with the majority of satellites lying on the near-side of the M31 tangent plane. From Fig. 9 (c), it is clear that the effect of the distance uncertainties lying along the line of sight is to create quite a broad distribution in the level of asymmetry about the tangent plane, though the average is markedly high at 20.3. To investigate the likelihood of this scenario arising from a random satellite distribution, we measure the average number of satellites on either side of the M31 tangent plane for each of 10,000 random realizations as per §2.2. The results are illustrated in Fig. 9 *d*). The observed profile is more-or-less as expected, with a maximum probability close to the minimum possible asymmetry at 14 and then a rapid fall off toward higher asymmetries. It is therefore clear that the distance uncertainties lying along the line of sight have no significant bearing on the orientation of the asymmetry. Yet the observed degree of asymmetry about the M31 tangent plane is equalled or exceeded in only 46 of the 10,000 random satellite realizations and hence is very significant. The possibility that this asymmetry may be a consequence of data incompleteness is currently being examined more closely (see Martin et al. 2012), although it seems very unlikely. The high degree of asymmetry is still observed even when only the brightest satellites are considered. Furthermore, the data incompleteness appears to be dominated by the boundaries of the PAndAS survey area and obstructed regions which are already taken into account by our analysis. Indeed, one would expect more satellites to be observed on the far side of the M31 tangent plane on account of the increased volume of space covered by the survey at greater distances, an effect clearly visible in Fig. 2.

3.3. *Subsets of Satellites*

It is perhaps not surprising that the satellite system of M31, when treated as a whole, is no more planar than one would expect from a random sample of comparable size. Indeed, a similar result was noted for the M31 system by Koch & Grebel (2006). The existence of outliers in our satellite sample was already clear from Fig. 1 and furthermore, if multiple planes of differing orientation are present as has been suggested for both the Milky Way's satellite system (e.g. Lynden-

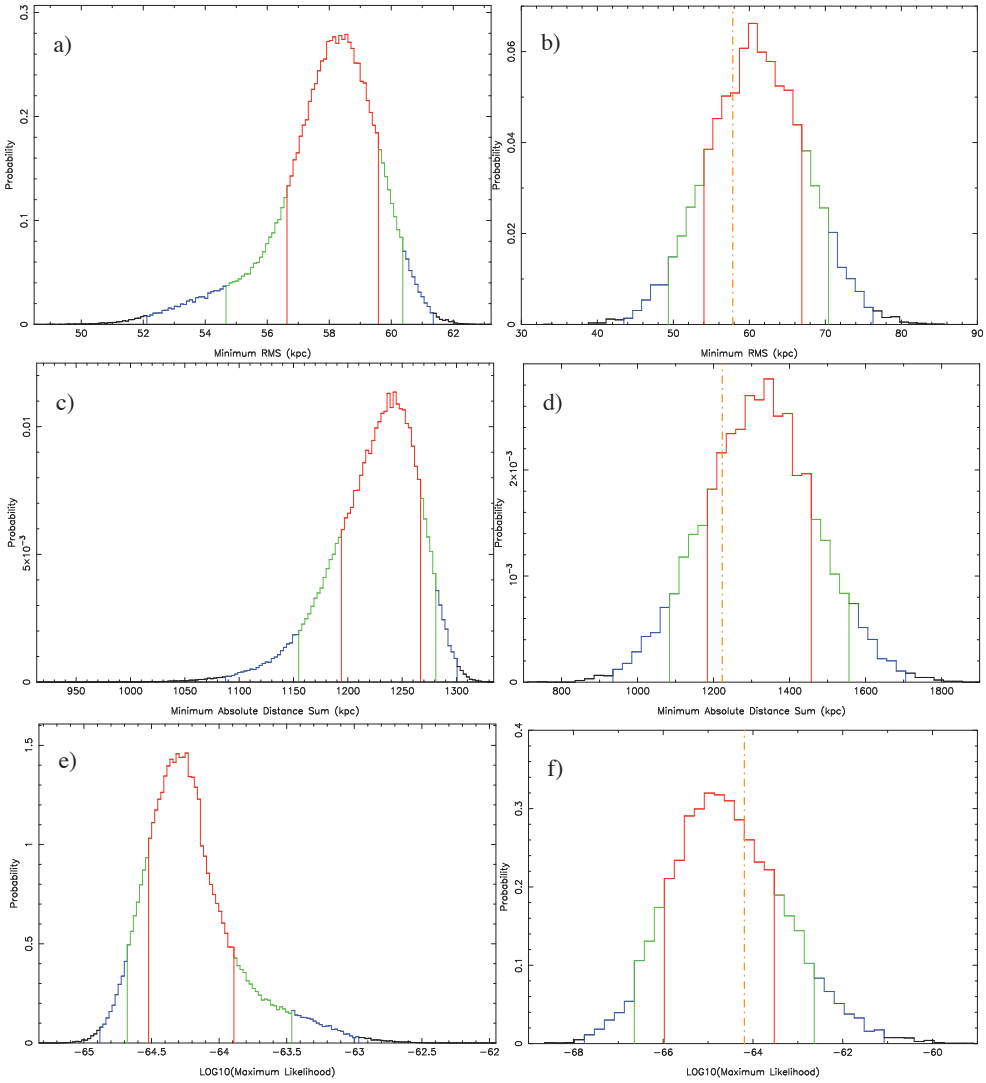


FIG. 5.— Probability distributions for the planarity of the entire satellite sample, as determined from three different measures of the plane goodness-of-fit. The left-hand column of figures gives the distribution of the goodness-of-fit statistic as obtained via plane fitting to 200,000 separate samplings of the *real* satellite sample. The right-hand column of figures summarizes the same procedure performed for 1,000 separate samplings of each of 10,000 *random* realizations of the satellites (as per §2.2). It is important to note that each histogram in this column has been generated by plotting the *average* values from the 10,000 individual histograms corresponding to each of the random realizations and hence they should only be compared with the *average* of the histograms in the left-hand column. The goodness-of-fit statistic for a) and b) is the distribution RMS; for c) and d) is the absolute distance sum and; for e) and f) is the sum of satellite likelihoods. The average of the histograms in (a), (c) and (e) are shown in (b), (d) and (f) respectively as dashed lines. Red, green and blue lines denote the extent of 1σ (68.2%), 90% and 99% credibility intervals respectively.

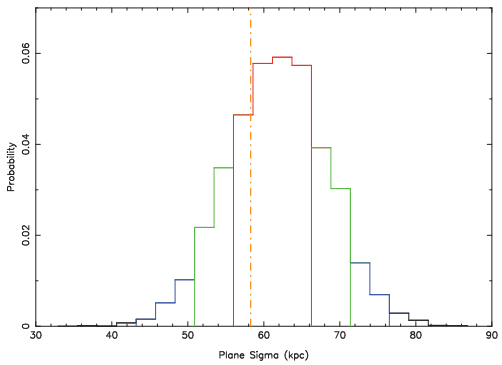


FIG. 6.— The probability distribution for the *average* 1σ width as determined from 10,000 random distributions of 27 satellites. This figure is generated from the same run as Fig. 5 f) and is the result of marginalizing over the plane-orientation model parameters.

Bell 1982; Pawlowski, Pflamm-Altenburg, & Kroupa 2012B) and the M31 system (McConnachie & Irwin 2006), then the goodness of fit of the best-fit plane to the entire distribution is of little consequence. For this reason, we now concentrate our analysis on subsets or *combinations* of satellites. Specifically, we perform a pole-count analysis by determining the pole of the best-fit plane to every possible satellite combination of a particular size that can be drawn from the entire sample.

A pole-count analysis is an excellent way of mapping the degree of prominence of various planes that exist within the distribution as a whole, whatever their orientation may be. The choice of combination size is not trivial however. The number of combinations s of a particular number of satellites k that can be drawn from the entire sample of n satellites can

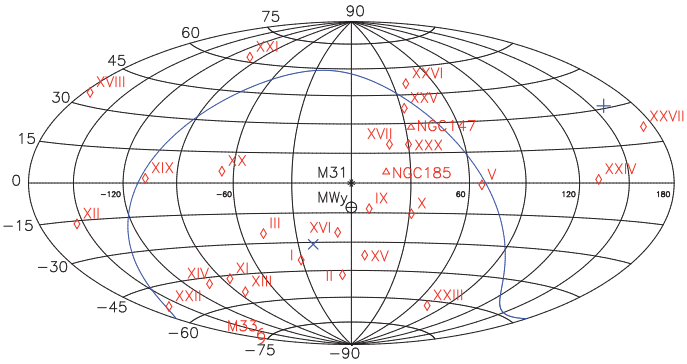


FIG. 7.— An Aitoff-Hammer projection showing the plane of maximum asymmetry identified from the full sample of best-fit satellite positions. It divides the distribution such that 21 satellites lie in one hemisphere, but only 6 in the other. The anti-pole of the maximum asymmetry plane lies just 28.1° from the Milky Way as viewed from the center of M31.

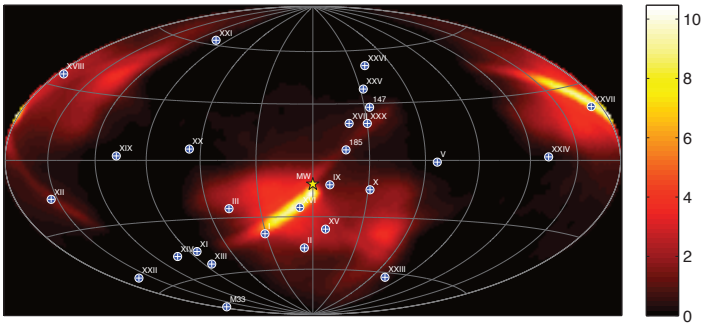


FIG. 8.— A pole-density map showing the effective uncertainty in the location of the maximum asymmetry plane to the whole satellite sample. The poles of the maximum asymmetry planes derived for 200,000 possible realizations of the data are plotted, along with their corresponding anti-poles. The elongated distributions that run through the pole and anti-pole determined from the best-fit distribution (see Fig. 7) arise due to the orientation of the uncertainty trails of the individual satellite positions, as presented in Fig. 1. Note that the probability of the anti-pole of the asymmetry lying within a couple of degrees of the direction of the Milky Way is close to a maximum.

be determined as follows:

$$s = \frac{n!}{k!(n-k)!} \tag{3}$$

For reasons that shall be discussed shortly, we will effectively be working with a sample of 25 satellite positions. It is clear from this equation however that with 25 satellites forming the entire sample, the total number of combinations that can be drawn may be very large, depending on the number of satellites forming the combinations. For instance, if $n = 25$ and $k = 13$, there are over 5.2 million possible combinations that can be drawn. Additionally, if we are to properly account for the uncertainties in the satellite positions, it will be necessary to sample from the distance distributions of each satellite a large number of times for every combination. Given that we must test every possible plane orientation (as per §2.1) for every rendition of every combination, the computation times can become impracticable. It is therefore necessary to limit our combination sizes as much as possible. We note however, that the final pole-plot distribution showing the poles of the best-fit planes to each combination, is not so dependent on the combination size as might at first be thought.

With all the planes tested as per §2.1 having to pass through the center of M31, the minimum number of satellites that can not be fitted exactly is 3. This is therefore the smallest combination size we consider. There are 2,300 combinations of 3 satellites that can be drawn from the full sample of 25 satellites. If we increase the combination size considerably to 7 satellites, there are 480,700 satellite combinations that can be drawn. Due to an excessive number of combinations beyond this point, this is the largest combination size we consider. But it is critical to note that even if we produce our pole-plot map from combinations of only 3 satellites we *do not* exclusively find planes consisting of 3 satellites. If a plane of 7 satellites exists for instance, then by Eq. 3, such a plane will produce 35 poles at the same location on the pole plot, where a plane consisting of only 3 satellites would contribute only one pole. Conversely if we take combinations of 7 satellites, despite the larger number of possible combinations in total, we become less sensitive to planes made up of less than 7 satellites. So in a sense, the combination size we choose depends on the satellite planes we wish to be most sensitive to. In practice, we have found that the smaller combination sizes of 3 and 4 satellites are particularly useful for identifying the lowest RMS planes congregating around the band of satellites visible in Fig. 1. The larger combination sizes of 5, 6 and 7 satellites gradually shift toward finding planes closer to the best-fit plane to the entire satellite sample illustrated in Fig. 3.

Noting these points, we proceed as follows. First, the number of satellites per combination k is chosen ($3 \leq k \leq 7$) and then for each combination, distances are drawn for each of the satellites from their respective posterior distance distributions as provided in CIL12. To give a satisfactory representation of the form of the distributions, each combination is sampled 100 times. As such, each satellite combination contributes not 1 pole to the pole density map for the chosen combination size but 100, with the spread of poles relating the possible orientations of the best-fit plane to the combination, given the error in the individual satellite positions. The contribution of each pole to the density map is also weighted by the RMS of the best-fit plane it represents. Thus each pole does not contribute 1 count, but rather some fraction, depending on how good a fit the plane it represents is to the satellites in the combination. This fraction is also further divided by 100, since it represents only 1% of the samples for the combination, as just discussed.

As stated above, it should also be noted that we effectively limit the total number of satellites in our sample to 25 for all analysis in this subsection. This is to account for the bound group of satellites consisting of NGC147, NGC185 and And XXX (henceforth the NGC147 group). Since we suspect that these satellites orbit M31 as a group and since they all lie along the apparent plane identified in Fig. 1, it is preferable to treat the group as a single object when we are not concerned with measurements of the significance of particular planes. To do this, we take the luminosity weighted centre as an approximation for the center of mass, and treat this determined position as though it were the location of a single satellite. To calculate the luminosity weighted center, we can ignore the contribution from And XXX since it is negligible compared with the contributions of the two dwarf ellipticals. From the Third Reference Catalogue of Bright Galaxies (de Vaucouleurs et al. 1991), NGC185 is 0.2 magnitudes brighter than NGC147 in the V-band. Each time the NGC147 group is chosen as one of the 'satellites' for a combination, distances to each of NGC147 and NGC185 are sampled from their respective distributions and the luminosity weighted center of

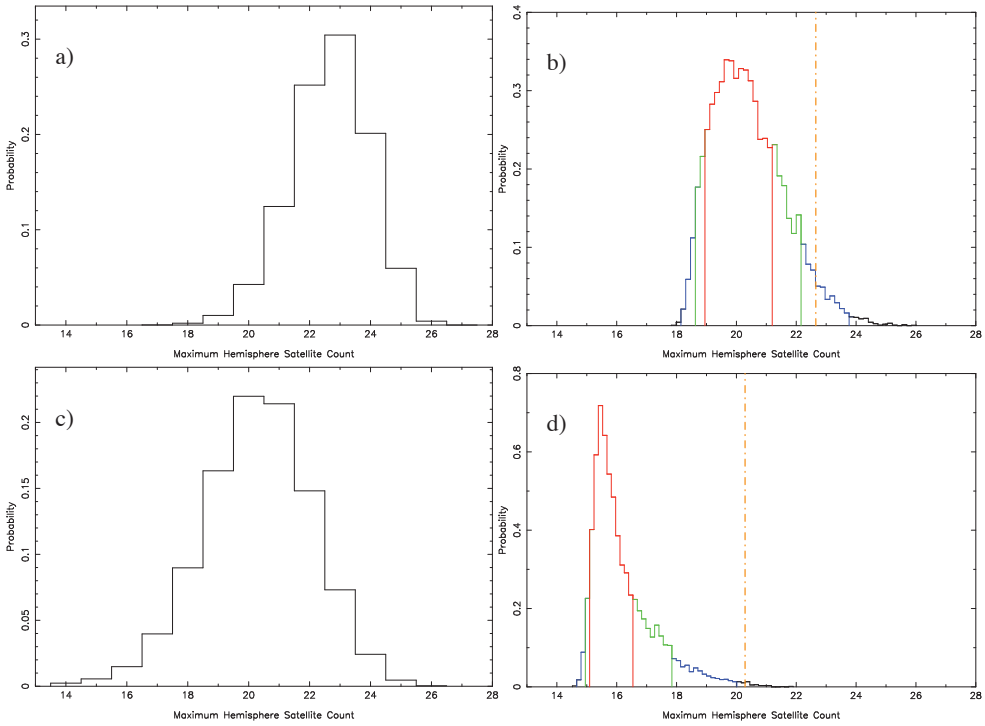


FIG. 9.— Asymmetry probability distributions. The top two histograms plot probability distributions for the greatest number of satellites that can be found in one hemisphere, as generated from (a) 200,000 samplings of satellite positions possible from the data and; (b) the *average* of 1000 samplings from each of 10,000 random realizations of the satellites generated as per §2.2. Figures (c) and (d) give the equivalent distributions when the maximum asymmetry plane is replaced with the fixed M31 tangent plane. As for Fig. 5, the histograms in the right-hand column should only be compared with the average of the corresponding histogram in the left column. The average value of the histograms of (a) and (c) are shown in (b) and (d) respectively as a dashed line.

the group is determined. As for any other combination, this position, along with all other satellites in the combination, is sampled 100 times.

The results of applying the above procedure to all combinations of 3, 4, 5, 6 and 7 satellites that can be drawn from the total sample is presented in Fig. 10. The left-hand column shows the fit to the most planar combination determined from the best-fit positions whilst the right-hand column shows the corresponding pole density plots for all combinations of that particular number of satellites, based on 100 samples of each combination as per the discussion above. It is noteworthy that the best-fit planes to the most planar combinations are almost identical in every case, except for that of the 3 satellite combinations, where the RMS values are so small for so many combinations as to make this result not particularly important. It should also be noted that these best-fit planes trace out the same approximate great circle as the prominent plane indicated in Fig. 1, a result that shall be investigated a little later in §3.4. It is particularly interesting that the pole shared by each of these planes, located at $l_{M31} \approx -80°, b_{M31} \approx 40°$ corresponds to a pole count maximum in each of the pole plots. This indicates that many of the satellite combinations are aligned along this plane, hence further suggesting that the plane applies to more satellites than the combination sizes tested here. The other, lower latitude principle maximum in the pole plots is that corresponding approximately to the best fit to *all* the satellites and hence it grows more prominent in the plots made from larger combination sizes as discussed earlier.

Besides the pole count maxima that are strongly indicative of a highly planar subset of satellites, the other principle feature of the pole plots in Fig. 10 is the great circle along which the pole count density is highest. This great circle is very prominent but great caution must be exercised in attributing any significance to it. It is centered on the Milky Way indicating that the constituent poles result from a majority of satellites lying along the Earth to M31 line of site. But this reflects the anisotropy predicted from Fig. 2, the result of the bias incurred by the finite area of the PAndAS survey. Hence it would seem that the progenitor of this prominent great circle is not physical but rather the result of selection effects. To further investigate the significance of the patterns observed in the pole plots, 1000 random realizations of 25 satellites were generated as per §2.2, and a similar pole count analysis performed on each of them. Specifically, the pole density distribution resulting from the best fit planes to all combinations of 5 satellites was generated for each of them. The resulting pole plots for 8 of the 1000 random realizations (chosen at random) are presented in Fig. 11 along with an enlarged version of the equivalent plot from Fig. 10 generated from the real distribution. A bias toward a similar high-density great circle is indeed observed in these plots, but the plot generated from the actual data features a conspicuously narrower great circle, and a much more constrained distribution in general. This appears to be primarily the result of the large fraction of satellites that lie along the prominent plane that is repeatedly identified and plotted in the left-hand column of Fig. 10. It should also be noted that this plane, whilst being oriented perfectly edge-on with respect to the Earth, contains a significant fraction of satellites lying well outside the region of the M31 sky where the detection bias is large, and hence it is unlikely that its prominence is due to our observational constraints.

Figure 12 provides for a comparison between the concentration of poles around the principle maximum in the pole dis-

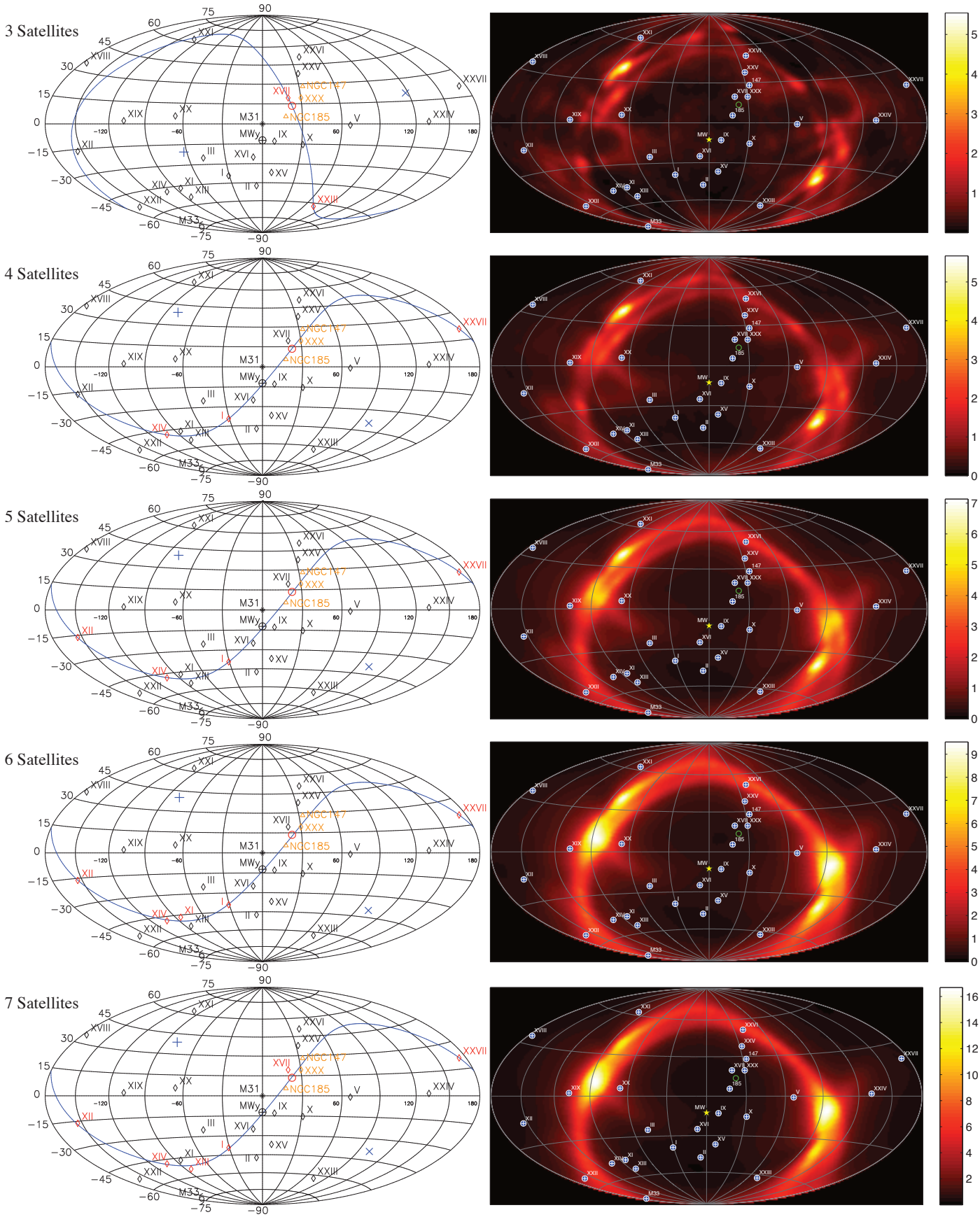


FIG. 10.— Best fit planes and pole density maps for combinations of 3 through 7 satellites. The left-hand column shows the best-fit plane through the combination of satellites that can be fit with the lowest RMS. Satellites included in the best-fit combination are colored red. The centre of the NGC147 group is marked with a circle, and lies on the best-fit plane in every case. The three members of this group are colored orange. Only the best-fit satellite positions are considered for these plots. The right-hand column shows the corresponding pole density plot for the poles of *all* satellite combinations. These plots have been weighted by the RMS of each pole and fully account for the uncertainty in the satellite positions.

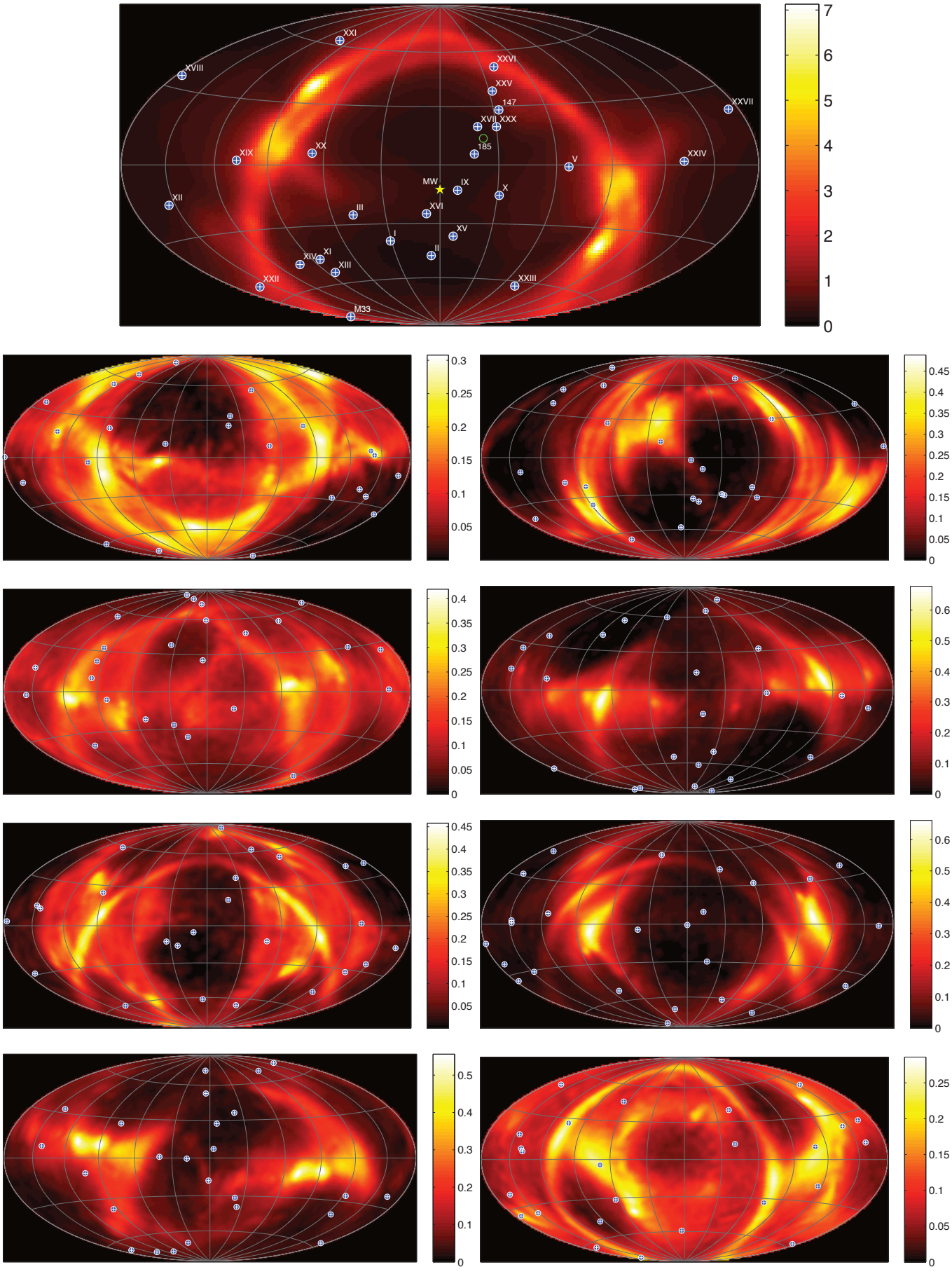


FIG. 11.— Pole density maps for 8 random realizations of 25 satellites. The maps plot the poles for the best-fit planes to all combinations of 5 satellites. The contribution of each pole is weighted by the RMS of the plane it represents. The map resulting from all combinations of 5 satellites drawn from the real data is shown again at the top for comparison.

tributions of the actual satellite distribution and the average of the 1000 random satellite distributions. From line (a) in Fig. 12 we see that 21.5% of all combinations of the actual satellite positions are fitted by a best-fit plane with pole within 15° of the principal maximum (located at $l_{M31} = -78.7°, b_{M31} = 38.4°$). This is in stark contrast to the 12.0% that lie within 15° of the principal maximum for the average random realization of satellite positions (Fig. 12 line (b)). Furthermore, we find that only 117 of the 1000 random realizations exhibited the degree of concentration of poles within 15° of the principal maximum that was observed for the actual satellite distribution. Hence it would seem that a large percentage of satellite combinations are fitted by best-fit planes that all have strikingly similar orientations when compared with what one could expect from a random distribution of satellites. Again, this points toward a significant plane of satellites that includes a large fraction of the whole satellite sample.

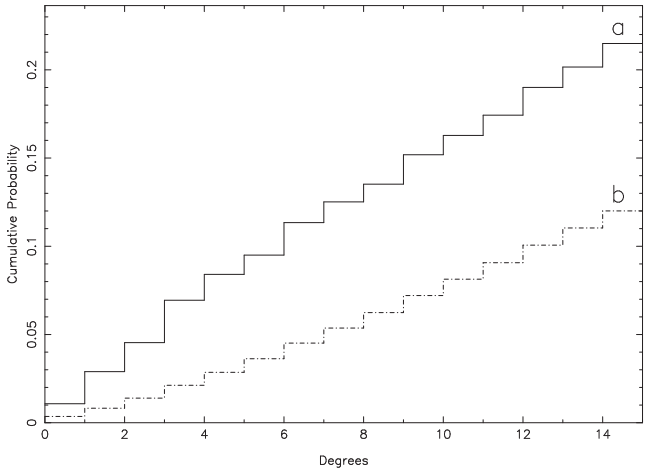


FIG. 12.— Radial density profiles showing the percentage of all poles lying within n degrees of the densest point in the pole count distributions (the principal maximum) for a) the actual satellite distribution and b) the average of 1000 random satellite distributions. The profile for the actual satellite distribution is generated from the same pole distribution as illustrated for 5 satellites in Fig. 10 and at the top of Fig. 11. Note that the relative linearity of (b) compared with (a) is simply a result of the averaging of a large number of individual profiles undertaken to produce the former.

In order to obtain a better understanding of the satellites that this plane consists of, it is of particular interest to explore the number of times each satellite is included in a combination that is best fit by a plane with pole in close proximity to the principal maximum in the pole distribution for the entire sample. Once again, we use the pole distribution for all combinations of 5 satellites, and we count the number of times each satellite contributes to a pole within 3° of the principal maximum at $l_{M31} = -78.7°, b_{M31} = 38.4°$. The counts are divided by 100 to account for the 100 samples that are taken of each combination. The result can be seen in Fig. 13. From this figure, it can be seen that the main contributors to the principal maximum in pole counts are those same satellites identified as forming a prominent plane in Fig 1, namely Andromedas I, XI, XII, XIII, XIV, XVI, XVII, XXV, XXVI, XXVII and the NGC147 group, along with Andromeda III and Andromeda IX. Hence the conclusion of our analysis thus far must be that there is indeed a significant plane in the satellite distribution of M31 and that it broadly consists of the aforesaid satellites. We therefore investigate the numerical significance of the best-fit plane to these satellites in §3.4. As yet there is still more to be gleaned from a study of the pole density distribution however.

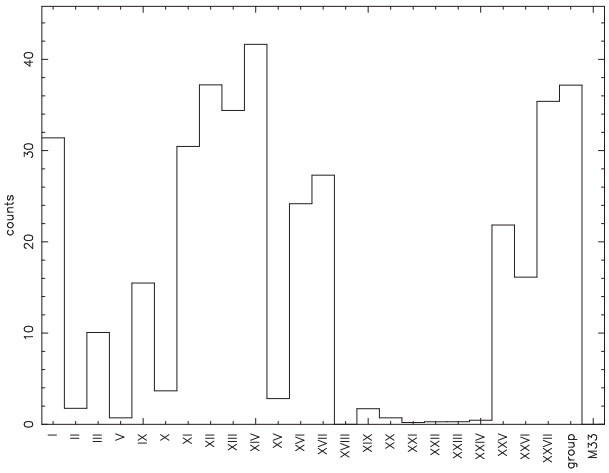


FIG. 13.— Histogram showing the relative contribution of each satellite to the pole density within 3° of the principal maximum at $l_{M31} = -78.7°, b_{M31} = 38.4°$. The histogram is generated from the same pole distribution as illustrated for 5 satellites in Fig. 10 and at the top of Fig. 11.

From Fig. 13 we have been able to determine the principle contributing satellites to the principal maximum in the pole density distribution, but what of the remaining satellites? Do the positions of these satellites follow any particular trend? The best way to determine this is to construct pole density plots of the two halves of the complete sample, namely the major contributors to the principal maximum and the minor contributors. The resulting pole plots are presented in Fig. 14.

The left-hand plot of Fig. 14 shows the pole density distribution generated from the major contributing satellites to the principal maximum at $l_{M31} = -78.7°, b_{M31} = 38.4°$. This half-sample includes Andromedas I, III, IX, XI, XII, XIII, XIV, XVI, XVII, XXV, XXVI, XXVII and the NGC147 group. As expected, this plot reflects the existence of the aforementioned plane with all combination poles lying in the vicinity of the principal maximum. The right-hand plot, with poles generated from the remaining 12 satellites, namely Andromedas II, V, X, XV, XVIII, XIX, XX, XXI, XXII, XXIII, XXIV and M33, paints a very different picture however. There is a much greater spread in the distribution of poles, with the great circle induced by the survey area bias once again conspicuous. Also prominent in this plot are 2 density maxima with their corresponding mirror images in the opposite hemisphere. The maximum lying midway between Andromedas XIX and XX lies very close to the pole of maximum detection bias at $l_{M31} = -90°, b_{M31} = 0°$ and so it is not unexpected, now that the prominent plane of satellites is effectively removed from the distribution. The elongated maximum passing through $l_{M31} \approx 45°, b_{M31} \approx 45°$ is more interesting however, and suggests the possibility of a second plane, roughly orthogonal to the major plane represented in the left-hand plot, though much less conspicuous. The planes represented by this maximum pass close to the error trails on the M31 sky of Andromedas II, III, XIX, XX, XXIII and XXIV. This maximum is faintly discernible in the pole distribution for combinations of 6 satellites presented in Fig. 10 but is no more pronounced than anywhere else along the high-density great circle in any of the other pole plots. On account of this, it

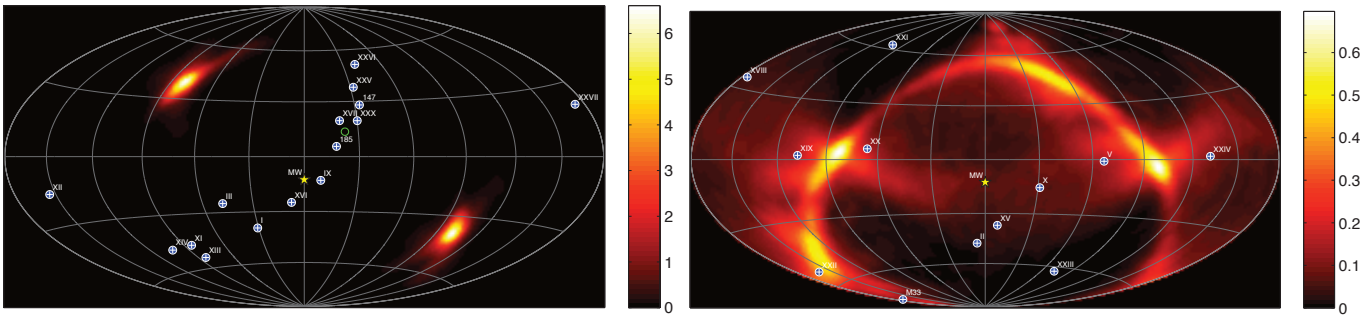


FIG. 14.— Pole density distributions generated from all combinations of 5 satellites possible from: Left) the satellites contributing significantly to the principal maximum at $l_{M31} = -78.7°, b_{M31} = 38.4°$ as per Fig. 13 and Right) the remaining 12 satellites.

would appear that this plane is likely no more significant than one would expect to find from a random satellite distribution subject to the same detection biases, such as those illustrated in Fig. 11.

3.4. *A Great Plane of Satellites*

Throughout the investigation undertaken thus far, all evidence has repeatedly pointed toward a conspicuously planar sub-set of satellites consisting of roughly half the total sample of satellites. Andromedas I, XI, XII, XIII, XIV, XVI, XVII, XXV, XXVI, XXVII and XXX as well as the dwarf ellipticals NGC147 and NGC185 all appeared to lie along a plane in Fig. 1. The reality of this co-planarity was verified in §3.3 and in particular Fig. 13, which also suggested that Andromeda III and Andromeda IX should be considered as plane members. Hence it is of great interest to ascertain whether this 'great plane' is in fact significant. To do this, it is necessary to determine how likely such a plane is to arise from a random satellite distribution subject to the same selection biases. The plane itself and the satellites of which it is constituted are illustrated in Fig. 15. The plane shown is that calculated from the best-fit satellite positions and has equation of the form: $0.158x + 0.769y + 0.620z = 0$ with pole at $(l_{M31}, b_{M31}) = (-78.4°, 38.3°)$. Note that for this section, we re-instate NGC147, NGC185 and Andromeda XXX as separate objects since we are again concerned with measurements of the significance of the planarity of the distribution. Our 'great plane' thus consists of 15 satellites out of the entire sample of 27.

Using the method of §2.2, we again generate 10,000 independent random realizations of 27 satellites and seek the most planar combination of 15 satellites from each. For each random realization, we sample 1000 possible positions for each satellite as in previous sections and take the average value for the RMS of the best fit plane through the most planar combination. Since there are more than 17 million ways that 15 satellites can be drawn from 27, and since we are not concerned with the orientation of each fitted plane as we have been in all previous sections, we depart from the plane fitting method of §2.1 for this section and instead proceed as follows. For each sample of satellite positions from each realization, 10,000 randomized planes are generated and the 15 closest satellites of the 27 to the plane are stored in each case and the RMS recorded. The lowest RMS achieved is hence taken to be that for the most planar combination of 15 satellites in the sample. These minimum RMS values from each of the 1000 samples of the particular random realization are then averaged to provide the best representation for the realization,

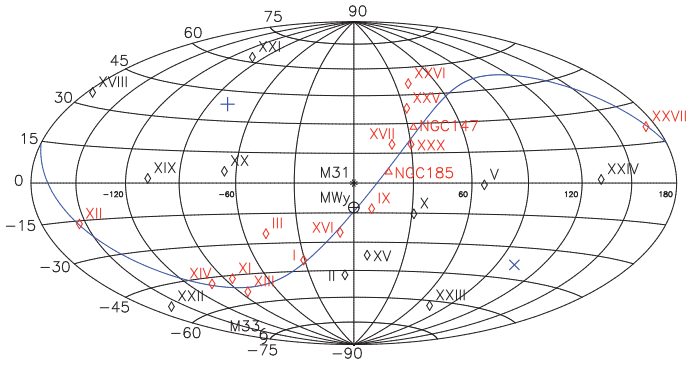


FIG. 15.— A Great Plane of Satellites consisting of Andromedas I, III, IX, XI, XII, XIII, XIV, XVI, XVII, XXV, XXVI, XXVII, XXX, NGC147 and NGC185. The plane shown is that derived from the best-fit satellite positions. The pole is located at $(l_{M31}, b_{M31}) = (-78.4°, 38.3°)$.

given the distance uncertainties. Fig. 16 provides probability distributions in the RMS for the observed 'great plane' (*a*) together with those for the average RMS for the most planar combination from each random realization (*b*). The average RMS for the observed plane is plotted in (b) for comparison.

As can be seen from Fig. 16, the RMS for the observed plane is very low compared to what one could reasonably expect from a chance alignment. Indeed, the average RMS of 12.58 kpc for the observed plane is found to be equalled or exceeded in only 36 out of the 10,000 random realizations. The chances of obtaining such a planar group of 15 satellites from a sample of 27 at random is thus estimated as 0.36%. Hence we can conclude from this test that the observed plane is very unlikely to be a chance alignment, but rather the result of some underlying physical mechanism. Note that an independent but equivalent investigation is presented in ILC12 where such an alignment is found to occur in only 0.15% of instances. This is due to the larger central obstruction adopted in that analysis (19.6 vs. 7.9 sq. deg.) which rejects more satellites in close proximity to the plane pivot point (M31) where small plane distances are most likely.

4. DISCUSSION

Throughout the analysis conducted in §3, the presence of a prominent plane of satellites has been a consistent feature. This is not the first time that a significant plane of satellites has been identified from among the denizens of the M31 halo however. Koch & Grebel (2006) identified a highly significant plane lying within 5° to 7° of being polar. Further-

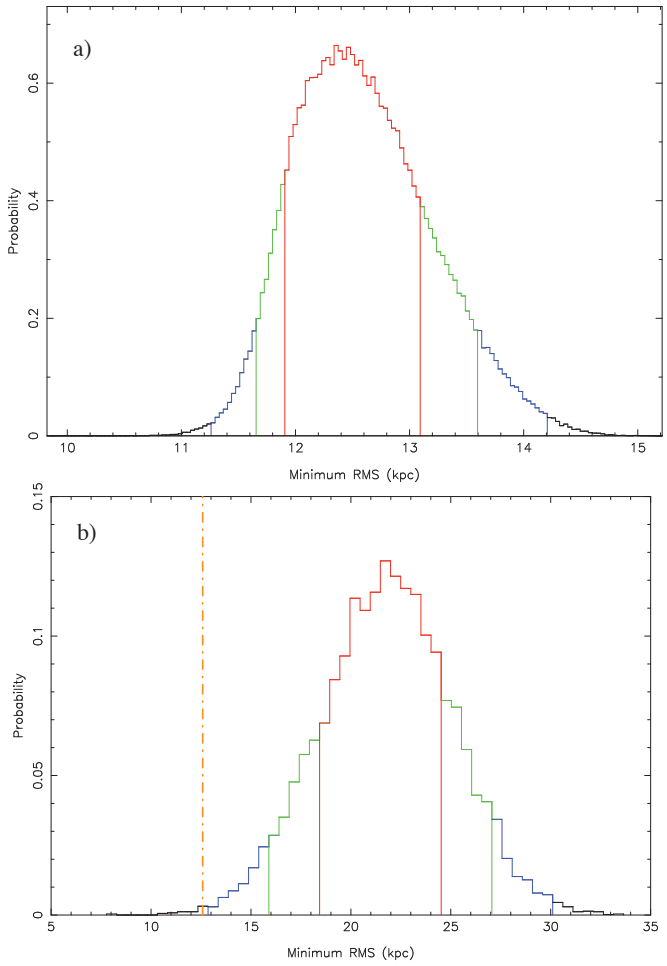


FIG. 16.— Determining the significance of the observed 'great plane' of satellites (see Fig. 15). Figure (a) gives the distribution of possible values of the RMS obtainable from 200,000 realizations of possible positions of the 15 plane members, given their respective distance probability distributions. Figure (b) plots the *average* RMS of the best fit plane through the most planar combination of 15 satellites for each of 10,000 random realizations of 27 satellites. These satellites are subject to the same selection biases as the real data. As for Fig. 5, histogram (b) should only be compared with the average of histogram (a), which is plotted in (b) as a dashed line. It is thus clear that the planarity observed for our 'great plane' of satellites is very unlikely to arise by chance. The 1σ (68.2%), 90% and 99% credibility intervals are shown as red, green and blue lines respectively.

more, they identify a subset of 9 satellites from this plane lying within a thin disk with an RMS of 16 kpc. Metz, Kroupa, & Jerjen (2007) and later Metz, Kroupa, & Jerjen (2009) similarly identify a disk of satellites, this time not so markedly polar, with pole (in our coordinate system) at $(l_{M31}, b_{M31}) = (-70.2°, 32.9°)$. They find this disk to have an RMS height of 39.2 kpc. This disk is clearly the same structure that we identify here, being tilted by only 8.6° with respect to our 'great plane.' Our plane is found to have a much smaller RMS of just $12.34^{+0.75}_{-0.43}$ kpc however, despite including a comparable number of satellites. It is particularly noteworthy however, that their satellite sample is significantly different to that used here, with their disk including M32, NGC205, IC10, LGS3 and IC1613 - all of which lie outside the portion of the PAndAS survey region used in this study (see Fig. 10 (c) of CIL12). Indeed, it is clear from Fig. 4 of McConnachie & Irwin (2006) that the galaxies M32, IC10, LGS3 and IC1613 all lie along the same great circle as our 'great plane' in Fig. 15, as do their entire error trails. Their conformity along with Andromeda I to a thin disk is noted in the said paper as one of 8 possible 'streams of satellites,' thus providing another early detection of the plane identified by this study. Majewski et al. (2007) also draw attention to the linear distribution of many of the plane-member satellites on the sky, a consequence of the edge-on orientation of the plane as indicated by the present study. The plane of Metz, Kroupa, & Jerjen (2009) does however include a significant number of satellites that, whilst included in our sample, we exclude due to their looser association with our plane. This then accounts for the much smaller RMS height observed in our study.

Unlike previous studies of the M31 satellite system, we have a significant advantage in this study on account of the greatly improved sample of satellites available to us. Our sample is not only more numerous, but the positions are all determined via the same method applied to the same data as per CLI11 and CIL12. We are thus afforded unprecedented knowledge of the satellite detection biases, as well as the uncertainties in the object positions and have factored this knowledge into the analysis. An understanding of this bias is of particular importance when it comes to ascertaining the significance of any substructure identified, since a physically homogeneous satellite distribution will inevitably appear anisotropic after 'folding in' the selection function and it is important that we do not attribute physical significance to this anisotropy.

Even after taking these effects into account however, there can be little doubt that the plane described in §3.4 is a real physical object. The component satellites extend well into the regions of low detection bias in Fig. 2 and the analysis of the last section makes it clear that such a thin disk of satellites has very little chance of arising within a random satellite distribution of the same size, even when subject to the same observation biases. Furthermore, it should be noted that the study of the plane's significance in §3.4 is likely to be conservative, given that if the satellites M32, IC10, LGS3, IC1613 and NGC205 were to be included in the analysis, the significance of our observed plane would likely grow still further. What is also particularly interesting is that subsequent research has shown 13 of the 15 objects to be co-rotating. This result is discussed in more detail in ILC12.

What then could be the progenitor of this 'great plane'? The polar orientation one might expect to arise had the satellites formed within the dark matter halo or had the dynamical friction proposed by Quinn & Goodman (1986) had sufficient time to take effect is not observed. Similarly, the findings of Metz et al. (2009) seemingly preclude the possibility that the structure might be the result of the accretion of an external galactic association. Furthermore, there is apparently no marked distinction in the metallicities of the disk members compared with the non-disk members as one might expect from this scenario. There remains however the possibility that the satellites trace out the tidal debris of a galaxy merger. This is a particularly interesting possibility, especially since the plane, when projected onto the M31 tangent plane, is in close alignment with the Giant Stellar Stream. Indeed, Hammer et al. (2010) show that the Giant Stellar Stream could feasibly be the product of a major merger event that began around 9 Gyr ago, sustained by the returning stars from a tidal tail oriented similarly to our 'great plane.'

The observed asymmetry of the system does however pose a problem for this scenario. It is of particular interest that, of the 13 co-rotating satellites in the plane, all but one lie on the

near side of the M31 tangent plane. Indeed, if we removed all of the plane member-satellites from the system, the remaining satellite distribution would no longer be significantly asymmetric. With almost all of the satellites currently on the near side of M31, it would seem that the progenitor event could not have occurred substantially more than a typical orbital time ago or else the satellites would have had sufficient time to disperse. This suggests the event responsible must have occurred within the last 5 Gyr. Another plausible alternative is that a strong drag is induced on the orbiting satellites by an overdensity in the dark matter halo broadly lying along the Milky-Way-to-M31 separation vector. The result is analogous to gas passing through a galaxy's spiral arms. This scenario would account for the direction of the asymmetry but would lead to rapid orbital decay however and hence again would imply that the structure is relatively short lived. In any case, how such a thin rotating structure could survive for an extended length of time in a traditional triaxial dark matter halo remains unclear.

There is also another striking characteristic of the observed plane. As one will note from examination of Fig. 15 (and indeed the left-hand column of plots in Fig. 10), it is oriented perfectly edge-on with respect to the Milky Way. Whilst there is a noted bias toward detection of satellites positioned along planes oriented in this way, it must be remembered that this bias arises primarily due to the propensity for detecting satellites close to the line of sight passing through M31. Many of the satellites observed to lie on our plane are located a good distance from this line of sight however and well into the low-bias portions of the M31 sky. In any case, the random realizations of §3.4 suffer from the same biases and yet show unequivocally that the observed plane is very unlikely to arise by chance. Hence if we are to accept these results, we must also accept the plane's orientation.

Further to this strikingly edge-on orientation, it is also noteworthy that the plane is approximately perpendicular to the Milky Way disk. This fact can be easily seen if the constituent satellites are traced out in Galactic coordinates (i.e. all lie on approximately the same Galactic longitude). This of course raises the question - how does the orientation of the Milky Way's polar plane of satellites compare with this plane? Noting that the average pole of the 'Vast Polar Structure' described by Pawlowski, Pflamm-Altenburg, & Kroupa (2012B) points roughly in the direction of M31, the two planes are approximately orthogonal. These precise alignments are discussed in more detail in ILC12, but suffice to say here that this alignment is particularly interesting and suggests that the Milky Way and M31 halos should not necessarily be viewed as fully isolated structures. It is entirely conceivable that our current ignorance as to the coupling between such structures may be to blame for our inability to pin down the precise mechanism by which such planes arise.

5. CONCLUSIONS

It is clear that whilst the satellites of M31 when taken as a whole are no more planar than one can expect from a random distribution, a subset consisting of roughly half the sample *is* remarkably planar. The presence of this thin disk of satellites has been conspicuous throughout the analysis contained in this paper. The degree of asymmetry determined from the satellite distribution is also found to be relatively high. Of particular note, the orientation of the asymmetry is very significant, being aligned very strongly in the direction of the Milky Way. When this fact is combined with the apparent orthogonality observed between the Milky Way and M31 satellite distributions and the Milky Way disk, it appears that the two halos may in fact be coupled. Regardless, the great plane of satellites identified in this study, and its clear degree of significance, provides persuasive evidence that thin disks of satellites are a ubiquitous feature of galaxy dark matter halos.

A. R. C. would like to thank Macquarie University for their financial support through the Macquarie University Research Excellence Scholarship (MQRES) and both the University of Sydney and Université de Strasbourg for the use of computational and other facilities. G. F. L. thanks the Australian Research Council for support through his Future Fellowship (FT100100268) and Discovery Project (DP110100678). R. A. I. and D. V. G. gratefully acknowledge support from the Agence Nationale de la Recherche though the grant POMMME (ANR 09-BLAN-0228). Based on observations obtained with MegaPrime/MegaCam, a joint project of CFHT and CEA/DAPNIA, at the Canada-France-Hawaii Telescope (CFHT) which is operated by the National Research Council (NRC) of Canada, the Institut National des Science de l'Univers of the Centre National de la Recherche Scientifique (CNRS) of France, and the University of Hawaii.

REFERENCES

Conn, A. R., Lewis, G. F., Ibata, R. A., et al. 2011, ApJ, 740, 69 (CLI11) 2
Conn, A. R., Ibata, R. A., Lewis, G. F., et al. 2012, ApJ, 758, 11 (CIL12) 2
Hammer F., Yang Y. B., Wang J. L., Puech M., Flores H., Fouquet S., 2010, ApJ, 725, 542 14
Hartwick F. D. A., 2000, AJ, 119, 2248 1
Holmberg E., 1969, ArA, 5, 305 2
Ibata, R. A., Lewis, G. F., Conn, A. R., et al. 2012, Natur, ?,? accepted for publication (ILC12) 2
Knebe A., Gill S. P. D., Gibson B. K., Lewis G. F., Ibata R. A., Dopita M. A., 2004, ApJ, 603, 7 2
Koch A., Grebel E. K., 2006, AJ, 131, 1405 2, 5, 6, 13
Kroupa P., Theis C., Boily C. M., 2005, A&A, 431, 517 1
Lovell M. R., Eke V. R., Frenk C. S., Jenkins A., 2011, MNRAS, 413, 3013 1
Lynden-Bell D., 1976, MNRAS, 174, 695 1
Lynden-Bell D., 1982, Obs, 102, 202 1, 6
Lynden-Bell D., Lynden-Bell R. M., 1995, MNRAS, 275, 429 2
Majewski, S. R., Beaton, R. L., Patterson, R. J., et al. 2007, ApJ, 670, L9 14
Martin N. F., et al. 2012, in preparation 6
McConnachie, A. W., Irwin, M. J., 2006, MNRAS, 365, 902 2, 3, 7, 14
McConnachie, A. W., Irwin, M. J., Ibata, R. A., et al. 2009, Natur, 461, 66 2
Metz M., Kroupa P., Jerjen H., 2007, MNRAS, 374, 1125 14
Metz M., Kroupa P., Libeskind N. I., 2008, ApJ, 680, 287 1
Metz M., Kroupa P., Theis C., Hensler G., Jerjen H., 2009, ApJ, 697, 269 2, 14
Metz M., Kroupa P., Jerjen H., 2009, MNRAS, 394, 2223 14
Navarro J. F., Abadi M. G., Steinmetz M., 2004, ApJ, 613, L41 1
Palma C., Majewski S. R., Johnston K. V., 2002, ApJ, 564, 736 2
Pawlowski M. S., Kroupa P., Angus G., de Boer K. S., Famaey B., Hensler G., 2012, MNRAS, 424, 80 1, 2
Pawlowski M. S., Pflamm-Altenburg J., Kroupa P., 2012, MNRAS, 423, 1109 7, 15
Quinn P. J., Goodman J., 1986, ApJ, 309, 472 2, 14
Springel, V., Wang, J., Vogelsberger, M., et al. 2008, MNRAS, 391, 1685 1
Starkenburg, E., Helmi, A., De Lucia, G., et al. 2012, arXiv, arXiv:1206.0020 2
de Vaucouleurs G., 1958, ApJ, 128, 465 3
de Vaucouleurs G., de Vaucouleurs A., Corwin H. G., Jr., Buta R. J., Paturel G., Fouqué P., 1991, Third Reference Catalogue of Bright Galaxies (Berlin: Springer) 8
Zentner A. R., Kravtsov A. V., Gnedin O. Y., Klypin A. A., 2005, ApJ, 629, 219 1

"We live in a changing universe, and few things are changing faster than our conception of it."

Timothy Ferris, "The Whole Shebang" (1997)

6

Conclusions

The contribution to the field of galactic archaeology embodied in this work has essentially been twofold. Firstly, a robust new technique has been developed for ascertaining distances via the tip of the red giant branch, a technique which is stand-alone in terms of its diverse applicability. Secondly, this technique has been applied to the satellite system of M31 to reveal an inhomogeneous structure which is somewhat at odds with our current understanding of galaxy formation. The key outcomes of the thesis are summarized as follows:

I A powerful new Bayesian technique has been developed for determining the distance probability distribution of an object from the tip of its red giant branch. The technique is best suited to older, metal poor structures that are sufficiently close as to facilitate accurate photometric measurements to a depth exceeding that of the RGB tip by at least 0.5 magnitudes.

II A 'density' matched-filter has been developed to compliment the technique of **I**. This matched filter was developed specifically for the satellite galaxies of M31 and as such, is not applicable to extended structures (such as streams). It effectively acts to improve the contrast of the RGB tip in the object's luminosity function by weighting the component stars with respect to their position within the object's density profile.

III An angle-specific density prior has been devised specifically for the M31 halo and incorporated into the technique of **I**. It effectively equates each position along the line of sight to one of the satellites with some probability, based on the assumed sub-halo density at the associated radius from the center of the halo.

IV Accurate distance probability distributions have been obtained for 27 of the satellites of M31 as well as for M31 itself via incorporating the priors of **II** and **III** into the technique of **I**.

V The distance distributions of **IV** have been converted into M31-centric 3D positions, providing the largest homogeneous sample of satellite galaxy positions for any galaxy halo.

VI The M31 satellite distribution has been found to be approximately isothermal. When the 15 most Gaussian distance distributions are considered, the satellite density profile is found to follow a power law with $\rho(r) \propto r^{-\alpha}$ where $\alpha = 1.87^{+0.46}_{-0.42}$.

VII The satellite distribution as a whole has been shown to be no more planar than one would expect from a random distribution of points.

VIII A large subset of the satellites, 15 out of the total sample of 27, has been found to be remarkably planar, with a root-mean-square thickness of just $12.34^{+0.75}_{-0.43}$ kpc. The probability of obtaining such a large, thin structure in a random distribution of equal size is found to be only 0.36%. The orientation of this plane is intriguing. It is found to lie almost perfectly edge-on with respect to the Milky Way, and approximately perpendicular to the Milky Way disk. It is also roughly orthogonal to the planar distribution of satellites regularly reported for the Milky Way, and 13 of the 15 satellites have subsequently been identified as co-rotating.

IX The asymmetry of the distribution as a whole has been shown to be considerably larger than one would expect by chance. After factoring in the uncertainty in the satellite positions, it is most likely that the sample can be divided such that 23 of the 27 satellites all lie in a single hemisphere. The probability of the observed asymmetry arising in a random distribution is 4.22%.

X The asymmetry about the M31 tangent plane has been found to be particularly high, with 20 of the 27 satellites most likely lying on the Milky Way side. The probability of the observed degree of asymmetry about this plane arising by chance is just 0.46%. It is noteworthy that if the 15 satellites belonging to the plane of **VIII** are omitted, the asymmetry about the M31 tangent plane is no longer significant.

In light of the above outcomes, there are several avenues of future investigation that warrant attention. The first concerns the future application of the RGB tip finding technique in its current form. The PAndAS survey region is awash with the relics of past accretion events and is the obvious starting place. There are many streams of stars that are well within reach of the technique. Furthermore, it should be possible to divide most of these streams into segments and obtain distance measurements to each individually. The result would be the effective conversion of the key structures of the PAndAS survey into a three dimensional network of streams and interspersed satellite galaxies. This would facilitate a study of unprecedented scale into the distribution of mass within the M31 halo.

The RGB tip finding technique is of course also readily applicable to the denizens of the Milky Way halo. There are more than 25 satellite galaxies and more than 150 globular clusters that orbit within the halo of our own galaxy. Distance measurements already exist for almost all of them, but there would be significant advantages to a sample of distances obtained via the systematic application of a single measurement technique.

In addition to the possible future applications of the tip finding technique, there are also a variety of means by which it might be improved. In particular, the method in its current form is most suited to more metal poor structures where there is minimum variation of the i-band tip luminosity with metallicity. The method could be made more versatile by replacing the one-dimensional model of the object's luminosity function with a two-dimensional model

of its CMD. Following this approach, stellar isochrones could be incorporated to model the correct form of the object's red giant branch in colour-magnitude space, making the method more robust in its treatment of more metal rich objects, and those containing more than one stellar population. Along similar lines, one could also apply a matched-filter to the object's CMD. This could be achieved by fitting a 2D surface to the object's CMD and dividing it by that fitted to the CMD of a suitable comparison (background) field, creating an effective 'flat field' tailored to suit the object. Each star in the object's CMD could thus be weighted by its probability of being a true member of the object's red giant branch.

The structure of the M31 satellite distribution as revealed by this study, also presents a number of opportunities to further our understanding of the local universe. The most immediate course to pursue would seem to lie in the application of this knowledge to a new study of the galaxy's mass distribution via modeling of its rotation curve. The plane of co-rotating satellites (**VIII**) provides the perfect starting point for such a study. With the plane fortuitously aligned edge-on with respect to the Earth, it will be possible to calculate the tangential component of the satellites' orbital velocities directly from the radial velocities via simple trigonometry. Thus, if the satellites are approximated to follow circular orbits, it will be relatively straightforward to obtain probability distributions for the mass enclosed by each satellite orbit, and in so doing, extend the known M31 rotation curve significantly. It would also be possible to obtain estimates of the enclosed mass for the non-plane members via a maximum likelihood approach, after marginalizing over the two tangential components of the orbital velocity.

More than anything else, this study has highlighted the limitations of existing theories of galaxy formation and evolution. It is very difficult to explain how such a large, thin structure as that identified in **VIII** can remain intact for any length of time, let alone how it came to exist in the first place. Add to that the bizarre orientation and the high degree of asymmetry, and we are left with an intriguing enigma. The onus then is on unlocking this enigma, for in so doing we shall undoubtedly learn a great deal about galaxy evolution and M31's past, as well as that of our own galaxy.

An Introduction to the Appendices

As a PhD student, perhaps 90 % of my time has been occupied with the development of programs (principally in Fortran) designed to perform the analysis necessary for my research. In this sense, the written component of the thesis really is just the tip of the iceberg, and it therefore seemed both fitting and rather useful to record some of the source code for reference. When I embarked to do this however, I did not realize the shear volume of code I had amassed and subsequently found it necessary to condense the code substantially. The code that *is* presented in these appendices therefore represents only a fraction of all the programs written during my PhD candidature. Nevertheless, I have endeavored to reproduce here the most important programs and subroutines in as logical a way as possible and with minimum repetition. Each appendix is devoted to code pertinent to a particular chapter, and each program is introduced with a brief description of its purpose and functionality as well as a link to the thesis content to which it relates. Note that some of the programs make a very localized contribution to the material presented in the thesis while others are much broader in scope and may apply to a number of chapters. Many subroutines have also been omitted either to avoid repetition or to remove portions of code which are secondary to the principal functionality of the parent program. In summary, it is intended that the programs presented in these appendices serve to provide further clarification of the precise way in which the analysis discussed in the thesis has been implemented. The code is not intended for ready implementation on other systems and hence may in many instances require substantial modification to be usable. All code is however well commented and should be reasonably intuitive even for those not well acquainted with Fortran.

A

Chapter Two Programs

Program: EdgeFinder7.f95

Creation Date: 3 September 2009

Relevant Section: 2.2

Notes: This program represents one of my earliest investigations into potential TRGB-finding algorithms. It is really a number of stand alone algorithms rolled into one program. An artificial 'kink' is induced in a simple luminosity function and this kink is sort out by a number of methods: 1. By fitting a polynomial to the data and finding where the second derivative of that polynomial has the largest absolute value; 2. By Finding the largest positive gradient between two neighboring bins of the luminosity function and; 3. By taking the angle subtended by each subsequent set of 3 luminosity function bins. The identified location of the tip is outputted along with the value of the particular measurement statistic.

```
1  PROGRAM EdgeFinder
2  IMPLICIT NONE
3
4  !A polynomial of degree ma - 1 is fitted to the read-in data and a '.p' file is
5  !generated so that the read-in data can be instantly plotted using gnuplot along
6  !with the fitted polynomial. The RandReal subroutine then generates a mock data
7  !set based on the fitted polynomial and the d2ydx2max subroutine finds where the
8  !maximum rate of change of the gradient occurs which is symbolic of an 'edge'
9  !or sudden discontinuity in the fitted polynomial.
10
11 !Later adapted to use pgplot
12
13 INTEGER :: ma,mp,ndata,ndat,np
14 parameter (np = 20)
15 parameter (mp = 1000)
16 parameter (ndat = 1000)
17 parameter (ma = np)
18 DOUBLE PRECISION :: chisq,a(ma),sig(ndat),u(mp,np),v(np,np),w(np),x(ndat)
19 DOUBLE PRECISION :: y(ndat), z(ndat), integral_max, dummy, e(ndat), f(ndat)
20 EXTERNAL :: funcs
21
22 INTEGER :: ios, i, j
23
24 integer :: ma_max,ma_used
25 parameter (ma_max=100)
26 DOUBLE PRECISION :: pass_a(ma_max)
27
28 common/pass_block2/pass_a, ma_used
29
30 ma_used=ma
31
32 x=0. ; y=0. ; z=0. ; sig=0.00001
33 OPEN (unit = 1, file = 'luminosity_function2.dat', status = 'old')
34 OPEN (unit = 2, file = 'lffit.dat', status = 'unknown')
35 i=0
36 DO WHILE (.TRUE.)  !Reads data until end of input file and puts it into arrays
37    i=i+1
38    READ (1, *, IOSTAT = ios) dummy,x(i),dummy, dummy, dummy, dummy, y(i)
39    if (ios == 0) then ;
```

```
40      else if (ios == -1) then ;
41         i=i-1
42         exit ;
43      else if (ios > 0) then ;
44         i=i-1
45         cycle
46      end if
47   x(i) = x(i)/5.
48   IF(x(i) > -0.2) then
49      i = i-1
50   else if (abs(x(i)).lt.0.1) then
51   !    i = i-1
52   end if
53   END DO
54
55
56
57   DO j = 1, i    !Outputs exclusively the chosen data to 'lffit.dat'
58      WRITE (2, '(2ES20.5)') x(j), y(j)
59   END DO
60
61   ndata=i
62
63   CALL svdfit(x,y,sig,ndata,a,ma,u,v,w,mp,np,chisq,funcs) !SVD fitting program
64
65   do j=1,ma
66      pass_a(j)=a(j)
67   end do
68
69   PRINT *, a
70
71   OPEN (unit = 3, file = "lffit.p", status = 'unknown')
72
73   CALL RandReal(ma, a, x, i, sig, ndata, u, v, w, mp, np, chisq, funcs)
74
75   !CALL d2ydx2max(ma, a)
76
77   END PROGRAM EdgeFinder
78
79   !-------------------------------------------------------------------------
80
81   SUBROUTINE RandReal(ma, a, x, i, sig, ndata, u, v, w, mp, np, chisq, funcs)
82   !Random realization mock data generator
83
84   INTEGER :: ma, i, q, l, val, ndat
85   PARAMETER (val = 200)
86   PARAMETER (ndat = 1000)
87   INTEGER :: idum = 0
88   DOUBLE PRECISION :: a(ma), area, max_x(ma + 1), min_x(ma + 1), x(i)
89   DOUBLE PRECISION :: ran1, randnum, sig(ndat), u(mp,np),v(np,np),w(np)
90   DOUBLE PRECISION :: b(ma + 1), rtr(ma), rti(ma), chisq
91   DOUBLE PRECISION :: mock_x(val), mock_y(val), mock_y_at_l(ma)
92
93   OPEN (unit = 1, file = "mockdata.dat", status = 'unknown')
94   OPEN (unit = 2, file = "mockdata.p", status = 'unknown')
95
96   DO q = 1, ma
97      b(q+1) = a(q)/q      !Transfers from coefficients of p(x) to those of integral
98   END DO
99   b(1) = 0
100
```

```
101 DO q = ma + 1, 1, -1                !Calculates the variable 'area' - the
102     min_x(q) = b(q) * x(i)**(q-1)   !area under the polynomial between
103     max_x(q) = b(q) * x(1)**(q-1)   !x(1) and x(i) - i.e. the range of the
104 END DO                              !integral for the chosen x-value
105 area = SUM(max_x) - SUM(min_x)      !domain.
106
107 DO q = 1, val
108     randnum = ran1(idum)
109     b(1) = -(SUM(min_x) + randnum * area) !Generates a random y value between the value of the integral at x(1) and at x(i).
110     CALL zrhqr(b,ma,rtr,rti)  ! Finds roots of integral for given y value
111     DO l = 1, ma
112         IF(rti(l) == 0.) THEN  ! Only use the real roots
113             IF(rtr(l) .gt. MINVAL(x)) THEN        !Make sure the chosen root
114                 IF(rtr(l) .lt. MAXVAL(x)) THEN    !is in the domain used
115                     mock_x(q) = rtr(l)
116                 END IF
117             END IF
118         END IF
119     END DO
120 END DO
121
122 DO q = 1, val
123     DO l = 1, ma
124         mock_y_at_l(l) = a(l) * mock_x(q)**(l-1)
125     END DO
126     mock_y(q) = SUM(mock_y_at_l)
127     WRITE (1, '(2ES20.5)') mock_x(q), mock_y(q)
128 END DO
129
130 WRITE (2,*) 'plot␣\'                                  !Prints fitted
131 DO j = ma, 2, -1                                      !polynomial to
132     WRITE (2,*) a(j), '*␣x**', j-1, '+␣\'             !a '.p' file for
133 END DO                                                !plotting with
134 WRITE (2,*) a(1), 'title␣''svdfit'',␣"mockdata.dat"'  !gnuplot
135
136 CALL Kink(ma,a,mock_x,mock_y,val,sig,ndata,u,v,w,mp,np,chisq,funcs)
137
138 END SUBROUTINE RandReal
139
140 !-------------------------------------------------------------------------
141
142 SUBROUTINE Kink(ma,a,mock_x,mock_y,val,sig,ndata,u,v,w,mp,np,chisq,funcs)
143 !Generates a new set of mock data points with a kink at offset of 0.25
144
145 INTEGER :: val, q, ma, l, h, ndat, ndata, ios = 0
146 PARAMETER (ndat = 1000)
147 DOUBLE PRECISION :: a(ma), mock_x(val), mock_y(val), e(ndat), f(ndat)
148 DOUBLE PRECISION :: shift_x(val), shift_y(val), shift_y_at_l(ma), new_y(val)
149 INTEGER :: mp, np
150 DOUBLE PRECISION :: chisq,sig(ndat),u(mp,np),v(np,np),w(np)
151 EXTERNAL :: funcs
152
153 DOUBLE PRECISION :: yp1,ypn,ya2(ndat),x,y, der_abs(ndat)
154 INTEGER :: indx(ndat)
155 DOUBLE PRECISION :: xa(ndat), ya(ndat)
156
157 OPEN (unit = 1, file = "kink.dat", status = 'unknown')
158
159 DO q = 1, val                                         !
160     shift_x(q) = mock_x(q) + 0.25                     !Offsets mockdata
161     DO l = 1, ma                                      !along x-axis by 0.25
```

```
162         shift_y_at_l(l) = a(l) * shift_x(q)**(l-1)          !and then adds these
163       END DO                                                !new mock data points
164       shift_y(q) = SUM(shift_y_at_l)                        !to the poly fitted
165       new_y(q) = (mock_y(q) + 5.0*MINVAL(mock_y)) + shift_y(q) !to the previous ones.
166       WRITE (1, *) shift_x(q), new_y(q)                     !5*MINVAL(mock_y)
167   END DO                                                    !makes large kink.
168
169   DO q = 1, val                                 !
170       IF (mock_x(q) .lt. MINVAL(shift_x)) THEN  !Outputs original mockdata points
171           WRITE (1, *) mock_x(q), mock_y(q)     !for mock_x points less than the
172       END IF                                    !minimum shift_x value.
173   END DO                                        !
174
175   REWIND(1)
176
177   e = 0. ; f = 0. ; h = 0
178   DO WHILE (.TRUE.)  !Reads data until end of input file and puts it into arrays
179       h=h+1
180       READ (1, *, IOSTAT = ios) e(h), f(h)  !i.e. Read shift_x(q), new_y(q)
181       if (ios == 0) then ;
182       else if (ios == -1) then ;
183           h=h-1
184           exit ;
185       else if (ios > 0) then ;
186           h=h-1
187           cycle
188       end if
189   END DO
190
191   ndata = h
192
193   CALL indexx(ndata,e,indx)  !Creates array indx(1:ndata) whose elements are
194                              !indicies to the elements of e in chronological order
195
196   DO j=1,ndata            !
197       xa(j)=e(indx(j))    !Makes xa and ya equal to the ordered versions of
198       ya(j)=f(indx(j))    !e and f respectively.
199       write (*,*) xa(j), ya(j)
200   END DO
201   WRITE (*,*) MINVAL(xa), MAXVAL(xa), MINVAL(ya), MAXVAL(ya)
202
203   call pgbegin(0,'?',1,1)
204
205   call pgenv(REAL(MINVAL(xa)), REAL(MAXVAL(xa)), &
206         0., REAL(MAXVAL(ya)), 0, 0)
207
208   call pgpt(ndata, REAL(xa), REAL(ya), 1)
209
210   call pgend
211
212
213   !CALL PolyTest(xa,ya,sig,ndat,ndata,a,ma,u,v,w,mp,np,chisq,funcs)
214   CALL SplineTest(xa,ya,ndata,mock_x,val)
215   CALL GradTest(xa, ya, ndata, mock_x, val)
216   CALL AngleTest(xa, ya, ndata, mock_x, val)
217   CALL LegPrint(ma, a)
218
219   END SUBROUTINE Kink
220
221   !-----------------------------------------------------------------------
222
```

```
223 SUBROUTINE PolyTest(xa,ya,sig,ndat,ndata,a,ma,u,v,w,mp,np,chisq,funcs)
224 !Use to test the polynomials ability to find the kink
225
226 INTEGER :: ndat, ndata, ma, mp, np
227 DOUBLE PRECISION :: sig(ndat), a(ma), u(mp,np), v(np,np), w(np), chisq, x(ndata)
228 DOUBLE PRECISION :: y(ndata), xa(ndata), ya(ndata)
229 EXTERNAL :: funcs
230
231 x = xa; y = ya   !Don't want high precission here
232
233 OPEN (unit = 1, file = "kinkpoly.p", status = 'unknown')
234
235 a = 0.
236
237 CALL svdfit(x,y,sig,ndata,a,ma,u,v,w,mp,np,chisq,funcs) !SVD fitting program
238
239 WRITE (1,*) 'set xr [-1.0 : -0.2]'
240 WRITE (1,*) 'plot \'                                  !Prints fitted
241 DO j = ma, 2, -1                                      !polynomial to
242    WRITE (1,*) a(j), '*x**', j-1, '+ \'               !a '.p' file for
243 END DO                                                !plotting with gnuplot
244 WRITE (1,*) a(1), 'title ''svdfit'', "kink.dat"'
245
246 CALL LegPlot2(xa, ya, ndata)
247
248 END SUBROUTINE PolyTest
249
250 !-------------------------------------------------------------------------
251
252 SUBROUTINE SplineTest(xa,ya,ndata,mock_x,val)
253 !Use to test the spline functions ability to find the kink
254
255 INTEGER :: ndata, indx(ndata), val
256 DOUBLE PRECISION :: yp1, ypn, ya2(ndata), x, y, der_abs(ndata), a(ndata)
257 DOUBLE PRECISION :: b(ndata), mock_x(val), xa(ndata), ya(ndata)
258
259 a = xa; b = ya  !Don't want high precision here
260
261 yp1=1.1e30   !Makes 2nd derivative (ya2) equal to zero at either
262 ypn=1.1e30   !end of the spline-interpolated function
263 CALL spline_NR(a,b,ndata,yp1,ypn,ya2) !Finds 2nd der. of tabulated fn f(e)
264
265 der_abs = 0.                                          !
266 DO j = 1, ndata                                       !
267    IF (xa(j) .gt. -0.8) THEN                          !
268       IF(xa(j) .lt. -0.2) THEN                        !Find location and
269          der_abs(j) = ABS(ya2(j))                     !absolute value of
270       END IF                                          !the maximum
271    END IF                                             !second derivative
272 END DO                                                !
273 PRINT *, 'Max. abs. val. of 2nd der. is', MAXVAL(der_abs) !
274 PRINT *, 'This occurs at x =', xa(MAXLOC(der_abs) - 1)    !
275 PRINT *, 'Giving offset:', xa(MAXLOC(der_abs)) - MINVAL(mock_x)
276
277 OPEN (1, file= 'spline.dat', status = 'unknown')
278 DO j=1,1000
279    x=-0.9+j*0.7/1000.0 !x range and interval size for outputted spline data
280    CALL splint_NR(a,b,ya2,ndata,x,y)!Returns cubic-spline interpolated value y
281    WRITE(1,*) x,y
282 END DO
283 CLOSE (1)
```

```
284
285 OPEN (2, file = 'kinkspline.p', status = 'unknown')
286 WRITE (2,*) 'set xr [-1.0 : -0.2]'
287 WRITE (2,*) 'plot "kink.dat", "spline.dat" with lines'
288
289 END SUBROUTINE SplineTest
290
291 !-------------------------------------------------------------------------
292
293 SUBROUTINE GradTest(xa, ya, ndata, mock_x, val)
294 INTEGER :: ndata, j, val
295 DOUBLE PRECISION :: grad(ndata - 1), graddiff(ndata - 2), mock_x(val)
296 DOUBLE PRECISION :: xa(ndata), ya(ndata)
297
298 !This subroutine takes the gradient of the line joining each two data points
299 !and then compares it to the gradient between the left most of the two data
300 !points and the data point to its left. Where the greatest difference occurs,
301 !an 'edge' or sharp gradient discontinuity must exist in the data. This is
302 !very similar to the second derivative method but more easily tailored to the
303 !specific nature of the inputted data.
304
305 grad = 0.                                                    !Find
306 DO j = 1, ndata - 1                                          !gradient
307    IF (xa(j) .gt. -1.0) THEN                                 !between
308       IF (xa(j) .lt. -0.2) THEN                              !each two
309          grad(j) = (ya(j + 1) - ya(j))/(xa(j+1) - xa(j))     !adjacent
310       END IF                                                 !points
311    END IF                                                    !in the range
312 END DO                                                       !-1.0 < x < -0.2
313
314 graddiff = 0.                                               !Find difference
315 DO j = 1, ndata - 2                                          !between each
316    IF (grad(j) .ne. 0.) THEN                                 !two adjacent
317       IF (grad(j + 1) .ne. 0.) THEN                          !gradients so
318          graddiff(j) = grad(j + 1) - grad(j)                 !long as neither
319       END IF                                                 !of the two
320    END IF                                                    !gradients are
321 END DO                                                       !equal to zero.
322
323 !Note: absolute values are not used in either of the above loops as we are
324 !specifically looking for positive gradients and for the edge where the
325 !gradient goes from small to large with increasing x.
326
327 PRINT *, 'Largest gradient difference is at x =', xa(MAXLOC(graddiff) + 1)
328 PRINT *, 'With magnitude', MAXVAL(graddiff)
329 PRINT *, 'This gives an offset of', xa(MAXLOC(graddiff) + 2) - MINVAL(mock_x)
330
331 END SUBROUTINE GradTest
332
333 !-------------------------------------------------------------------------
334
335 SUBROUTINE AngleTest(xa, ya, ndata, mock_x, val)
336
337 !This subroutine is designed to find the angle between lines connecting
338 !adjacent data points. Starting from the smallest x value (say point 1), the
339 !line joining point 2 and point 3 is denoted length 'a', that joining point 1
340 !and point 2 is denoted length 'b' and that connecting points 1 and 3 is
341 !denoted length 'c,' thus setting up a triangle. The trigonometric rule
342 ! c = a + b - 2abcosC is then used to find angle C. The algorithm then shifts
343 !to concentrate on the triangle made by points 2, 3 and 4 and so on until the
344 !end of the data. The sharpest gradient change occurs where angle C is smallest.
```

```
345  !Note that only positive gradient changes from left to right are considered.
346
347  INTEGER :: ndata, j, val
348  DOUBLE PRECISION :: mock_x(val)
349  DOUBLE PRECISION :: xa(ndata), ya(ndata), grad(ndata-1)
350  DOUBLE PRECISION :: a(ndata-2), b(ndata-2), c(ndata-2), angle_c(ndata-2)
351
352  grad = 0.                                                    !Find
353  DO j = 1, ndata - 1                                          !gradient
354     IF (xa(j) .gt. -1.0) THEN                                 !between
355        IF (xa(j) .lt. -0.2) THEN                              !each two
356           grad(j) = (ya(j + 1) - ya(j))/(xa(j+1) - xa(j))     !adjacent
357        END IF                                                 !points
358     END IF                                                    !in the range
359  END DO
360
361  angle_c = 7.  !Makes angle_c larger than 2*pi for grad(j+1) < grad(j)
362  DO j = 1, ndata-2                                            !Populates
363     a(j) = SQRT((xa(j+2) - xa(j+1))**2 + (ya(j+2) - ya(j+1))**2)  !array
364     b(j) = SQRT((xa(j+1) - xa(j))**2 + (ya(j+1) - ya(j))**2)      !'angle_c'
365     c(j) = SQRT((xa(j+2) - xa(j))**2 + (ya(j+2) - ya(j))**2)      !with the
366     IF (grad(j+1) .gt. grad(j)) THEN                          !angles
367        angle_c(j) = ACOS(-(c(j)**2 - a(j)**2 - b(j)**2)/(2*a(j)*b(j)))  !between
368     END IF                                                    !each set
369  END DO                                                       !of points
370
371  PRINT *, 'Smallest angle occurs for x =', xa(MINLOC(angle_c) + 1)
372  PRINT *, 'With magnitude', MINVAL(angle_c), 'radians'
373  PRINT *, 'This recovers an offset of', xa(MINLOC(angle_c) + 2) - MINVAL(mock_x)
374
375  END SUBROUTINE AngleTest
376
377  !-------------------------------------------------------------------------------
378
379  SUBROUTINE LegPrint(ma, a) !For printing Legendre polynomials
380  implicit none
381  REAL*8 :: fac1, fac2, fac3, fac4
382  INTEGER :: ma, n, k, j
383  INTEGER :: ma_max,ma_used
384  PARAMETER (ma_max=100)
385  INTEGER :: LegEx(0:ma_max-1, 0:(ma_max/2))
386  DOUBLE PRECISION :: a(ma)
387  DOUBLE PRECISION :: LegCo(0:ma_max-1, 0:(ma_max/2)), PolyCo(ma_max)
388  CHARACTER(LEN=100), DIMENSION(0:ma_max-1, 0:(ma_max/2)) :: P
389  CHARACTER(LEN=100), DIMENSION(ma_max) :: PolyElement
390  CHARACTER(LEN=1000) :: Polynomial = ' '
391
392  COMMON /PASS/ PolyCo, ma_used
393
394  PolyCo = 0.
395
396  if (ma.gt.ma_max) then
397     write(*,*) 'ma too large' ; stop
398  end if
399  ma_used=ma
400
401  !||Calculate Legendre polynomial
402  !\/exponents and coefficients
403  DO n = 0, ma-1
404     DO k = 0, n/2
405        fac1 = 1; fac2 = 1; fac3 = 1; fac4 = 1
```

```
406         DO j = 1, 2*n - 2*k
407             fac1 = fac1 * j
408         END DO
409         DO j = 1, k
410             fac2 = fac2 * j
411         END DO
412         DO j = 1, n - k
413             fac3 = fac3 * j
414         END DO
415         DO j = 1, n - 2*k
416             fac4 = fac4 * j
417         END DO
418         LegCo(n,k) = a(n+1)*(((-1.d0)**k)*(fac1/((2.d0**n) * fac2 * fac3 * fac4)))
419         LegEx(n,k) = n - 2.d0*k
420         WRITE(P(n,k),*) LegCo(n,k), '*x**', LegEx(n,k)
421         PRINT *, TRIM(P(n,k))
422     END DO
423     PRINT *, ' '
424 END DO
425
426 DO j = 1, ma
427     DO n = 0, ma-1
428         DO k = 0, n/2
429             IF (LegEx(n,k) == j-1) THEN
430                 PolyCo(j) = PolyCo(j) + LegCo(n,k)
431             END IF
432         END DO
433     END DO
434 END DO
435
436 OPEN (unit = 1, file = "lffit2.p", status = 'unknown')
437 WRITE(1,*) 'plot \'
438
439 DO j = ma, 1, -1
440     IF (j .ne. 1) THEN
441         WRITE(1,*) PolyCo(j), '*x**', j-1, '+ \'
442         IF (PolyCo(j - 1) .gt. 0.d0) THEN
443             WRITE(PolyElement(j), *) PolyCo(j), '*x**', j-1, '+'
444         END IF
445         IF (PolyCo(j - 1) .lt. 0.d0) THEN
446             WRITE(PolyElement(j), *) PolyCo(j), '*x**', j-1
447         END IF
448     END IF
449     IF (j == 1) THEN
450         WRITE(1,*) PolyCo(j), 'title ''svdfit'', "lffit.dat"'
451         WRITE(PolyElement(j),*) PolyCo(j)
452     END IF
453     Polynomial = TRIM(Polynomial) // TRIM(PolyElement(j))
454 END DO
455
456 PRINT *, TRIM(Polynomial)
457
458
459 END SUBROUTINE LegPrint
460
461 !---------------------------------------------------------------------------
462
463 SUBROUTINE LegPlot2(xa, ya, ndata) !For plotting Legendre polynomials with pgplot
464 IMPLICIT NONE
465
466 INTEGER :: ndata, j
```

```
467  DOUBLE PRECISION :: xa(ndata), ya(ndata)
468  REAL :: xasp(ndata), yasp(ndata), Legendre_new, dummy
469  EXTERNAL :: Legendre_new
470
471  xasp = xa ; yasp = ya
472
473  dummy = Legendre_new(-0.5)
474
475  DO j= 1, ndata
476     PRINT *, j, xasp(j), yasp(j)
477  END DO
478
479  CALL pgbegin(0,'?',1,1)
480
481  CALL pgfunx(Legendre_new,100,-0.9,-0.2,0)
482
483  CALL pgpt(ndata, xasp, yasp, 228)
484
485  CALL pgend
486
487  END SUBROUTINE LegPlot2
488
489  !-------------------------------------------------------------------------
490
491  SUBROUTINE d2ydx2max(ma, a) !Finds the turning pts of the 2nd derivative of p(x)
492  INTEGER :: ma, j
493  DOUBLE PRECISION :: a(ma), c(ma-2), d(ma-3), rtr(ma-3), rti(ma-3)
494
495  DO j = 1, ma-2
496     c(j) = a(j+2) * (j+1) * (j) !Finds the coefficients for d2ydx2
497  END DO
498
499  DO j = 1, ma-3
500     d(j) = c(j+1) * j  !Finds the coefficients for the 3rd derivative (d3ydx3)
501  END DO
502
503  CALL zrhqr(d,1,rtr,rti)  !Finds the roots for the maximum rate of change of the
504  DO j=1, ma - 3           !gradient (d3ydx3 = 0) and d2ydx2's magnitude there.
505     IF(rti(j) == 0) THEN     !Include only real roots
506        PRINT *, 'turning point at x=', rtr(j)
507     END IF
508  END DO
509
510  END SUBROUTINE d2ydx2max
511
512  !-------------------------------------------------------------------------
513  !----------------------------Libpress algorithms--------------------------
```

Program: RGBPeakFinder6.f95

Creation Date: 7 December 2009

Relevant Section: 2.2

Notes: This is another RGB finder which combines elements of 'edge-finding' and model fitting. Stars in a small region around Andromeda I are read in and a smoothed 'Luminosity Probability Distribution' is produced from the individual stellar magnitudes via a Gaussian smoothing of the luminosity function. The second derivative of this distribution is produced with the peaks denoting inflection points in the gradient of the smoothed luminosity function. See Figs. 2.1 and 2.2.

```
1  MODULE Global3 !Define all
2  IMPLICIT NONE  !Variables
3
4  INTEGER :: ndata_max, ndata, ndata_t, i, j, k, n, ios, idum = 0, randnum
5  INTEGER :: ndata2, ndata_sub, div_per_mag = 100
6  PARAMETER (ndata_max = 100000)
7  REAL :: xmin, xmax, min_mag = 19.5d0, max_mag = 21.5d0
8  REAL*8 :: temp_x(ndata_max), temp_y(ndata_max), temp_e(ndata_max), dummy
9  REAL*8 :: mag(10000), phi(10000), mag2(10000), phi2(10000)
10 REAL*8 :: mag3(10000), phi3(10000), phi4(10000), phi4_max
11
12 END MODULE Global3
13
14 !-------------------------------------------------------------------------
15
16 PROGRAM RGBPeakFinder6 !Finds greatest peak of (d2phi/dm2)/(phi) between min_mag and max_mag.
17 USE GLOBAL3
18 IMPLICIT NONE
19
20 INTEGER :: check(100000)
21 REAL*8 :: xh1(ndata_max), yh1(ndata_max), eh1(ndata_max)
22
23 temp_x = 0.d0 ; temp_y = 20.d0
24
25 OPEN (unit = 1, file = './m31_fields_stellar/ANDI_box_small.dat', status = 'old')
26 i = 0 ; ios = 0
27 DO WHILE (.TRUE.)  !Reads data until end of input file and puts it into arrays
28    i=i+1
29    READ (1, *, IOSTAT = ios) temp_x(i), temp_y(i) !x: g magnitude
30                                                   !y: i magnitude
31
32    if (ios == 0) then ;
33    else if (ios == -1) then ;
34       i=i-1
35       exit ;
36    else if (ios > 0) then ;
37       i=i-1
38       cycle
39    end if
40    IF (temp_x(i) == 0. .or. temp_y(i) == 0.) THEN
41       i=i-1
42       cycle
43    END IF
```

```
44  END DO
45
46  ndata = i
47  PRINT *, "Number of sources =", ndata
48
49  DO j = 1, ndata
50     temp_x(j) = temp_x(j) - temp_y(j)
51  END DO
52
53  CALL CutPlot
54  CALL Smooth
55
56  END PROGRAM RGBPeakFinder6
57
58  !-----------------------------------------------------------------------
59
60  SUBROUTINE CutPlot  !Produce colour cuts
61  USE Global3
62  IMPLICIT NONE
63
64  REAL :: div, var(1000)
65  REAL :: uppercut(1000), lowercut(1000)
66
67  div = 2*(INT(MAXVAL(temp_x)) - INT(MINVAL(temp_x)))/1000.
68
69  DO j = 1, 1000
70     var(j) = INT(MINVAL(temp_x)) + div*j
71     uppercut(j) = 24.5 - 3*(var(j))
72     lowercut(j) = 27.0 - 3*(var(j))
73  END DO
74
75  !-----------------------------------------Upper and Lower Cuts plot
76
77  CALL pgbegin(0,'temp1.ps/CPS',1,1)
78
79  CALL pgenv(MINVAL(REAL(temp_x)), MAXVAL(REAL(temp_x)), MAXVAL(REAL(temp_y)), MINVAL(REAL(temp_y)), 0, 0)
80  CALL pgpt(ndata, REAL(temp_x), REAL(temp_y), 1)
81  CALL pgsci(2)
82  CALL pgline(1000, var, uppercut)
83  CALL pgline(1000, var, lowercut)
84  CALL pgsci(1)
85  CALL pglab('(g - i)\d0\u', 'i\d0\u', '')
86
87  CALL pgend
88  !---------------------------------------
89
90  END SUBROUTINE CutPlot
91
92  !-----------------------------------------------------------------------
93
94  SUBROUTINE Smooth   !Apply Gaussian smoothing to LF and
95  USE Global3         !find inflection points.
96  IMPLICIT NONE
97
98  REAL*8 :: x(ndata), y(ndata), xa(ndata), ya(ndata), pi = 2*ACOS(0.d0)
99  REAL*8 :: e(ndata), err(ndata), ave_phi, ave_mag, ave_phi4
100 REAL*8 :: xa_sel(ndata), ya_sel(ndata), err_sel(ndata)
101 REAL :: yasp(ndata), phisp(10000), magsp(10000)
102 INTEGER :: indx(ndata), place, place2, place3, counts, denominator
103
104 DO j = 1, ndata
```

```
105     x(j) = temp_x(j)
106     y(j) = temp_y(j)
107  END DO
108
109  CALL indexx(ndata,y,indx)  !Creates array indx(1:ndata) whose elements are
110                             !indicies to the elements of y in chronological order
111
112  DO j=1,ndata
113     xa(j)=x(indx(j))     !Makes xa, ya & err equal to the ordered versions of
114     ya(j)=y(indx(j))     !x, y & e respectively.
115     err(j) = 0.1d0
116  END DO
117
118  counts = 0 ; xa_sel = 0.d0 ; ya_sel = 0.d0
119  DO j=1, ndata
120     IF (ya(j) .gt. 24.5 - 3*(xa(j))) THEN      !Throw away stars
121        IF (ya(j) .lt. 27.0 - 3*(xa(j))) THEN !outside of colour cut
122           counts = counts + 1   !counts is the total number of accepted stars
123           xa_sel(counts) = xa(j)
124           ya_sel(counts) = ya(j)
125           err_sel(counts) = err(j)
126        END IF
127     END IF
128  END DO
129
130  !--------------------------------------------Accepted Data Points Plot
131  CALL pgbegin(0,'?',1,1)
132
133  CALL pgenv(MINVAL(REAL(temp_x)), MAXVAL(REAL(temp_x)), MAXVAL(REAL(temp_y)), MINVAL(REAL(temp_y)), 0, 0)
134  CALL pgpt(counts, REAL(xa_sel), REAL(ya_sel), 1)
135  CALL pglab('(g - i)\d0\u', 'i\d0\u', '')
136
137  CALL pgend
138  !--------------------------------------------
139
140
141  !Produce Luminosity Probability Distribution (LPD)
142  place = 0 ; phi = 0.d0 ; mag = 0.d0
143  DO j = 100*(INT(MINVAL(ya)) - 1), 100*(INT(MAXVAL(ya)) + 1), (100/div_per_mag)
144     place = place + 1
145     mag(place) = j/100.d0
146     DO k = 1, counts                  !Perform Gaussian smoothing at magnitude by summing contributions
147        phi(place) = phi(place) + &    !of each star represented by a normalized Gaussian
148        (1.d0/(SQRT(2.d0 * pi) * err_sel(k))) * EXP(-((ya_sel(k) - j/100.d0)**2.d0)/ (2.d0*(err_sel(k)**2.d0)))
149     END DO
150  END DO
151
152  !Produce derivative of LPD
153  place2 = 0 ; phi2 = 0.d0 ; mag2 = 0.d0
154  DO j = 100*(INT(MINVAL(ya)) - 1), 100*(INT(MAXVAL(ya)) + 1), (100/div_per_mag)
155     place2 = place2 + 1
156     mag2(place2) = j/100.d0
157     DO k = 1, counts
158        phi2(place2) = phi2(place2) + &
159        (1.d0/(SQRT(2.d0 * pi) * (err_sel(k))*3)) * EXP(-((ya_sel(k) - j/100.d0)**2.d0)/ (2.d0*(err_sel(k)**2.d0))) * (ya_sel(k) - (j
               /100.d0))
160     END DO
161  END DO
162
163  !Produce second derivative of LPD
164  place3 = 0 ; phi3 = 0.d0 ; mag3 = 0.d0
```

```
DO j = 100*(INT(MINVAL(ya)) - 1), 100*(INT(MAXVAL(ya)) + 1), (100/div_per_mag)
    place3 = place3 + 1
    mag3(place3) = j/100.d0
    DO k = 1, counts
       phi3(place3) = phi3(place3) + &
       (1.d0/(SQRT(2.d0 * pi) * (err_sel(k))*5)) * EXP(-((ya_sel(k) - j/100.d0)**2.d0)/ (2.d0*(err_sel(k)**2.d0))) * (ya_sel(k)**2 - 2*
            ya_sel(k)*(j/100.d0) - err_sel(k)**2 + (j/100.d0)**2)
    END DO
END DO



phi4 = (phi3/(ABS(phi)+0.1))  !Use 2nd der. of LPD divided by LPD to locate potential TRGBs

ndata2 = div_per_mag*(INT(MAXVAL(ya)) - INT(MINVAL(ya)) + 2) + 1
ndata_sub = (div_per_mag * (max_mag - min_mag)) + 1

ave_phi = SUM(phi)/ ndata2
ave_mag = SUM(mag)/ ndata2
ave_phi4 = 0. ; denominator = 0 ; phi4_max = 0.d0
DO j = 1, ndata2                          !
    IF (mag(j) .ge. min_mag) THEN         !
       IF (mag(j) .le. max_mag) THEN      !
          ave_phi4 = ave_phi4 + phi4(j)   !Find maximum value of array phi4 and
          denominator = denominator + 1   !the averagage value of array phi4 in
          IF (phi4(j) .gt. phi4_max) THEN !the region where the TRGB could
             phi4_max = phi4(j)           !feasibly be located. These can then
          END IF                          !be used for scaling graphs and
       END IF                             !determining the strenghts of possible
    END IF                                !TRGBs.
END DO                                    !
ave_phi4 = ave_phi4/ denominator          !

yasp = ya ; magsp = mag
phisp = phi*(phi4_max/ MAXVAL(phi)) !Scale REAL(phi) for plotting with phi4

DO j = ndata2 + 1, 10000
    phi(j) = ave_phi      !Since SIZE(phi) = SIZE(phisp) = SIZE(mag) = SIZE(magsp)
    mag(j) = ave_mag      !                = 10000 .not. ndata2
    phisp(j) = 0.         !it is necessary to make all array elements outside of
    magsp(j) = ave_mag    !ndata2 equal to some intermediate value so that the
END DO                    !minval & maxval functions are still useable.

xmin = INT(MINVAL(yasp)) - 1. ; xmax = INT(MAXVAL(yasp)) + 1.

CALL TurningPoints

!-----------------------------------------!Main Plot (i.e. smoothed luminosity
                                          !function with inflection points)
CALL pgbegin(0,'temp2.ps/CPS',1,1)

CALL pgenv(18., 26., MINVAL(phisp), REAL(1.5*phi4_max), 0, 0)
CALL pgsci(1)
CALL pgline(ndata2, magsp, phisp)
CALL pgsci(2)
CALL pgline(ndata2, REAL(mag3), REAL(phi4))
CALL pgsci(4)
CALL pgsch(4.0)
CALL pgpt(1, 20.77, 0.001, 2264)
CALL pgsch(1.0)
CALL pgsci(1)
```

```
225  CALL pglab('i\d0\u', 'relative probability', '')
226  CALL pgend
227  !-------------------------------------------
228
229  END SUBROUTINE Smooth
230
231  !-----------------------------------------------------------------------------
232
233  SUBROUTINE TurningPoints  !Prints location of potential RGB tips (in magnitudes)
234  USE Global3               !and assigns to them a stength, based on the change in
235  IMPLICIT NONE             !LF slope at that magnitude
236
237  INTEGER :: TRGB_found = 0
238
239  PRINT *, "-------------------------------------------------"
240  DO j = 1, ndata2
241     IF (mag(j) .ge. min_mag) THEN
242        IF (mag(j) .le. max_mag) THEN
243           IF (phi4(j) .gt. phi4(j-1) .and. phi4(j) .gt. phi4(j+1)) THEN
244              PRINT *, "Potential TRGB at", mag(j), &
245                       "Strength =", REAL(phi4(j)/ phi4_max)
246              TRGB_found = 1
247           END IF
248        END IF
249     END IF
250  END DO
251
252  IF (TRGB_found == 0) THEN
253     PRINT *, "No TRGB could be located."
254  END IF
255  PRINT *, "-------------------------------------------------"
256
257  END SUBROUTINE TurningPoints
258
259  !-----------------------------------------------------------------------------
260
261  SUBROUTINE RandSplit(check)
262  USE Global3
263  IMPLICIT NONE
264
265  !Subroutine creates a randomized index to the read-in data set. It is called
266  !by subroutine 'TestSeparate' to split the data in two halves and process them.
267
268  INTEGER :: check(100000), count, num = 0
269  REAL*8 :: ran1
270
271  check(100000) = 0 ; j = 0              !Insure array check has all elements 0
272  DO WHILE (j .lt. ndata_t - 1)          !
273     j = j + 1                           !
274     randnum = INT(ndata_t*ran1(idum))   !
275     count = 0                           !
276     DO k = 1, j                         !
277        IF (randnum .eq. check(k)) THEN  !
278           count = count + 1             !
279        END IF                           !Loop generates randomised index with
280     END DO                              !number of entries equal to ndata. All
281     IF (count == 0) THEN                !entries are unique integers from 1 to
282        check(j) = randnum ;             !ndata. Index is outputted to 'check'
283     END IF                              !
284     IF (count .ne. 0) THEN              !
285        j = j - 1                        !
```

```
286         cycle                              !
287     END IF                                 !
288 END DO                                     !
289 check(ndata_t) = ndata_t                   !Final element of 'check' is ndata
290
291 DO j = 1, ndata_t                         ! Check that
292     DO k = 1, ndata_t                      ! all integer
293         IF(check(k) == j) THEN             ! values between
294             num = num + 1                  ! 1 and ndata
295         END IF                             ! can be
296     END DO                                 ! found in the
297 END DO                                     ! array 'check'
298
299 PRINT *, "number of unique integers =", num
300
301 END SUBROUTINE RandSplit
302
303 !-------------------------------------------------------------------------------
304 !----------------------------Libpress Algorithms--------------------------------
```

Program: spikes.f95

Creation Date: 1 February 2011

Relevant Section: 2.3

Notes: I created this program to illustrate the precise way in which the posterior probability distribution of a parameter is produced via maximum likelihood model fitting. An artificial luminosity function is created from a step function, where the user specifies the number of stars to be produced as well as the position of the step and the relative proportions of the 'background' and 'signal' components. Various types of priors can then be applied to the resulting posterior distributions. Fig. 2.4 illustrates the way in which the posterior distributions in the tip position are created. Figs. 2.3, 2.5 and 2.6 were created using this code.

```
1  MODULE Global  !Define all
2  IMPLICIT NONE  !variables
3
4  INTEGER :: i, j, indx(20001)
5  REAL*8 :: like(1000), posterior(1000,2), likex1(1000,20000), data(20001), rand_num, ylim = 0.d0, x(1000)
6  REAL*8 :: hist(101,2) = 0.d0, temp(101,2), ord_data(20001)
7
8  INTEGER :: ndata = 1000                    !<= Enter number of stars
9  REAL*8 :: TRGB ; PARAMETER (TRGB = 0.4d0)  !<= Enter tip position 0 < TRGB < 1
10 REAL*8 :: f = 0.3d0                        !<= Enter fraction background
11
12 !----------For No Prior on tip------------
13 REAL*8 :: u(1000)
14
15 !--------For Gaussian Prior on tip--------
16 REAL*8 :: tip_expt = TRGB       !Tip magnitude expected for structure
17 REAL*8 :: gauss_hwhm = 0.25d0   !Magnitudes on either side of expected tip magnitude to explore
18 REAL*8 :: gauss_expo = 4.d0     !Sharpness of edges of Gaussian prior profile
19 REAL*8 :: prior_sig, g(1000)
20
21 !----
22 REAL*8 :: gap, maxgap, counts, d(1000)
23
24 REAL :: BG_counts, RGB_counts
25
26 END MODULE Global
27
28 !-----------------------------------------------------------------------------------------------------------------------
29
30 PROGRAM spikes   !
31 USE Global       !Main Program
32 IMPLICIT NONE   !
33
34 CALL random_seed    ! Insures stars are at different magnitudes each time
35
36 DO i = 1, ndata                                  !
37    CALL random_number(rand_num)                  !
38    IF (rand_num .gt. TRGB * f) THEN              !
39       CALL random_number(rand_num)               !Draw ndata stars at
40       rand_num = TRGB + (1.00d0 - TRGB) * rand_num!random from a luminosity
41    ELSE                                          !function with tip at TRGB
```

```
42          CALL random_number(rand_num)                !and background height = f.
43          rand_num = TRGB * rand_num                  !
44       END IF                                         !
45       rand_num = NINT(rand_num * 100.d0)             !
46       data(i) = rand_num/ 100.d0                     !
47    END DO                                            !
48
49    DO i = 1, 101
50       hist(i,1) = REAL(i-1)/100.d0
51       DO j = 1, ndata
52          IF (data(j) .eq. hist(i,1)) THEN
53             hist(i,2) = hist(i,2) + 1.d0
54          END IF
55       END DO
56    END DO
57
58    hist(101,2) = hist(101,2) * 2.e0  !Account for the bin width of the last bin
59
60    BG_counts = (REAL(ndata) * REAL(f))/100.e0 !Function height before step
61    RGB_counts = (REAL(ndata) * REAL(1.d0 - f))/(REAL(1.d0-TRGB) * 100.e0)
62    !/\ Function height
63    !|| after step
64
65    !---------------Plot histogram of data points---------------
66
67    CALL pgbegin(0,'tempLF.ps/CPS',1,1)
68
69    CALL pgenv(0.0, 1.0, 0., 1.1*MAXVAL(REAL(hist(:,2))), 0, 0)
70    CALL pgbin (101, REAL(hist(:,1)), REAL(hist(:,2)), .true.)
71    CALL pglab('star_magnitude', 'counts', '')
72
73    !||Plot model
74    !\/over LF
75    CALL pgsci(2)
76    CALL pgsls(3)
77    CALL pgslw(5)
78    CALL pgline(2, (/0.0e0, REAL(TRGB)/), (/ BG_counts, BG_counts /))
79    CALL pgline(2, (/REAL(TRGB), 1.e0/), (/ BG_counts + RGB_counts, BG_counts + RGB_counts /))
80    CALL pgline(2, (/REAL(TRGB), REAL(TRGB)/), (/ BG_counts, BG_counts + RGB_counts /))
81    CALL pgslw(1)
82    CALL pgsls(1)
83    CALL pgsci(1)
84    !/\
85    !||
86
87    CALL pgend
88    !-------------------------------------------------------------
89
90    IF (ndata .ge. NINT(1/f)) THEN
91       !CALL remove
92    END IF
93
94    !---------------Plot histogram of data points---------------
95    CALL pgbegin(0,'?',1,1)
96
97    CALL pgenv(0.0, 1.0, 0., 1.1*MAXVAL(REAL(hist(:,2))), 0, 0)
98    CALL pgbin (101, REAL(hist(:,1)), REAL(hist(:,2)), .true.)
99    CALL pglab('star_magnitude', 'counts', 'BG-removed_Luminosity_Function')
100
101   CALL pgend
102   !-------------------------------------------------------------
```

```
103
104 CALL u_prior
105 !CALL d_prior        ! Choose prior here, for no prior -> CALL u_prior
106 !CALL g_prior
107
108 like = 1.d0                          !Set initial values of likelihood array elements to 1
109 DO i = 1, ndata                      !For each star...
110    DO j = 1, 1000                    !For TRGB set at bin j...
111       IF (data(i).gt. REAL(j)/1000.d0) THEN                                    !
112          like(j) = like(j) * (f + (1.d0 - f)*(1.d0/(REAL(1001-j)/1000.d0)))    !Multiply the current likelihood value for
113          likex1(j,i) = (f + (1.d0 - f)*(1.d0/(REAL(1001-j)/1000.d0)))          !bin j [like(j)], by the likelihood for
114       ELSE                                                                     !star i. For the purpose of plotting likelihoods
115          like(j) = like(j) * f                                                 !for the TRGB being at each j for each individual
116          likex1(j,i) = f                                                       !star i, likex1(j,i) is also calculated.
117       END IF                                                                   !
118    END DO
119    like = like * u !* d * g                          !Multiply 'like' by chosen prior function
120    likex1(:,i) = likex1(:,i) * u !* d * g            !Multiply 'likex1(:,i)' by chosen prior function
121    like = like/ SUM(like)                      !Normalize 'like'
122    likex1(:,i) = likex1(:,i)/ SUM(likex1(:,i))      !Normalize 'likex1(:,i)'
123 END DO
124
125 DO j = 1, 1000                        !
126    posterior(j,1) = REAL(j)/1000.d0    !Build final likelihood
127    posterior(j,2) = like(j)            !distribution ready for plotting
128 END DO                                !
129
130
131 !----------------Plot Individual Likelihoods-----------------
132
133 CALL pgbegin(0,'ind_like.ps/CPS',1,1)
134
135 DO i = 1, ndata          !<---------------------------------!
136    IF (MAXVAL(likex1(:,i)) .gt. ylim) THEN                  !
137       ylim = MAXVAL(likex1(:,i))                            !
138    END IF                                                   !Change stars plotted here
139 END DO                                                     !
140 CALL pgenv(0., 1., 0., 1.1 * REAL(ylim), 0, 0)            !
141 DO i = 1, ndata          !<---------------------------------!
142    CALL pgsci(i+1)
143    CALL pgbin (1000, REAL(posterior(:,1)), REAL(likex1(:,i)), .true.)
144 END DO
145 CALL pgsci(1)
146 CALL pglab('Proposed_tip_magnitude', 'Probability', '')
147
148 CALL pgend
149
150 !----------------Plot Posterior Distribution-----------------
151
152 CALL pgbegin(0,'post_dis.ps/CPS',1,1)
153
154 CALL pgenv(0.0, 1.0, 0., 1.1*MAXVAL(REAL(posterior(:,2))), 0, 0)
155 CALL pgbin (1000, REAL(posterior(:,1)), REAL(posterior(:,2)), .true.)
156 CALL pglab('Proposed_tip_magnitude', 'Probability', '')
157
158 CALL pgend
159
160 END PROGRAM spikes
161
162 !-----------------------------------------------------------------------------------------------------------------------
163
```

```
164 SUBROUTINE u_prior    !
165 USE Global            !Generates Uniform prior function -> i.e. u = 1
166 IMPLICIT NONE         !
167
168 DO i = 1, 1000
169    x(i) = REAL(i) / 1000.d0
170 END DO
171
172 !---------------No Prior-----------------
173
174 DO i = 1, 1000
175    u(i) = 1.d0
176 END DO
177
178 CALL pgbegin(0,'?',1,1)
179
180 CALL pgenv(0., 1., 0., 1.1, 0, 0)
181 CALL pgbin (1000, REAL(x), REAL(u), .true.)
182 CALL pglab('x', 'y', 'Uniform Prior Applied')
183
184 CALL pgend
185
186 END SUBROUTINE u_prior
187
188 !-------------------------------------------------------------------------------------------------------------------------
189
190 SUBROUTINE g_prior !
191 USE Global          !Generates Gaussian prior function -> parameters of Gaussian changed in MODULE Global.
192 IMPLICIT NONE       !
193
194 DO i = 1, 1000
195    x(i) = REAL(i) / 1000.d0
196 END DO
197
198 !------------Gaussian Prior---------------
199 prior_sig = ABS(gauss_hwhm ** (0.5d0 * gauss_expo))
200
201 DO i = 1, 1000
202    g(i) = exp((-(REAL(x(i)) - tip_expt)**gauss_expo)/(2.d0*(prior_sig)**2.d0))
203 END DO
204
205 CALL pgbegin(0,'?',1,1)
206
207 CALL pgenv(0., 1., 0., 1.1, 0, 0)
208 CALL pgbin (1000, REAL(x), REAL(g), .true.)
209 CALL pglab('magnitude', 'weight', 'Gaussian Prior Applied')
210
211 CALL pgend
212
213 END SUBROUTINE g_prior
214
215 !-------------------------------------------------------------------------------------------------------------------------
216
217 SUBROUTINE d_prior  !For applying a density prior. A window is chosen with width equal to the greatest separation between any
218 USE Global          !2 data points in the array "data." The window then slides across the LF at 1-bin increments and the number
219 IMPLICIT NONE       !of stars lying within the window is placed in the bin corresponding to either the LHS, RHS
220                     !or middle of the window.
221 DO i = 1, 1000
222    x(i) = REAL(i) / 1000.d0
223 END DO
224
```

```
225  CALL indexx(ndata, REAL(data), indx)
226
227  DO i = 1, ndata
228      ord_data(i) = data(indx(i))
229  END DO
230
231  maxgap = 0.d0
232  DO i = 2, ndata
233      gap = ord_data(i) - ord_data(i - 1)
234      IF (gap .gt. maxgap) THEN
235          maxgap = gap
236      END IF
237  END DO
238
239  maxgap = NINT(maxgap*100.d0)
240
241  !-----Density applied to left edge of window
242  DO i = 1, 101 - NINT(maxgap)
243      counts = 0.d0
244      DO j = i, i + NINT(maxgap)
245          counts = counts + hist(j,2)
246      END DO
247      d(10*i - 9:10*i) = counts
248      IF (i .eq. 101 - NINT(maxgap)) THEN
249          d(10*i:1000) = counts
250      END IF
251  END DO
252
253  d = d / (MAXVAL(d))
254
255  !-------Plot density prior--------
256  CALL pgbegin(0,'?',1,1)
257
258  CALL pgenv(0., 1., 0., 1.1 * MAXVAL(REAL(d)), 0, 0)
259  CALL pgbin (1000, REAL(x), REAL(d), .true.)
260  CALL pglab('magnitude', 'weight', 'Density␣Prior␣Applied')
261
262  CALL pgend
263
264  END SUBROUTINE d_prior
265
266  !-----------------------------------------------------------------------------------------------------------------------------------
267
268  SUBROUTINE remove !Remove f*ndata data points
269  USE Global        !spread randomly over magnitude
270  IMPLICIT NONE     !space from data array.
271
272  i = 0
273  DO
274      CALL random_number(rand_num)
275      rand_num = NINT(rand_num * 100) + 1.d0
276      IF (hist(NINT(rand_num), 2) .ge. 1.d0) THEN
277          i = i+1
278          hist(NINT(rand_num), 2) = hist(NINT(rand_num), 2) - 1.d0
279      END IF
280      IF (i .ge. NINT(f * ndata)) THEN
281          exit;
282      END IF
283  END DO
284
285  temp = hist
```

```
286
287   ndata = ndata - (INT(f * ndata) + 1)
288
289   data = 0.d0
290   j = 0
291   DO i = 1, 101
292      DO WHILE (temp(i,2) .gt. 0.d0)
293         j = j + 1
294         data(j) = temp(i,1)
295         temp(i,2) = temp(i,2) - 1.d0
296      END DO
297   END DO
298
299   END SUBROUTINE remove
300
301   !-----------------------------------------------------------------------------------------------------------------------
```

B

Chapter Three Programs

Program: BayesianTRGB_ANDI.f95

Creation Date: 23 September 2010 (first version 6 Mar 2010)

Relevant Sections: 2.4, 3

Notes: This program lies at the heart of the material presented in Paper I (Ch. 3) and hence is described there in much more detail. Note also that §2 of Paper II (presented in Ch. 4) also provides a useful summary as to its workings. The program is an example of my original TRGB finding algorithm. At this stage I had not yet generalized the code so that the parameters for different objects could be fed in, and hence each object had its own separate code - that shown here is for Andromeda I. In summary, a circular field is taken centered on user-specified coordinates and all stars within that field are then plotted on a Colour-Magnitude Diagram. The user then provides the coordinates for the corners of a polygon to be used as a colour-cut to isolate the stars of the object's Red Giant Branch. The colour-cut should extend at least half a magnitude brightward of the estimated TRGB magnitude to give a reasonable portion of pure background luminosity function (LF) for the algorithm to fit, and it should span an equal colour range as a function of magnitude so as not to distort the LF. The user must also provide a 'background field' from which the algorithm generates a LF which it then fits with a polynomial to give the functional form of the background component of the model LF. The same colour-cut is imposed on the background field as was chosen for the object or 'signal' field. By calculating the number of stars in both the signal and background fields and dividing by their respective areas, the expected ratio of the two components in the signal field's LF is determined and the components are scaled accordingly in the model LF generated. An MCMC algorithm is then used to find the parameters of the model which best fit the data and posterior distributions of these parameters are plotted. The object's distance posterior distribution can then be determined by sampling the distribution in the tip magnitude along with those for the uncertainty in the absolute magnitude of the tip and the extinction. This is done using another, purpose-written program, a version of which can be found in Appendix C ('Multi_MCMC_Result_Plotter.f95').

```
1  MODULE Global !Defines all variables used by BayesianTRGB
2  IMPLICIT NONE
3
4  !----------------General Program Parameters-----------------
5  INTEGER :: i, j, k, l, eval, idum = -9999, it, nit
6  INTEGER :: ndata_max, nsamples, binspm, nbins, cmod_nbins, ghw, mm, ios
```

```
7   PARAMETER (ndata_max = 20000000, nsamples = 100)
8   PARAMETER (binspm = 100)
9   PARAMETER (nbins = 8*binspm + 1)
10  PARAMETER (nit = 3000000)
11  INTEGER :: ndata, ndata2
12  INTEGER :: d1, d2, d3, d4
13  REAL*8 :: blim, flim, pi
14  PARAMETER (blim = 19.5d0)
15  PARAMETER (flim = 23.5d0)
16  PARAMETER (pi = ACOS(-1.e0))
17  INTEGER :: blimBins = INT(REAL((blim - 18.d0) * binspm)) + 1
18  INTEGER :: flimBins = INT(REAL((flim - 18.d0) * binspm)) + 1
19  REAL*8 :: randnum1, randnum2, randnum3, r1, r2, spotR, hb = 0.005d0
20  REAL*8 :: model(nbins,2), cmodel(nbins,2), magnitude(ndata_max)
21  REAL*8 :: histo_fine(nbins,2), histo_coarse(INT(0.25*(nbins-1.d0)) + 1,2)
22  REAL*8 :: data(ndata_max), cumulative_cmodel(nbins,2), f, f_hold, bfm(nbins)
23  REAL*8 :: mag_tip, mag, mag_cutoff = 24.e0, a
24  REAL*8 :: area, area2
25  REAL*8 :: modelnoise(nbins,2), noise(nbins) = 0.d0
26  REAL*8 :: kernel(nbins,2) = 0.e0, scale, uplim, lowlim, gx
27  REAL*8 :: temp(nbins,2) = 0.e0, t
28  REAL*8 :: logL, prob, LikeA, LikeB
29  REAL*8 :: tip(nsamples), tip_ord(nsamples), maxlogL(nsamples) = -999999999999.
30  REAL*8 :: tip_rec, tip_offset, tip_psigma, tip_msigma, Toffset_kpc, Tsigma_kpc
31  REAL*8 :: f_offset, tip_kpc, kpc_perr, kpc_merr, f_sigma, a_offset, a_sigma
32  REAL*8 :: f_rec, a_rec, tip_counts, f_counts, a_counts
33  REAL*8 :: tipminsig, tiplusig, fminsig, fplusig, aminsig, aplusig
34  REAL*8 :: mcounts, pcounts
35  REAL*8 :: x1(nit), x2(nit), x3(nit), p(3), time(nit), r
36  REAL*8 :: post_y1(10*(nbins-1)+1) = 0.d0, post_x1(10*(nbins-1)+1), mlim
37  REAL*8 :: d_blim, bg_blim, d_flim, bg_flim
38  REAL*8 :: post_y2(nbins) = 0.d0, post_x2(nbins)
39  REAL*8 :: post_y3(2*nbins - 1) = 0.d0, post_x3(2*nbins - 1)
40  REAL*8 :: PPD_peak, Best_Combo(6)
41  CHARACTER :: argv*10, field*30, ch1*9, ch2*9, ch3*9, ch4*9, ch5*9, string*60
42
43  !-----------------For reading in PAndAS data-----------------
44  INTEGER :: iCCDt, clsg, clsi, ifieldt, iacc_t
45  REAL*4 :: xgt, ygt, g, dg, im, dim, xki_t, eta_t, FeH_phot_t, diff_tip_t, E_BV_t
46  REAL*8 :: ra_t, de_t
47  REAL*4 :: dummy
48
49  REAL*4 :: mag_g(ndata_max), mag_i(ndata_max), xki(ndata_max), eta(ndata_max)
50  REAL*4 :: g_min_i(ndata_max), mag_i_poly(ndata_max), g_min_i_poly(ndata_max)
51  REAL*4 :: gmi
52
53  !--Additional parameters for calculating background stats---
54  INTEGER :: bg_ndata, bg_ndata2, bg_ndata3
55  REAL*4 :: bg_mag_g(ndata_max), bg_mag_i(ndata_max), bg_xki(ndata_max), bg_eta(ndata_max)
56  REAL*4 :: bg_g_min_i(ndata_max), bg_mag_i_poly(ndata_max), bg_g_min_i_poly(ndata_max)
57  REAL*4 :: bg_gmi
58  REAL*8 :: bg_data(ndata_max)
59
60    !--SVD fitting of background--
61    INTEGER ma, mp, np, ndat
62    PARAMETER (ndat = INT(0.25*(nbins-1.d0)) + 1)
63    PARAMETER (np = 8)
64    PARAMETER (mp = ndat)
65    PARAMETER (ma = np)
66    REAL :: chisq, ay(ma), sig(ndat), u(mp,np), v(np,np), w(np), xa(ndat), ya(ndat)
67    REAL :: xt(ndat), yt(ndat)
```

```
68    REAL*8 :: bg_histo_coarse(ndat,2)
69    EXTERNAL :: funcs
70
71  !--Additional parameters for specifying object coordinates--
72  INTEGER :: Jop
73  REAL*8 :: XIop, ETAop
74  REAL*8 :: RAh, RAm, RAs, DecD, DecM, DecS, RA_rad, Dec_rad
75  REAL*8 :: tpRAh, tpRAm, tpRAs, tpDecD, tpDecM, tpDecS, tpRA_rad, tpDec_rad
76
77  !----------------------When f is known-----------------------
78  INTEGER :: bg_stars, sig_stars
79  REAL*8 :: bg_area, sig_area
80  REAL*8 :: known_f, bg_stars_in_sig_field
81  REAL*8 :: sig_field_radius = 0.1d0, bg_low_xi = -10.d0, bg_up_xi = 10.d0
82
83  END MODULE Global
84
85  !------------------------------------------------------------------------------
86
87  PROGRAM BayesianTRGBsatellite     !Master program
88  USE Global
89  IMPLICIT NONE
90
91  WRITE (field,*) 'Andromeda1'
92
93  string = TRIM(ADJUSTL(field)) // '/results.dat'
94  OPEN(3, file=TRIM(ADJUSTL(string)), status = 'unknown')
95  WRITE (3,*) "Field Name:", field
96
97  CALL positionFinder   !
98                        !
99  CALL random_seed      !
100                       !
101 CALL M31DataReader    !
102 CALL SVDFitter        !
103 CALL NoiseMake        !
104 !CALL NoisePlot       !CALL
105 CALL MCMC             !
106 !CALL PosteriorPlot   !SUBROUTINES
107                       !
108 CALL TipAndSigma      !
109 CALL PosteriorPlot    !
110 CALL OtherPlots       !
111 CALL DataHist         !
112
113 WRITE (3, '(3a11)') "   tip_mag:", "  +sigma:  ", "  -sigma: "      !
114 WRITE (3, '(3F10.3)') tip_rec, tip_psigma, tip_msigma                 !
115 WRITE (3, '(2a11)') "   f:      ", "   sigma:  "                     !
116 WRITE (3, '(2F10.3)') f_rec, f_sigma                                  !Write results
117 WRITE (3, '(2a11)') "   a:      ", "   sigma:  "                     !to file
118 WRITE (3, '(2F10.3)') a_rec, a_sigma                                  !
119 WRITE (3,*) "Distance =", REAL(tip_kpc), "kpc"                        !
120 WRITE (3,*) "Error = +", kpc_perr, "kpc -", kpc_merr, "kpc"           !
121
122 END PROGRAM BayesianTRGBsatellite
123
124 !------------------------------------------------------------------------------
125
126 SUBROUTINE PositionFinder  !Converts object's position in RA and Dec into
127 USE Global                 !its position on the M31 tangent plane
128 IMPLICIT NONE
```

```
129
130  RAh = 0.d0          !
131  RAm = 45.d0         !
132  RAs = 39.8d0        !Object coordinates
133  DecD = 38.d0        !in RA and Dec
134  DecM = 2.d0         !
135  DecS = 28.d0        !
136
137  tpRAh = 0.d0        !
138  tpRAm = 42.d0       !
139  tpRAs = 44.33d0     !Tangent point (i.e. M31)
140  tpDecD = 41.d0      !coordinates in RA and Dec
141  tpDecM = 16.d0      !
142  tpDecS = 7.5d0      !
143
144  !||Perform
145  !\/Conversion
146  RA_rad = (pi/180.d0) * (RAh * 15.d0 + RAm * (15.d0/60.d0) + RAs * (15.d0/3600.d0))
147
148  Dec_rad = (pi/180.d0) * (DecD + DecM/60.d0 + DecS/3600.d0)
149
150  tpRA_rad = (pi/180.d0) * (tpRAh * 15.d0 + tpRAm * (15.d0/60.d0) + tpRAs * (15.d0/3600.d0))
151
152  tpDec_rad = (pi/180.d0) * (tpDecD + tpDecM/60.d0 + tpDecS/3600.d0)
153
154  CALL sla_DS2TP (RA_rad, Dec_rad, tpRA_rad, tpDec_rad, XIop, ETAop, Jop)
155  !/\
156  !||
157
158  XIop = XIop * (180.d0/pi)   !tangent plane coordinates
159  ETAop = ETAop * (180.d0/pi) !(i.e. PAndAS xi and eta)
160
161
162  WRITE (3,*) "C.O.F. Xi =", XIop, "C.O.F. Eta =", ETAop
163
164  END SUBROUTINE PositionFinder
165
166  !-------------------------------------------------------------------------
167  SUBROUTINE M31DataReader !The field to be analysed is specified here
168  USE Global
169  IMPLICIT NONE
170
171  OPEN(1,file='../../../PANDAS/M31_unique_con.dat',form='unformatted',status='old')
172
173  i = 0 ; j = 0
174
175  DO WHILE (.true.)
176     READ(1,IOSTAT=ios) ra_t,de_t,iCCDt,xgt,ygt, & !Read in data
177     g,dg,clsg,im,dim,clsi,ifieldt,xki_t,eta_t, &  !from binary
178     dummy,FeH_phot_t,diff_tip_t,E_BV_t,iacc_t     !format data file
179
180     IF (ios.ne.0) exit
181
182     g=g-3.793*E_BV_t              !Extinction
183     im=im-2.086*E_BV_t            !Corrections
184     gmi = g - im
185
186     if (clsi.ne.-1 .and. clsi.ne.-2) cycle   !Rejects
187     if (clsg.ne.-1 .and. clsg.ne.-2) cycle   !non stars
188     if (18.0.le.im.and.im.le.24.) then       !Specifies
189     else                                     !magnitude
```

```
190         cycle                                !range to
191       end if                                 !include
192       if (-2.5.le.FeH_phot_t.and.FeH_phot_t.le.-1.5) then !
193       else                                                !Specifies metallicity
194         ! cycle                                           !range to include
195       end if                                              !
196
197       spotR = SQRT((ABS(eta_t - (ETAop)))**2 + (ABS(xki_t - (XIop)))**2)
198
199       IF (spotR .lt. sig_field_radius) THEN
200          i = i + 1
201
202          IF (i .gt. ndata_max) exit
203
204          mag_g(i)=g             !
205          mag_i(i)=im            !
206          g_min_i(i)=gmi         !If all conditions are met, add star data to signal arrays
207          xki(i)=xki_t           !
208          eta(i)=eta_t           !
209
210
211       ELSE IF (xki_t .ge. bg_low_xi .and. xki_t .le. bg_up_xi) THEN
212          IF (xki_t .lt. -3.5 .or. xki_t .gt. 2.5) THEN
213          IF (eta_t .ge. ETAop - 0.2d0 .and. eta_t .le. ETAop + 0.2d0) THEN
214             j = j + 1
215
216             IF (j .gt. ndata_max) exit
217
218             bg_mag_g(j)=g       !
219             bg_mag_i(j)=im      !
220             bg_g_min_i(j)=gmi   !If all conditions are met, add star data to bckgrnd arrays
221             bg_xki(j)=xki_t     !
222             bg_eta(j)=eta_t     !
223          END IF
224          END IF
225       END IF
226
227    END DO
228
229    ndata = i ; bg_ndata = j
230
231    sig_area = pi * (sig_field_radius ** 2.d0)  !Calculate area of signal field
232    bg_area = 0.4d0 * (bg_up_xi - bg_low_xi) - 2.4d0  !Calculate area of bacground field
233
234    DO i = 1, ndata
235       data(i) = mag_i(i)      !Object stars before colour cut
236    END DO
237
238    DO j = 1, bg_ndata
239       bg_data(j) = bg_mag_i(j)  !Background stars before colour cut
240    END DO
241
242    CALL M31DataPlotter
243
244    END SUBROUTINE M31DataReader
245
246    !-------------------------------------------------------------------------
247
248    SUBROUTINE M31DataPlotter  !Plot object and backgound field star positions
249    USE Global                 !on the sky and CMDs for each object (g-i vs. i)
250    IMPLICIT NONE
```

```
251
252  !----------------------Signal-Field-------------------------
253   string = TRIM(ADJUSTL(field)) // '/sig_field.ps/CPS'
254  CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
255
256  CALL pgenv(MAXVAL(xki, mask = xki .ne. 0.), MINVAL(xki, mask = xki .ne. 0.), &
257  MINVAL(eta, mask = eta .ne. 0.), MAXVAL(eta, mask = eta .ne. 0.), 1, 0)
258  CALL pgpt (ndata, xki, eta, -1)
259  CALL pglab('\(0640) (degrees)', '\(0633) (degrees)', '')
260
261  CALL pgend
262
263  !----------------------Signal-CMD--------------------------
264   string = TRIM(ADJUSTL(field)) // '/sig_cmd.ps/CPS'
265  CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
266
267  CALL pgenv(MINVAL(g_min_i, mask = g_min_i .ne. 0.), MAXVAL(g_min_i), &
268  MAXVAL(mag_i), MINVAL(mag_i, mask = mag_i .ne. 0.), 0, 0)
269  CALL pgpt (ndata, g_min_i, mag_i, -1)
270  CALL pglab('(g-i)\d0\u', 'i\d0\u', '')
271
272  CALL PolySelect    !Apply colour cut to signal CMD
273
274  CALL pgend
275
276  !----------------------Bckgrnd-Field-------------------------
277   string = TRIM(ADJUSTL(field)) // '/bg_field.ps/CPS'
278  CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
279
280  CALL pgenv(MAXVAL(bg_xki, mask = bg_xki .ne. 0.), &
281             MINVAL(bg_xki, mask = bg_xki .ne. 0.), &
282             MINVAL(bg_eta, mask = bg_eta .ne. 0.), &
283             MAXVAL(bg_eta, mask = bg_eta .ne. 0.), 1, 0)
284  CALL pgpt (bg_ndata, bg_xki, bg_eta, -1)
285  CALL pglab('\(0640) (degrees)', '\(0633) (degrees)', '')
286
287  CALL pgend
288
289  !----------------------Bckgrnd-CMD--------------------------
290   string = TRIM(ADJUSTL(field)) // '/bg_cmd.ps/CPS'
291  CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
292
293  CALL pgenv(MINVAL(bg_g_min_i, mask = bg_g_min_i .ne. 0.), &
294             MAXVAL(bg_g_min_i), MAXVAL(bg_mag_i), &
295             MINVAL(bg_mag_i, mask = bg_mag_i .ne. 0.), 0, 0)
296  CALL pgpt (bg_ndata, bg_g_min_i, bg_mag_i, -1)
297  CALL pglab('(g-i)\d0\u', 'i\d0\u', '')
298
299  CALL PolySelect
300
301  CALL pgend
302
303  !-------------Input selected data into 'data'-----------
304  !||Removes stars from the signal and backround fields that
305  !\/lie outside of the chosen colour cut
306  data = 0.d0 ; xki = 0.d0 ; eta = 0.d0 ; d_blim = 100.d0 ; d_flim = 0.d0
307  DO i = 1, ndata2
308      data(i) = mag_i_poly(i)
309      IF (data(i) .lt. d_blim) THEN
310          d_blim = data(i)
311      END IF
```

```
312     IF (data(i) .gt. d_flim) THEN
313         d_flim = data(i)
314     END IF
315 END DO
316
317 bg_data = 0.d0 ; bg_blim = 100.d0 ; bg_flim = 0.d0
318 DO i = 1, bg_ndata2
319     bg_data(i) = bg_mag_i_poly(i)
320     IF (bg_data(i) .lt. bg_blim) THEN
321         bg_blim = bg_data(i)
322     END IF
323     IF (bg_data(i) .gt. bg_flim) THEN
324         bg_flim = bg_data(i)
325     END IF
326 END DO
327 !/\
328 !||
329
330 !---Set parameters for calculation of background height---
331
332 sig_stars = ndata2      !Total number of stars in signal field
333 bg_stars = bg_ndata3    !Number of stars in background field
334 bg_stars_in_sig_field = REAL(bg_stars) * (sig_area/bg_area)
335 !Number of Background stars in signal field
336
337 WRITE (3,*) "Number of data points:", sig_stars
338 WRITE (3,*) "SNR:", (REAL(sig_stars) - bg_stars_in_sig_field) / bg_stars_in_sig_field
339
340 !---------Make coarse data histogram for bckgrnd---------
341
342 DO i = 1, INT(0.25*(nbins-1.d0)) + 1
343     bg_histo_coarse(i,1) = 18.d0 + (i-1.d0)/REAL(0.25*binspm)
344 END DO
345
346 DO i = 1, bg_ndata2
347     bg_histo_coarse(INT((bg_data(i)-18.d0)*0.25*binspm) + 1, 2) = &
348     bg_histo_coarse(INT((bg_data(i)-18.d0)*0.25*binspm) + 1, 2) + 1.d0
349 END DO
350
351 !|| Fill empty bright edge of array with
352 !\/ artificial data for improved fitting
353 DO i = 1, INT((bg_blim - 18.d0) * REAL(binspm/4.d0)) + 4
354     bg_histo_coarse(i,2) = bg_histo_coarse(INT((bg_blim - 18.d0) * REAL(binspm/4.d0)) + 4, 2)
355 END DO
356
357 END SUBROUTINE M31DataPlotter
358
359 !-------------------------------------------------------------------------
360
361 SUBROUTINE SVDFitter  !Fits a polynomial to the background luminosity function
362 USE Global
363 IMPLICIT NONE
364
365 INTEGER  :: ntmp
366
367 xa = bg_histo_coarse(:,1)
368 ya = bg_histo_coarse(:,2)
369 xt = xa
370 yt = ya
371 sig = 1.e0
372
```

```
373
374  ! Shift the array in steps of 1 until the first element does not contain a zero
375
376  shiftloop: do
377     xt = cshift(xt,1)
378     yt = cshift(yt,1)
379     if ( yt(1) > 0.1 ) exit shiftloop
380  end do shiftloop
381
382  ntmp = 0
383  countloop: do i = 1 , ndat
384     if ( yt(i) > 0.1 ) then
385        ntmp = ntmp + 1
386     else
387        exit countloop
388     end if
389  end do countloop
390
391    xt = xt - 21.
392
393    CALL svdfit(xt,yt,sig,ntmp-1,ay,ma,u,v,w,mp,np,chisq,funcs)
394
395  CALL BG_DataHist
396
397  END SUBROUTINE SVDFitter
398
399  !-------------------------------------------------------------------------
400
401  SUBROUTINE BG_DataHist !Plots background luminosity function together with
402  USE Global             !fitted polynomial
403  IMPLICIT NONE
404
405  bfm = 0.d0
406
407  DO i = 1, ndat
408     DO j = 1 , np
409        bfm(i) = bfm(i) + ay(j) * (xa(i)-21.) ** (j-1)
410     END DO
411  END DO
412
413
414  !-----------------------Plots best fit model over coarse histogram
415  string = TRIM(ADJUSTL(field)) // '/bckgrdfit.ps/CPS'
416  CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
417
418  CALL pgenv(19.5, 23.5, 0., 1.1*MAXVAL(REAL(bg_histo_coarse(:,2))), 0, 0)
419  CALL pgbin (ndat, REAL(bg_histo_coarse(:,1)), REAL(bg_histo_coarse(:,2)), .true.)
420  CALL pgsci(2)
421  CALL pgline (ndat, xa, REAL(bfm))
422  CALL pgsci(1)
423  CALL pglab('i\d0\u', 'counts', '')
424
425  CALL pgend
426
427  END SUBROUTINE BG_DataHist
428
429  !-------------------------------------------------------------------------
430
431  SUBROUTINE MCMC  !The master Markov Chain MonteCarlo routine
432  USE Global       !creates a new model at each iteration and then compares
433  IMPLICIT NONE    !the quality of the fit to the data
```

```
434                 !*** Most subroutines are called from 'MCMC' ***
435
436  REAL*8 :: gasdev
437
438  known_f = (REAL(bg_stars) * sig_area)/(REAL(sig_stars) * bg_area)
439
440  x1(1) = 20.88d0; x2(1) = known_f; x3(1) = 0.27d0 ; time(1) = 1
441
442  mag_tip = x1(1) ; f = x2(1) ; a = x3(1)
443  CALL ModelMake     !Make model and
444  CALL Convolution   !evaluate goodness of fit
445  CALL Loglike       !for initial parameter choices
446  LikeA = logL
447  LikeB = 0.d0
448
449  x1(2) = x1(1) ; x2(2) = x2(1) ; x3(2) = x3(1)
450
451  Best_Combo(6) = -9.d99
452
453  string = TRIM(ADJUSTL(field)) // '/MCMC_steps.dat'
454  OPEN(2, file=TRIM(ADJUSTL(string)), status = 'unknown')
455  WRITE(2,*) "___Iteration___________mag_tip_____________f_____________a__________LikeA___________LikeB"
456  DO it = 2, nit
457     time(it) = it
458     p(1) = x1(it) + 0.03d0*gasdev(idum)  !Gaussian weighted steps from initial
459     p(2) = x2(it)! + 0.02d0*gasdev(idum)  !parameters for the tip magnitude (p(1))
460     p(3) = x3(it) + 0.02d0*gasdev(idum)  !noise ratio (p(2)) and slope (p(3))
461
462     IF (p(1) .lt. blim .or. p(1) .gt. flim) THEN      !
463        r = 0.d0                                       !
464     else IF (p(2) .le. 0.d0 .or. p(2) .ge. 1.d0) THEN !Restrictions on
465        r = 0.d0                                       !whether proposed step
466     else IF (p(3) .le. 0.d0 .or. p(3) .ge. 2.d0) THEN !
467        r = 0.d0                                       !
468     else                                              !
469        mag_tip = p(1) ; f = p(2) ; a = p(3)
470        CALL ModelMake       !Make model and
471        CALL Convolution     !evaluate the
472        CALL Loglike         !goodness of fit
473        LikeB = logL
474        r = 10**(LikeB-LikeA)
475     end IF
476     CALL random_number(randnum3)                                !
477     IF (it .lt. nit) THEN                                       !
478     IF (randnum3 .le. r) THEN                                   !
479        x1(it+1) = p(1) ; x2(it+1) = p(2) ; x3(it+1) = p(3)      !
480        likeA = likeB                                            !Decide whether
481     ELSE                                                        !to take step
482        x1(it+1) = x1(it) ; x2(it+1) = x2(it) ; x3(it+1) = x3(it) !or not
483        likeA = likeA                                            !
484     END IF                                                      !
485     END IF                                                      !
486     post_y1(INT((x1(it) - 18.d0)*10*binspm + 1)) = &          !
487     post_y1(INT((x1(it) - 18.d0)*10*binspm + 1)) + 1.d0       !
488     post_y2(INT(x2(it) * (nbins - 1)) + 1) = &                !Generate posterior plot
489     post_y2(INT(x2(it) * (nbins - 1)) + 1) + 1.d0             !for mag_tip, f and a
490     post_y3(INT(x3(it) * (nbins - 1)) + 1) = &                !
491     post_y3(INT(x3(it) * (nbins - 1)) + 1) + 1.d0             !
492
493     WRITE (2, '(6F16.5)') time(it), x1(it), x2(it), x3(it), LikeA, LikeB
494
```

```
495       IF (LikeB .gt. Best_Combo(6)) THEN
496          Best_Combo(1) = time(it) ; Best_Combo(2) = p(1)
497          Best_Combo(3) = p(2)     ; Best_Combo(4) = p(3)
498          Best_Combo(5) = LikeA    ; Best_Combo(6) = LikeB
499       END IF
500
501    END DO
502
503    WRITE (2, '(6F16.5)') Best_Combo(1), Best_Combo(2), Best_Combo(3), &
504                          Best_Combo(4), Best_Combo(5), Best_Combo(6)
505
506    DO i = 1, 10*(nbins-1)+1                                    !
507       post_x1(i) = 18.d0 + (REAL(i) - 1.d0)/REAL(10*binspm)    !
508    END DO                                                      !
509                                                                !
510    DO i = 1, nbins                                             !x-values of posterior
511       post_x2(i) = (REAL(i) - 1.d0)/REAL(nbins - 1)            !histograms created above
512    END DO                                                      !
513                                                                !
514    DO i = 1, 2*nbins - 1                                       !
515       post_x3(i) = (REAL(i) - 1.d0)/REAL(nbins - 1)            !
516    END DO                                                      !
517
518    END SUBROUTINE MCMC
519
520    !-------------------------------------------------------------------------
521
522    SUBROUTINE PosteriorPlot
523    USE Global
524    IMPLICIT NONE
525
526    post_y1 = post_y1/nit ; post_y2 = post_y2/nit ; post_y3 = post_y3/nit
527
528    !------------------------------Plots mag_tip posterior plot
529    string = TRIM(ADJUSTL(field)) // '/mag_tip_postplot.ps/CPS'
530    CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
531
532    CALL pgenv(REAL(MINVAL(x1))-0.01, REAL(MAXVAL(x1))+0.01, &
533         0., 1.1*REAL(MAXVAL(post_y1)), 0, 0)
534
535    CALL pgbin (10*(nbins-1)+1, REAL(post_x1), REAL(post_y1),.true.)
536    CALL pglab('Proposed i\d0\u tip magnitude', 'Probability', '')
537
538    CALL pgend
539
540    !------------------------------Plots f and a posterior plots
541    string = TRIM(ADJUSTL(field)) // '/f_and_a_postplot.ps/CPS'
542    CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
543
544    IF (MAXVAL(post_y3) .ge. MAXVAL(post_y2)) THEN
545       CALL pgenv(0.,  2., 0., 1.1*REAL(MAXVAL(post_y3)), 0, 0)
546    ELSE
547       CALL pgenv(0.,  2., 0., 1.1*REAL(MAXVAL(post_y2)), 0, 0)
548    END IF
549
550    CALL pgsci(2)
551    CALL pgbin (nbins, REAL(post_x2), REAL(post_y2),.true.)
552    CALL pgsci(3)
553    CALL pgbin (2*nbins-1, REAL(post_x3), REAL(post_y3),.true.)
554    CALL pgsci(1)
555    CALL pglab('Proposed value', 'Probability: f (red) a (green)', '')
```

```
556
557 CALL pgend
558
559 post_y1 = post_y1*nit ; post_y2 = post_y2*nit ; post_y3 = post_y3*nit
560
561 END SUBROUTINE PosteriorPlot
562
563 !------------------------------------------------------------------------------
564
565 SUBROUTINE OtherPlots
566 USE Global
567 IMPLICIT NONE
568
569 !-----------------------------Variation of 'mag_tip' with iteration #
570 string = TRIM(ADJUSTL(field)) // '/mag_tip_val_vs_it.ps/CPS'
571 CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
572
573 CALL pgenv(0., REAL(nit), REAL(MINVAL(x1))-0.01, REAL(MAXVAL(x1))+0.01, 0, 0)
574 CALL pgline (nit, REAL(time), REAL(x1))
575 CALL pglab('Iteration␣number', 'Proposed␣i\d0\u␣tip␣magnitude', '')
576
577 CALL pgend
578
579 !-----------------------------Variation of 'f' and 'a' with iteration #
580 string = TRIM(ADJUSTL(field)) // '/f_and_a_val_vs_it.ps/CPS'
581 CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
582
583 CALL pgenv(0., REAL(nit), 0., 2., 0, 0)
584 CALL pgsci(2)
585 CALL pgline (nit, REAL(time), REAL(x2))
586 CALL pgsci(3)
587 CALL pgline (nit, REAL(time), REAL(x3))
588 CALL pgsci(1)
589 CALL pglab('Iteration␣number', 'Proposed␣value:␣f␣(red)␣a␣(green)', '')
590
591 CALL pgend
592
593 !-----------------------------Values of 'f' for each value of 'mag_tip'
594 string = TRIM(ADJUSTL(field)) // '/f_vs_mag_tip.ps/CPS'
595 CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
596
597 CALL pgenv(0.99*REAL(MINVAL(x1)), 1.01*REAL(MAXVAL(x1)), 0.9*REAL(MINVAL(x2)), 1.1*REAL(MAXVAL(x2)), 0, 0)
598 CALL pgpoint (nit, REAL(x1), REAL(x2), -1)
599 CALL pglab('Proposed␣i\d0\u␣tip␣magnitude', 'Proposed␣value␣of␣f', '')
600
601 CALL pgend
602
603 !-----------------------------Values of 'a' for each value of 'mag_tip'
604 string = TRIM(ADJUSTL(field)) // '/a_vs_mag_tip.ps/CPS'
605 CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
606 CALL pgenv(REAL(MINVAL(x1))-0.01, REAL(MAXVAL(x1))+0.01, REAL(MINVAL(x3))-0.01, REAL(MAXVAL(x3))+0.01, 0, 0)
607 CALL pgslw(3)
608 CALL pgpoint (nit, REAL(x1), REAL(x3), -1)
609 CALL pgslw(1)
610 CALL pglab('Proposed␣i\d0\u␣tip␣magnitude', 'Proposed␣value␣of␣a', '')
611
612 CALL pgend
613
614 !-----------------------------Values of 'f' for each value of 'a'
615 string = TRIM(ADJUSTL(field)) // '/a_vs_f.ps/CPS'
616 CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
```

```
617
618  CALL pgenv(0.9*REAL(MINVAL(x2)),  1.1*REAL(MAXVAL(x2)), 0.9*REAL(MINVAL(x3)), 1.1*REAL(MAXVAL(x3)), 0, 0)
619  CALL pgpoint (nit, REAL(x2), REAL(x3), -1)
620  CALL pglab('Proposed value of f', 'Proposed value of a', '')
621
622  CALL pgend
623
624  END SUBROUTINE OtherPlots
625
626  !-------------------------------------------------------------------------
627
628  SUBROUTINE NoiseMake !Generates a polynomial of degree 7 that follows the
629  USE Global           !functional form of the GSS background LF. The polynomial
630  IMPLICIT NONE        !coefficients were derived in 'BackgroundPolyFit' using
631                       !'svdfit' from Numerical Recipes.
632  area2 = 0.d0
633
634  DO i = 1, 8 * binspm + 1
635     modelnoise(i,1) = 18.d0 + (i-1.d0)/REAL(binspm)
636     modelnoise(i,2) = 0.d0
637     DO j = 1, np                      !Set background counts
638        modelnoise(i,2) = modelnoise(i,2) + ay(j) * (modelnoise(i,1) - 21.d0) ** (j - 1)
639     END DO
640     IF (modelnoise(i,2) .lt. 0.d0) THEN  !
641        modelnoise(i,2) = 0.d0         !Insure no negative counts
642     END IF                            !
643     IF (i .ge. (blim-18.d0)*binspm + 1 .and. i .le. flimBins) THEN
644        area2 = area2 + modelnoise(i,2)   !Used for normalization in 'ModelMake'
645     END IF
646  END DO
647
648  END SUBROUTINE NoiseMake
649
650  !-------------------------------------------------------------------------
651  SUBROUTINE NoisePlot !Plots the unscaled form of the background LF
652  USE Global
653  IMPLICIT NONE
654
655  CALL pgbegin(0,'?',1,1)
656
657  CALL pgenv(18.,  REAL(mag_cutoff), 0., 1.1*REAL(MAXVAL(modelnoise(:,2), mask = modelnoise(:,1) .le. 23.5)), 0, 0)
658  CALL pgbin (nbins - INT(2.5*binspm), REAL(modelnoise(:,1)), REAL(modelnoise(:,2)), .true.)
659  CALL pglab('i\d0\u', 'Counts', '')
660
661  CALL pgend
662
663  END SUBROUTINE NoisePlot
664
665  !-------------------------------------------------------------------------
666
667  SUBROUTINE ModelMake   !Initial Model (i.e. model before convolution)
668  USE Global
669  IMPLICIT NONE
670
671  REAL*8 :: func_i
672
673  noise = modelnoise(:,2) * (f/area2) !Calculate background height
674  area = 0.d0
675  DO i = 1, nbins
676     model(i,1) = 18.d0 + (i-1.d0)/REAL(binspm)
677     IF (model(i,1) + hb .gt. mag_tip .and. model(i,1) - hb .le. mag_tip) THEN
```

```
678         model(i,2) = ((10.d0**(a*(model(i,1) + hb - mag_tip)))/(a*LOG(10.))) - &
679                      (1.d0/(a*LOG(10.)))                    !Model value at tip
680         area = area + model(i,2)       !Used to calculate noise in master program
681       ELSE IF (model(i,1) .gt. mag_tip) THEN          !Model value faintward of tip
682         model(i,2) = ((10.d0**(a*(model(i,1) + hb - mag_tip)))/(a*LOG(10.))) - &
683                      ((10.d0**(a*(model(i,1) - hb - mag_tip)))/(a*LOG(10.)))
684         IF (i .ge. (blim-18.d0)*binspm + 1 .and. i .le. flimBins) THEN
685           area = area + model(i,2)    !Used to calculate noise in master program
686         ELSE
687           cycle;
688         END IF
689       ELSE
690         model(i,2) = 0.d0                          !Model value brightward of tip
691       END IF
692     END DO
693
694     model(:,2) = (model(:,2)/area) * (1.d0-f)  !Normalize
695     model(:,2) = model(:,2) + noise      !Add noise
696
697     END SUBROUTINE ModelMake
698
699     !------------------------------------------------------------------------------
700
701     SUBROUTINE ModelPrint !Prints model before convolution
702     USE Global
703     IMPLICIT NONE
704
705     CALL pgbegin(0,'?',1,1)
706
707     CALL pgenv(REAL(mag_tip) - 3., REAL(mag_cutoff), 0., 1.1*REAL(model(INT(5.5*binspm),2)), 0, 0)
708     CALL pgbin (nbins - INT(2.5*binspm), REAL(model(:,1)), REAL(model(:,2)), .true.)
709     CALL pglab('i\d0\u', 'Counts', '')
710
711     CALL pgend
712
713     END SUBROUTINE ModelPrint
714
715     !------------------------------------------------------------------------------
716
717     SUBROUTINE GaussianKernel !Generates a Gaussian kernel 'kernel' with
718     USE Global                !HWHM (sigma) changing with magnitude in
719     IMPLICIT NONE             !accordance with func_i. Kernel is defined from
720                               !gx = -5*sigma to gx = +5*sigma.
721     REAL*8 :: func_i
722
723     temp = 0.d0 ; kernel = 0.d0
724     gx=0.
725     j=0
726     DO WHILE (gx .le. 5.e0*func_i(t))                          !
727       j=j+1                                                    !
728       gx = 0.e0 + (j-1.e0)/binspm                              !Creates half of
729       temp(j,1) = gx                                           !the kernel ('temp')
730       temp(j,2) = exp(-((gx)**2.e0)/(2.e0*(func_i(t)**2.e0)))!
731     END DO                                                     !
732
733     ghw = j - 1.d0      !The first non-zero bin of 'cmodel' will be the first
734                         !non-zero bin of 'model' minus ghw
735
736     DO k = 1, j
737       kernel(k,:) = temp(j - (k-1),:)  !Create 'kernel' by concatenating
738       kernel(j+k,2) = temp(k+1,2)      !'temp' with a reflected version
```

```
739     kernel(j+k,1) = -temp(k+1,1)      !of itself
740  END DO
741  !Note: temp(2*j,2) = 0.d0 ; temp(2*j,1) = -0.d0
742
743  kernel(:,2) = kernel(:,2)/SUM(kernel(:,2))
744
745  END SUBROUTINE GaussianKernel
746
747  !-------------------------------------------------------------------------
748
749  SUBROUTINE GaussianKernelPrint !Prints Gaussian Kernel at given magnitude
750  USE Global
751  IMPLICIT NONE
752
753  REAL*8 :: func_i
754
755  CALL pgbegin(0,'?',1,1)
756
757  CALL pgenv(-5.5 * REAL(func_i(t)), 5.5 * REAL(func_i(t)), 0., 1.1*MAXVAL(REAL(kernel(:,2))), 0, 0)
758  CALL pgbin (2*ghw+1, REAL(kernel(:,1)), REAL(kernel(:,2)), .true.)
759  CALL pglab('Magnitude_offset', 'Strength', '')
760
761  CALL pgend
762
763  END SUBROUTINE GaussianKernelPrint
764
765  !-------------------------------------------------------------------------
766
767  SUBROUTINE Convolution  !Convolves initial model with a Gaussian kernel
768  Use Global              !whose width is equal to the photometric error
769  IMPLICIT NONE           !and hence expands with increasing magnitude
770
771  cmodel = 0.d0
772
773  DO i = 1, nbins
774     t = 18.d0 + (i - 1.d0)/REAL(binspm)     !Convert bin number to magnitude
775     cmodel(i,1) = t                         !This then derives the current
776
777     CALL GaussianKernel                     !width of the Gaussian kernel
778     DO j = -ghw, ghw, +1                                              !
779        IF (i .gt. ghw .and. i .lt. nbins - ghw) THEN                  !Convolve
780           cmodel(i+j,2) = cmodel(i+j,2) + kernel(ghw+j+1,2)*model(i,2)!model with
781        END IF                                                         !gaussian
782     END DO                                                            !
783  END DO
784
785  DO i = nbins, flimBins+1, -1          !Set the faint limit
786        cmodel(i,2) = 0.d0              !of the final convolved
787  END DO                                !model at flim.
788  cmod_nbins = flimBins
789
790  !Normalize the convolved model
791  cmodel(:,2) = cmodel(:,2)/SUM(cmodel(:,2), mask = cmodel(:,1) .ge. blim)
792
793  !Note the above step is very important - normalization must only be over the
794  !range of magnitudes in the 'data' array - i.e. down to mlim -> not right the
795  !way to 18 unless mlim = 18. This was a difficult bug to find!
796
797  END SUBROUTINE Convolution
798
799  !-------------------------------------------------------------------------
```

```
800
801 SUBROUTINE ConvolutionPrint !Prints convolved version of model
802 USE Global
803 IMPLICIT NONE
804
805 CALL pgbegin(0,'?',1,1)
806
807 CALL pgenv(REAL(mag_tip) - 0.5, 25., 0., 1.1*MAXVAL(REAL(cmodel(:,2))), 0, 0)
808 CALL pgbin (nbins, REAL(cmodel(:,1)), REAL(cmodel(:,2)), .true.)
809 CALL pglab('i\d0\u', 'Relative probability', '')
810
811 CALL pgend
812
813 END SUBROUTINE ConvolutionPrint
814
815 !------------------------------------------------------------------------
816
817 SUBROUTINE DataHist !Generates finely and coarsely binned histograms and
818 USE Global          !overlays them with the best fit model determined by
819 IMPLICIT NONE       !the MCMC
820
821 REAL*8 :: scaled_f_rec
822
823 histo_fine(:,1) = model(:,1)
824 DO i = 1, INT(0.25*(nbins-1.d0)) + 1
825    histo_coarse(i,1) = 18.d0 + (i-1.d0)/REAL(0.25*binspm)
826 END DO
827
828 DO i = 1, ndata2                                                    !
829    histo_fine(INT(REAL((data(i)-18.d0)*binspm) + 1.d0),2) = &       !Generates
830    histo_fine(INT(REAL((data(i)-18.d0)*binspm) + 1.d0),2) + 1.d0    !
831    histo_coarse(INT(REAL((data(i)-18.d0)*0.25*binspm) + 1.d0),2) = & !Histograms
832    histo_coarse(INT(REAL((data(i)-18.d0)*0.25*binspm) + 1.d0),2) + 1.d0 !
833 END DO                                                              !
834
835 histo_coarse(INT(5.5*REAL(binspm/4.d0)) + 1,2) = &    !See paragraph
836 histo_coarse(INT(5.5*REAL(binspm/4.d0)) + 1,2) * 2.d0 !below
837
838 !For graphing purposes, the last bin of the coarse histogram is doubled since
839 !this bin lies half outside the range of interest and so is depleted by
840 !roughly one half. This is for graphing only and has no bearing on the
841 !determined best fit model.
842
843 !|| Plot Best Fit Model
844 !\/ over histogram
845 mag_tip = tip_rec ; f = f_rec ; a = a_rec          !
846 CALL ModelMake                                     !Generate best fit function
847 CALL Convolution                                   !
848
849 bfm = cmodel(:,2)                                  !bfm = best fit model
850 bfm = bfm * (SUM(histo_fine(:,2))/SUM(bfm))        !Scale bfm to match histogram
851
852 !------------------------Plots best fit model over fine histogram
853 string = TRIM(ADJUSTL(field)) // '/model_fit_vs_data_fine.ps/CPS'
854 CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
855
856 CALL pgenv(REAL(blim), REAL(flim), 0., 1.1*MAXVAL(real(histo_fine(:,2))), 0, 0)
857 CALL pgbin (nbins, REAL(histo_fine(:,1)), REAL(histo_fine(:,2)), .false.)
858 CALL pgsci(2)
859 CALL pgslw(5)
860 CALL pgline (nbins, REAL(histo_fine(:,1)), REAL(bfm))
```

```
861  CALL pgsci(1)
862  CALL pgslw(1)
863  CALL pglab('i\d0\u', 'Counts', '')
864
865  CALL pgend
866
867  bfm = bfm * 4.d0                    !Scale bfm to match coarse histogram
868
869  !-----------------------Plots best fit model over coarse histogram
870  string = TRIM(ADJUSTL(field)) // '/model_fit_vs_data_coarse.ps/CPS'
871  CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
872
873  CALL pgenv(REAL(blim), REAL(flim), 0., 1.1*MAXVAL(real(histo_coarse(:,2))), 0, 0)
874  CALL pgbin (INT(0.25*(nbins-1.d0)) + 1, REAL(histo_coarse(:,1)), &
875              REAL(histo_coarse(:,2)), .false.)
876  CALL pgsci(2)
877  CALL pgline (nbins, REAL(histo_fine(:,1)), REAL(bfm))
878  CALL pgsci(1)
879  CALL pglab('i\d0\u', 'Counts', '')
880
881  CALL pgend
882
883  END SUBROUTINE DataHist
884
885  !-------------------------------------------------------------------------
886
887  SUBROUTINE LogLike !Generates the log of the likelihood fn
888  USE Global         ! - gives the likelihood of the tip being at
889  IMPLICIT NONE      !each magnitude given the dataset
890
891  logL = 0.d0
892  DO i = 1, ndata2
893     prob = (data(i) - 18.d0)*binspm + 1.d0
894     prob = cmodel(INT(prob),2)
895     logL = logL + LOG10(prob)
896  END DO
897
898  END SUBROUTINE LogLike
899
900  !-------------------------------------------------------------------------
901
902  SUBROUTINE TipAndSigma !Identifies the best parameter values and
903  USE Global             !their associated 1 sigma errors from the
904  IMPLICIT NONE          !respective posterior plots.
905
906  PPD_peak = 0.d0                     !
907  DO i = 1, 10*(nbins-1)+1            !
908     IF (post_y1(i) .gt. PPD_peak) THEN  !
909        PPD_peak = post_y1(i)         !Find best fit TRGB value
910        tip_rec = post_x1(i)          !
911     END IF                           !
912  END DO                              !
913
914  PPD_peak = 0.d0                     !
915  DO i = 1, nbins                     !
916     IF (post_y2(i) .gt. PPD_peak) THEN  !
917        PPD_peak = post_y2(i)         !Find best fit f value
918        f_rec = post_x2(i)            !
919     END IF                           !
920  END DO                              !
921
```

```
922  PPD_peak = 0.d0                          !
923  DO i = 1, 2*nbins - 1                    !
924     IF (post_y3(i) .gt. PPD_peak) THEN    !
925        PPD_peak = post_y3(i)              !Find best fit a value
926        a_rec = post_x3(i)                 !
927     END IF                                !
928  END DO                                   !
929
930  tip_kpc = (100.d0**((tip_rec + 3.44d0)/10.d0))/100.d0 !Distance inferred from
931                                                        !tip magnitude in kpc
932
933  tip_counts = 0.d0 ; mcounts = 0.d0                    !
934  DO i = MAXLOC(post_y1, DIM = 1), 1, -1                !
935     mcounts = mcounts + post_y1(i)                     !
936  END DO                                                !
937  DO i = MAXLOC(post_y1, DIM = 1), 1, -1                !
938     tip_counts = tip_counts + post_y1(i)               !Finds negative one sigma
939     IF (tip_counts .ge. 0.682*mcounts) THEN            !error in magnitudes
940        tip_msigma = ((REAL(i) - 1.d0)/REAL(10*binspm)) + 18.d0 !
941        tip_msigma = tip_rec - tip_msigma               !
942        exit                                            !
943     END IF                                             !
944  END DO                                                !
945
946  tip_counts = 0.d0 ; pcounts = 0.d0                    !
947  DO i = MAXLOC(post_y1, DIM = 1), 10*(nbins-1)+1       !
948     pcounts = pcounts + post_y1(i)                     !
949  END DO                                                !
950  DO i = MAXLOC(post_y1, DIM = 1), 10*(nbins-1)+1       !
951     tip_counts = tip_counts + post_y1(i)               !Finds positive one sigma
952     IF (tip_counts .ge. 0.682*pcounts) THEN            !error in magnitudes
953        tip_psigma = ((REAL(i) - 1.d0)/REAL(10*binspm)) + 18.d0 !
954        tip_psigma = tip_psigma - tip_rec               !
955        exit                                            !
956     END IF                                             !
957  END DO                                                !
958
959
960  d1 = 0 ; d2 = 0 ; d3 = 0 ; d4 = 0
961  f_counts = 0.d0 ; a_counts = 0.d0                !
962  DO i = 1, nbins                                  !
963     f_counts = f_counts + post_y2(i)              !
964     a_counts = a_counts + post_y3(i)              !
965     IF (f_counts .ge. 0.159*nit .and. d1 .eq. 0) THEN !
966        fminsig = post_x2(i)                       !
967        d1 = 1                                     !
968     END IF                                        !
969     IF (f_counts .ge. 0.841*nit .and. d2 .eq. 0) THEN !For f and a:
970        fplusig = post_x2(i)                       !Finds upper and lower
971        d2 = 1                                     !bounds for posterior
972     END IF                                        !distribution within one
973     IF (a_counts .ge. 0.159*nit .and. d3 .eq. 0) THEN !sigma of maximum.
974        aminsig = post_x3(i)                       !
975        d3 = 1                                     !
976     END IF                                        !
977     IF (a_counts .ge. 0.841*nit .and. d4 .eq. 0) THEN !
978        aplusig = post_x3(i)                       !
979        d4 = 1                                     !
980     END IF                                        !
981  END DO                                           !
982
```

```
983
984  f_sigma = 0.5d0*(fplusig - fminsig)          !Hence calculates 1 sigma error
985  a_sigma = 0.5d0*(aplusig - aminsig)          !for f and a
986
987  kpc_merr = tip_kpc*100.d0**(tip_msigma/10.d0) - tip_kpc !minus tip error in kpc
988  kpc_perr = tip_kpc*100.d0**(tip_psigma/10.d0) - tip_kpc !plus tip error in kpc
989
990  END SUBROUTINE TipAndSigma
991
992  !-------------------------------------------------------------------------
993
994  FUNCTION func_i(m)    !This function feeds the photometric error as a function
995  USE Global            !of magnitude to the 'GaussianKernel' subroutine.
996  IMPLICIT NONE
997
998  REAL*8 :: func_i, m, c1, c2, c3
999
1000 c1 = 0.001
1001 c3 = log(0.24) - log(0.11)
1002 c2 = c3*25.0 - log(0.24)
1003
1004 func_i = c1 + exp(c3*m - c2)
1005
1006 END FUNCTION
1007
1008 !-------------------------Rodrigo's poly selection tool-----------------------
1009
1010 SUBROUTINE PolySelect     !Used for selection of appropriate colour cut
1011 USE Global                !in colour-magnitude space
1012 IMPLICIT NONE
1013
1014
1015 integer MAXPT, ipol
1016 integer NPT_ggr, NPT_spatial
1017 parameter (MAXPT=100)
1018 real*4   XCOL_ggr(MAXPT), YMAG_ggr(MAXPT)
1019 real*4   X_spatial(MAXPT), Y_spatial(MAXPT)
1020 logical refine_CMDsel_ggr, refine_spatialsel
1021 !parameter (refine_CMDsel_ggr=.true.)
1022 parameter (refine_CMDsel_ggr=.false.)
1023 !parameter (refine_spatialsel=.true.)
1024 parameter (refine_spatialsel=.false.)
1025
1026 logical in_poly
1027 external in_poly
1028
1029 npt_ggr=0
1030 if (refine_CMDsel_ggr) then
1031    call pgsls(2)
1032    call pgmove(0.2,26.0)
1033    call pgdraw(0.2,15.0)
1034    call pgsls(1)
1035    call pglcur(MAXPT, NPT_ggr, XCOL_ggr, YMAG_ggr)
1036    open(2, file='ANDI.CMD', status='unknown')
1037    write(2,*) NPT_ggr
1038    do ipol=1,NPT_ggr
1039       write(2,*) XCOL_ggr(ipol), YMAG_ggr(ipol)
1040    end do
1041    close(2)
1042    call pgsci(1)
1043    call pgadvance
```

```
1044  else
1045     open(2,file='ANDI.CMD',status='old')
1046     read(2,*) NPT_ggr
1047     do ipol=1,NPT_ggr
1048        read(2,*) XCOL_ggr(ipol),YMAG_ggr(ipol)
1049     end do
1050     close(2)
1051     call pgsci(2)
1052     call pgslw(5)
1053     call pgline(NPT_ggr,XCOL_ggr,YMAG_ggr)
1054     call pgsci(1)
1055     call pgslw(1)
1056  end if
1057
1058  !------------------Make colour cut to Signal Field------------------
1059  j=0                                                              !
1060  DO i = 1, ndata                                                  !Makes new
1061     IF (in_poly(g_min_i(i),mag_i(i),NPT_ggr,XCOL_ggr,YMAG_ggr)) THEN!arrays for
1062        IF (mag_i(i) .le. flim .AND. mag_i(i) .ge. blim) THEN      !i and g-i
1063           j = j+1                                                !containing
1064           mag_i_poly(j) = mag_i(i)                               !only stars
1065           g_min_i_poly(j) = g_min_i(i)                           !within
1066        END IF                                                    !polygon
1067     END IF                                                       !
1068  END DO                                                          !
1069
1070  ndata2 = j     !New number of stars in dataset after colour cut
1071
1072  !------------------Make colour cut to Bckgrnd Field-----------------
1073  j=0 ; k = 0                                                                !
1074  DO i = 1, bg_ndata                                                         !
1075     IF (in_poly(bg_g_min_i(i),bg_mag_i(i),NPT_ggr,XCOL_ggr,YMAG_ggr)) THEN  !
1076        IF (bg_mag_i(i) .le. 24.d0) THEN                                    !Makes new
1077           IF (bg_mag_i(i) .le. flim .AND. bg_mag_i(i) .ge. blim) THEN      !arrays for
1078              k = k+1                                                       !i and g-i
1079           END IF                                                           !containing
1080           j = j+1                                                          !only stars
1081           bg_mag_i_poly(j) = bg_mag_i(i)                                   !within
1082           bg_g_min_i_poly(j) = bg_g_min_i(i)                               !polygon
1083        END IF                                                              !
1084     END IF                                                                 !
1085  END DO                                                                    !
1086
1087  bg_ndata2 = j ; bg_ndata3 = k  !Stars in bckgrnd ; Stars in bckgrnd between blim & flim
1088
1089  END SUBROUTINE PolySelect
1090
1091  !-----------------------------------------
1092
1093  logical function in_poly(x,y,np,xp,yp) !Used by PolySelect subroutine
1094  implicit none
1095
1096  real*4 x,y
1097  integer np
1098  real*4 xp(np),yp(np)
1099  real*4 tiny,xs,xe,ys,ye
1100  parameter (tiny=1.e-5)
1101
1102  real*4 simag,fimag
1103  external fimag
1104  integer j
```

```
1105
1106  simag=0.0
1107  do j=1,np
1108     if (j.lt.np) then
1109        xe=xp(j+1)
1110        xs=xp(j)
1111        ye=yp(j+1)
1112        ys=yp(j)
1113     else
1114        xe=xp(1)
1115        xs=xp(j)
1116        ye=yp(1)
1117        ys=yp(j)
1118     end if
1119     simag=simag+ fimag(x,xs,xe,y,ys,ye)
1120  end do
1121  if (abs(simag).gt.tiny) then
1122     in_poly=.true.
1123  else
1124     in_poly=.false.
1125  end if
1126
1127  end
1128
1129  !------------------------------------------
1130
1131  real*4 function fimag(x0,xs,xe,y0,ys,ye)  !Used by PolySelect subroutine
1132  implicit none
1133
1134  real*4 x0,xs,xe,y0,ys,ye
1135  real*4 top,bot
1136
1137  top= -(xe-x0)*(ys-y0)+(ye-y0)*(xs-x0)
1138
1139  bot=  (xe-x0)*(xs-x0)+(ye-y0)*(ys-y0)
1140
1141  fimag=atan2(top,bot)
1142
1143  end
1144
1145  !-------------------------------------------------------------------------------
1146  !------------------------------Libpress Algorithms------------------------------
```

Program: MCMCTRGBTester2.f95

Creation Date: 30 July 2010 (first version 25 Mar 2010)

Relevant Section: §2.3 of Paper I (Ch. 3)

Notes: This program was written to test the performance of the TRGB algorithm (i.e. BayesianTRGB_ANDI.f95) for different luminosity functions (LFs) that might be encountered. A model LF is created with both the tip magnitude and RGB slope constant at $mag\_tip = 20.5$ and $a = 0.3$ respectively. The fraction of background stars (f) in the LF is varied however as is the number of stars populating the LF (*ndata*). In practice, a perl script was written to run this code for all combinations of f and *ndata*, where $f \in \{0.1, 0.2, \ldots 0.9\}$ and $ndata \in \{10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10000, 20000\}$. Many of the subroutines in this program are omitted for the sake of brevity, but their form can be ascertained from the 'BayesianTRGB_ANDI.f95' program. The subroutine that actually generates the artificial stars from the model LF is however shown - 'DataMake.'

```
1  MODULE Global !Defines all variables used by BayesianTRGB
2  IMPLICIT NONE
3
4  INTEGER :: i, j, k, l, eval, idum = -9999, it, nit
5  INTEGER :: ndata_max, nsamples, binspm, nbins, cmod_nbins, ghw, mm, ios
6  PARAMETER (ndata_max = 20000000, nsamples = 100)
7  PARAMETER (binspm = 100)
8  PARAMETER (nbins = 8*binspm + 1)
9  PARAMETER (nit = 200000)
10 INTEGER :: ndata, ndata2
11 INTEGER :: d1, d2, d3, d4, d5, d6, field_num
12 INTEGER :: flimBins, blimBins
13 REAL*8 :: blim, flim, array(ndata_max)
14 REAL*8 :: c_o_f(2)
15 REAL*8 :: randnum1, randnum2, randnum3, randnum4, r1, r2, B1, B2, B3, B4, hb = 0.005d0
16 REAL*8 :: model(nbins,2), cmodel(nbins,2), magnitude(ndata_max)
17 REAL*8 :: histo_fine(nbins,2), histo_coarse(INT(0.25*(nbins-1.d0)) + 1,2)
18 REAL*8 :: data(ndata_max), cumulative_cmodel(nbins,2), f, f_hold, bfm(nbins)
19 REAL*8 :: mag_tip, mag, mag_cutoff = 24.e0, a, inputs(4)
20 REAL*8 :: area, area2
21 REAL*8 :: modelnoise(nbins,2), noise(nbins) = 0.d0, pi = ACOS(-1.e0)
22 REAL*8 :: kernel(nbins,2) = 0.e0, scale, uplim, lowlim, gx
23 REAL*8 :: temp(nbins,2) = 0.e0, t
24 REAL*8 :: logL, prob, LikeA, LikeB
25 REAL*8 :: tip(nsamples), tip_ord(nsamples), maxlogL(nsamples) = -999999999999.
26 REAL*8 :: tip_rec, tip_offset, tip_sigma, Toffset_kpc, Tsigma_kpc
27 REAL*8 :: f_offset, tip_kpc, kpc_err, f_sigma, a_offset, a_sigma
28 REAL*8 :: f_rec, a_rec, tip_counts, f_counts, a_counts
29 REAL*8 :: tipminsig, tiplusig, fminsig, fplusig, aminsig, aplusig
30 REAL*8 :: x1(nit), x2(nit), x3(nit), p(3), time(nit), r
31 REAL*8 :: post_y1(10*(nbins-1)+1) = 0.d0, post_x1(10*(nbins-1)+1), mlim
32 REAL*8 :: post_y2(nbins) = 0.d0, post_x2(nbins)
33 REAL*8 :: post_y3(2*nbins - 1) = 0.d0, post_x3(2*nbins - 1)
34 REAL*8 :: offset_kpc, PPD_peak
35 CHARACTER :: argv*10, test*40, ch1*9, ch2*9, ch3*9, ch4*9, ch5*9, string*80
```

```
36
37
38  INTEGER :: iCCDt, clsg, clsi, ifieldt, iacc_t
39  REAL*4 :: xgt, ygt, g, dg, im, dim, xki_t, eta_t, FeH_phot_t, diff_tip_t, E_BV_t
40  REAL*8 :: ra_t, de_t
41
42  REAL*4 :: mag_g(ndata_max), mag_i(ndata_max), xki(ndata_max), eta(ndata_max)
43  REAL*4 :: g_min_i(ndata_max), mag_i_poly(ndata_max), g_min_i_poly(ndata_max)
44  REAL*4 :: gmi
45
46  END MODULE Global
47
48  !-------------------------------------------------------------------------
49
50  PROGRAM MCMCTRGBTester2      !Master program
51  USE Global
52  IMPLICIT NONE
53
54  mm = IARGC()
55
56  IF (mm==4) THEN                                  !
57     CALL GETARG(1, argv)                          !
58     READ (argv,*,iostat=ios) mag_tip              !
59     CALL GETARG(2, argv)                          !
60     READ (argv,*,iostat=ios) a                    !
61     CALL GETARG(3, argv)                          !
62     READ (argv,*,iostat=ios) ndata                !Indicates the arguments to be
63     CALL GETARG(4, argv)                          !set in the command line
64     READ (argv,*,iostat=ios) f                    !
65  ELSE                                             !
66     WRITE(*,*) "You must enter 4 arguments:"      !
67     stop ;                                        !
68  END IF                                           !
69
70  WRITE (*,*) " "
71  WRITE (*,*) "Model mag_tip/slope/# sources/background height =", mag_tip, a, ndata, f
72
73  WRITE (ch1,*) mag_tip        !
74  WRITE (ch2,*) a              !
75  WRITE (ch3,*) ndata          !
76  IF (f .eq. 0.d0) THEN        !Generate test identifying character string
77     WRITE (ch4,*) '0'         !to become file name using mag_tip, ndata and f
78  ELSE                         !e.g. 'MCMC_Test/T_20.5-0.3-1000-0.2'
79     WRITE (ch4,*) f           !
80  END IF                       !
81
82  inputs(1) = mag_tip
83  inputs(2) = a
84  inputs(3) = ndata
85  inputs(4) = f
86
87  ndata2 = 0
88
89  WRITE (test,*) 'MCMC_Test/T_' // TRIM(ADJUSTL(ch1)) &
90                                // '-' // TRIM(ADJUSTL(ch2)) &
91                                // '-' // TRIM(ADJUSTL(ch3)) &
92                                // '-' // TRIM(ADJUSTL(ch4))
93  CALL random_seed       !
94                         !
95
96  flimBins = INT(REAL((23.5d0 - 18.d0)) * binspm) + 1
```

```
97  blim = 18.0d0 ; flim = 23.5d0 ; mlim = blim
98
99  CALL NoiseMake      !Generates the convoluted model for the inputted
100 CALL ModelMake      !tip magnitude and then uses to generate sets
101 CALL Convolution    !of data points in 'DataMake' subroutine
102
103 blim = 20.0d0 ; flim = 21.0d0 ; mlim = blim
104 flimBins = INT(REAL((flim - 18.d0)) * binspm) + 1
105 blimBins = INT(REAL((blim - 18.d0)) * binspm) + 1
106
107 CALL DataMake
108
109 DO i = 1, ndata
110    IF (data(i) .ge. blim .AND. data(i) .le. flim) THEN
111       ndata2 = ndata2 + 1
112       data(ndata2) = data(i)
113    END IF
114 END DO
115 WRITE (*,*) "Number␣of␣stars␣in␣fitted␣range:", ndata2
116
117 CALL NoiseMake         !
118 CALL MCMC              !CALL
119 CALL TipAndSigma       !MCMC
120 CALL PosteriorPlot     !SUBROUTINES
121 CALL OtherPlots        !
122 CALL DataHist          !
123
124 WRITE (*, '(3a11)') "␣␣␣tip␣mag:", "␣␣␣sigma:", "␣␣␣␣offset:"   !
125 WRITE (*, '(3F11.3)') tip_rec, tip_sigma, tip_offset            !
126 WRITE (*, '(3a11)') "␣␣␣f:␣␣␣␣␣␣", "␣␣␣sigma:", "␣␣␣␣offset:"   !Write results
127 WRITE (*, '(3F11.3)') f_rec, f_sigma, f_offset                  !to file
128 WRITE (*, '(3a11)') "␣␣␣a:␣␣␣␣␣␣", "␣␣␣sigma:", "␣␣␣␣offset:"   !
129 WRITE (*, '(3F11.3)') a_rec, a_sigma, a_offset                  !
130 WRITE (*,*) "Distance␣=", REAL(tip_kpc), "kpc␣p/m", REAL(kpc_err), "kpc"
131 WRITE (*,*) "Distance␣Offset␣=", REAL(offset_kpc), "kpc"
132
133 END PROGRAM MCMCTRGBTester2
134
135 !-------------------------------------------------------------------------
136
137 SUBROUTINE DataMake     !Generates data points from the convolved model
138 USE Global
139 IMPLICIT NONE
140
141 real*8 :: ran1
142
143 cumulative_cmodel(:,1) = cmodel(:,1)
144
145 cumulative_cmodel(1,2) = cmodel(1,2)                              !Effective
146 DO i = 2, cmod_nbins                                             !integral of
147    cumulative_cmodel(i,2) = cumulative_cmodel(i-1,2) + cmodel(i,2)!convolved
148 END DO                                                           !model
149
150
151 DO i = 1, ndata                                    !
152    CALL random_number(randnum4)                    !
153    randnum4 = cumulative_cmodel(blimBins,2) + &    !
154               randnum4 * (cumulative_cmodel(flimBins,2) - cumulative_cmodel(blimBins,2))   !
155    DO j = flimBins, blimBins, -1                   !
156       IF (randnum4 .le. cumulative_cmodel(j,2)) THEN       !Generates 'ndata'
157          IF (randnum4 .gt. cumulative_cmodel(j-1,2)) THEN  !datapoints from
```

```
158                 data(i) = cumulative_cmodel(j,1)                !the convolved
159                 exit;                                           !model
160               END IF                                            !
161             END IF                                              !
162           END DO                                                !
163         END DO                                                  !
164
165         END SUBROUTINE DataMake
166
167         !-------------------------------------------------------------------------------
168         !-----------------------------Libpress Algorithms-------------------------------
```

Program: BayesianTRGBTestPlotterMCMC.f95

Creation Date: 29 April 2010

Relevant Section: Figs. 10 & 11 of Paper I (Ch. 3) and Figs. 4 & 5 of Paper II (Ch. 4)

Notes: This program is used to plot the results returned by 'MCMCTRGBTester2.f95,' namely the one sigma uncertainties and the offset of the recovered tip magnitude from $mag\_tip = 20.5$ for each combination of f and $ndata$. Pixels are created with bounds $X1, X2, Y1, Y2$ with the added complication of a log scale for the x-axis. These pixels are then assigned a shade of grey based on the magnitude of the quantity they represent.

```
1  PROGRAM BayesianTRGBTestPlotterMCMC  !Plots results of tests for different combinations of
2  IMPLICIT NONE                        !f vs. ndata (generates two plots: tip offset for each
3                                       !combination and sigma for each combination)
4  INTEGER :: i, ios, j, k, x(11,9) = 3
5  REAL :: A1(11,9), A2(11,9)
6  REAL :: ndata(1000), ndata_actual(1000), f(1000), offmag(1000), v
7  REAL :: offkpc(1000), sigmag(1000), sigkpc(1000), temp, X1, X2, Y1, Y2
8  REAL :: ALEV(100), TR(6), stars(11,9), noise(11,9)
9  REAL :: grey, xmin, xmax
10 CHARACTER(LEN=15) :: C1(11,9), C2(11,9)
11
12 OPEN (unit = 1, file = './summary.dat', status = 'old')
13 i = 0 ; ios = 0
14 DO WHILE (.TRUE.)  !Reads data until end of input file and puts it into arrays
15    i=i+1
16    READ (1, *, IOSTAT = ios) ndata(i), f(i), sigkpc(i), offkpc(i), temp, temp, temp
17    if (ios == 0) then ;
18    else if (ios == -1) then ;
19       i=i-1
20       exit ;
21    else if (ios > 0) then ;
22       i=i-1
23       cycle
24    end if
25 END DO
26
27 DO j = 1, 11                          !
28    DO k = 1, 9                        !
29       A1(j,k) = sigkpc(((j-1)*9)+k)   !
30       IF (A1(j,k) .eq. 0.d0) THEN     !
31          A1(j,k) = 0.001d0            !
32       END IF                          !Puts results
33       A2(j,k) = offkpc(((j-1)*9)+k)   !
34       IF (A2(j,k) .eq. 0.d0) THEN     !into ndata x f
35          A2(j,k) = 0.001d0            !
36       END IF                          !arrays for
37       stars(j,k) = ndata(((j-1)*9)+k) !
38       noise(j,k) = f(((j-1)*9)+k)     !easy plotting
39       WRITE (C1(j,k),*) NINT(A1(j,k)) !
40       WRITE (C2(j,k),*) NINT(A2(j,k)) !
41    END DO                             !
42 END DO                               !
43
44 !-------------------------------------------------------------------
45
```

```
46  CALL pgbegin(0,'MF_TRGBSigmaMCMC.ps/CPS',1,1)  !Generates sigma plot
47
48  CALL pgenv(0.8, 4.5, 0., 1., 0, 10)
49
50  xmin = MINVAL(A1)
51  xmax = MAXVAL(A1)
52
53  DO i = 1, 11
54     DO j = 1, 9
55
56        IF (MOD(i,3) .eq. 1) THEN                    !
57           X1 = LOG10(10.**(((i-1)/3)+1)) - 0.13     !
58           X2 = LOG10(10.**(((i-1)/3)+1)) + 0.13     !
59        END IF                                       !Generate pixel
60                                                     !
61        IF (MOD(i,3) .eq. 2) THEN                    !x boundaries for
62           X1 = LOG10(2. * 10.**(((i-1)/3)+1)) - 0.13 !
63           X2 = LOG10(2. * 10.**(((i-1)/3)+1)) + 0.13 !log x axis
64        END IF                                       !
65                                                     !(3 different width
66        IF (MOD(i,3) .eq. 0) THEN                    !calculations required)
67           X1 = LOG10(5. * 10.**(((i-1)/3)+1)) - 0.13 !
68           X2 = LOG10(5. * 10.**(((i-1)/3)+1)) + 0.13 !
69        END IF                                       !
70
71        Y1 = j * 0.1e0 - 0.04                        !Generate
72                                                     !pixel
73        Y2 = j * 0.1e0 + 0.04                        !y boundaries
74
75        grey = (xmax - A1(i,j)) / (xmax - xmin)      !Determine shade of grey
76
77        CALL pgscr(3, grey, grey, grey)
78
79        CALL pgpixl(x, 11, 9, i, i, j, j, X1, X2, Y1, Y2)  !make pixels
80
81        CALL pgsci(2)
82
83        CALL pgptxt(LOG10(stars(i,j)), noise(i,j), 0., 0.5, C1(i,j)) !put value in pixels
84     END DO
85  END DO
86
87  CALL pgsci(1)
88  CALL pglab('number␣of␣stars', 'proportion␣of␣background␣stars', &
89             'Sigma␣(kpc)␣-␣One␣Sigma␣Error')
90
91  CALL pgend
92
93  !-----------------------------------------------------------------
94
95  CALL pgbegin(0,'MF_TRGBOffsetMCMC.ps/CPS',1,1)  !Generates offset plot
96  CALL pgenv(0.8, 4.5, 0., 1., 0, 10)
97
98  A2 = ABS(A2)
99  xmin = MINVAL(A2)
100 xmax = MAXVAL(A2)
101
102 DO i = 1, 11
103    DO j = 1, 9
104
105       IF (MOD(i,3) .eq. 1) THEN                    !
106          X1 = LOG10(10.**(((i-1)/3)+1)) - 0.13     !
```

```
107             X2 = LOG10(10.**(((i-1)/3)+1)) + 0.13          !
108         END IF                                             !Generate pixel
109                                                            !
110         IF (MOD(i,3) .eq. 2) THEN                          !x boundaries for
111             X1 = LOG10(2. * 10.**(((i-1)/3)+1)) - 0.13     !
112             X2 = LOG10(2. * 10.**(((i-1)/3)+1)) + 0.13     !log x axis
113         END IF                                             !
114                                                            !(3 different width
115         IF (MOD(i,3) .eq. 0) THEN                          !calculations required)
116             X1 = LOG10(5. * 10.**(((i-1)/3)+1)) - 0.13     !
117             X2 = LOG10(5. * 10.**(((i-1)/3)+1)) + 0.13     !
118         END IF                                             !
119
120         Y1 = j * 0.1e0 - 0.04                              !Generate
121                                                            !pixel
122         Y2 = j * 0.1e0 + 0.04                              !y boundaries
123
124         grey = (xmax - A2(i,j)) / (xmax - xmin)            !Determine shade of grey
125
126         CALL pgscr(3, grey, grey, grey)
127
128         CALL pgpixl(x, 11, 9, i, i, j, j, X1, X2, Y1, Y2) !make pixels
129
130         CALL pgsci(2)
131
132         CALL pgptxt(LOG10(stars(i,j)), noise(i,j), 0., 0.5, C2(i,j)) !put value in pixels
133     END DO
134 END DO
135
136 CALL pgsci(1)
137 CALL pglab('number␣of␣stars', 'proportion␣of␣background␣stars', &
138             'Offset␣(kpc)␣-␣Radial␣Distance␣Offset')
139
140 CALL pgend
141
142 !---------------------------------------------------------------------
143
144
145 END PROGRAM BayesianTRGBTestPlotterMCMC
```

C

Chapter Four Programs

Program: MF_TRGB.f95

Creation Date: 31 August 2011 (first version 8 Dec 2010) Many modifications.

Relevant Section: Ch. 4

Notes: This program is the successor of 'BayesianTRGB_ANDI.f95' in Appendix B and similarly lies at the heart of the material presented in Paper II (Ch. 4). The principal difference between the two is that this new version incorporates a density matched filter weighting scheme, where by stars are given a weight based on their position with in the object's density profile. In this way, stars that are more likely to be true object member's are given greater consideration during the luminosity function fitting. The actual weighting itself is taken care of by the 'Weighter' subroutine, but other subroutines have been modified significantly to handle it. The background component of the LF (built in 'NoiseMake') for instance is no longer added to the model LF in the 'ModelMake' subroutine. This is because with the weighting switched on, each star effectively has its own model LF with the ratio of background to RGB component tailored to suit the star's probability of being a true object member. Hence, these ratios are now taken into account in the 'LogLike' subroutine on a star-by-star basis. There are many other additions. A new LF plotting subroutine 'w_DataHist' plots the weighted LF and a plot of the object density profile is created in the 'Weighter' subroutine. Parallel tempering has been added to the 'MCMC' subroutine and the run-speed of the whole algorithm has be greatly improved by fixing up a design flaw in the way the convolution step was being done (The 'GaussianKernel' subroutine is now called just once and the values are saved). The program also now takes command line input so that the one set of code can be used for all objects. Due to the large number of changes made to most of the subroutines originally written for 'BayesianTRGB_ANDI.f95,' I have reproduced the whole program here rather than omitting the duplicate subroutines. Note that the command line inputs for each satellite are provided as the next item in this appendix for completeness. For a more in depth description of the workings of the program in general, see Paper II - particularly §2 and §3.1.

```
1  MODULE Global !Defines all variables used by BayesianTRGB
2  IMPLICIT NONE
3
4  !--------------------General Program Parameters----------------------
5  INTEGER :: i, j, k, l, eval, idum = -9999, it, nit
6  INTEGER :: ndata_max, nsamples, binspm, nbins, cmod_nbins, mm, ios
```

```
7   PARAMETER (ndata_max = 10000000, nsamples = 100)
8   PARAMETER (binspm = 100)
9   PARAMETER (nbins = 8*binspm + 1)
10  PARAMETER (nit = 500000)
11  INTEGER :: ndata, ndata2
12  INTEGER :: d1, d2, d3, d4
13  INTEGER :: ghw(nbins)
14  REAL*8 :: blim, flim, pi
15  PARAMETER (blim = 19.5d0)
16  PARAMETER (flim = 23.5d0)
17  PARAMETER (pi = ACOS(-1.e0))
18  INTEGER :: blimBins = INT(REAL((blim - 18.d0) * binspm)) + 1
19  INTEGER :: flimBins = INT(REAL((flim - 18.d0) * binspm)) + 1
20  REAL*8 :: randnum1, randnum2, randnum3, randnum4, randnum5
21  INTEGER :: randint
22  REAL*8 :: r1, r2, spotR, hb = 0.005d0
23  REAL*8 :: model(nbins,2), cmodel(nbins,2), magnitude(ndata_max)
24  REAL*8 :: histo_fine(nbins,2), histo_coarse(INT(0.25*(nbins-1.d0)) + 1,2)
25  REAL*8 :: w_histo_fine(nbins,2), w_histo_coarse(INT(0.25*(nbins-1.d0)) + 1,2)
26  REAL*8 :: data(ndata_max), cumulative_cmodel(nbins,2), f, f_hold, bfm(nbins)
27  REAL*8 :: mag_tip, mag, mag_cutoff = 24.e0, a
28  REAL*8 :: area, area2
29  REAL*8 :: modelnoise(nbins,2), noise(nbins) = 0.d0 , bg(nbins) = 0.d0
30  REAL*8 :: kernel(nbins,2,nbins) = 0.e0, scale, uplim, lowlim, gx
31  REAL*8 :: temp(nbins,2) = 0.e0, t
32  INTEGER :: starbin
33  REAL*8 :: tip(nsamples), tip_ord(nsamples), maxlogL(nsamples) = -999999999999.
34  REAL*8 :: tip_rec, tip_offset, tip_psigma, tip_msigma, Toffset_kpc, Tsigma_kpc
35  REAL*8 :: f_offset, tip_kpc, kpc_perr, kpc_merr, f_sigma, a_offset, a_sigma
36  REAL*8 :: f_rec, a_rec, tip_counts, f_counts, a_counts
37  REAL*8 :: tipminsig, tiplusig, fminsig, fplusig, aminsig, aplusig
38  REAL*8 :: mcounts, pcounts
39  INTEGER :: num_chains, cn, chain_compare, swap_count
40  PARAMETER (num_chains = 4)
41  REAL*8 :: swaprate = 1.d0/ 30.d0, logL(num_chains), LikeA(num_chains), LikeB(num_chains)
42  REAL*8 :: prob, sig_prob, bg_prob
43  REAL*8 :: beta, betaholder(num_chains) = (/ 1.d0, 0.25d0, 0.111d0, 0.001d0 /)
44  REAL*8 :: m_step(num_chains) = (/ 0.03d0, 0.06d0, 0.12d0, 0.3d0 /)
45  REAL*8 :: f_step(num_chains) = (/ 0.02d0, 0.04d0, 0.08d0, 0.2d0 /)
46  REAL*8 :: a_step(num_chains) = (/ 0.02d0, 0.04d0, 0.08d0, 0.2d0 /)
47  REAL*8 :: PTAR, par_hold(4)
48  REAL*8 :: x1(nit,num_chains), x2(nit,num_chains), x3(nit,num_chains), p(3), time(nit), r
49  REAL*8 :: post_y1(10*(nbins-1)+1) = 0.d0, post_x1(10*(nbins-1)+1), mlim
50  REAL*8 :: d_blim, bg_blim, d_flim, bg_flim
51  REAL*8 :: post_y2(nbins) = 0.d0, post_x2(nbins)
52  REAL*8 :: post_y3(2*nbins - 1) = 0.d0, post_x3(2*nbins - 1)
53  REAL*8 :: PPD_peak, Best_Combo(6)
54  CHARACTER :: argv*30, field*30, colcut*30, ch1*9, ch2*9, ch3*9, ch4*9, ch5*9, string*60, command*200
55  INTEGER :: scout_counts
56  LOGICAL :: not_scout
57
58  !--------------------For reading in PAndAS data--------------------
59  INTEGER :: iCCDt, clsg, clsi, ifieldt, iacc_t
60  REAL*4 :: xgt, ygt, g, dg, im, dim, xki_t, eta_t, FeH_phot_alan_t, FeH_phot_t, diff_tip_t, E_BV_t
61  REAL*8 :: ra_t, de_t
62
63  REAL*4 :: mag_g(ndata_max), mag_i(ndata_max), xki(ndata_max), eta(ndata_max)
64  REAL*4 :: g_min_i(ndata_max), mag_i_poly(ndata_max), g_min_i_poly(ndata_max)
65  REAL*4 :: mag_g_poly(ndata_max), xi_poly(ndata_max), eta_poly(ndata_max)
66  REAL*4 :: gmi, xi_all(ndata_max), eta_all(ndata_max)
67
```

```
68  LOGICAL :: truestar(ndata_max), truestar_poly(ndata_max)
69
70  !------Additional parameters for calculating background stats-------
71  INTEGER :: bg_ndata, bg_ndata2, bg_ndata3
72  REAL*4 :: bg_mag_g(ndata_max), bg_mag_i(ndata_max), bg_xki(ndata_max), bg_eta(ndata_max)
73  REAL*4 :: bg_g_min_i(ndata_max), bg_mag_i_poly(ndata_max), bg_g_min_i_poly(ndata_max)
74  REAL*4 :: bg_mag_g_poly(ndata_max), bg_gmi
75  REAL*8 :: bg_data(ndata_max)
76
77    !--SVD fitting of background--
78    INTEGER ma, mp, np, ndat
79    PARAMETER (ndat = INT(0.25*(nbins-1.d0)) + 1)
80    PARAMETER (np = 8)
81    PARAMETER (mp = ndat)
82    PARAMETER (ma = np)
83    REAL :: chisq, ay(ma), sig(ndat), u(mp,np), v(np,np), w(np), xa(ndat), ya(ndat)
84    REAL :: xt(ndat), yt(ndat)
85    REAL*8 :: bg_histo_coarse(ndat,2)
86    EXTERNAL :: funcs
87
88  !------Additional parameters for specifying object coordinates------
89  INTEGER :: Jop
90  REAL*8 :: XIop, ETAop
91  REAL*8 :: RAh, RAm, RAs, DecD, DecM, DecS, RA_rad, Dec_rad
92  REAL*8 :: tpRAh, tpRAm, tpRAs, tpDecD, tpDecM, tpDecS, tpRA_rad, tpDec_rad
93
94  !--Additional parameters for Matched Filters Subroutine 'Weighter'--
95  INTEGER :: rhobins, rhobins2
96  PARAMETER (rhobins = 40)
97  REAL*4 :: C_O_F_dist(ndata_max), Density(rhobins,2), Den_sig(rhobins), rhofit(rhobins,2), weightplot(500,2)
98  REAL*8 :: weight(ndata_max), scaled_a(ndata_max), scaled_a_all(ndata_max), crowded_rad, ellipse_stars, ellipse_area
99  REAL*8 :: ellip, HLR, PA, xdash, ydash, maxweight, maxa, SR, den_prof_scale, outer_rad
100 REAL*8 :: weightsum, Densitysum
101
102 !-------------------------When f is known--------------------------
103 INTEGER :: bg_stars, sig_stars
104 REAL*8 :: bg_area, sig_area
105 REAL*8 :: known_f, bg_stars_in_sig_field
106 REAL*8 :: sig_field_radius, bg_low_xi, bg_up_xi
107
108 END MODULE Global
109
110 !-----------------------------------------------------------------------
111
112 PROGRAM BayesianTRGBsatellite     !Master program
113 USE Global
114 IMPLICIT NONE
115
116 mm = IARGC()
117
118 IF (mm==16) THEN                              !
119    CALL GETARG(1, argv)                       !
120    READ (argv,*,iostat=ios) field             !
121    CALL GETARG(2, argv)                       !
122    READ (argv,*,iostat=ios) RAh               !
123    CALL GETARG(3, argv)                       !
124    READ (argv,*,iostat=ios) RAm               !
125    CALL GETARG(4, argv)                       !
126    READ (argv,*,iostat=ios) RAs               !
127    CALL GETARG(5, argv)                       !
128    READ (argv,*,iostat=ios) DecD              !
```

```
129     CALL GETARG(6, argv)                        !
130     READ (argv,*,iostat=ios) DecM               !
131     CALL GETARG(7, argv)                        !
132     READ (argv,*,iostat=ios) DecS               !
133     CALL GETARG(8, argv)                        !
134     READ (argv,*,iostat=ios) ellip              !Indicates the arguments to be
135     CALL GETARG(9, argv)                        !set in the command line
136     READ (argv,*,iostat=ios) HLR                !
137     CALL GETARG(10, argv)                       !
138     READ (argv,*,iostat=ios) PA                 !
139     CALL GETARG(11, argv)                       !
140     READ (argv,*,iostat=ios) crowded_rad        !
141     CALL GETARG(12, argv)                       !
142     READ (argv,*,iostat=ios) outer_rad          !
143     CALL GETARG(13, argv)                       !
144     READ (argv,*,iostat=ios) sig_field_radius   !
145     CALL GETARG(14, argv)                       !
146     READ (argv,*,iostat=ios) bg_low_xi          !
147     CALL GETARG(15, argv)                       !
148     READ (argv,*,iostat=ios) bg_up_xi           !
149     CALL GETARG(16, argv)                       !
150     READ (argv,*,iostat=ios) colcut             !
151 ELSE                                            !
152     WRITE(*,*) "You must enter 16 arguments:"   !
153     stop ;                                      !
154 END IF                                          !
155
156  string = TRIM(ADJUSTL(field)) // '/results.dat'
157 OPEN(3, file=TRIM(ADJUSTL(string)), status = 'unknown')
158 WRITE (3,*) "Field Name:", field
159
160 CALL positionFinder   !
161                       !
162 CALL random_seed      !
163                       !
164 CALL M31DataReader    !
165 CALL Weighter         !
166 CALL SVDFitter        !
167 CALL GaussianKernel   !
168 CALL NoiseMake        !CALL
169 CALL MCMC             !
170 !CALL PosteriorPlot   !SUBROUTINES
171                       !
172 CALL TipAndSigma      !
173 CALL PosteriorPlot    !
174 CALL OtherPlots       !
175 CALL DataHist         !
176 CALL w_DataHist       !
177
178 IF (num_chains .ne. 1) THEN
179 WRITE (3,*) "Proposed Swaps with Cold Sampler Chain:", chain_compare
180 WRITE (3,*) "Accepted Swaps with Cold Sampler Chain:", swap_count
181 WRITE (3,*) "Parallel Tempering Acceptance Rate:", REAL(swap_count)/ REAL(chain_compare)
182 END IF
183 WRITE (3, '(3a11)') "   tip_mag:", "  +sigma:  ", "  -sigma: "      !
184 WRITE (3, '(3F10.3)') tip_rec, tip_psigma, tip_msigma                 !
185 WRITE (3, '(2a11)') "   f:      ", "   sigma:  "                     !
186 WRITE (3, '(2F10.3)') f_rec, f_sigma                                 !Write results
187 WRITE (3, '(2a11)') "   a:      ", "   sigma:  "                     !to file
188 WRITE (3, '(2F10.3)') a_rec, a_sigma                                 !
189 WRITE (3,*) "Distance =", REAL(tip_kpc), "kpc"                       !
```

```
190  WRITE (3,*) "Error = +", kpc_perr, "kpc -", kpc_merr, "kpc"          !
191
192  END PROGRAM BayesianTRGBsatellite
193
194  !-------------------------------------------------------------------------
195
196  SUBROUTINE PositionFinder !Converts object RA and Dec into M31 tangent plane coordinates xi and
197  USE Global                !eta. These are used in the next subroutine for reading in all stars
198  IMPLICIT NONE             !from the PAndAS survey with in some radius of the object center
199
200  tpRAh = 0.d0    !
201  tpRAm = 42.d0   !
202  tpRAs = 44.33d0 !RA and Dec of tangent point
203  tpDecD = 41.d0  !(i.e. M31)
204  tpDecM = 16.d0  !
205  tpDecS = 7.5d0  !
206
207  !||Perform
208  !\/Conversion
209  RA_rad = (pi/180.d0) * (RAh * 15.d0 + RAm * (15.d0/60.d0) + RAs * (15.d0/3600.d0))
210
211  Dec_rad = (pi/180.d0) * (DecD + DecM/60.d0 + DecS/3600.d0)
212
213  tpRA_rad = (pi/180.d0) * (tpRAh * 15.d0 + tpRAm * (15.d0/60.d0) + tpRAs * (15.d0/3600.d0))
214
215  tpDec_rad = (pi/180.d0) * (tpDecD + tpDecM/60.d0 + tpDecS/3600.d0)
216
217  CALL sla_DS2TP (RA_rad, Dec_rad, tpRA_rad, tpDec_rad, XIop, ETAop, Jop)
218  !/\
219  !||
220
221  XIop = XIop * (180.d0/pi)   !tangent plane coordinates
222  ETAop = ETAop * (180.d0/pi) !(i.e. PAndAS xi and eta)
223
224  WRITE (3,*) "C.O.F. Xi =", XIop, "C.O.F. Eta =", ETAop
225
226  END SUBROUTINE PositionFinder
227
228  !-------------------------------------------------------------------------
229  SUBROUTINE M31DataReader !The field to be analysed is specified here
230  USE Global
231  IMPLICIT NONE
232
233  OPEN(1,file='../../../PANDAS/M31_unique_con.dat',form='unformatted',status='old')
234
235  i = 0 ; j = 0
236
237  DO WHILE (.true.)
238     READ(1,IOSTAT=ios) ra_t,de_t,iCCDt,xgt,ygt, &          !Read in data
239     g,dg,clsg,im,dim,clsi,ifieldt,xki_t,eta_t, &           !from binary
240     FeH_phot_alan_t, FeH_phot_t,diff_tip_t,E_BV_t,iacc_t   !format data file
241
242     IF (ios.ne.0) exit
243
244     g=g-3.793*E_BV_t              !Extinction
245     im=im-2.086*E_BV_t            !Corrections
246     gmi = g - im
247
248     if (clsi.ne.-1 .and. clsi.ne.-2) cycle   !Regects
249     if (clsg.ne.-1 .and. clsg.ne.-2) cycle   !non stars
250     if (iacc_t .ne. 1) cycle
```

```
251     if (18.0.le.im.and.im.le.24.0) then     !Specifies
252     else                                    !magnitude
253        cycle                                !range to
254     end if                                  !include
255     if (-2.5.le.FeH_phot_alan_t.and.FeH_phot_alan_t.le.-1.5) then !
256     else                                                !Specifies metallicity
257       ! cycle                                           !range to include
258     end if                                              !
259
260     spotR = SQRT((ABS(eta_t - (ETAop)))**2 + (ABS(xki_t - (XIop)))**2)
261
262     IF (spotR .lt. sig_field_radius) THEN
263        i = i + 1
264
265        IF (i .gt. ndata_max) exit
266
267        mag_g(i)=g            !
268        mag_i(i)=im           !
269        g_min_i(i)=gmi        !If all conditions are met, add star data to signal arrays
270        xki(i)=xki_t          !
271        eta(i)=eta_t          !
272        IF (ifieldt .ge. 0) THEN        !
273           truestar(i) = .true.         !Distinguish between
274        ELSE                            !real data and artificial
275           truestar(i) = .false.        !background
276        END IF                          !
277
278     ELSE IF (xki_t .ge. bg_low_xi .and. xki_t .le. bg_up_xi) THEN
279        IF (eta_t .ge. ETAop - 0.5d0 .and. eta_t .le. ETAop + 0.5d0) THEN
280           j = j + 1
281
282           IF (j .gt. ndata_max) exit
283
284           bg_mag_g(j)=g       !
285           bg_mag_i(j)=im      !
286           bg_g_min_i(j)=gmi   !If all conditions are met, add star data to bckgrnd arrays
287           bg_xki(j)=xki_t     !
288           bg_eta(j)=eta_t     !
289        END IF
290     END IF
291
292  END DO
293
294  ndata = i ; bg_ndata = j
295
296  sig_area = pi * (sig_field_radius ** 2.d0)  !Calculate area of signal field
297  bg_area = 1.d0 * (bg_up_xi - bg_low_xi) - (pi * (sig_field_radius ** 2.d0)) !Calculate area of BG field
298
299  DO i = 1, ndata
300     data(i) = mag_i(i)  !
301     xi_all(i) = xki(i)  !Object stars before applying colour cut
302     eta_all(i) = eta(i)
303  END DO
304
305  DO j = 1, bg_ndata
306     bg_data(j) = bg_mag_i(j) !BG stars before applying colour cut
307  END DO
308
309  CALL M31DataPlotter
310
311  END SUBROUTINE M31DataReader
```

```
312
313 !------------------------------------------------------------------------------
314
315 SUBROUTINE M31DataPlotter !Produces plots of the object and background fields
316 USE Global                !as well as their Colour-Magnitude Diagrams. Colour
317 IMPLICIT NONE             !cuts are also implemented in this subroutine
318
319 !---------------------Signal-Field-------------------------
320 string = TRIM(ADJUSTL(field)) // '/sig_field.ps/CPS'
321 CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
322
323 CALL pgenv(MAXVAL(xki, mask = xki .ne. 0.), MINVAL(xki, mask = xki .ne. 0.), &
324 MINVAL(eta, mask = eta .ne. 0.), MAXVAL(eta, mask = eta .ne. 0.), 1, 0)
325 CALL pgslw(3)
326 DO i = 1, ndata
327    IF (truestar(i)) THEN
328       CALL pgpt (1, xki(i), eta(i), -1)
329    ELSE
330       CALL pgpt (1, xki(i), eta(i), 225)
331    END IF
332 END DO
333 CALL pgslw(1)
334 CALL pglab('\(0640) (degrees)', '\(0633) (degrees)', '')
335
336 CALL pgend
337
338 WRITE (command,*) 'convert -rotate 90 ./' // TRIM(ADJUSTL(field)) // &
339                    '/sig_field.ps ./' // TRIM(ADJUSTL(field)) // &
340                    '/sig_field.jpg'
341
342 call system(command)
343
344 !---------------------Signal-CMD-------------------------
345 string = TRIM(ADJUSTL(field)) // '/sig_cmd.ps/CPS'
346 CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
347
348 CALL pgenv(MINVAL(g_min_i, mask = g_min_i .ne. 0.), MAXVAL(g_min_i), &
349 MAXVAL(mag_i), MINVAL(mag_i, mask = mag_i .ne. 0.), 0, 0)
350 CALL pgslw(3)
351 DO i = 1, ndata
352    IF (truestar(i)) THEN
353       CALL pgpt (1, g_min_i(i), mag_i(i), -1)
354    ELSE
355       CALL pgpt (1, g_min_i(i), mag_i(i), 225)
356    END IF
357 END DO
358 CALL pgslw(1)
359 CALL pglab('(g - i)\d0\u', 'i\d0\u', '')
360
361 CALL PolySelect !For CMD colour-cut
362
363 CALL pgend
364
365 WRITE (command,*) 'convert -rotate 90 ./' // TRIM(ADJUSTL(field)) // &
366                    '/sig_cmd.ps ./' // TRIM(ADJUSTL(field)) // &
367                    '/sig_cmd.jpg'
368
369 call system(command)
370
371 !---------------------Bckgrnd-Field-------------------------
372 string = TRIM(ADJUSTL(field)) // '/bg_field.ps/CPS'
```

```
373  CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
374
375  CALL pgenv(MAXVAL(bg_xki, mask = bg_xki .ne. 0.), &
376             MINVAL(bg_xki, mask = bg_xki .ne. 0.), &
377             MINVAL(bg_eta, mask = bg_eta .ne. 0.), &
378             MAXVAL(bg_eta, mask = bg_eta .ne. 0.), 1, 0)
379  CALL pgslw(2)
380  CALL pgpt (bg_ndata, bg_xki, bg_eta, -1)
381  CALL pgslw(1)
382  CALL pglab('\(0640) (degrees)', '\(0633) (degrees)', '')
383
384  CALL pgend
385
386  WRITE (command,*) 'convert -rotate 90 ./' // TRIM(ADJUSTL(field)) // &
387                          '/bg_field.ps ./' // TRIM(ADJUSTL(field)) // &
388                          '/bg_field.jpg'
389
390   call system(command)
391
392  !-----------------------Bckgrnd-CMD-------------------------
393   string = TRIM(ADJUSTL(field)) // '/bg_cmd.ps/CPS'
394  CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
395
396  CALL pgenv(MINVAL(bg_g_min_i, mask = bg_g_min_i .ne. 0.), &
397             MAXVAL(bg_g_min_i), MAXVAL(bg_mag_i), &
398             MINVAL(bg_mag_i, mask = bg_mag_i .ne. 0.), 0, 0)
399  CALL pgslw(3)
400  CALL pgpt (bg_ndata, bg_g_min_i, bg_mag_i, -1)
401  CALL pgslw(1)
402  CALL pglab('(g - i)\d0\u', 'i\d0\u', '')
403
404  CALL PolySelect !For CMD colour-cut
405
406  CALL pgend
407
408  WRITE (command,*) 'convert -rotate 90 ./' // TRIM(ADJUSTL(field)) // &
409                          '/bg_cmd.ps ./' // TRIM(ADJUSTL(field)) // &
410                          '/bg_cmd.jpg'
411
412   call system(command)
413
414  !--------------Input selected data into 'data'-----------
415   string = TRIM(ADJUSTL(field)) // '/i_and_g_in_cut.dat'
416  OPEN(7, file=TRIM(ADJUSTL(string)), status = 'unknown')
417  WRITE(7,*) "          i                  g                   number of stars:", ndata2
418  data = 0.d0 ; xki = 0.d0 ; eta = 0.d0 ; d_blim = 100.d0 ; d_flim = 0.d0
419  DO i = 1, ndata2
420     data(i) = mag_i_poly(i)
421     xki(i) = xi_poly(i)
422     eta(i) = eta_poly(i)
423     IF (data(i) .lt. d_blim) THEN
424        d_blim = data(i)
425     END IF
426     IF (data(i) .gt. d_flim) THEN
427        d_flim = data(i)
428     END IF
429  WRITE (7, '(2F16.5)') mag_i_poly(i), mag_g_poly(i)
430  END DO
431
432  !-------------Signal-Field after colour cut----------------
433   string = TRIM(ADJUSTL(field)) // '/sig_field_cc.ps/CPS'
```

```
434 CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
435
436 CALL pgenv(MAXVAL(xki, mask = xki .ne. 0.), MINVAL(xki, mask = xki .ne. 0.), &
437 MINVAL(eta, mask = eta .ne. 0.), MAXVAL(eta, mask = eta .ne. 0.), 1, 0)
438 CALL pgslw(3)
439 DO i = 1, ndata2
440    IF (truestar_poly(i)) THEN
441       CALL pgpt (1, xki(i), eta(i), -1)
442    ELSE
443       CALL pgpt (1, xki(i), eta(i), 225)
444    END IF
445 END DO
446 CALL pgslw(1)
447 CALL pglab('\(0640) (degrees)', '\(0633) (degrees)', '')
448
449 CALL pgend
450
451 WRITE (command,*) 'convert -rotate 90 ./' // TRIM(ADJUSTL(field)) // &
452                    '/sig_field_cc.ps ./' // TRIM(ADJUSTL(field)) // &
453                    '/sig_field_cc.jpg'
454
455 call system(command)
456
457 !-------Input selected background data into 'bg_data'-------
458 bg_data = 0.d0 ; bg_blim = 100.d0 ; bg_flim = 0.d0
459 DO i = 1, bg_ndata2
460    bg_data(i) = bg_mag_i_poly(i)
461    IF (bg_data(i) .lt. bg_blim) THEN
462       bg_blim = bg_data(i)
463    END IF
464    IF (bg_data(i) .gt. bg_flim) THEN
465       bg_flim = bg_data(i)
466    END IF
467 END DO
468
469 !---Set parameters for calculation of background height---
470
471 sig_stars = ndata2      !Total number of stars in signal field
472 bg_stars = bg_ndata3    !Number of stars in background field
473 bg_stars_in_sig_field = REAL(bg_stars) * (sig_area/bg_area)!  ; bg_stars = 0.d0
474 !Number of Background stars in signal field
475
476 WRITE (3,*) "Number of data points:", sig_stars
477 WRITE (3,*) "Average Field SNR:", (REAL(sig_stars) - bg_stars_in_sig_field) / bg_stars_in_sig_field
478
479 !---------Make coarse data histogram for bckgrnd---------
480
481 DO i = 1, INT(0.25*(nbins-1.d0)) + 1
482    bg_histo_coarse(i,1) = 18.d0 + (i-1.d0)/REAL(0.25*binspm)
483 END DO
484
485 DO i = 1, bg_ndata2
486    bg_histo_coarse(INT((bg_data(i)-18.d0)*0.25*binspm) + 1, 2) = &
487    bg_histo_coarse(INT((bg_data(i)-18.d0)*0.25*binspm) + 1, 2) + 1.d0
488 END DO
489
490 !|| Fill empty bright edge of array with
491 !\/ artificial data for improved fitting
492 DO i = 1, INT((bg_blim - 18.d0) * REAL(binspm/4.d0)) + 4
493    bg_histo_coarse(i,2) = bg_histo_coarse(INT((bg_blim - 18.d0) * REAL(binspm/4.d0)) + 4, 2)
494 END DO
```

```
495
496 END SUBROUTINE M31DataPlotter
497
498 !-------------------------------------------------------------------------
499
500 SUBROUTINE Weighter       !Implements an elliptical weighting scheme using the
501 USE Global                !ellipticity, half-light radius and position angle
502 IMPLICIT NONE             !of the satellite
503
504 HLR = HLR * (1.d0/ 60.d0)          !Convert half-light radius from minutes to degrees
505 SR = HLR / 1.678                   !Convert from half light radius to exponential scale radius
506 PA = PA * (pi/180.d0)              !Convert position angle from degrees to radians
507 ellipse_area = pi * outer_rad * outer_rad * (1.d0 - ellip) !Area of outer ellipse
508
509 ellipse_stars = 0.d0 ; maxweight = 0.d0 ; maxa = 0.d0
510
511 ellipse_area = ellipse_area - pi * crowded_rad * crowded_rad * (1.d0 - ellip) !Subtract area of inner ellipse
512
513
514 DO i = 1, ndata2                                                    !Rotation of coordinates to match orientation
515    xdash = (xki(i) - (XIop)) * cos(PA) - (eta(i) - (ETAop)) * sin(PA)    !of satellite. The new coordinates are then used
516    ydash = (xki(i) - (XIop)) * sin(PA) + (eta(i) - (ETAop)) * cos(PA)    !to find the major axis of the ellipse a given
517    scaled_a(i) = SQRT(ydash**2.d0 + (xdash**2.d0 / (1.d0 - ellip)**2.d0))!star lies on - this equates to circular radius.
518    IF (scaled_a(i) .le. outer_rad .and. scaled_a(i) .ge. crowded_rad) THEN   !
519       ellipse_stars = ellipse_stars + 1.d0                              !Count stars in ellipse_area
520    END IF                                                               !
521    IF (scaled_a(i) .gt. maxa) THEN              !
522       maxa = scaled_a(i)                        !Find major-axis of largest ellipse that passes through field
523    END IF                                       !   (weighting will be smallest for stars on this ellipse)
524 END DO
525
526 WRITE (3,*) "Number_of_data_points_in_annulus:", ellipse_stars
527 WRITE (3,*) "Average_annulus_SNR:", (ellipse_stars - (REAL(bg_stars) * (ellipse_area/bg_area))) / (REAL(bg_stars) * (ellipse_area/
       bg_area))
528
529 den_prof_scale = ((ellipse_stars/ellipse_area) - (bg_stars/bg_area)) * ellipse_area
530 den_prof_scale = den_prof_scale / (2.d0 * pi * SR * ((exp(-crowded_rad/SR) * (SR + crowded_rad)) - (exp(-outer_rad/SR) * (SR +
       outer_rad))))
531 den_prof_scale = den_prof_scale/ (1.d0 - ellip)
532 !/\Calculate scaling factor of elliptical function of shape defined by HLR, ellipticity and PA.
533 !||This is calculated by insuring the total number of stars under the curve matches the number of stars in ellipse_area
534 !Where there is an inner or outer cutoff radius, it is not absolutely necessary to account for this in the scaling
535 !as the exponential profile should account for the variations in density across annuli but for completeness, scaling is achieved
536 !by only measuring the density and number of stars in the annulus used (this becomes very important if a huge HLR is set to remove
537 !weighting as the profile will no longer account for the variation in density in this case!)
538
539
540 DO i = 1, ndata2                                                    !
541    IF (truestar_poly(i) .and. scaled_a(i) .ge. crowded_rad .and. scaled_a(i) .le. outer_rad) THEN
542       weight(i) = exp(-1.d0 * scaled_a(i)/SR) * den_prof_scale      !Apply weights to stars based on elliptical
543    ELSE                                                             !contour they lie on.
544       weight(i) = exp(-1.d0 * maxa/SR) * den_prof_scale             !Artificial stars are given the lowest possible
545    END IF                                                           !weight in this step.
546    IF (weight(i) .gt. maxweight) THEN                               !
547       maxweight = weight(i)                                         !Determine maximum weight for use in
548    END IF                                                           !plotting the weighted LF
549 END DO                                                              !
550
551
552 !||  All subsequent lines in this subroutine are for plotting the density function
553 !\/ "density.ps" and for checking the scaling relative to the data.
```

```
554
555  Density = 0.e0
556  maxa = sig_field_radius
557
558  DO i = 1, ndata2  !Calculate number of stars in each density bin (i.e. elliptical annulus)
559     IF (scaled_a(i) .lt. maxa) THEN
560        Density(INT((scaled_a(i)/maxa) * (rhobins)) + 1, 2) = &
561        Density(INT((scaled_a(i)/maxa) * (rhobins)) + 1, 2) + 1.e0
562     END IF
563     IF (scaled_a(i) .eq. maxa) THEN
564        Density(rhobins,2) = Density(rhobins,2) + 1.d0
565     END IF
566  END DO
567
568  Densitysum = 0.d0
569
570  DO i = 1, rhobins  !Calculate density of stars in each density bin
571     Density(i,1) = (REAL(i)/REAL(rhobins)) * REAL(maxa)  !radius of bin
572     IF (i .eq. 1) THEN
573     Den_sig(i) = SQRT(Density(i,2))/ (pi*(Density(i,1)**2.e0)*(1.e0 - ellip)) !<---Error bars for bin-------------------------!
574     Density(i,2) = Density(i,2)/ (pi*(Density(i,1)**2.e0)*(1.e0 - ellip)) !<---Density of bin-----------------------------------!
575     Densitysum = (Density(i,2) - (bg_stars/bg_area)) * (pi*(Density(i,1)**2.e0)*(1.e0 - ellip))                              ! !
576     ELSE                                                                                                                     ! !
577     Den_sig(i) = SQRT(Density(i,2))/ ((pi*(Density(i,1)**2.e0)*(1.e0 - ellip)) - (pi*(Density(i-1,1)**2.e0)*(1.e0 - ellip)))!<-! !
578     Density(i,2) = Density(i,2)/ ((pi*(Density(i,1)**2.e0)*(1.e0 - ellip)) - (pi*(Density(i-1,1)**2.e0)*(1.e0 - ellip)))!<-------!
579     Densitysum = Densitysum + (Density(i,2) - (bg_stars/bg_area)) * ((pi*(Density(i,1)**2.e0)*(1.e0 - ellip)) - (pi*(Density(i-1,1)**2.
          e0)*(1.e0 - ellip)))
580     END IF
581  END DO
582
583  weightsum = 0.d0
584  DO i = 1, 500    !Calculate values of fitted density profile
585     weightplot(i,1) = (REAL(i)/500.e0) * maxa
586     weightplot(i,2) = exp(-1.e0*(weightplot(i,1))/SR) * den_prof_scale
587     IF(i .eq. 1) THEN
588        weightsum = weightplot(i,2) * (pi*(weightplot(i,1)**2.e0)*(1.e0 - ellip))
589     ELSE
590        weightsum = weightsum + weightplot(i,2) * ((pi*(weightplot(i,1)**2.e0)*(1.e0 - ellip)) - (pi*(weightplot(i-1,1)**2.e0)*(1.e0 -
             ellip)))
591     END IF
592  END DO
593  weightplot(:,2) = weightplot(:,2) + (bg_stars/bg_area)
594
595  WRITE (3,*) "stars_in_model/stars_in_largest_field_ellipse:", weightsum/ Densitysum
596  WRITE (3,*) "stars_in_annulus/stars_in_largest_field_ellipse:", &
597  (((ellipse_stars/ellipse_area) - (bg_stars/bg_area)) * ellipse_area)/ Densitysum
598
599  CALL WeighterPlots
600
601  END SUBROUTINE Weighter
602
603  !-------------------------------------------------------------------------
604
605  SUBROUTINE WeighterPlots   !Plots (log) binned density profile of object and
606  USE Global                 !superimposes the best fit model to the
607  IMPLICIT NONE              !density profile
608
609  INTEGER :: lw
610
611  !----------------------Plots Density Profile histogram
612  string = TRIM(ADJUSTL(field)) // '/density.ps/CPS'
```

```
613  CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
614
615  CALL pgenv(0., 1.1 * REAL(maxa), 0., 1.1*MAXVAL((/MAXVAL(weightplot(:,2)),MAXVAL(Density(:,2) + Den_sig)/)), 0, 0)
616  CALL pgslw(10)
617  CALL pgsci(1)
618
619  DO i = 1, rhobins !Plot density bin values
620     IF (Density(i,1) .le. crowded_rad .or. Density(i,1) .gt. outer_rad) THEN
621        CALL pgsci(5)  !Outside of fitted region
622        CALL pgpt (1, Density(i,1), Density(i,2), -1)
623     ELSE
624        CALL pgsci(1)  !In fitted region
625        CALL pgpt (1, Density(i,1), Density(i,2), -1)
626     END IF
627  END DO
628
629  CALL pgsci(1)
630  CALL pgslw(1)
631
632  DO i = 1, rhobins  !Plot error bars for density bin values
633     IF (Density(i,1) .le. crowded_rad .or. Density(i,1) .gt. outer_rad) THEN
634        CALL pgsci(5)  !Outside of fitted region
635        CALL pgerry(1, Density(i,1), Density(i,2) + Den_sig(i), Density(i,2) - Den_sig(i), 5.)
636     ELSE
637        CALL pgsci(1)  !In fitted region
638        CALL pgerry(1, Density(i,1), Density(i,2) + Den_sig(i), Density(i,2) - Den_sig(i), 5.)
639     END IF
640  END DO
641
642  CALL pgsci(2)  !Plot fit to density bins of object
643  CALL pgline (500, weightplot(:,1), weightplot(:,2))
644  CALL pgsci(3)
645  CALL pgline (2, (/0., REAL(sig_field_radius)/), (/REAL(bg_stars/bg_area), REAL(bg_stars/bg_area)/))
646  CALL pgptxt(0.05*REAL(sig_field_radius), 1.2*REAL(bg_stars/bg_area), 0., 0.5, 'BG')
647  CALL pgsci(1)
648  CALL pglab('Elliptical Radius (degrees)', 'Object stars per sq. degree', '')
649
650  CALL pgend
651
652  WRITE (command,*) 'convert -rotate 90 ./' // TRIM(ADJUSTL(field)) // &
653                    '/density.ps ./' // TRIM(ADJUSTL(field)) // &
654                    '/density.jpg'
655
656  call system(command)
657
658  !------------------Signal-Field-(weighted)------------------
659  string = TRIM(ADJUSTL(field)) // '/sig_field_cc_w.ps/CPS'
660  CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
661
662  CALL pgenv(MAXVAL(xki, mask = xki .ne. 0.), MINVAL(xki, mask = xki .ne. 0.), &
663  MINVAL(eta, mask = eta .ne. 0.), MAXVAL(eta, mask = eta .ne. 0.), 1, 0)
664
665  DO i = 1, 20                                                    !
666     CALL pgscr(i, 0.5*(SIN(1.0*REAL((i+10) * pi/10)) + 1.), &    !
667                   0.5*(SIN(1.0*REAL((i-5) * pi/10)) + 1.), &     !Assign colours to indicies
668                   0.5*(SIN(1.0*REAL((i) * pi/10)) + 1.))         !
669  END DO                                                          !
670
671  DO i = 1, ndata2
672     lw = nint((weight(i)/maxweight) * 20.d0) + 1
673     CALL pgslw(lw+5)
```

```
674     CALL pgsci(22 - lw)
675     IF (truestar_poly(i)) THEN      !If star is real (not artifical) ...
676        IF (scaled_a(i) .ge. crowded_rad .and. scaled_a(i) .le. outer_rad) THEN !Fitted stars
677           CALL pgpt (1, xki(i), eta(i), -1)
678        ELSE
679           CALL pgslw(1)
680           IF (scaled_a(i) .lt. crowded_rad) THEN   !stars inside crowded region
681           CALL pgsci(3)
682           CALL pgpt (1, xki(i), eta(i), 2779)
683           END IF
684           IF (scaled_a(i) .gt. outer_rad) THEN   !stars outside fitted region
685           CALL pgsci(20)
686           CALL pgpt (1, xki(i), eta(i), 227)!2281)
687           END IF
688        END IF
689     ELSE     !If star is artificial...
690        CALL pgslw(1)
691        CALL pgsci(21)
692        CALL pgpt (1, xki(i), eta(i), 225)
693     END IF
694  END DO
695  CALL pgscr(1, 0., 0., 0.)
696  CALL pgslw(1)
697  CALL pgsci(1)
698  CALL pglab('\(0640) (degrees)', '\(0633) (degrees)', '')
699
700  CALL pgend
701
702  WRITE (command,*) 'convert -rotate 90 ./' // TRIM(ADJUSTL(field)) // &
703                          '/sig_field_cc_w.ps ./' // TRIM(ADJUSTL(field)) // &
704                          '/sig_field_cc_w.jpg'
705
706  call system(command)
707
708  !-------------------Signal-CMD-(weighted)-------------------
709  string = TRIM(ADJUSTL(field)) // '/sig_cmd_w.ps/CPS'
710  CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
711                                                          !
712  CALL pgenv(MINVAL(g_min_i, mask = g_min_i .ne. 0.), MAXVAL(g_min_i), &
713  MAXVAL(mag_i), MINVAL(mag_i, mask = mag_i .ne. 0.), 0, 0)
714
715  DO i = 1, 20                                                !
716     CALL pgscr(i, 0.5*(SIN(1.0*REAL((i+10) * pi/10)) + 1.), &        !
717                   0.5*(SIN(1.0*REAL((i-5) * pi/10)) + 1.), &         !Assign colours to indicies
718                   0.5*(SIN(1.0*REAL((i) * pi/10)) + 1.))             !
719  END DO                                                      !
720
721  DO i = 1, ndata2
722     lw = nint((weight(i)/maxweight) * 20.d0) + 1
723     CALL pgslw(lw+5)
724     CALL pgsci(22 - lw)
725     IF (truestar_poly(i)) THEN     !If star is real (not artifical) ...
726        IF (scaled_a(i) .ge. crowded_rad .and. scaled_a(i) .le. outer_rad) THEN !Fitted stars
727           CALL pgpt (1, REAL(g_min_i_poly(i)), REAL(data(i)), -1)
728        ELSE
729           CALL pgslw(1)
730           IF (scaled_a(i) .lt. crowded_rad) THEN !stars inside crowded region
731           CALL pgsci(3)
732           CALL pgpt (1, REAL(g_min_i_poly(i)), REAL(data(i)), 2779)
733           END IF
734           IF (scaled_a(i) .gt. outer_rad) THEN !stars outside fitted region
```

```
735             CALL pgsci(20)
736             CALL pgpt (1, REAL(g_min_i_poly(i)), REAL(data(i)), 227)
737             END IF
738          END IF
739       ELSE     !If star is artificial...
740          CALL pgslw(1)
741          CALL pgsci(21)
742          CALL pgpt (1, REAL(g_min_i_poly(i)), REAL(data(i)), 225)
743       END IF
744    END DO
745    CALL pgscr(1, 0., 0., 0.)
746    CALL pgscr(2, 1., 0., 0.)
747    CALL pgslw(3)
748    CALL pgsci(1)
749    CALL pgpt (ndata, g_min_i, mag_i, -1)
750    CALL pgslw(1)
751    CALL pglab('(g-i)\d0\u', 'i\d0\u', '')
752
753    CALL PolySelect
754
755    CALL pgend
756
757    WRITE (command,*) 'convert -rotate 90 ./' // TRIM(ADJUSTL(field)) // &
758                           '/sig_cmd_w.ps ./' // TRIM(ADJUSTL(field)) // &
759                           '/sig_cmd_w.jpg'
760
761    call system(command)
762
763    !--------------------Signal-CMD-(included stars only)--------------------
764
765    DO i = 1, ndata                                                    !Rotation of coordinates to match orientation
766       xdash = (xi_all(i) - (XIop)) * cos(PA) - (eta_all(i) - (ETAop)) * sin(PA)    !of satellite. The new coordinates are then used
767       ydash = (xi_all(i) - (XIop)) * sin(PA) + (eta_all(i) - (ETAop)) * cos(PA)    !to find the major axis of the ellipse a given
768       scaled_a_all(i) = SQRT(ydash**2.d0 + (xdash**2.d0 / (1.d0 - ellip)**2.d0))   !star lies on - this equates to circular radius.
769    END DO
770
771    string = TRIM(ADJUSTL(field)) // '/sig_cmd_used.ps/CPS'
772    CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
773                                                          !
774    CALL pgenv(MINVAL(g_min_i, mask = g_min_i .ne. 0.), MAXVAL(g_min_i), &
775    MAXVAL(mag_i), MINVAL(mag_i, mask = mag_i .ne. 0.), 0, 0)
776
777
778    DO i = 1, ndata
779       IF (scaled_a_all(i) .ge. crowded_rad .and. scaled_a_all(i) .le. outer_rad) THEN
780          CALL pgpt (1, REAL(g_min_i(i)), REAL(mag_i(i)), -1)
781       END IF
782    END DO
783
784    CALL pglab('(g-i)\d0\u', 'i\d0\u', '')
785
786    CALL PolySelect
787
788    CALL pgend
789
790    WRITE (command,*) 'convert -rotate 90 ./' // TRIM(ADJUSTL(field)) // &
791                           '/sig_cmd_used.ps ./' // TRIM(ADJUSTL(field)) // &
792                           '/sig_cmd_used.jpg'
793
794    call system(command)
795
```

```
796 END SUBROUTINE WeighterPlots
797
798 !-------------------------------------------------------------------------
799
800 SUBROUTINE SVDFitter !For fitting polynomial to the
801 USE Global           !background field luminosity function
802 IMPLICIT NONE
803
804 INTEGER  :: ntmp
805
806 xa = bg_histo_coarse(:,1)
807 ya = bg_histo_coarse(:,2)
808 xt = xa
809 yt = ya
810 sig = 1.e0
811
812
813 ! Shift the array in steps of 1 until the first element does not contain a zero
814
815 shiftloop: do
816    xt = cshift(xt,1)
817    yt = cshift(yt,1)
818    if ( yt(1) > 0.1 ) exit shiftloop
819 end do shiftloop
820
821 ntmp = 0
822 countloop: do i = 1 , ndat
823    if ( yt(i) > 0.1 ) then
824       ntmp = ntmp + 1
825    else
826       exit countloop
827    end if
828 end do countloop
829
830   xt = xt - 21.
831
832   CALL svdfit(xt,yt,sig,ntmp-1,ay,ma,u,v,w,mp,np,chisq,funcs)
833
834 CALL BG_DataHist
835
836 END SUBROUTINE SVDFitter
837
838 !-------------------------------------------------------------------------
839
840 SUBROUTINE BG_DataHist !Produces plot of background field luminosity
841 USE Global             !function and polynomial fit
842 IMPLICIT NONE
843
844 bfm = 0.d0
845
846 DO i = 1, ndat
847    DO j = 1 , np
848       bfm(i) = bfm(i) + ay(j) * (xa(i)-21.) ** (j-1)
849    END DO
850 END DO
851
852
853 !----------------------Plots best fit model over coarse histogram
854 string = TRIM(ADJUSTL(field)) // '/bckgrdfit.ps/CPS'
855 CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
856
```

```
857  CALL pgenv(18., 24., 0., 1.1*MAXVAL(REAL(bg_histo_coarse(:,2))), 0, 0)
858  CALL pgbin (ndat, REAL(bg_histo_coarse(:,1)), REAL(bg_histo_coarse(:,2)), .true.)
859  CALL pgsci(2)
860  CALL pgline (ndat, xa, REAL(bfm))
861  CALL pgsci(1)
862  CALL pglab('i\d0\u', 'counts', '')
863
864  CALL pgend
865
866  WRITE (command,*) 'convert -rotate 90 ./' // TRIM(ADJUSTL(field)) // &
867                          '/bckgrdfit.ps ./' // TRIM(ADJUSTL(field)) // &
868                          '/bckgrdfit.jpg'
869
870  call system(command)
871
872  END SUBROUTINE BG_DataHist
873
874  !-------------------------------------------------------------------------
875
876  SUBROUTINE MCMC  !The master Markov Chain MonteCarlo routine
877  USE Global       !creates a new model at each iteration and then compares
878  IMPLICIT NONE    !the quality of the fit to the data
879                   !*** Most subroutines are called from 'MCMC' ***
880
881  REAL*8 :: gasdev
882
883  string = TRIM(ADJUSTL(field)) // '/MCMC_steps.dat'
884  OPEN(2, file=TRIM(ADJUSTL(string)), status = 'unknown')
885  OPEN(8, file=TRIM(ADJUSTL(field)) // '/MCMC_steps_unf.dat', form = 'unformatted', status = 'unknown')
886  WRITE(2,*) "   Iteration          mag_tip                f                  a           LikeA           LikeB"
887
888  not_scout = .false. ; scout_counts = 0
889
890  known_f = (REAL(bg_stars) * sig_area)/(REAL(sig_stars) * bg_area)
891
892  x1(1,:) = 20.5d0; x2(1,:) = known_f; x3(1,:) = 0.3d0 ; time(1) = 1
893
894
895  1 IF (not_scout) THEN                                                    !Set values
896       x1(1,:) = x1(200,:); x2(1,:) = x2(200,:); x3(1,:) = x3(200,:) ; time(1) = 1   !after
897    END IF                                                                 !scout run
898
899  mag_tip = x1(1,1) ; f = x2(1,1) ; a = x3(1,1)
900  cn = 1 ; beta = betaholder(1)
901  CALL ModelMake                  !Make model and
902  CALL Convolution                !
903  DO j = 1, num_chains            !evaluate goodness of fit
904     cn = j ; beta = betaholder(cn)!
905     CALL Loglike                 !for initial parameter choices
906     LikeA(cn) = logL(cn)         !
907  END DO                          !
908  LikeB(:) = 0.d0
909
910  x1(2,:) = x1(1,:) ; x2(2,:) = x2(1,:) ; x3(2,:) = x3(1,:)
911
912  Best_Combo(5) = -9.d99
913  DO it = 2, nit
914     time(it) = it
915     DO cn = 1, num_chains
916        beta = betaholder(cn)
917        p(1) = x1(it,cn) + m_step(cn)*gasdev(idum)  !Gaussian weighted steps from initial
```

```
      p(2) = x2(it,cn)! + f_step(cn)*gasdev(idum) !parameters for the tip magnitude (p(1))
      p(3) = x3(it,cn) + a_step(cn)*gasdev(idum)  !noise ratio (p(2)) and slope (p(3))

      IF (p(1) .lt. blim .or. p(1) .gt. flim) THEN      !
         r = 0.d0                                       !
      else IF (p(2) .le. 0.d0 .or. p(2) .ge. 1.d0) THEN !Restrictions on
         r = 0.d0                                       !whether proposed
      else IF (p(3) .le. 0.d0 .or. p(3) .ge. 2.d0) THEN !step is taken
         r = 0.d0                                       !
      else                                              !
         mag_tip = p(1) ; f = p(2) ; a = p(3)
         CALL ModelMake         !Make model and
         CALL Convolution       !evaluate the
         CALL Loglike           !goodness of fit
         LikeB(cn) = logL(cn)
         r = 10**(LikeB(cn) - LikeA(cn))
      end IF

      IF (cn .eq. 1 .and. not_scout) THEN                    !Only counts after the scouting run contribute to the ppds
         post_y1(INT((x1(it,1) - 18.d0)*10*binspm + 1)) = &      !
         post_y1(INT((x1(it,1) - 18.d0)*10*binspm + 1)) + 1.d0   !
         post_y2(INT(x2(it,1) * (nbins - 1)) + 1) = &            !Generate posterior plot
         post_y2(INT(x2(it,1) * (nbins - 1)) + 1) + 1.d0         !for mag_tip, f and a
         post_y3(INT(x3(it,1) * (nbins - 1)) + 1) = &            !
         post_y3(INT(x3(it,1) * (nbins - 1)) + 1) + 1.d0         !

         WRITE (2, '(6F16.5)') time(it), x1(it,cn), x2(it,cn), x3(it,cn), LikeA(cn), LikeB(cn)
         WRITE (8) time(it), x1(it,cn), x2(it,cn), x3(it,cn), LikeA(cn), LikeB(cn)
         !/\ Prints current parameter values and their likelihood (LikeA) as
         !|| well as the likelihood of the current proposed swap (LikeB)
         IF (LikeA(cn) .gt. Best_Combo(5)) THEN                !
            Best_Combo(1) = time(it)    ; Best_Combo(2) = x1(it,cn)  !Update best likelihood
            Best_Combo(3) = x2(it,cn); Best_Combo(4) = x3(it,cn)     !combination encountered
            Best_Combo(5) = LikeA(cn)   ; Best_Combo(6) = LikeB(cn)  !
         END IF
      END IF

      CALL random_number(randnum3)          !
      IF (it .lt. nit) THEN                 !
         IF (randnum3 .le. r) THEN          !
            x1(it+1,cn) = p(1)              !
            x2(it+1,cn) = p(2)              !
            x3(it+1,cn) = p(3)              !Decide
            likeA(cn) = likeB(cn)           !whether
         ELSE                               !to take
            x1(it+1,cn) = x1(it,cn)         !step
            x2(it+1,cn) = x2(it,cn)         !
            x3(it+1,cn) = x3(it,cn)         !
            likeA(cn) = likeA(cn)           !
         END IF                             !
      END IF
   END DO
   !/\ RUN MULTIPLE
   !|| MCMC CHAINS

   IF (scout_counts .lt. 200) THEN              !
      scout_counts = scout_counts + 1           !
      cycle                                     !
   END IF                                       !200 iterations will be run at the beginning before the
   IF (scout_counts .eq. 200) THEN              !
      not_scout = .true.                        !'nit' used iterations in order to remove the lead in trail.
```

```
979         scout_counts = 1000000000                !
980         goto 1                                   !
981       END IF                                     !
982
983       !|| PARALLEL
984       !\/ TEMPERING
985       IF (num_chains .ne. 1 .and. it .ne. nit) THEN
986         CALL random_number(randnum4)
987         IF (randnum4 .le. swaprate) THEN
988           CALL random_number(randnum4)
989           randint = INT((num_chains - 1) * randnum4) + 1
990           IF (randint .eq. 1) THEN                    !Count number of proposed
991             chain_compare = chain_compare + 1.        !swaps with the
992           END IF                                      !cold sampler chain
993             PTAR = (Betaholder(randint)/ Betaholder(randint + 1)) * LikeA(randint + 1) + &
994                  (Betaholder(randint + 1)/ Betaholder(randint)) * LikeA(randint) - &
995                  LikeA(randint) - LikeA(randint + 1)
996           CALL random_number(randnum5)
997           IF (randnum5 .le. 10 ** PTAR) THEN
998             IF (randint .eq. 1) THEN          !Count number of accepted
999               swap_count = swap_count + 1     !swaps with the
1000            END IF                            !cold sampler chain
1001            par_hold(1) = x1(it+1, randint)                !
1002            par_hold(2) = x2(it+1, randint)                !
1003            par_hold(3) = x3(it+1, randint)                !
1004            par_hold(4) = LikeA(randint)                   !
1005            x1(it+1, randint) = x1(it+1, randint + 1)       !
1006            x2(it+1, randint) = x2(it+1, randint + 1)       !Swap the parameter
1007            x3(it+1, randint) = x3(it+1, randint + 1)       !values and
1008            LikeA(randint) = LikeA(randint + 1) * &         !likelihoods
1009            (Betaholder(randint)/Betaholder(randint + 1))   !between chains
1010            x1(it+1, randint + 1) = par_hold(1)             !
1011            x2(it+1, randint + 1) = par_hold(2)             !
1012            x3(it+1, randint + 1) = par_hold(3)             !
1013            LikeA(randint + 1) = par_hold(4) * &            !
1014            (Betaholder(randint + 1)/Betaholder(randint))   !
1015          END IF
1016        END IF
1017      END IF
1018
1019    END DO
1020
1021    x1(1,:) = x1(200,:) ; x2(1,:) = x2(200,:) ; x3(1,:) = x3(200,:)  !Remove initial
1022    x1(2,:) = x1(201,:) ; x2(2,:) = x2(201,:) ; x3(2,:) = x3(201,:)  !parameter values
1023
1024    WRITE (2, '(6F16.5)') Best_Combo(1), Best_Combo(2), Best_Combo(3), &
1025                          Best_Combo(4), Best_Combo(5), Best_Combo(6)
1026
1027    DO i = 1, 10*(nbins-1)+1                                  !
1028      post_x1(i) = 18.d0 + (REAL(i) - 1.d0)/REAL(10*binspm)!
1029    END DO                                                    !
1030                                                              !
1031    DO i = 1, nbins                                           !x-values of posterior
1032      post_x2(i) = (REAL(i) - 1.d0)/REAL(nbins - 1)           !histograms created above
1033    END DO                                                    !
1034                                                              !
1035    DO i = 1, 2*nbins - 1                                     !
1036      post_x3(i) = (REAL(i) - 1.d0)/REAL(nbins - 1)           !
1037    END DO                                                    !
1038
1039    END SUBROUTINE MCMC
```

```
1040
1041  !-------------------------------------------------------------------------------
1042
1043  SUBROUTINE PosteriorPlot !Produces histogram plots of the posterior distributions
1044  USE Global               !in the tip magnitude and LF slope
1045  IMPLICIT NONE
1046
1047  post_y1 = post_y1/nit ; post_y2 = post_y2/nit ; post_y3 = post_y3/nit
1048
1049  !------------------------------Plots mag_tip posterior plot
1050  string = TRIM(ADJUSTL(field)) // '/mag_tip_postplot.ps/CPS'
1051  CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
1052
1053  CALL pgenv(REAL(MINVAL(post_x1, mask = post_y1 .ne. 0.)), &
1054        REAL(MAXVAL(post_x1, mask = post_y1 .ne. 0.)), 0., &
1055        1.1*REAL(MAXVAL(post_y1)), 0, 0)
1056  CALL pgbin (10*(nbins-1)+1, REAL(post_x1), REAL(post_y1),.false.)
1057  CALL pglab('Proposed i\d0\u tip magnitude', 'Probability', '')
1058
1059  CALL pgend
1060
1061  WRITE (command,*) 'convert -rotate 90 ./' // TRIM(ADJUSTL(field)) // &
1062                    '/mag_tip_postplot.ps ./' // TRIM(ADJUSTL(field)) // &
1063                    '/mag_tip_postplot.jpg'
1064
1065  call system(command)
1066
1067  !------------------------------Plots f and a posterior plots
1068  string = TRIM(ADJUSTL(field)) // '/f_and_a_postplot.ps/CPS'
1069  CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
1070
1071  IF (MAXVAL(post_y3) .ge. MAXVAL(post_y2)) THEN
1072     CALL pgenv(0.,  2., 0., 1.1*REAL(MAXVAL(post_y3)), 0, 0)
1073  ELSE
1074     CALL pgenv(0.,  2., 0., 1.1*REAL(MAXVAL(post_y2)), 0, 0)
1075  END IF
1076
1077  CALL pgsci(2)
1078  CALL pgbin (nbins, REAL(post_x2), REAL(post_y2),.false.)
1079  CALL pgsci(3)
1080  CALL pgbin (2*nbins-1, REAL(post_x3), REAL(post_y3),.false.)
1081  CALL pgsci(1)
1082  CALL pglab('Proposed value', 'Probability: f (red) a (green)', '')
1083
1084  CALL pgend
1085
1086  WRITE (command,*) 'convert -rotate 90 ./' // TRIM(ADJUSTL(field)) // &
1087                    '/f_and_a_postplot.ps ./' // TRIM(ADJUSTL(field)) // &
1088                    '/f_and_a_postplot.jpg'
1089
1090  call system(command)
1091
1092  post_y1 = post_y1*nit ; post_y2 = post_y2*nit ; post_y3 = post_y3*nit
1093
1094  END SUBROUTINE PosteriorPlot
1095
1096  !-------------------------------------------------------------------------------
1097
1098  SUBROUTINE OtherPlots !MCMC related plots - i.e. plots each parameter vs.
1099  USE Global            !iteration number and vs. each other
1100  IMPLICIT NONE
```

```
1101
1102  !------------------------------Variation of 'mag_tip' with iteration #
1103   string = TRIM(ADJUSTL(field)) // '/mag_tip_val_vs_it.ps/CPS'
1104  CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
1105
1106  CALL pgenv(0., REAL(nit), REAL(MINVAL(x1(:,1)))-0.1, REAL(MAXVAL(x1(:,1)))+0.1, 0, 0)
1107  CALL pgline (nit, REAL(time), REAL(x1(:,1)))
1108  CALL pglab('Iteration number', 'Proposed i\d0\u tip magnitude', '')
1109
1110  CALL pgend
1111
1112  WRITE (command,*) 'convert -rotate 90 ./' // TRIM(ADJUSTL(field)) // &
1113                          '/mag_tip_val_vs_it.ps ./' // TRIM(ADJUSTL(field)) // &
1114                          '/mag_tip_val_vs_it.jpg'
1115
1116   call system(command)
1117
1118  !------------------------------Variation of 'f' and 'a' with iteration #
1119   string = TRIM(ADJUSTL(field)) // '/f_and_a_val_vs_it.ps/CPS'
1120  CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
1121
1122  CALL pgenv(0., REAL(nit), 0., 2., 0, 0)
1123  CALL pgsci(2)
1124  CALL pgline (nit, REAL(time), REAL(x2(:,1)))
1125  CALL pgsci(3)
1126  CALL pgline (nit, REAL(time), REAL(x3(:,1)))
1127  CALL pgsci(1)
1128  CALL pglab('Iteration number', 'Proposed value: f (red) a (green)', '')
1129
1130  CALL pgend
1131
1132  WRITE (command,*) 'convert -rotate 90 ./' // TRIM(ADJUSTL(field)) // &
1133                          '/f_and_a_val_vs_it.ps ./' // TRIM(ADJUSTL(field)) // &
1134                          '/f_and_a_val_vs_it.jpg'
1135
1136   call system(command)
1137
1138  !------------------------------Values of 'f' for each value of 'mag_tip'
1139   string = TRIM(ADJUSTL(field)) // '/f_vs_mag_tip.ps/CPS'
1140  CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
1141
1142  CALL pgenv(0.99*REAL(MINVAL(x1(:,1))), 1.01*REAL(MAXVAL(x1(:,1))), 0.9*REAL(MINVAL(x2(:,1))), 1.1*REAL(MAXVAL(x2(:,1))), 0, 0)
1143  CALL pgslw(3)
1144  CALL pgpoint (nit, REAL(x1(:,1)), REAL(x2(:,1)), -1)
1145  CALL pgslw(1)
1146  CALL pglab('Proposed i\d0\u tip magnitude', 'Proposed value of f', '')
1147
1148  CALL pgend
1149
1150  WRITE (command,*) 'convert -rotate 90 ./' // TRIM(ADJUSTL(field)) // &
1151                          '/f_vs_mag_tip.ps ./' // TRIM(ADJUSTL(field)) // &
1152                          '/f_vs_mag_tip.jpg'
1153
1154   call system(command)
1155
1156  !------------------------------Values of 'a' for each value of 'mag_tip'
1157   string = TRIM(ADJUSTL(field)) // '/a_vs_mag_tip.ps/CPS'
1158  CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
1159
1160  CALL pgenv(0.99*REAL(MINVAL(x1(:,1))), 1.01*REAL(MAXVAL(x1(:,1))), 0.9*REAL(MINVAL(x3(:,1))), 1.1*REAL(MAXVAL(x3(:,1))), 0, 0)
1161  CALL pgslw(3)
```

```
1162  CALL pgpoint (nit, REAL(x1(:,1)), REAL(x3(:,1)), -1)
1163  CALL pgslw(1)
1164  CALL pglab('Proposed i\d0\u tip magnitude', 'Proposed value of a', '')
1165
1166  CALL pgend
1167
1168  WRITE (command,*) 'convert -rotate 90 ./' // TRIM(ADJUSTL(field)) // &
1169                    '/a_vs_mag_tip.ps ./' // TRIM(ADJUSTL(field)) // &
1170                    '/a_vs_mag_tip.jpg'
1171
1172  call system(command)
1173
1174  !------------------------------Values of 'f' for each value of 'a'
1175  string = TRIM(ADJUSTL(field)) // '/a_vs_f.ps/CPS'
1176  CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
1177
1178  CALL pgenv(0.9*REAL(MINVAL(x2(:,1))),  1.1*REAL(MAXVAL(x2(:,1))), 0.9*REAL(MINVAL(x3(:,1))), 1.1*REAL(MAXVAL(x3(:,1))), 0, 0)
1179  CALL pgslw(3)
1180  CALL pgpoint (nit, REAL(x2(:,1)), REAL(x3(:,1)), -1)
1181  CALL pgslw(1)
1182  CALL pglab('Proposed value of f', 'Proposed value of a', '')
1183
1184  CALL pgend
1185
1186  WRITE (command,*) 'convert -rotate 90 ./' // TRIM(ADJUSTL(field)) // &
1187                    '/a_vs_f.ps ./' // TRIM(ADJUSTL(field)) // &
1188                    '/a_vs_f.jpg'
1189
1190  call system(command)
1191
1192  END SUBROUTINE OtherPlots
1193
1194  !-------------------------------------------------------------------------------
1195
1196  SUBROUTINE NoiseMake !Generates a polynomial of degree 7 that follows the
1197  USE Global           !functional form of the GSS background LF. The polynomial
1198  IMPLICIT NONE        !coefficients were derived in 'BackgroundPolyFit' using
1199                       !'svdfit' from Numerical Recipes.
1200  area2 = 0.d0
1201
1202  DO i = 1, 8 * binspm + 1
1203     modelnoise(i,1) = 18.d0 + (i-1.d0)/REAL(binspm)
1204     modelnoise(i,2) = 0.d0
1205     DO j = 1, np                        !Set background counts
1206        modelnoise(i,2) = modelnoise(i,2) + ay(j) * (modelnoise(i,1) - 21.d0) ** (j - 1)
1207     END DO
1208     IF (modelnoise(i,2) .lt. 0.d0) THEN  !
1209        modelnoise(i,2) = 0.d0            !Insure no negative counts
1210     END IF                               !
1211     IF (i .ge. blimBins .and. i .le. flimBins) THEN
1212        area2 = area2 + modelnoise(i,2)   !Used for normalization in 'ModelMake'
1213     END IF
1214  END DO
1215
1216  model(:,2) =  modelnoise(:,2) / area2
1217
1218  CALL Convolution
1219  noise = cmodel(:,2)
1220
1221  END SUBROUTINE NoiseMake
1222
```

```
1223  !-------------------------------------------------------------------------------
1224  SUBROUTINE NoisePlot !Plots the unscaled form of the background LF
1225  USE Global
1226  IMPLICIT NONE
1227
1228  CALL pgbegin(0,'?',1,1)
1229
1230  CALL pgenv(REAL(blim), REAL(mag_cutoff), 0., 1.1*REAL(MAXVAL(modelnoise(:,2), mask = modelnoise(:,1) .le. 23.5 .and. modelnoise(:,1)
          .ge. bg_blim)), 0, 0)
1231  CALL pgbin (nbins - INT(2.5*binspm), REAL(modelnoise(:,1)), REAL(modelnoise(:,2)), .true.)
1232  CALL pglab('i\d0\u', 'Counts', '')
1233
1234  CALL pgend
1235
1236  END SUBROUTINE NoisePlot
1237
1238  !-------------------------------------------------------------------------------
1239
1240  SUBROUTINE ModelMake   !Initial Model (i.e. model before convolution)
1241  USE Global
1242  IMPLICIT NONE
1243
1244  REAL*8 :: func_i
1245
1246  area = 0.d0
1247  DO i = 1, nbins
1248     model(i,1) = 18.d0 + (i-1.d0)/REAL(binspm)
1249     IF (model(i,1) + hb .gt. mag_tip .and. model(i,1) - hb .le. mag_tip) THEN
1250        model(i,2) = ((10.d0**(a*(model(i,1) + hb - mag_tip)))/(a*LOG(10.))) - &
1251                     (1.d0/(a*LOG(10.)))                    !Model value at tip
1252        area = area + model(i,2)       !Used for normalization
1253     ELSE IF (model(i,1) .gt. mag_tip) THEN        !Model value faintward of tip
1254        model(i,2) = ((10.d0**(a*(model(i,1) + hb - mag_tip)))/(a*LOG(10.))) - &
1255                     ((10.d0**(a*(model(i,1) - hb - mag_tip)))/(a*LOG(10.)))
1256     ELSE
1257        model(i,2) = 0.d0                          !Model value brightward of tip
1258     END IF
1259     IF (i .ge. blimBins .and. i .le. flimBins) THEN
1260        area = area + model(i,2)   !Used for normalization
1261     END IF
1262  END DO
1263
1264  model(:,2) = model(:,2) / area  !Normalize
1265
1266  END SUBROUTINE ModelMake
1267
1268  !-------------------------------------------------------------------------------
1269
1270  SUBROUTINE ModelPrint !Prints model before convolution
1271  USE Global
1272  IMPLICIT NONE
1273
1274  CALL pgbegin(0,'?',1,1)
1275
1276  CALL pgenv(REAL(mag_tip) - 3., REAL(mag_cutoff), 0., 1.1*REAL(model(INT(5.5*binspm),2)), 0, 0)
1277  CALL pgbin (nbins - INT(2.5*binspm), REAL(model(:,1)), REAL(model(:,2)), .true.)
1278  CALL pglab('i\d0\u', 'Counts', '')
1279
1280  CALL pgend
1281
1282  END SUBROUTINE ModelPrint
```

```
1283
1284 !------------------------------------------------------------------------------
1285
1286 SUBROUTINE GaussianKernel !Generates a Gaussian kernel 'kernel' with
1287 USE Global                !HWHM (sigma) changing with magnitude in
1288 IMPLICIT NONE             !accordance with func_i. Kernel is defined from
1289                           !gx = -5*sigma to gx = +5*sigma.
1290 REAL*8 :: func_i
1291
1292 kernel = 0.d0
1293
1294 DO i = 1, nbins
1295    t = 18.d0 + (i - 1.d0)/REAL(binspm)    !Convert bin number to magnitude
1296
1297    temp = 0.d0
1298    gx=0.
1299    j=0
1300    DO WHILE (gx .le. 5.e0*func_i(t))                          !
1301       j=j+1                                                   !
1302       gx = 0.e0 + (j-1.e0)/binspm                             !Creates half of
1303       temp(j,1) = gx                                          !the kernel ('temp')
1304       temp(j,2) = exp(-((gx)**2.e0)/(2.e0*(func_i(t)**2.e0)))!
1305    END DO                                                     !
1306
1307    ghw(i) = j - 1.d0     !The first non-zero bin of 'cmodel' will be the first
1308                          !non-zero bin of 'model' minus ghw
1309
1310    DO k = 1, j
1311       kernel(k,:,i) = temp(j - (k-1),:)  !Create 'kernel' by concatenating
1312       kernel(j+k,2,i) = temp(k+1,2)      !'temp' with a reflected version
1313       kernel(j+k,1,i) = -temp(k+1,1)     !of itself
1314    END DO
1315    !Note: temp(2*j,2) = 0.d0 ; temp(2*j,1) = -0.d0
1316
1317    kernel(:,2,i) = kernel(:,2,i)/SUM(kernel(:,2,i))
1318
1319 END DO
1320
1321
1322 END SUBROUTINE GaussianKernel
1323
1324 !------------------------------------------------------------------------------
1325
1326 SUBROUTINE GaussianKernelPrint !Prints Gaussian Kernel at given magnitude
1327 USE Global
1328 IMPLICIT NONE
1329
1330 REAL*8 :: func_i
1331
1332 CALL pgbegin(0,'?',1,1)
1333
1334 CALL pgenv(-5.5 * REAL(func_i(t)), 5.5 * REAL(func_i(t)), 0., 1.1*MAXVAL(REAL(kernel(:,2,i))), 0, 0)
1335 CALL pgbin (2*ghw(i)+1, REAL(kernel(:,1,i)), REAL(kernel(:,2,i)), .true.)
1336 CALL pglab('Magnitude_offset', 'Strength', '')
1337
1338 CALL pgend
1339
1340 END SUBROUTINE GaussianKernelPrint
1341
1342 !------------------------------------------------------------------------------
1343
```

```
1344  SUBROUTINE Convolution  !Convolves initial model with a Gaussian kernel
1345  Use Global              !whose width is equal to the photometric error
1346  IMPLICIT NONE           !and hence expands with increasing magnitude
1347
1348  cmodel = 0.d0
1349
1350  DO i = 1, nbins
1351     cmodel(i,1) = 18.d0 + (i - 1.d0)/REAL(binspm)
1352     DO j = -ghw(i), ghw(i), +1                                          !
1353        IF (i .gt. ghw(i) .and. i .lt. nbins - ghw(i)) THEN              !Convolve
1354           cmodel(i+j,2) = cmodel(i+j,2) + kernel(ghw(i)+j+1,2,i)*model(i,2)  !model with
1355        END IF                                                           !gaussian
1356     END DO                                                              !
1357  END DO
1358
1359  DO i = nbins, flimBins+1, -1          !Set the faint limit
1360        cmodel(i,2) = 0.d0              !of the final convolved
1361  END DO                                !model at flim.
1362  cmod_nbins = flimBins
1363
1364  !Normalize the convolved model
1365  cmodel(:,2) = cmodel(:,2)/SUM(cmodel(:,2), mask = cmodel(:,1) .ge. blim)
1366
1367  !Note the above step is very important - normalization must only be over the
1368  !range of magnitudes in the 'data' array - i.e. down to blim -> not right the
1369  !way to blim - 1.d0. This was a difficult bug to find!
1370
1371  END SUBROUTINE Convolution
1372
1373  !-------------------------------------------------------------------------
1374
1375  SUBROUTINE ConvolutionPrint !Prints convolved version of model
1376  USE Global
1377  IMPLICIT NONE
1378
1379  CALL pgbegin(0,'?',1,1)
1380
1381  CALL pgenv(REAL(mag_tip) - 0.5, 25., 0., 1.1*MAXVAL(REAL(cmodel(:,2))), 0, 0)
1382  CALL pgbin (nbins, REAL(cmodel(:,1)), REAL(cmodel(:,2)), .true.)
1383  CALL pglab('i\d0\u', 'Relative probability', '')
1384
1385  CALL pgend
1386
1387  END SUBROUTINE ConvolutionPrint
1388
1389  !-------------------------------------------------------------------------
1390
1391  SUBROUTINE DataHist !Generates finely and coarsely binned histograms and
1392  USE Global          !overlays them with the best fit model determined by
1393  IMPLICIT NONE       !the MCMC
1394
1395  REAL*8 :: scaled_f_rec
1396
1397  histo_fine(:,1) = model(:,1)
1398  DO i = 1, INT(0.25*(nbins-1.d0)) + 1
1399     histo_coarse(i,1) = 18.d0 + (i-1.d0)/REAL(0.25*binspm)
1400  END DO
1401
1402  DO i = 1, ndata2                                                  !
1403     histo_fine(INT(REAL((data(i)-18.d0)*binspm) + 1.d0),2) = &     !Generates
1404     histo_fine(INT(REAL((data(i)-18.d0)*binspm) + 1.d0),2) + 1.d0  !
```

```
1405      histo_coarse(INT(REAL((data(i)-18.d0)*0.25*binspm) + 1.d0),2) = &     !Histograms
1406      histo_coarse(INT(REAL((data(i)-18.d0)*0.25*binspm) + 1.d0),2) + 1.d0 !
1407  END DO                                                                    !
1408
1409
1410  histo_coarse(INT(REAL(blimbins)/4.e0) + 1,2) = &     !See paragraph
1411  histo_coarse(INT(REAL(blimbins)/4.e0) + 1,2) * 2.d0 !below
1412  histo_coarse(INT(REAL(flimbins)/4.e0) + 1,2) = &     !See paragraph
1413  histo_coarse(INT(REAL(flimbins)/4.e0) + 1,2) * 2.d0 !below
1414
1415  !For graphing purposes, the first and last bins of the coarse histogram are doubled since
1416  !these bin lies half outside the range of interest and so are depleted by
1417  !roughly one half. This is for graphing only and has no bearing on the
1418  !determined best fit model.
1419
1420  !|| Plot Best Fit Model
1421  !\/ over histogram
1422  mag_tip = tip_rec ; f = f_rec ; a = a_rec             !
1423  CALL ModelMake                                        !Generate best fit sig function
1424  CALL Convolution                                      !
1425
1426  bfm = 0.d0 ; bg = 0.d0                                !
1427
1428  bfm = cmodel(:,2) * (1.d0 - f)                        !bfm = best fit signal function
1429  bg = noise * f                                        !bg = back ground function
1430
1431  bfm = bfm + bg                                  !Add bfm and background together
1432  bfm = bfm * (SUM(histo_fine(:,2))/SUM(bfm, mask = cmodel(:,1) .ge. blim))   !Scale bfm to match histogram
1433
1434  !-------------------------Plots best fit model over fine histogram
1435  string = TRIM(ADJUSTL(field)) // '/model_fit_vs_data_fine.ps/CPS'
1436  CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
1437
1438  CALL pgenv(REAL(blim), REAL(flim), 0., 1.1*MAXVAL(real(histo_fine(:,2))), 0, 0)
1439  CALL pgbin (nbins, REAL(histo_fine(:,1)), REAL(histo_fine(:,2)), .false.)
1440  CALL pgsci(2)
1441  CALL pgslw(5)
1442  CALL pgline (nbins, REAL(histo_fine(:,1)), REAL(bfm))
1443  CALL pgsci(1)
1444  CALL pgslw(1)
1445  CALL pglab('i\d0\u', 'Counts', '')
1446
1447  CALL pgend
1448
1449  WRITE (command,*) 'convert -rotate 90 ./' // TRIM(ADJUSTL(field)) // &
1450                    '/model_fit_vs_data_fine.ps ./' // TRIM(ADJUSTL(field)) // &
1451                    '/model_fit_vs_data_fine.jpg'
1452
1453  call system(command)
1454
1455  bfm = bfm * 4.d0          !Scale bfm to match coarse histogram
1456
1457  !-----------------------Plots best fit model over coarse histogram
1458  string = TRIM(ADJUSTL(field)) // '/model_fit_vs_data_coarse.ps/CPS'
1459  CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
1460
1461  CALL pgenv(REAL(blim), REAL(flim), 0., 1.1*MAXVAL(real(histo_coarse(:,2))), 0, 0)
1462  CALL pgbin (INT(0.25*(nbins-1.d0)) + 1, REAL(histo_coarse(:,1)), &
1463              REAL(histo_coarse(:,2)), .false.)
1464  CALL pgsci(2)
1465  CALL pgline (nbins, REAL(histo_fine(:,1)), REAL(bfm))
```

```
1466  CALL pgsci(1)
1467  CALL pglab('i\d0\u', 'Counts', '')
1468
1469  CALL pgend
1470
1471  WRITE (command,*) 'convert -rotate 90 ./' // TRIM(ADJUSTL(field)) // &
1472                          '/model_fit_vs_data_coarse.ps ./' // TRIM(ADJUSTL(field)) // &
1473                          '/model_fit_vs_data_coarse.jpg'
1474
1475  call system(command)
1476
1477  END SUBROUTINE DataHist
1478
1479  !-------------------------------------------------------------------------
1480
1481  SUBROUTINE w_DataHist !Generates finely and coarsely binned (weighted) histograms
1482  USE Global            !and overlays them with the best fit model determined by
1483  IMPLICIT NONE         !the MCMC
1484
1485  REAL*8 :: scaled_f_rec
1486
1487  w_histo_fine(:,1) = model(:,1)
1488  DO i = 1, INT(0.25*(nbins-1.d0)) + 1
1489     w_histo_coarse(i,1) = 18.d0 + (i-1.d0)/REAL(0.25*binspm)
1490  END DO
1491
1492  DO i = 1, ndata2                                                    !
1493     IF (truestar_poly(i)) THEN                                       !
1494     IF (scaled_a(i) .ge. crowded_rad .and. scaled_a(i) .le. outer_rad) THEN
1495     w_histo_fine(INT(REAL((data(i)-18.d0)*binspm) + 1.d0),2) = &      !Generates Weighted Histograms.
1496     w_histo_fine(INT(REAL((data(i)-18.d0)*binspm) + 1.d0),2) + &      !Set so that stars at centre of
1497     (weight(i)/maxweight)                                            !field contribute 1.0 counts
1498     w_histo_coarse(INT(REAL((data(i)-18.d0)*0.25*binspm) + 1.d0),2) = &  !while stars at some radius give
1499     w_histo_coarse(INT(REAL((data(i)-18.d0)*0.25*binspm) + 1.d0),2) + &  !some fraction of 1.0 counts
1500     (weight(i)/maxweight)                                            !depending on the density profile.
1501     END IF
1502     END IF                                                           !
1503  END DO                                                              !
1504
1505
1506  w_histo_coarse(INT(REAL(blimbins)/4.e0) + 1,2) = &    !See paragraph
1507  w_histo_coarse(INT(REAL(blimbins)/4.e0) + 1,2) * 2.d0 !below
1508  w_histo_coarse(INT(REAL(flimbins)/4.e0) + 1,2) = &    !See paragraph
1509  w_histo_coarse(INT(REAL(flimbins)/4.e0) + 1,2) * 2.d0 !below
1510
1511  !For graphing purposes, the first and last bins of the coarse histogram are doubled since
1512  !these bin lies half outside the range of interest and so are depleted by
1513  !roughly one half. This is for graphing only and has no bearing on the
1514  !determined best fit model.
1515
1516  !|| Plot Best Fit Model
1517  !\/ over weighted histogram
1518  mag_tip = tip_rec ; f = f_rec ; a = a_rec        !
1519  CALL ModelMake                                   !Generate best fit signal function
1520  CALL Convolution                                 !
1521
1522  bfm = 0.d0 ; bg = 0.d0   !Apply weights to best fit model for  ||
1523  DO i = 1, ndata2         !each star and sum together.          \/
1524     IF (truestar_poly(i)) THEN
1525     IF (scaled_a(i) .ge. crowded_rad .and. scaled_a(i) .le. outer_rad) THEN
1526     bfm = bfm + cmodel(:,2) * (weight(i)/(weight(i) + (bg_stars / bg_area))) * weight(i)    !sum together RGB LFs from each star
```

```
1527      bg = bg + noise * ((bg_stars / bg_area)/(weight(i) + (bg_stars / bg_area))) * weight(i) !sum together BG LFs from each star
1528      END IF
1529      END IF
1530    END DO
1531
1532    bfm = bfm + bg                                    !Add bfm and background together
1533    bfm = bfm * (SUM(w_histo_fine(:,2))/SUM(bfm, mask = cmodel(:,1) .ge. blim))    !Scale bfm to match histogram
1534
1535    !-------------------------Plots best fit model over fine histogram
1536    string = TRIM(ADJUSTL(field)) // '/model_fit_vs_data_fine_w.ps/CPS'
1537    CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
1538
1539    CALL pgenv(REAL(blim), REAL(flim), 0., 1.1*MAXVAL(real(w_histo_fine(:,2))), 0, 0)
1540    CALL pgbin (nbins, REAL(w_histo_fine(:,1)), REAL(w_histo_fine(:,2)), .false.)
1541    CALL pgsci(2)
1542    CALL pgslw(5)
1543    CALL pgline (nbins, REAL(w_histo_fine(:,1)), REAL(bfm))
1544    CALL pgsci(1)
1545    CALL pgslw(1)
1546    CALL pglab('i\d0\u', 'Weighted Counts', '')
1547
1548    CALL pgend
1549
1550    WRITE (command,*) 'convert -rotate 90 ./' // TRIM(ADJUSTL(field)) // &
1551                      '/model_fit_vs_data_fine_w.ps ./' // TRIM(ADJUSTL(field)) // &
1552                      '/model_fit_vs_data_fine_w.jpg'
1553
1554    call system(command)
1555
1556    bfm = bfm * 4.d0          !Scale bfm to match coarse histogram
1557
1558    !-----------------------Plots best fit model over coarse histogram
1559    string = TRIM(ADJUSTL(field)) // '/model_fit_vs_data_coarse_w.ps/CPS'
1560    CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
1561
1562    CALL pgenv(REAL(blim), REAL(flim), 0., 1.1*MAXVAL(real(w_histo_coarse(:,2))), 0, 0)
1563    CALL pgbin (INT(0.25*(nbins-1.d0)) + 1, REAL(w_histo_coarse(:,1)), &
1564                REAL(w_histo_coarse(:,2)), .false.)
1565    CALL pgsci(2)
1566    CALL pgline (nbins, REAL(w_histo_fine(:,1)), REAL(bfm))
1567    CALL pgsci(1)
1568    CALL pglab('i\d0\u', 'Weighted Counts', '')
1569
1570    CALL pgend
1571
1572    WRITE (command,*) 'convert -rotate 90 ./' // TRIM(ADJUSTL(field)) // &
1573                      '/model_fit_vs_data_coarse_w.ps ./' // TRIM(ADJUSTL(field)) // &
1574                      '/model_fit_vs_data_coarse_w.jpg'
1575
1576    call system(command)
1577
1578    END SUBROUTINE w_DataHist
1579
1580    !-------------------------------------------------------------------------
1581
1582    SUBROUTINE LogLike     !Generates the log
1583    USE Global             !of the likelihood
1584    IMPLICIT NONE          !for a given model
1585
1586    logL(cn) = 0.d0
1587    DO i = 1, ndata2
```

```
1588      IF (truestar_poly(i)) THEN
1589      IF (scaled_a(i) .ge. crowded_rad .and. scaled_a(i) .le. outer_rad) THEN
1590      starbin = INT((data(i) - 18.d0)*binspm) + 1
1591      sig_prob = cmodel(starbin, 2) * weight(i)/(weight(i) + (bg_stars/ bg_area))      !Determine likelihood for star given current ratio
1592      bg_prob = noise(starbin) * (bg_stars/ bg_area)/(weight(i) + (bg_stars/ bg_area))!of the RGB LF vs. the BG LF due to star's weight.
1593      prob = (sig_prob + bg_prob) * weight(i)                                          !This also insures the total prob. of the model
1594      logL(cn) = logL(cn) + LOG10(prob)                                                !is 1. Add likelihoods together in log space.
1595      END IF
1596      ELSE
1597         cycle
1598      END IF
1599   END DO
1600   logL(cn) = logL(cn) * beta
1601
1602   END SUBROUTINE LogLike
1603
1604   !-------------------------------------------------------------------------
1605
1606   SUBROUTINE TipAndSigma !Identifies the best parameter values and
1607   USE Global             !their associated 1 sigma errors from the
1608   IMPLICIT NONE          !respective posterior plots.
1609
1610   PPD_peak = 0.d0                        !
1611   DO i = 1, 10*(nbins-1)+1               !
1612      IF (post_y1(i) .gt. PPD_peak) THEN  !
1613         PPD_peak = post_y1(i)            !Find best fit TRGB value
1614         tip_rec = post_x1(i)             !
1615      END IF                              !
1616   END DO                                 !
1617
1618   PPD_peak = 0.d0                        !
1619   DO i = 1, nbins                        !
1620      IF (post_y2(i) .gt. PPD_peak) THEN  !
1621         PPD_peak = post_y2(i)            !Find best fit f value
1622         f_rec = post_x2(i)               !
1623      END IF                              !
1624   END DO                                 !
1625
1626   PPD_peak = 0.d0                        !
1627   DO i = 1, 2*nbins - 1                  !
1628      IF (post_y3(i) .gt. PPD_peak) THEN  !
1629         PPD_peak = post_y3(i)            !Find best fit a value
1630         a_rec = post_x3(i)               !
1631      END IF                              !
1632   END DO                                 !
1633
1634   tip_kpc = (100.d0**((tip_rec + 3.44d0)/10.d0))/100.d0 !Distance inferred from
1635                                                         !tip magnitude in kpc
1636
1637   tip_counts = 0.d0 ; mcounts = 0.d0                          !
1638   DO i = MAXLOC(post_y1, DIM = 1), 1, -1                      !
1639      mcounts = mcounts + post_y1(i)                           !
1640   END DO                                                      !
1641   DO i = MAXLOC(post_y1, DIM = 1), 1, -1                      !
1642      tip_counts = tip_counts + post_y1(i)                     !Finds negative one sigma
1643      IF (tip_counts .ge. 0.682*mcounts) THEN                  !error in magnitudes
1644         tip_msigma = ((REAL(i) - 1.d0)/REAL(10*binspm)) + 18.d0 !
1645         tip_msigma = tip_rec - tip_msigma                     !
1646         exit                                                  !
1647      END IF                                                   !
1648   END DO                                                      !
```

```
1649
1650  tip_counts = 0.d0 ; pcounts = 0.d0                              !
1651  DO i = MAXLOC(post_y1, DIM = 1), 10*(nbins-1)+1                 !
1652     pcounts = pcounts + post_y1(i)                               !
1653  END DO                                                          !
1654  DO i = MAXLOC(post_y1, DIM = 1), 10*(nbins-1)+1                 !
1655     tip_counts = tip_counts + post_y1(i)                         !Finds positive one sigma
1656     IF (tip_counts .ge. 0.682*pcounts) THEN                      !error in magnitudes
1657        tip_psigma = ((REAL(i) - 1.d0)/REAL(10*binspm)) + 18.d0   !
1658        tip_psigma = tip_psigma - tip_rec                         !
1659        exit                                                      !
1660     END IF                                                       !
1661  END DO                                                          !
1662
1663
1664  d1 = 0 ; d2 = 0 ; d3 = 0 ; d4 = 0
1665  f_counts = 0.d0 ; a_counts = 0.d0                    !
1666  DO i = 1, nbins                                      !
1667     f_counts = f_counts + post_y2(i)                  !
1668     a_counts = a_counts + post_y3(i)                  !
1669     IF (f_counts .ge. 0.159*nit .and. d1 .eq. 0) THEN !
1670        fminsig = post_x2(i)                           !
1671        d1 = 1                                         !
1672     END IF                                            !
1673     IF (f_counts .ge. 0.841*nit .and. d2 .eq. 0) THEN !For f and a:
1674        fplusig = post_x2(i)                           !Finds upper and lower
1675        d2 = 1                                         !bounds for posterior
1676     END IF                                            !distribution within one
1677     IF (a_counts .ge. 0.159*nit .and. d3 .eq. 0) THEN !sigma of maximum.
1678        aminsig = post_x3(i)                           !
1679        d3 = 1                                         !
1680     END IF                                            !
1681     IF (a_counts .ge. 0.841*nit .and. d4 .eq. 0) THEN !
1682        aplusig = post_x3(i)                           !
1683        d4 = 1                                         !
1684     END IF                                            !
1685  END DO                                               !
1686
1687
1688  f_sigma = 0.5d0*(fplusig - fminsig)          !Hence calculates 1 sigma error
1689  a_sigma = 0.5d0*(aplusig - aminsig)          !for f and a
1690
1691  kpc_merr = tip_kpc*100.d0**(tip_msigma/10.d0) - tip_kpc !minus tip error in kpc
1692  kpc_perr = tip_kpc*100.d0**(tip_psigma/10.d0) - tip_kpc !plus tip error in kpc
1693
1694  END SUBROUTINE TipAndSigma
1695
1696  !-------------------------------------------------------------------------
1697
1698  FUNCTION func_i(m)    !This function feeds the photometric error as a function
1699  USE Global            !of magnitude to the 'GaussianKernel' subroutine.
1700  IMPLICIT NONE
1701
1702  REAL*8 :: func_i, m, c1, c2, c3
1703
1704  c1 = 0.001
1705  c3 = log(0.24) - log(0.11)
1706  c2 = c3*25.0 - log(0.24)
1707
1708  func_i = c1 + exp(c3*m - c2)
1709
```

```
1710  END FUNCTION
1711
1712  !-------------------------Rodrigo's poly selection tool-----------------------
1713
1714  SUBROUTINE PolySelect     !Used for selection of appropriate colour cut
1715  USE Global                !in colour-magnitude space
1716  IMPLICIT NONE
1717
1718
1719  integer MAXPT,ipol
1720  integer NPT_ggr,NPT_spatial
1721  parameter (MAXPT=100)
1722  real*4   XCOL_ggr(MAXPT),YMAG_ggr(MAXPT)
1723  real*4   X_spatial(MAXPT),Y_spatial(MAXPT)
1724  logical refine_CMDsel_ggr,refine_spatialsel
1725  !parameter (refine_CMDsel_ggr=.true.)
1726  parameter (refine_CMDsel_ggr=.false.)
1727  !parameter (refine_spatialsel=.true.)
1728  parameter (refine_spatialsel=.false.)
1729
1730  logical in_poly
1731  external in_poly
1732
1733  npt_ggr=0
1734  if (refine_CMDsel_ggr) then
1735     call pgsls(2)
1736     call pgmove(0.2,26.0)
1737     call pgdraw(0.2,15.0)
1738     call pgsls(1)
1739     call pglcur(MAXPT,NPT_ggr,XCOL_ggr,YMAG_ggr)
1740     open(2,file=TRIM(ADJUSTL(colcut)),status='unknown')
1741     write(2,*) NPT_ggr
1742     do ipol=1,NPT_ggr
1743        write(2,*) XCOL_ggr(ipol),YMAG_ggr(ipol)
1744     end do
1745     close(2)
1746     call pgsci(1)
1747     call pgadvance
1748  else
1749     open(2,file=TRIM(ADJUSTL(colcut)),status='old')
1750     read(2,*) NPT_ggr
1751     do ipol=1,NPT_ggr
1752        read(2,*) XCOL_ggr(ipol),YMAG_ggr(ipol)
1753     end do
1754     close(2)
1755     call pgsci(2)
1756     call pgslw(5)
1757     call pgline(NPT_ggr,XCOL_ggr,YMAG_ggr)
1758     call pgsci(1)
1759     call pgslw(1)
1760  end if
1761
1762  !-----------------Make colour cut to Signal Field-----------------
1763  j=0                                                              !
1764  DO i = 1, ndata                                                  !
1765     IF (in_poly(g_min_i(i),mag_i(i),NPT_ggr,XCOL_ggr,YMAG_ggr)) THEN!
1766        IF (mag_i(i) .le. flim .AND. mag_i(i) .ge. blim) THEN      !Makes new
1767           j = j+1                                                 !arrays
1768           mag_i_poly(j) = mag_i(i)                                !containing
1769           mag_g_poly(j) = mag_g(i)                                !only
1770           g_min_i_poly(j) = g_min_i(i)                            !stars
```

```
1771            xi_poly(j) = xki(i)                                    !within
1772            eta_poly(j) = eta(i)                                   !polygon
1773            truestar_poly(j) = truestar(i)                         !
1774         END IF                                                    !
1775      END IF                                                       !
1776   END DO                                                          !
1777
1778   ndata2 = j     !New number of stars in dataset after colour cut
1779
1780   !------------------Make colour cut to Bckgrnd Field-----------------
1781   j=0 ; k = 0                                                                !
1782   DO i = 1, bg_ndata                                                         !
1783      IF (in_poly(bg_g_min_i(i),bg_mag_i(i),NPT_ggr,XCOL_ggr,YMAG_ggr)) THEN  !
1784         IF (bg_mag_i(i) .le. 24.d0) THEN                                    !Makes new
1785            IF (bg_mag_i(i) .le. flim .AND. bg_mag_i(i) .ge. blim) THEN      !arrays for
1786               k = k+1                                                       !i and g-i
1787            END IF                                                           !containing
1788            j = j+1                                                          !only
1789            bg_mag_i_poly(j) = bg_mag_i(i)                                   !stars
1790            bg_mag_g_poly(j) = bg_mag_g(i)                                   !within
1791            bg_g_min_i_poly(j) = bg_g_min_i(i)                               !polygon
1792         END IF                                                              !
1793      END IF                                                                 !
1794   END DO                                                                    !
1795
1796   bg_ndata2 = j ; bg_ndata3 = k  !Stars in bckgrnd ; Stars in bckgrnd between blim & flim
1797
1798   END SUBROUTINE PolySelect
1799
1800   !-----------------------------------------
1801
1802   logical function in_poly(x,y,np,xp,yp)     !Used by PolySelect subroutine
1803   implicit none
1804
1805   real*4 x,y
1806   integer np
1807   real*4 xp(np),yp(np)
1808   real*4 tiny,xs,xe,ys,ye
1809   parameter (tiny=1.e-5)
1810
1811   real*4 simag,fimag
1812   external fimag
1813   integer j
1814
1815   simag=0.0
1816   do j=1,np
1817      if (j.lt.np) then
1818         xe=xp(j+1)
1819         xs=xp(j)
1820         ye=yp(j+1)
1821         ys=yp(j)
1822      else
1823         xe=xp(1)
1824         xs=xp(j)
1825         ye=yp(1)
1826         ys=yp(j)
1827      end if
1828      simag=simag+ fimag(x,xs,xe,y,ys,ye)
1829   end do
1830   if (abs(simag).gt.tiny) then
1831      in_poly=.true.
```

```
1832  else
1833      in_poly=.false.
1834  end if
1835
1836  end
1837
1838  !-----------------------------------------
1839
1840  real*4 function fimag(x0,xs,xe,y0,ys,ye)    !Used by PolySelect subroutine
1841  implicit none
1842
1843  real*4 x0,xs,xe,y0,ys,ye
1844  real*4 top,bot
1845
1846  top= -(xe-x0) * (ys-y0) + (ye-y0) * (xs-x0)
1847
1848  bot=  (xe-x0) * (xs-x0) + (ye-y0) * (ys-y0)
1849
1850  fimag=atan2(top,bot)
1851
1852  end
1853
1854  !-------------------------------------------------------------------------------
1855  !----------------------------Libpress Algorithms--------------------------------
```

Program: MF_TRGB_Feed.pl

Creation Date: 23 January 2012

Relevant Section: Ch. 4

Notes: This Perl scrip shows the individual parameters for each satellite fed to the program 'MF_TRGB.f95.' I have included it as it provides information specific to each satellite that is not given in Ch. 4. For each satellite, there are 16 inputs in the order described below. Note that for the dwarf spheroidal satellites, values for parameters 2 - 10 were provided by Nicolas Martin (Observatoire Astronomique, Universite de Strasbourg) and are due for publication in the near future. As an aside, it is worth noting that the weighting can effectively be turned off by specifying a very large Half-Light Radius, which produces an essentially flat object density profile across the field of view.

1. Object Name
2. Right Ascension Coordinate (hours)
3. Right Ascension Coordinate (minutes)
4. Right Ascension Coordinate (seconds)
5. Declination Coordinate (degrees)
6. Declination Coordinate (minutes)
7. Declination Coordinate (seconds)
8. Object Ellipticity
9. Object Half-Light Radius
10. Object Position Angle
11. Inner Cutoff Radius
12. Outer Cutoff Radius
13. Object Field Radius
14. Background Field Right Edge (Xi)
15. Background Field Left Edge (Xi)
16. File Name for Colour-Cut Polygon

```
#!/usr/bin/perl
system("./MF_TRGB.e AndromedaIe 0.0 45.0 40.0 38.0 2.0 18.5 0.26 3.9 25.0 0.0 0.3 0.3 0.0 2.0 ANDI.CMD"); #final
print "AndromedaI done.\n";
system("./MF_TRGB.e AndromedaIIe 1.0 16.0 26.9 33.0 26.0 1.9 0.13 5.0 27.0 0.0 0.4 0.4 5.0 8.0 ANDII.CMD"); #final
print "AndromedaII done.\n";
system("./MF_TRGB.e AndromedaIIIe 0.0 35.0 30.6 36.0 30.0 3.5 0.61 1.7 138.0 0.0175 0.2 0.2 -2.5 0.5 ANDIII.CMD"); #final
print "AndromedaIII done.\n";
system("./MF_TRGB.e AndromedaVe 1.0 10.0 17.1 47.0 37.0 45.4 0.27 1.6 41.0 0.011 0.2 0.2 3.0 6.0 ANDV.CMD"); #final
print "AndromedaV done.\n";
system("./MF_TRGB.e AndromedaIXe 0.0 52.0 52.5 43.0 11.0 58.2 0.0 1.9 105.0 0.0 0.15 0.15 1.45 2.25 ANDIX.CMD"); #final
print "AndromedaIX done.\n";
```

```
12 system(" ./MF_TRGB.e AndromedaXe 1.0 6.0 35.5 44.0 48.0 30.9 0.41 1.4 33.0 0.0 0.15 0.15 3.2 5.2 ANDX.CMD"); #final
13 print "AndromedaX done.\n";
14 system(" ./MF_TRGB.e AndromedaXIe 0.0 46.0 19.6 33.0 48.0 8.6 0.04 0.7 43.0 0.0 0.15 0.15 -0.75 2.25 ANDXI.CMD"); #final
15 print "AndromedaXI done.\n";
16 system(" ./MF_TRGB.e AndromedaXIIe 0.0 47.0 27.0 34.0 22.0 29.0 0.0 1.1 0.0 0.0 0.15 0.15 -0.5 2.5 ANDXII.CMD"); #final
17 print "AndromedaXII done.\n";
18 system(" ./MF_TRGB.e AndromedaXIIIe 0.0 51.0 50.9 33.0 0.0 14.5 0.60 0.8 -25.0 0.0 0.15 0.15 0.4 3.4 ANDXIII.CMD"); #final
19 print "AndromedaXIII done.\n";
20 system(" ./MF_TRGB.e AndromedaXIVe 0.0 51.0 35.1 29.0 41.0 14.1 0.31 1.6 -7.0 0.0 0.2 0.2 1.0 3.0 ANDXIV.CMD"); #final
21 print "AndromedaXIV done.\n";
22 system(" ./MF_TRGB.e AndromedaXVe 1.0 14.0 18.8 38.0 7.0 18.0 0.23 1.4 33.0 0.0 0.2 0.2 4.7 7.7 ANDXV.CMD"); #final
23 print "AndromedaXV done.\n";
24 system(" ./MF_TRGB.e AndromedaXVIe 0.0 59.0 30.1 32.0 22.0 32.9 0.27 0.92 104.0 0.005 0.2 0.2 1.7 4.7 ANDXVI.CMD"); #final
25 print "AndromedaXVI done.\n";
26 system(" ./MF_TRGB.e AndromedaXVIIe 0.0 37.0 6.5 44.0 19.0 20.1 0.36 1.3 121.0 0.0 0.2 0.2 -2.5 0.5 ANDXVII.CMD"); #final
27 print "AndromedaXVII done.\n";
28 system(" ./MF_TRGB.e AndromedaXVIIIe 0.0 2.0 16.0 45.0 5.0 33.2 0.15 0.72 91.0 0.0 0.1 0.1 -8.5 -6.0 ANDXVIII.CMD"); #final
29 print "AndromedaXVIII done.\n";
30 system(" ./MF_TRGB.e AndromedaXIXe 0.0 19.0 32.1 35.0 2.0 37.1 0.17 6.2 37.0 0.0 0.2 0.2 -7.5 -2.0 ANDXIX.CMD");
31 print "AndromedaXIX done.\n";
32 system(" ./MF_TRGB.e AndromedaXXe 0.0 7.0 30.5 35.0 7.0 39.4 0.07 0.48 69.0 0.0 0.15 0.15 -8.3 -6.0 ANDXX.CMD"); #final
33 print "AndromedaXX done.\n";
34 system(" ./MF_TRGB.e AndromedaXXIe 23.0 54.0 46.8 42.0 28.0 16.9 0.23 4.2 -36.0 0.0 0.3 0.3 -9.5 -6.5 ANDXXI.CMD"); #final
35 print "AndromedaXXI done.\n";
36 system(" ./MF_TRGB.e AndromedaXXIIe 1.0 27.0 40.2 28.0 5.0 26.0 0.62 0.92 -65.0 0.0 0.08 0.08 10.2 13.2 ANDXXII.CMD"); #final
37 print "AndromedaXXII done.\n";
38 system(" ./MF_TRGB.e AndromedaXXIIIe 1.0 29.0 20.8 38.0 43.0 27.8 0.39 5.1 -42.0 0.0 0.2 0.2 7.2 10.2 ANDXXIII.CMD"); #final
39 print "AndromedaXXIII done.\n";
40 system(" ./MF_TRGB.e AndromedaXXIVe 1.0 18.0 31.4 46.0 22.0 19.3 0.0 2.3 -87.0 0.0 0.125 0.125 5.0 8.0 ANDXXIV.CMD"); #final
41 print "AndromedaXXIV done.\n";
42 system(" ./MF_TRGB.e AndromedaXXVe 0.0 30.0 11.0 46.0 51.0 20.6 0.17 3.1 -3.0 0.0 0.2 0.2 -3.5 -1.0 ANDXXV.CMD"); #final
43 print "AndromedaXXV done.\n";
44 system(" ./MF_TRGB.e AndromedaXXVIe 0.0 23.0 45.7 47.0 54.0 43.6 0.55 1.3 -31.0 0.0 0.15 0.15 -4.2 -2.2 ANDXXVI.CMD"); #final
45 print "AndromedaXXVI done.\n";
46 system(" ./MF_TRGB.e AndromedaXXVIIe 0.0 37.0 36.4 45.0 22.0 19.0 0.75 15.8 -59.0 0.0 0.3 0.3 -2.5 0.5 ANDXXVII.CMD");# final
47 print "AndromedaXXVII done.\n";
48 system(" ./MF_TRGB.e AndromedaXXXe 0.0 36.0 34.7 49.0 38.0 47.0 0.33 1.5 -63.0 0.0 0.15 0.15 -1.5 -0.4 ANDXXX.CMD"); #final
49 print "AndromedaXXX done.\n";
50 #system(" ./MF_TRGB_NGC147e.e NGC147e_outer 0.0 33.0 12.0 48.0 30.0 31.0 0.44 10.0 28.0 0.28 0.33 0.6 -4.2 -2.2 NGC147.CMD");
51 #print "NGC147e_outer done.\n";
52 #system(" ./MF_TRGB_NGC147e.e NGC147e_inner 0.0 33.0 12.0 48.0 30.0 31.0 0.44 10.0 28.0 0.12 0.18 0.6 -4.0 -2.1 NGC147.CMD");
53 #print "NGC147e_inner done.\n";
54 #system(" ./MF_TRGB.e NGC147stream 0.0 33.0 12.0 48.0 30.0 31.0 0.0 6000.0 0.0 0.0 0.5 0.5 -4.0 -2.7 NGC147stream.CMD");
55 #print "NGC147stream done.\n";
56 #system(" ./MF_TRGB_NGC185e.e NGC185e_outer 0.0 38.0 57.97 48.0 20.0 14.56 0.26 6.0 41.0 0.18 0.26 0.6 0.0 1.0 NGC185.CMD");
57 #print "NGC185e_outer done.\n";
58 #system(" ./MF_TRGB_NGC205e.e NGC205e 0.0 40.0 22.075 41.0 41.0 7.08 0.50 13.0 -35.0 0.38 0.4 0.4 -1.9 -0.9 NGC205.CMD");
59 #print "NGC205e done.\n";
60 #system(" ./MF_TRGB_M33e.e M33e 1.0 33.0 50.904 30.0 39.0 35.79 0.4 6000.0 17.0 0.75 0.9 1.0 12.4 13.4 M33_ellipse.CMD");
61 #print "M33 done.\n";
62 #system(" ./MF_TRGB_M31e.e M31e 0.0 42.0 44.33 41.0 16.0 7.50 0.68 6000.0 37.0 2.45 2.5 2.5 9.0 10.0 M31_ellipse.CMD");
63 #print "M31 done.\n";
64 #system(" ./MF_TRGB_M31e_NE.e M31e_NE 0.0 42.0 44.33 41.0 16.0 7.50 0.68 6000.0 37.0 2.45 2.5 2.5 9.0 10.0 M31_ellipse.CMD");
65 #print "M31e_NE done.\n";
66 #system(" ./MF_TRGB_M31e_NW.e M31e_NW 0.0 42.0 44.33 41.0 16.0 7.50 0.68 6000.0 37.0 2.45 2.5 2.5 9.0 10.0 M31_ellipse.CMD");
67 #print "M31e_NW done.\n";
68 #system(" ./MF_TRGB_M31e_SE.e M31e_SE 0.0 42.0 44.33 41.0 16.0 7.50 0.68 6000.0 37.0 2.45 2.5 2.5 9.0 10.0 M31_ellipse.CMD");
69 #print "M31e_SE done.\n";
70 #system(" ./MF_TRGB_M31e_SW.e M31e_SW 0.0 42.0 44.33 41.0 16.0 7.50 0.68 6000.0 37.0 2.45 2.5 2.5 9.0 10.0 M31_ellipse.CMD");
71 #print "M31e_SW done.\n";
```

Program: MF_TRGB_Tester.f95

Creation Date: 8 December 2010

Relevant Section: §3.2 of Paper II (Ch. 4)

Notes: This program is the equivalent of 'MCMCTRGBTester2.f95' provided in Appendix B, but it has been updated for use with 'MF_TRGB.f95' and thus also provides the artificial stars with a radius representing their distance from the object's center. For simplicity, an ellipticity of 0 is assumed. For the sake of brevity, only the 'DataMaker' subroutine is shown, but the other subroutines called can be found in 'MF_TRGB.f95.'

```
1  MODULE Global !Defines all variables used by BayesianTRGB
2  IMPLICIT NONE
3
4  !--------------------General Program Parameters----------------------
5  INTEGER :: i, j, k, l, eval, idum = -9999, it, nit, trial
6  INTEGER :: ndata_max, nsamples, binspm, nbins, cmod_nbins, ghw, mm, ios
7  PARAMETER (ndata_max = 20000000, nsamples = 100)
8  PARAMETER (binspm = 100)
9  PARAMETER (nbins = 8*binspm + 1)
10 PARAMETER (nit = 50000)
11 INTEGER :: ndata, ndata2
12 INTEGER :: d1, d2, d3, d4
13 REAL*8 :: blim, flim, pi
14 PARAMETER (blim = 19.5d0)
15 PARAMETER (flim = 23.5d0)
16 PARAMETER (pi = ACOS(-1.e0))
17 INTEGER :: blimBins = INT(REAL((blim - 18.d0) * binspm)) + 1
18 INTEGER :: flimBins = INT(REAL((flim - 18.d0) * binspm)) + 1
19 REAL*8 :: randnum1, randnum2, randnum3, randnum4, randnum5, randnum6, randnum7
20 INTEGER :: randint
21 REAL*8 :: r1, r2, spotR, hb = 0.005d0
22 REAL*8 :: model(nbins,2), cmodel(nbins,2), magnitude(ndata_max)
23 REAL*8 :: histo_fine(nbins,2), histo_coarse(INT(0.25*(nbins-1.d0)) + 1,2)
24 REAL*8 :: w_histo_fine(nbins,2), w_histo_coarse(INT(0.25*(nbins-1.d0)) + 1,2)
25 REAL*8 :: data(ndata_max), cumulative_cmodel(nbins,2), f, f_hold, bfm(nbins)
26 REAL*8 :: cumulative_dist(2000,2)
27 REAL*8 :: mag_tip, mag, mag_cutoff = 24.e0, a
28 REAL*8 :: area, area2
29 REAL*8 :: modelnoise(nbins,2), noise(nbins) = 0.d0 , bg(nbins) = 0.d0
30 REAL*8 :: kernel(nbins,2) = 0.e0, scale, uplim, lowlim, gx
31 REAL*8 :: temp(nbins,2) = 0.e0, t
32 INTEGER :: starbin
33 REAL*8 :: tip(nsamples), tip_ord(nsamples), maxlogL(nsamples) = -999999999999.
34 REAL*8 :: tip_rec, tip_offset, tip_psigma, tip_msigma, Toffset_kpc, Tsigma_kpc
35 REAL*8 :: f_offset, tip_kpc, kpc_perr, kpc_merr, f_sigma, a_offset, a_sigma
36 REAL*8 :: f_rec, a_rec, tip_counts, f_counts, a_counts
37 REAL*8 :: tipminsig, tiplusig, fminsig, fplusig, aminsig, aplusig
38 REAL*8 :: mcounts, pcounts
39 INTEGER :: num_chains, cn, chain_compare, swap_count
40 PARAMETER (num_chains = 4)
41 REAL*8 :: swaprate = 1.d0/ 30.d0, logL(num_chains), LikeA(num_chains), LikeB(num_chains)
42 REAL*8 :: prob, sig_prob, bg_prob
43 REAL*8 :: beta, betaholder(num_chains) = (/ 1.d0, 0.25d0, 0.111d0, 0.001d0 /)
44 REAL*8 :: m_step(num_chains) = (/ 0.03d0, 0.06d0, 0.12d0, 0.3d0 /)
45 REAL*8 :: f_step(num_chains) = (/ 0.02d0, 0.04d0, 0.08d0, 0.2d0 /)
```

```
46  REAL*8 :: a_step(num_chains) = (/ 0.02d0, 0.04d0, 0.08d0, 0.2d0 /)
47  REAL*8 :: PTAR, par_hold(4)
48  REAL*8 :: x1(nit,num_chains), x2(nit,num_chains), x3(nit,num_chains), p(3), time(nit), r
49  REAL*8 :: post_y1(10*(nbins-1)+1) = 0.d0, post_x1(10*(nbins-1)+1), mlim
50  REAL*8 :: d_blim, bg_blim, d_flim, bg_flim
51  REAL*8 :: post_y2(nbins) = 0.d0, post_x2(nbins)
52  REAL*8 :: post_y3(2*nbins - 1) = 0.d0, post_x3(2*nbins - 1)
53  REAL*8 :: PPD_peak, Best_Combo(6)
54  CHARACTER :: argv*10, field*60, ch1*9, ch2*9, ch3*9, ch4*9, ch5*9, ch6*9, string*90
55
56  !----------------------For reading in PAndAS data--------------------
57  INTEGER :: iCCDt, clsg, clsi, ifieldt, iacc_t
58  REAL*4 :: xgt, ygt, g, dg, im, dim, xki_t, eta_t, FeH_phot_t, diff_tip_t, E_BV_t
59  REAL*8 :: ra_t, de_t
60
61  REAL*4 :: mag_g(ndata_max), mag_i(ndata_max), xki(ndata_max), eta(ndata_max)
62  REAL*4 :: g_min_i(ndata_max), mag_i_poly(ndata_max), g_min_i_poly(ndata_max)
63  REAL*4 :: xi_poly(ndata_max), eta_poly(ndata_max)
64  REAL*4 :: gmi
65
66  !------Additional parameters for calculating background stats-------
67  INTEGER :: bg_ndata, bg_ndata2, bg_ndata3
68  REAL*4 :: bg_mag_g(ndata_max), bg_mag_i(ndata_max), bg_xki(ndata_max), bg_eta(ndata_max)
69  REAL*4 :: bg_g_min_i(ndata_max), bg_mag_i_poly(ndata_max), bg_g_min_i_poly(ndata_max)
70  REAL*4 :: bg_gmi
71  REAL*8 :: bg_data(ndata_max)
72
73    !--SVD fitting of background--
74    INTEGER ma, mp, np, ndat
75    PARAMETER (ndat = INT(0.25*(nbins-1.d0)) + 1)
76    PARAMETER (np = 8)
77    PARAMETER (mp = ndat)
78    PARAMETER (ma = np)
79    REAL :: chisq, ay(ma), sig(ndat), u(mp,np), v(np,np), w(np), xa(ndat), ya(ndat)
80    REAL :: xt(ndat), yt(ndat)
81    REAL*8 :: bg_histo_coarse(ndat,2)
82    EXTERNAL :: funcs
83
84  !------Additional parameters for specifying object coordinates------
85  INTEGER :: Jop
86  REAL*8 :: XIop, ETAop
87  REAL*8 :: RAh, RAm, RAs, DecD, DecM, DecS, RA_rad, Dec_rad
88  REAL*8 :: tpRAh, tpRAm, tpRAs, tpDecD, tpDecM, tpDecS, tpRA_rad, tpDec_rad
89
90  !--Additional parameters for Matched Filters Subroutine 'Weighter'--
91  INTEGER :: rhobins, rhobins2
92  PARAMETER (rhobins = 10)
93  REAL*4 :: C_O_F_dist(ndata_max), Density(rhobins,2), rhofit(rhobins,2)
94  REAL*8 :: weight(ndata_max)
95
96    !--Fitting to Density Profile--
97    INTEGER :: mwt, ndat2
98    PARAMETER (ndat2 = rhobins)
99    REAL offset, gradient, chi2, q, siga, sigb, sigma(ndat2)
100
101 !------------------------When f is known--------------------------
102 INTEGER :: bg_stars, sig_stars
103 REAL*8 :: bg_area, sig_area, bg_density
104 REAL*8 :: known_f, bg_stars_in_sig_field
105 REAL*8 :: sig_field_radius = 0.2d0, bg_low_xi = -5.d0, bg_up_xi = 13.d0
106
```

```
107 END MODULE Global
108
109 !-------------------------------------------------------------------------
110
111 PROGRAM BayesianTRGBsatellite     !Master program
112 USE Global
113 IMPLICIT NONE
114
115 mm = IARGC()
116
117 IF (mm==4) THEN                              !
118    CALL GETARG(1, argv)                      !
119    READ (argv,*,iostat=ios) mag_tip          !
120    CALL GETARG(2, argv)                      !
121    READ (argv,*,iostat=ios) a                !
122    CALL GETARG(3, argv)                      !
123    READ (argv,*,iostat=ios) ndata            !Indicates the arguments to be
124    CALL GETARG(4, argv)                      !set in the command line
125    READ (argv,*,iostat=ios) f                !
126 ELSE                                         !
127    WRITE(*,*) "You must enter 4 arguments:"  !
128    stop ;                                    !
129 END IF                                       !
130
131 WRITE (ch1,*) mag_tip       !
132 WRITE (ch2,*) a             !
133 WRITE (ch3,*) ndata         !
134 IF (f .eq. 0.d0) THEN       !Generate test identifying character string
135    WRITE (ch4,*) '0'         !to become file name using mag_tip, ndata and f
136 ELSE                        !e.g. 'MCMC_Test/T_20.5-0.3-1000-0.2'
137    WRITE (ch4,*) f           !
138 END IF                      !
139
140 ndata2 = 0
141
142 WRITE (field,*) 'MF_MCMC_Test/T_' // TRIM(ADJUSTL(ch1)) &
143                              // '-' // TRIM(ADJUSTL(ch2)) &
144                              // '-' // TRIM(ADJUSTL(ch3)) &
145                              // '-' // TRIM(ADJUSTL(ch4))
146
147 string = TRIM(ADJUSTL(field)) // '/test.dat'
148 OPEN(3, file=TRIM(ADJUSTL(string)), status = 'unknown')
149 WRITE (3,*) "Field Name:", field
150 WRITE (3,*) "ndata =", TRIM(ADJUSTL(ch3))
151 WRITE (3,*) "f =", TRIM(ADJUSTL(ch4))
152
153                        !
154 CALL random_seed       !
155
156 CALL NoiseMake         !
157 CALL ModelMake         !
158 CALL Convolution       !
159
160 cmodel(:,2) = (1.d0 - f) * cmodel(:,2) + f * noise
161
162 CALL DataMaker         !
163 CALL Weighter          !
164 CALL NoiseMake         !CALL
165 CALL MCMC              !
166 CALL TipAndSigma       !SUBROUTINES
167 CALL PosteriorPlot     !
```

```
168  CALL OtherPlots       !
169  CALL DataHist         !
170  CALL w_DataHist       !
171
172
173  IF (num_chains .ne. 1) THEN
174  WRITE (3,*) "Proposed Swaps with Cold Sampler Chain:", chain_compare
175  WRITE (3,*) "Accepted Swaps with Cold Sampler Chain:", swap_count
176  WRITE (3,*) "Parallel Tempering Acceptance Rate:", REAL(swap_count)/ REAL(chain_compare)
177  END IF
178  WRITE (3, '(3a11)') "   tip_mag:", "  +sigma:  ", "  -sigma: "       !
179  WRITE (3, '(3F10.3)') tip_rec, tip_psigma, tip_msigma                  !
180  WRITE (3, '(2a11)') "   f:       ", "   sigma:  "                      !
181  WRITE (3, '(2F10.3)') f_rec, f_sigma                                   !Write results
182  WRITE (3, '(2a11)') "   a:       ", "   sigma:  "                      !to file
183  WRITE (3, '(2F10.3)') a_rec, a_sigma                                   !
184  WRITE (3,*) "Distance =", REAL(tip_kpc), "kpc"                         !
185  WRITE (3,*) "Error = +", REAL(kpc_perr), "kpc -", REAL(kpc_merr), "kpc"          !
186  WRITE (3,*) "Average Error =", REAL((ABS(kpc_perr) + ABS(kpc_merr))/2.d0), "kpc"
187  WRITE (3,*) "Tip_Mag_&_Error =", tip_rec, REAL(tip_psigma), REAL(tip_msigma)
188  WRITE (3,*) "Offset =", REAL(tip_rec - 20.5d0), "=", REAL(tip_kpc - (100.d0**((20.5d0 + 3.44d0)/10.d0))/100.d0), "kpc"
189
190  END PROGRAM BayesianTRGBsatellite
191
192  !-------------------------------------------------------------------------
193
194  SUBROUTINE DataMaker  !Generates artificial stars with magnitudes from
195  USE Global            !model luminosity function and positions from
196  IMPLICIT NONE         !model density profile
197
198  cumulative_cmodel(:,1) = cmodel(:,1)
199
200  cumulative_cmodel(1,2) = cmodel(1,2)                              !Effective
201  DO i = 2, cmod_nbins                                              !integral of
202     cumulative_cmodel(i,2) = cumulative_cmodel(i-1,2) + cmodel(i,2)!convolved
203  END DO                                                            !model
204
205  DO i = 1, ndata                                          !
206     CALL random_number(randnum6)                          !
207     randnum6 = cumulative_cmodel(blimBins,2) + &
208                randnum6 * (cumulative_cmodel(flimBins,2) - cumulative_cmodel(blimBins,2))
209
210     DO j = flimBins, blimBins, -1                         !Generates 'ndata'
211        IF (randnum6 .le. cumulative_cmodel(j,2)) THEN     !magnitude datapoints
212           IF (randnum6 .gt. cumulative_cmodel(j-1,2)) THEN!from the convolved
213              data(i) = cumulative_cmodel(j-1,1)          !model
214              exit;                                       !
215           END IF                                         !
216        END IF                                            !
217     END DO                                               !
218  END DO                                                  !
219
220  cumulative_dist = 0.d0                                        !
221  cumulative_dist(1,1) = 0.0001                                 !Generates model radial density
222  cumulative_dist(1,2) = 10.d0 ** (5.610696 - 0.0013362497)     !profile based on that fitted
223                                                                !to Andromeda II (with approximation
224  DO i = 2, 2000                                                !of zero ellipticity)
225  cumulative_dist(i,1) = i * 0.0001d0                           !
226  cumulative_dist(i,2) = cumulative_dist(i-1,2) + (10.d0 ** (5.610696 - 13.362497 * cumulative_dist(i,1)))
227  END DO
228
```

```
229 DO i = 1, ndata                                                    !
230    CALL random_number(randnum6)                                    !
231    IF (data(i) .ge. mag_tip) THEN                                  !
232       CALL random_number(randnum7)                                 !Draws random
233       IF (randnum7 .gt. f * (flim - mag_tip)/(flim - blim)) THEN   !
234          randnum6 = randnum6 * cumulative_dist(2000,2)             !radial distance
235          DO j = 2000, 2, -1                                        !
236             IF (randnum6 .le. cumulative_dist(j,2) .and. randnum6 .gt. cumulative_dist(j-1,2)) THEN
237                C_O_F_dist(i) = cumulative_dist(j-1,1)              !
238                exit;                                               !for each
239             END IF                                                 !
240          END DO                                                    !star based
241       ELSE                                                         !
242          C_O_F_dist(i) = SQRT((randnum6 * (pi * sig_field_radius ** 2))/ pi)!on above
243       END IF                                                       !
244    ELSE                                                            !model
245       C_O_F_dist(i) = SQRT((randnum6 * (pi * sig_field_radius ** 2))/ pi)  !
246    END IF                                                          !
247 END DO                                                            !
248
249 sig_area = pi * (sig_field_radius ** 2.d0) !
250 sig_stars = ndata                          !For calculating ratio
251 ndata2 = ndata                             !of RGB to background
252 bg_density = f * REAL(sig_stars)/ sig_area !
253
254
255 !---------------plot magnitude vs. radius-------------------
256 string = TRIM(ADJUSTL(field)) // '/mag_vs_rad.ps/CPS'
257
258 CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
259
260 CALL pgenv(19.5, 23.5, 0., MAXVAL(REAL(C_O_F_dist)), 0, 0)
261 CALL pgslw(3)
262 CALL pgpt (ndata, REAL(data), REAL(C_O_F_dist), -1)
263 CALL pgslw(1)
264 CALL pglab('magnitude', 'radius', '')
265
266 CALL pgend
267
268 END SUBROUTINE DataMaker
269
270 !-------------------------------------------------------------------------------
```

Program: Multi_MCMC_Result_Plotter.f95

Creation Date: 6 Feb 2012 (first version 10 Dec 2010)

Relevant Sections: Ch. 3 & Ch. 4

Notes: This program was created to take the posterior distributions generated by the TRGB algorithm (e.g. 'MF_TRGB.f95') and produce more polished versions of the figures for use in papers I and II. In particular, it colour-codes the distributions to indicate the 1σ, 90% and 99% credibility intervals. It also generates a contour map of the distribution of the tip magnitude verses the RGB slope model parameters (see Fig. 8 of Paper I; Ch. 3) - i.e. a 3D surface from which the individual parameter posterior distributions are created by marginalizing over the other parameter. The actual distance posterior distributions for each object are also created by this program. This is achieved by sampling the posterior distribution in the tip magnitude along with the probability distributions for the absolute magnitude of the tip and the extinction along the line of sight (see the 'Dist_Error' subroutine). The halo density prior (see §3.3 of Paper II; Ch. 4) is also generated and applied in this program, as are the hundredth-percentile tables of the object distance distributions published alongside Paper II (see Table 1 of Paper II for example).

```
1  MODULE Global !Defines all variables used by BayesianTRGB
2  IMPLICIT NONE
3
4  INTEGER :: i, ios, j, k, ndata_max, ndata, ndata_M31, nbins, binspm, d1, d2, d3, d4, mm
5  PARAMETER (ndata_max = 7100000, binspm = 100)
6  PARAMETER (nbins = 8 * binspm)
7  REAL*8 :: pi
8  PARAMETER (pi = ACOS(-1.e0))
9  REAL :: it(ndata_max), mag_tip(ndata_max), f(ndata_max), a(ndata_max)
10 REAL :: LikeA(ndata_max), LikeB(ndata_max), M31_dist_ppd(ndata_max)
11 REAL :: tip_PPD(10*(nbins-1)+1, 2) = 0.d0
12 REAL :: f_PPD(nbins, 2) = 0.d0, a_PPD(2*nbins - 1, 2) = 0.d0
13 REAL :: mag_tip_ll, mag_tip_ul
14 REAL :: f_ll = 0., f_ul = 1.
15 REAL :: a_ll, a_ul
16 REAL :: tip_rec, f_rec, a_rec, PPD_peak, tip_kpc, tip_counts, mcounts, pcounts, tip_msigma, tip_psigma
17 REAL :: f_sigma, a_sigma, kpc_merr, kpc_perr, f_counts, a_counts, fminsig, fplusig, aminsig, aplusig
18 REAL :: tip_m90, tip_p90, tip_m99, tip_p99, xpts(2), ypts(2)
19 REAL :: xi_coord, eta_coord
20 REAL*8 :: RA, DEC, xi_dble, eta_dble
21 CHARACTER :: argv*20, field*60, plot_dir*60, string*200, string2*200, command1*200, command2*300
22
23 !---------------For Contour Plot-----------------
24 INTEGER :: cont_bins, ncontours
25 PARAMETER (cont_bins = 75)
26 PARAMETER (ncontours = 20)
27 REAL :: Cont(cont_bins, cont_bins), clevels(ncontours), TR(6)
28
29 !----------For Distance Distribution-------------
```

```
30  INTEGER :: idum = -9999, nsamples, DistBins, prior_type
31  PARAMETER (nsamples = 500000)
32  REAL :: M_TRGB, Ext, Ext_0, Tip, Dist_PPD, Dist_PPDx(4000), Dist_PPDy(4000), Dist_PPD_min, Dist_PPD_max
33  REAL :: dist_rec, dist_counts, dist_msigma, dist_psigma, dist_m90, dist_p90, dist_m99, dist_p99
34  REAL :: DistPrior(4000), alpha, slope, flat, hwhm, theta
35  REAL :: M31_to_obj_x(4001), M31_to_obj_y(4001), M31_to_obj, m31_dist
36  REAL :: M31_dist_rec, M31_dist_psigma, M31_dist_msigma
37  REAL*8 :: randnum
38
39  !---For converting distances back to magnitudes---
40  REAL :: dist2mag_rec, d2m_msigma, d2m_psigma, d2m_m90, d2m_p90, d2m_m99, d2m_p99
41
42  !--------For Hundredth Percentile Table-----------
43  REAL :: cum_dist, perc, dist_at_perc(100,2)
44  LOGICAL next
45
46  END MODULE Global
47
48  !-------------------------------------------------------------------------------------------------
49
50  PROGRAM MCMC_Result_Plotter
51  USE Global
52  IMPLICIT NONE
53
54  DOUBLE PRECISION :: sla_DSEP
55
56  mm = IARGC()
57
58  IF (mm==9) THEN                              !
59     CALL GETARG(1, argv)                      !
60     READ (argv,*,iostat=ios) field            !
61     CALL GETARG(2, argv)                      !
62     READ (argv,*,iostat=ios) Ext_0            !
63     CALL GETARG(3, argv)                      !
64     READ (argv,*,iostat=ios) mag_tip_ll       !
65     CALL GETARG(4, argv)                      !
66     READ (argv,*,iostat=ios) mag_tip_ul       !Indicates the arguments to be
67     CALL GETARG(5, argv)                      !set in the command line
68     READ (argv,*,iostat=ios) a_ll             !
69     CALL GETARG(6, argv)                      !
70     READ (argv,*,iostat=ios) a_ul             !
71     CALL GETARG(7, argv)                      !
72     READ (argv,*,iostat=ios) xi_coord         !
73     CALL GETARG(8, argv)                      !
74     READ (argv,*,iostat=ios) eta_coord        !
75     CALL GETARG(9, argv)                      !
76     READ (argv,*,iostat=ios) plot_dir         !
77  ELSE                                         !
78     WRITE(*,*) "You must enter 9 arguments:"  !
79     stop ;                                    !
80  END IF                                       !
81
82
83     xi_coord = xi_coord * (pi/180.e0)            !Convert angles from
84     eta_coord = eta_coord * (pi/180.e0)          !degrees to radians
85
86     xi_dble = xi_coord ; eta_dble = eta_coord                !
87     CALL sla_DTP2S(xi_dble, eta_dble, 0.d0, 0.d0, RA, DEC)   !Convert tangent plane
88     IF (xi_dble .lt. 0.d0) then                              !projection angles into
89        RA = RA - (2.e0 * pi)                                 !their true angles using
90     END IF                                                   !sla_DTP2S
```

```
91      xi_coord = RA                                           !
92      eta_coord = DEC                                         !
93
94      xi_dble = xi_coord                                  !Find the true angle
95      eta_dble = eta_coord                                !theta - the angle on
96      theta = sla_DSEP(0.d0, 0.d0, xi_dble, eta_dble)     !the sky between M31
97                                                          !and the object
98                                                          !(uses sla_DSEP)
99
100     xi_coord = xi_coord * (180.e0/pi)               !Convert back
101     eta_coord = eta_coord * (180.e0/pi)             !to degrees
102
103     WRITE (*,*) xi_coord, eta_coord, theta * (180.e0/pi)
104
105   WRITE (string,*) './' // TRIM(ADJUSTL(field)) // '/' // TRIM(ADJUSTL(plot_dir))
106
107   WRITE (command1,*) 'mkdir ' // TRIM(ADJUSTL(string))
108
109   call system(command1)
110
111   OPEN (unit = 1, file = './' // TRIM(ADJUSTL(field)) // '/MCMC_steps.dat', status = 'old')  !Open input
112   OPEN (unit = 2, file = './' // TRIM(ADJUSTL(string)) // '/results.dat', status = 'unknown')!and output
113   OPEN (unit = 3, file = './M31e/other_plots/M31_Distance_PPD.dat', status = 'old')          !files
114
115   WRITE (2,*) "Field: ", TRIM(ADJUSTL(field))                       !
116   WRITE (2,*) "Coordinates: xi = ", xi_coord, ", eta = ", eta_coord   !
117   WRITE (2,*) "Plot Directory: ", TRIM(ADJUSTL(string))             !Print basic object
118   WRITE (2,*) " "                                                   !info to file
119   WRITE (2,*) "Extinction in SDSS i: ", Ext_0                       !
120   WRITE (2,*) "E(B-V): ", Ext_0 / 2.086e0                           !
121
122   i = 0 ; ios = 0
123   DO WHILE (.TRUE.)  !Reads data until end of input file and puts it into arrays
124      i=i+1
125      READ (1, *, IOSTAT = ios) it(i), mag_tip(i), f(i), a(i), LikeA(i), LikeB(i)
126      if (ios == 0) then ;
127      else if (ios == -1) then ;
128         i=i-1
129         exit ;
130      else if (ios > 0) then ;
131         i=i-1
132         cycle
133      end if
134   END DO
135   ndata = i - 1
136
137   i = 0 ; ios = 0
138   DO WHILE (.TRUE.)  !Reads M31 distance sample data until end of input file and puts it into an array
139      i=i+1
140      READ (3, *, IOSTAT = ios) M31_dist_ppd(i)
141      if (ios == 0) then ;
142      else if (ios == -1) then ;
143         i=i-1
144         exit ;
145      else if (ios > 0) then ;
146         i=i-1
147         cycle
148      end if
149   END DO
150   ndata_M31 = i - 1
151
```

```
152 CALL random_seed     !
153 CALL PosteriorBuild  !CALL
154 CALL PosteriorPlot   !
155 CALL OtherPlots      !SUBROUTINES
156 CALL Dist_Error      !
157
158 END PROGRAM MCMC_Result_Plotter
159
160 !-----------------------------------------------------------------------------------------
161
162 SUBROUTINE PosteriorBuild
163 USE Global
164 IMPLICIT NONE
165
166 DO i = 1, 10*(nbins-1)+1                                    !
167     tip_PPD(i, 1) = 18.d0 + (REAL(i) - 1.d0)/REAL(10*binspm) !
168 END DO                                                      !
169                                                             !
170 DO i = 1, nbins                                            !x-values of
171     f_PPD(i, 1) = (REAL(i) - 1.d0)/REAL(nbins - 1)          !PPD histograms
172 END DO                                                      !
173                                                             !
174 DO i = 1, 2*nbins - 1                                      !
175     a_PPD(i, 1) = (REAL(i) - 1.d0)/REAL(nbins - 1)          !
176 END DO                                                      !
177
178 DO i = 1, ndata                                            !
179 tip_PPD(INT((mag_tip(i) - 18.d0)*10*binspm + 1), 2) = &     !
180 tip_PPD(INT((mag_tip(i) - 18.d0)*10*binspm + 1), 2) + 1.d0 !
181 f_PPD(INT(f(i) * (nbins - 1)) + 1, 2) = &                  !y-values of
182 f_PPD(INT(f(i) * (nbins - 1)) + 1, 2) + 1.d0               !PPD histograms
183 a_PPD(INT(a(i) * (nbins - 1)) + 1, 2) = &                  !
184 a_PPD(INT(a(i) * (nbins - 1)) + 1, 2) + 1.d0               !
185 END DO                                                     !
186
187 CALL Confidence
188
189 END SUBROUTINE PosteriorBuild
190
191 !-----------------------------------------------------------------------------------------
192
193 SUBROUTINE PosteriorPlot  !Plots posterior distributions in tip magnitude and a
194 USE Global                !tip magnitude PPD is plotted with credibility intervals
195 IMPLICIT NONE
196
197 tip_PPD(:,2) = tip_PPD(:,2)/ndata ; f_PPD(:,2) = f_PPD(:,2)/ndata
198 a_PPD(:,2) = a_PPD(:,2)/ndata
199
200 !-----------------------------Plots mag_tip posterior plot
201 string2 = TRIM(ADJUSTL(string)) // '/bw_mag_tip_postplot.ps/CPS'
202 CALL pgbegin(0,TRIM(ADJUSTL(string2)),1,1)
203
204 CALL pgenv(mag_tip_ll, mag_tip_ul, 0., 1.1*MAXVAL(tip_PPD(:,2)), 0, 0)
205 CALL pgbin (10*(nbins-1)+1, tip_PPD(:,1), tip_PPD(:,2),.false.)
206 CALL pglab('Proposed_i\d0\u_tip_magnitude', 'Probability', '')
207
208 CALL pgend
209
210 WRITE (command2,*) 'convert_-rotate_90_' // TRIM(ADJUSTL(string)) // &
211                    '/bw_mag_tip_postplot.ps_' // TRIM(ADJUSTL(string)) // &
212                    '/bw_mag_tip_postplot.jpg'
```

```
213
214  call system(command2)
215
216  !------------------------------Plots mag_tip posterior plot with confidence levels
217  string2 = TRIM(ADJUSTL(string)) // '/mag_tip_postplot.ps/CPS'
218  CALL pgbegin(0,TRIM(ADJUSTL(string2)),1,1)
219
220  CALL pgenv(mag_tip_ll, mag_tip_ul, 0., 1.1*MAXVAL(tip_PPD(:,2)), 0, 0)
221
222  DO i = 1, 10*(nbins-1)+1
223     IF (tip_PPD(i,1) .ge. tip_rec - tip_msigma .and. tip_PPD(i,1) .lt. tip_rec + tip_psigma) THEN
224        CALL pgsci(2)                                          !
225        CALL pgbin (2, tip_PPD(i,1), tip_PPD(i,2),.false.)     !
226        IF (tip_PPD(i,1) .eq. tip_rec - tip_msigma) THEN       !
227           xpts = tip_PPD(i,1)                                 !
228           ypts(1) = 0.e0 ; ypts(2) = tip_PPD(i,2)             !One Sigma
229           CALL pgline (2, xpts, ypts)                         !
230        END IF                                                 !Credibility
231        IF (tip_PPD(i+1,1) .eq. tip_rec + tip_psigma) THEN     !
232           xpts = tip_PPD(i+1,1)                               !Interval
233           ypts(1) = 0.e0 ; ypts(2) = tip_PPD(i,2)             !
234           CALL pgline (2, xpts, ypts)                         !
235        END IF                                                 !
236     ELSE IF (tip_PPD(i,1) .ge. tip_rec - tip_m90 .and. tip_PPD(i,1) .lt. tip_rec + tip_p90) THEN
237        CALL pgsci(3)                                          !
238        CALL pgbin (2, tip_PPD(i,1), tip_PPD(i,2),.false.)     !
239        IF (tip_PPD(i,1) .eq. tip_rec - tip_m90) THEN          !
240           xpts = tip_PPD(i,1)                                 !
241           ypts(1) = 0.e0 ; ypts(2) = tip_PPD(i,2)             !90 percent
242           CALL pgline (2, xpts, ypts)                         !
243        END IF                                                 !Credibility
244        IF (tip_PPD(i+1,1) .eq. tip_rec + tip_p90) THEN        !
245           xpts = tip_PPD(i+1,1)                               !Interval
246           ypts(1) = 0.e0 ; ypts(2) = tip_PPD(i,2)             !
247           CALL pgline (2, xpts, ypts)                         !
248        END IF                                                 !
249     ELSE IF (tip_PPD(i,1) .ge. tip_rec - tip_m99 .and. tip_PPD(i,1) .lt. tip_rec + tip_p99) THEN
250        CALL pgsci(4)                                          !
251        CALL pgbin (2, tip_PPD(i,1), tip_PPD(i,2),.false.)     !
252        IF (tip_PPD(i,1) .eq. tip_rec - tip_m99) THEN          !
253           xpts = tip_PPD(i,1)                                 !
254           ypts(1) = 0.e0 ; ypts(2) = tip_PPD(i,2)             !99 percent
255           CALL pgline (2, xpts, ypts)                         !
256        END IF                                                 !Credibility
257        IF (tip_PPD(i+1,1) .eq. tip_rec + tip_p99) THEN        !
258           xpts = tip_PPD(i+1,1)                               !Interval
259           ypts(1) = 0.e0 ; ypts(2) = tip_PPD(i,2)             !
260           CALL pgline (2, xpts, ypts)                         !
261        END IF                                                 !
262     ELSE
263        CALL pgsci(1)                                          !Distribution
264        CALL pgbin (2, tip_PPD(i,1), tip_PPD(i,2),.false.)     !outside of 99 %
265     END IF                                                    !Cred. Interval
266  END DO
267
268  CALL pgsci(1)
269
270  CALL pglab('Proposed i\d0\u tip magnitude', 'Probability', '')
271
272  CALL pgend
273
```

```
274  WRITE (command2,*) 'convert -rotate 90 ' // TRIM(ADJUSTL(string)) // &
275                         '/mag_tip_postplot.ps ' // TRIM(ADJUSTL(string)) // &
276                         '/mag_tip_postplot.jpg'
277
278  call system(command2)
279
280  !------------------------------Plots a posterior plot
281  string2 = TRIM(ADJUSTL(string)) // '/a_postplot.ps/CPS'
282  CALL pgbegin(0,TRIM(ADJUSTL(string2)),1,1)
283
284  CALL pgenv(a_ll, a_ul, 0., 1.1*MAXVAL(a_PPD(:,2)), 0, 0)
285  CALL pgbin (2*nbins-1, a_PPD(:,1), a_PPD(:,2),.false.)
286  CALL pglab('Proposed value for LF slope (a)', 'Probability', '')
287
288  CALL pgend
289
290  WRITE (command2,*) 'convert -rotate 90 ' // TRIM(ADJUSTL(string)) // &
291                         '/a_postplot.ps ' // TRIM(ADJUSTL(string)) // &
292                         '/a_postplot.jpg'
293
294  call system(command2)
295
296  tip_PPD(:,2) = tip_PPD(:,2)*ndata ; f_PPD(:,2) = f_PPD(:,2)*ndata
297  a_PPD(:,2) = a_PPD(:,2)*ndata
298
299  END SUBROUTINE PosteriorPlot
300
301  !-------------------------------------------------------------------------------------------
302
303  SUBROUTINE OtherPlots
304  USE Global
305  IMPLICIT NONE
306
307  !------------Values of 'a' for each value of 'mag_tip' - contour plot
308
309  Cont = 0.e0
310  TR = 0.e0
311  TR(1) = mag_tip_ll ; TR(2) = (mag_tip_ul - mag_tip_ll)/REAL(cont_bins) ; TR(4) = a_ll ; TR(6) = (a_ul - a_ll)/REAL(cont_bins)
312
313        DO k = 1, ndata
314           i = INT((mag_tip(k) - TR(1))/TR(2)) + 1
315           j = INT((a(k) - TR(4))/TR(6)) + 1
316           if ( i>0 .and. i<=cont_bins .and. j>0 .and. j<=cont_bins) Cont(i,j) = Cont(i,j) + 1.e0
317        END DO
318
319  DO i = 1, ncontours
320     clevels(i) = 0.e0 + i*MAXVAL(Cont)/REAL(ncontours)
321  END DO
322
323  string2 = TRIM(ADJUSTL(string)) // '/m_vs_a_contour.ps/CPS'
324  CALL pgbegin(0,TRIM(ADJUSTL(string2)),1,1)
325
326  CALL pgenv(mag_tip_ll,  mag_tip_ul, a_ll, a_ul, 0, 0)
327  CALL PGCONT (Cont, cont_bins, cont_bins, 1, cont_bins, 1, cont_bins, clevels, ncontours, TR)
328  CALL pglab('Proposed i\d0\u tip magnitude', 'Proposed value of a', '')
329
330  CALL pgend
331
332  WRITE (command2,*) 'convert -rotate 90 ' // TRIM(ADJUSTL(string)) // &
333                         '/m_vs_a_contour.ps ' // TRIM(ADJUSTL(string)) // &
334                         '/m_vs_a_contour.jpg'
```

```
335
336  call system(command2)
337
338  END SUBROUTINE OtherPlots
339
340  !-------------------------------------------------------------------------------------------
341
342  SUBROUTINE Confidence  !Identifies the best parameter values and
343  USE Global             !their associated 1 sigma errors from the
344  IMPLICIT NONE          !respective posterior plots.
345
346  PPD_peak = 0.d0                          !
347  DO i = 1, 10*(nbins-1)+1                 !
348     IF (tip_PPD(i,2) .gt. PPD_peak) THEN  !
349        PPD_peak = tip_PPD(i,2)            !Find best fit TRGB value
350        tip_rec = tip_PPD(i,1)             !
351     END IF                                !
352  END DO                                   !
353
354  PPD_peak = 0.d0                          !
355  DO i = 1, nbins                          !
356     IF (f_PPD(i,2) .gt. PPD_peak) THEN    !
357        PPD_peak = f_PPD(i,2)              !Find best fit f value
358        f_rec = f_PPD(i,1)                 !
359     END IF                                !
360  END DO                                   !
361
362  PPD_peak = 0.d0                          !
363  DO i = 1, 2*nbins - 1                    !
364     IF (a_PPD(i,2) .gt. PPD_peak) THEN    !
365        PPD_peak = a_PPD(i,2)              !Find best fit a value
366        a_rec = a_PPD(i,1)                 !
367     END IF                                !
368  END DO                                   !
369
370  tip_kpc = (100.d0**((tip_rec + 3.44d0)/10.d0))/100.d0 !Distance inferred from
371                                                        !tip magnitude in kpc
372
373  tip_counts = 0.d0 ; mcounts = 0.d0                          !
374  DO i = MAXLOC(tip_PPD(:,2), DIM = 1), 1, -1                 !
375     mcounts = mcounts + tip_PPD(i,2)                         !
376  END DO                                                      !
377  DO i = MAXLOC(tip_PPD(:,2), DIM = 1), 1, -1                 !
378     tip_counts = tip_counts + tip_PPD(i,2)                   !Finds negative one sigma
379     IF (tip_counts .ge. 0.682*mcounts) THEN                  !error in magnitudes
380        tip_msigma = ((REAL(i) - 1.d0)/REAL(10*binspm)) + 18.d0 !
381        tip_msigma = tip_rec - tip_msigma                     !
382        exit                                                  !
383     END IF                                                   !
384  END DO                                                      !
385
386  tip_counts = 0.d0 ; pcounts = 0.d0                          !
387  DO i = MAXLOC(tip_PPD(:,2), DIM = 1), 10*(nbins-1)+1        !
388     pcounts = pcounts + tip_PPD(i,2)                         !
389  END DO                                                      !
390  DO i = MAXLOC(tip_PPD(:,2), DIM = 1), 10*(nbins-1)+1        !
391     tip_counts = tip_counts + tip_PPD(i,2)                   !Finds positive one sigma
392     IF (tip_counts .ge. 0.682*pcounts) THEN                  !error in magnitudes
393        tip_psigma = ((REAL(i) - 1.d0)/REAL(10*binspm)) + 18.d0 !
394        tip_psigma = tip_psigma - tip_rec                     !
395        exit                                                  !
```

```
396     END IF                                                  !
397   END DO                                                    !
398
399   tip_counts = 0.d0 ; mcounts = 0.d0                        !
400   DO i = MAXLOC(tip_PPD(:,2), DIM = 1), 1, -1               !
401     mcounts = mcounts + tip_PPD(i,2)                        !
402   END DO                                                    !
403   DO i = MAXLOC(tip_PPD(:,2), DIM = 1), 1, -1               !
404     tip_counts = tip_counts + tip_PPD(i,2)                  !Finds negative 90% confidence
405     IF (tip_counts .ge. 0.9d0*mcounts) THEN                 !error in magnitudes
406       tip_m90 = ((REAL(i) - 1.d0)/REAL(10*binspm)) + 18.d0  !
407       tip_m90 = tip_rec - tip_m90                           !
408       exit                                                  !
409     END IF                                                  !
410   END DO                                                    !
411
412   tip_counts = 0.d0 ; pcounts = 0.d0                        !
413   DO i = MAXLOC(tip_PPD(:,2), DIM = 1), 10*(nbins-1)+1      !
414     pcounts = pcounts + tip_PPD(i,2)                        !
415   END DO                                                    !
416   DO i = MAXLOC(tip_PPD(:,2), DIM = 1), 10*(nbins-1)+1      !
417     tip_counts = tip_counts + tip_PPD(i,2)                  !Finds positive 90% confidence
418     IF (tip_counts .ge. 0.9d0*pcounts) THEN                 !error in magnitudes
419       tip_p90 = ((REAL(i) - 1.d0)/REAL(10*binspm)) + 18.d0  !
420       tip_p90 = tip_p90 - tip_rec                           !
421       exit                                                  !
422     END IF                                                  !
423   END DO                                                    !
424
425   tip_counts = 0.d0 ; mcounts = 0.d0                        !
426   DO i = MAXLOC(tip_PPD(:,2), DIM = 1), 1, -1               !
427     mcounts = mcounts + tip_PPD(i,2)                        !
428   END DO                                                    !
429   DO i = MAXLOC(tip_PPD(:,2), DIM = 1), 1, -1               !
430     tip_counts = tip_counts + tip_PPD(i,2)                  !Finds negative 99% confidence
431     IF (tip_counts .ge. 0.99d0*mcounts) THEN                !error in magnitudes
432       tip_m99 = ((REAL(i) - 1.d0)/REAL(10*binspm)) + 18.d0  !
433       tip_m99 = tip_rec - tip_m99                           !
434       exit                                                  !
435     END IF                                                  !
436   END DO                                                    !
437
438   tip_counts = 0.d0 ; pcounts = 0.d0                        !
439   DO i = MAXLOC(tip_PPD(:,2), DIM = 1), 10*(nbins-1)+1      !
440     pcounts = pcounts + tip_PPD(i,2)                        !
441   END DO                                                    !
442   DO i = MAXLOC(tip_PPD(:,2), DIM = 1), 10*(nbins-1)+1      !
443     tip_counts = tip_counts + tip_PPD(i,2)                  !Finds positive 99% confidence
444     IF (tip_counts .ge. 0.99d0*pcounts) THEN                !error in magnitudes
445       tip_p99 = ((REAL(i) - 1.d0)/REAL(10*binspm)) + 18.d0  !
446       tip_p99 = tip_p99 - tip_rec                           !
447       exit                                                  !
448     END IF                                                  !
449   END DO                                                    !
450
451   d1 = 0 ; d2 = 0 ; d3 = 0 ; d4 = 0
452   f_counts = 0.d0 ; a_counts = 0.d0                   !
453   DO i = 1, nbins                                     !
454     f_counts = f_counts + f_PPD(i,2)                  !
455     a_counts = a_counts + a_PPD(i,2)                  !
456     IF (f_counts .ge. 0.159*ndata .and. d1 .eq. 0) THEN!
```

```
457         fminsig = f_PPD(i,1)                                !
458         d1 = 1                                              !
459      END IF                                                 !
460      IF (f_counts .ge. 0.841*ndata .and. d2 .eq. 0) THEN!For f and a:
461         fplusig = f_PPD(i,1)                                !Finds upper and lower
462         d2 = 1                                              !bounds for posterior
463      END IF                                                 !distribution within one
464      IF (a_counts .ge. 0.159*ndata .and. d3 .eq. 0) THEN!sigma of maximum.
465         aminsig = a_PPD(i,1)                                !
466         d3 = 1                                              !
467      END IF                                                 !
468      IF (a_counts .ge. 0.841*ndata .and. d4 .eq. 0) THEN!
469         aplusig = a_PPD(i,1)                                !
470         d4 = 1                                              !
471      END IF                                                 !
472   END DO                                                    !
473
474   f_sigma = 0.5d0*(fplusig - fminsig)          !Hence calculates 1 sigma error
475   a_sigma = 0.5d0*(aplusig - aminsig)          !for f and a
476
477   kpc_merr = tip_kpc*100.d0**(tip_msigma/10.d0) - tip_kpc !minus tip error in kpc
478   kpc_perr = tip_kpc*100.d0**(tip_psigma/10.d0) - tip_kpc !plus tip error in kpc
479
480   WRITE (2,*) "␣"
481   WRITE (2,*) "Distance␣Modulus:", tip_rec + 3.44e0
482   WRITE (2,*) "+sigma␣-sigma:", tip_psigma, tip_msigma
483   WRITE (2,*) "+90␣-90:", tip_p90, tip_m90
484   WRITE (2,*) "+99␣-99:", tip_p99, tip_m99
485   WRITE (2,*) "tip␣+sigma␣-sigma:", tip_rec, tip_rec + tip_psigma, tip_rec - tip_msigma
486   WRITE (2,*) "tip␣+90␣-90:", tip_rec, tip_rec + tip_p90, tip_rec - tip_m90
487   WRITE (2,*) "tip␣+99␣-99:", tip_rec, tip_rec + tip_p99, tip_rec - tip_m99
488
489   END SUBROUTINE Confidence
490
491   !-------------------------------------------------------------------------------------------
492
493   SUBROUTINE Dist_Error       !Samples Distance likelihood space
494   USE Global                  !using samples of m_TRGB, A_lambda and M_TRGB
495   IMPLICIT NONE               !from their respective likelihood distributions
496
497   REAL*8 :: gasdev
498
499   !||Don't forget to reinstate
500   !\/writing distances to files 13 & 14
501
502   string2 = TRIM(ADJUSTL(string)) // '/Sampled_MWy_Distances.dat'
503   OPEN (unit = 13, file = TRIM(ADJUSTL(string2)), status = 'unknown')
504   string2 = TRIM(ADJUSTL(string)) // '/Sampled_M31_Distances.dat'
505   OPEN (unit = 14, file = TRIM(ADJUSTL(string2)), status = 'unknown')
506
507   Dist_PPDx = 0.e0            !Pre-set Distance likelihood
508   Dist_PPDy = 0.e0            !distribution histogram to 0.
509   Dist_PPD_min = (100.e0**((MINVAL(mag_tip, mask = mag_tip .ne. 0.) - 0.3e0*Ext_0 + 3.14)/10.e0))/100.e0
510   Dist_PPD_max = (100.e0**((MAXVAL(mag_tip, mask = mag_tip .ne. 0.) + 0.3e0*Ext_0 + 3.74)/10.e0))/100.e0
511   M31_to_obj_x = 0.e0         !Pre-set M31 to object histogram x values to 0.
512   M31_to_obj_y = 0.e0         !Pre-set M31 to object distance histogram values to 0.
513
514   DistBins = 0                                                !
515   DO i = NINT(Dist_PPD_min) - 1, NINT(Dist_PPD_max) + 1       !Generate 'x' values (MWy distances)
516      DistBins = DistBins + 1                                  !for distribution histogram and count
517      Dist_PPDx(DistBins) = REAL(i)                            !number of bins
```

```
518 END DO                                                                    !
519
520 DO i = -2000, 2000                                                        !Generate 'x' values (M31 distances)
521    M31_to_obj_x(i+2001) = REAL(i)                                         !for distribution histogram
522 END DO                                                                    !
523
524 DO i = 1, nsamples                                                       !
525    M_TRGB = 3.44e0 + 0.05e0*gasdev(idum)                                  !
526    Ext = Ext_0 + 0.1e0*Ext_0*gasdev(idum)                                 !
527    CALL random_number(randnum)                                            !Take 'nsamples' samples of the distance
528    Tip = mag_tip(NINT(randnum*0.9999d0*ndata)+1) + Ext_0                  !using values of m_TRGB, A_lambda and M_TRG
529    Dist_PPD = (100.e0**((Tip - Ext + M_TRGB)/10.e0))/100.e0               !from their respective likelihood distributions.
530    m31_dist = M31_dist_ppd(NINT(randnum*0.9999d0*ndata_M31)+1)            !m31_dist is sampled directly from the M31 dist PPD each iteration
531    M31_to_Obj = ((Dist_PPD ** 2.e0) + (m31_dist ** 2.e0) - &              !
532                  2.e0 * Dist_PPD * m31_dist * cos(theta)) ** 0.5e0 !
533    WRITE (13,*) Dist_PPD
534    WRITE (14,*) M31_to_Obj
535
536    Dist_PPDy(NINT(Dist_PPD) - (NINT(Dist_PPD_min) - 2)) = &                !Tally up number of counts
537    Dist_PPDy(NINT(Dist_PPD) - (NINT(Dist_PPD_min) - 2)) + 1.e0             !in each Earth distance bin
538
539    M31_to_obj_y(2001 + NINT(M31_to_Obj)) = &     !Tally up number of counts
540    M31_to_obj_y(2001 + NINT(M31_to_Obj)) + 1.e0  !in each M31 distance bin
541 END DO
542
543 !-----------One Hundredth Percentiles before prior-----------
544
545 dist_at_perc = 0.e0
546 cum_dist = 0.e0 ; perc = 0.e0 ; next = .true.
547
548 DO i = 1, 4000
549    cum_dist = cum_dist + Dist_PPDy(i)                                     !
550 1    IF (next) THEN                                                      !Note, this routine now accounts for the fact that
551       perc = perc + 1.e0                                                 !a single bin can contain more than 1% of the
552       next = .false.                                                     !data. i.e. - cum_dist does not progress until
553    END IF                                                                !the percentage of the PPD surpasses it. Otherwise
554    IF (cum_dist .ge. (perc/100.e0)*SUM(Dist_PPDy)) THEN                  !cum_dist overtakes it and the second if statement
555       dist_at_perc(NINT(perc),1) = Dist_PPDx(i)                          !is always true.
556       next = .true.                                                      !
557       goto 1
558    END IF
559 END DO
560
561 !-----------Apply distance prior--------------
562 CALL DistancePrior
563
564 Dist_PPDy = Dist_PPDy * DistPrior
565
566 Dist_PPDy = Dist_PPDy/ SUM(Dist_PPDy)
567
568 !-----------One Hundredth Percentiles after prior-----------
569
570 cum_dist = 0.e0 ; perc = 0.e0 ; next = .true.
571
572 DO i = 1, 4000
573    cum_dist = cum_dist + Dist_PPDy(i)                                     !
574 2    IF (next) THEN                                                      !Note, this routine now accounts for the fact that
575       perc = perc + 1.e0                                                 !a single bin can contain more than 1% of the
576       next = .false.                                                     !data. i.e. - cum_dist does not progress until
577    END IF                                                                !the percentage of the PPD surpasses it. Otherwise
578    IF (cum_dist .ge. (perc/100.e0)*SUM(Dist_PPDy)) THEN                  !cum_dist overtakes it and the second if statement
```

```
579         dist_at_perc(NINT(perc),2) = Dist_PPDx(i)              !is always true.
580         next = .true.                                          !
581         goto 2
582       END IF
583   END DO
584
585   !---------Create table of One Hundredth Percentiles---------
586
587   string2 = TRIM(ADJUSTL(string)) // '/Hundredth_Percentiles.dat'
588   OPEN (unit = 14, file = TRIM(ADJUSTL(string2)), status = 'unknown')
589
590   DO i = 1, 100
591      WRITE(14,'(3i7)') i, NINT(dist_at_perc(i,1)), NINT(dist_at_perc(i,2))
592   END DO
593
594   !-----------------------------------------------
595
596   CALL Confidence2                        !Calculate 68.3%, 90% and 99% plus/ minus credibility intervals
597   Dist_PPDy = Dist_PPDy/ SUM(Dist_PPDy)   !normalize distribution
598
599   !-----------------------------------------------
600
601   CALL Confidence3                                !Calculate 68.3%credibility intervals
602   M31_to_obj_y = M31_to_obj_y/ SUM(M31_to_obj_y)  !normalize distribution
603
604   !-------------------------------------------------Plots Distance Distribution w/o credibility intervals
605   string2 = TRIM(ADJUSTL(string)) // '/bw_dist_PPD.ps/CPS'
606   CALL pgbegin(0,TRIM(ADJUSTL(string2)),1,1)
607
608   CALL pgenv(MINVAL(DIST_PPDx, mask = DIST_PPDx .ne. 0.) - 1, MAXVAL(DIST_PPDx) + 1, 0., 1.1*MAXVAL(Dist_PPDy), 0, 0)
609   CALL pgbin (DistBins, Dist_PPDx, Dist_PPDy, .false.)
610   CALL pglab('Proposed Distance (kpc)', 'Probability', '')
611
612   CALL pgend
613
614   WRITE (command2,*) 'convert -rotate 90 ' // TRIM(ADJUSTL(string)) // &
615                      '/bw_dist_PPD.ps ' // TRIM(ADJUSTL(string)) // &
616                      '/bw_dist_PPD.jpg'
617
618   call system(command2)
619
620   !------------------------------Plots Distance Distribution with credibility intervals
621   string2 = TRIM(ADJUSTL(string)) // '/dist_PPD.ps/CPS'
622   CALL pgbegin(0,TRIM(ADJUSTL(string2)),1,1)
623
624   CALL pgenv(MINVAL(DIST_PPDx, mask = DIST_PPDx .ne. 0.) - 1, MAXVAL(DIST_PPDx) + 1, 0., 1.1*MAXVAL(Dist_PPDy), 0, 0)
625
626   DO i = 1, DistBins
627      IF (Dist_PPDx(i) .ge. dist_rec - dist_msigma .and. Dist_PPDx(i) .lt. dist_rec + dist_psigma) THEN
628         CALL pgsci(2)                                    !
629         CALL pgbin (2, Dist_PPDx(i), Dist_PPDy(i),.false.)  !
630         IF (Dist_PPDx(i) .eq. dist_rec - dist_msigma) THEN  !
631            xpts = Dist_PPDx(i)                           !
632            ypts(1) = 0.e0 ; ypts(2) = Dist_PPDy(i)       !One Sigma
633            CALL pgline (2, xpts, ypts)                   !
634         END IF                                           !Credibility
635         IF (Dist_PPDx(i+1) .eq. dist_rec + dist_psigma) THEN !
636            xpts = Dist_PPDx(i+1)                         !Interval
637            ypts(1) = 0.e0 ; ypts(2) = Dist_PPDy(i)       !
638            CALL pgline (2, xpts, ypts)                   !
639         END IF                                           !
```

```
640     ELSE IF (Dist_PPDx(i) .ge. dist_rec - dist_m90 .and. Dist_PPDx(i) .lt. dist_rec + dist_p90) THEN
641        CALL pgsci(3)                                          !
642        CALL pgbin (2, Dist_PPDx(i), Dist_PPDy(i),.false.)     !
643        IF (Dist_PPDx(i) .eq. dist_rec - dist_m90) THEN        !
644           xpts = Dist_PPDx(i)                                 !
645           ypts(1) = 0.e0 ; ypts(2) = Dist_PPDy(i)             !90 percent
646           CALL pgline (2, xpts, ypts)                         !
647        END IF                                                 !Credibility
648        IF (Dist_PPDx(i+1) .eq. dist_rec + dist_p90) THEN      !
649           xpts = Dist_PPDx(i+1)                               !Interval
650           ypts(1) = 0.e0 ; ypts(2) = Dist_PPDy(i)             !
651           CALL pgline (2, xpts, ypts)                         !
652        END IF                                                 !
653     ELSE IF (Dist_PPDx(i) .ge. dist_rec - dist_m99 .and. Dist_PPDx(i) .lt. dist_rec + dist_p99) THEN
654        CALL pgsci(4)                                          !
655        CALL pgbin (2, Dist_PPDx(i), Dist_PPDy(i),.false.)     !
656        IF (Dist_PPDx(i) .eq. dist_rec - dist_m99) THEN        !
657           xpts = Dist_PPDx(i)                                 !
658           ypts(1) = 0.e0 ; ypts(2) = Dist_PPDy(i)             !99 percent
659           CALL pgline (2, xpts, ypts)                         !
660        END IF                                                 !Credibility
661        IF (Dist_PPDx(i+1) .eq. dist_rec + dist_p99) THEN      !
662           xpts = Dist_PPDx(i+1)                               !Interval
663           ypts(1) = 0.e0 ; ypts(2) = Dist_PPDy(i)             !
664           CALL pgline (2, xpts, ypts)                         !
665        END IF                                                 !
666     ELSE
667        CALL pgsci(1)                                          !Distribution
668        CALL pgbin (2, Dist_PPDx(i), Dist_PPDy(i),.false.)     !outside of 99 %
669     END IF                                                    !Cred. Interval
670  END DO
671
672  CALL pgsci(1)
673
674  CALL pglab('Proposed Distance (kpc)', 'Probability', '')
675
676  CALL pgend
677
678  WRITE (command2,*) 'convert -rotate 90 ' // TRIM(ADJUSTL(string)) // &
679                     '/dist_PPD.ps ' // TRIM(ADJUSTL(string)) // &
680                     '/dist_PPD.jpg'
681
682  call system(command2)
683
684  !------------------------Plots M31 to Object Distance Distribution w/o credibility intervals
685  string2 = TRIM(ADJUSTL(string)) // '/bw_M31dist_PPD.ps/CPS'
686  CALL pgbegin(0,TRIM(ADJUSTL(string2)),1,1)
687
688  CALL pgenv(MINVAL(M31_to_obj_x, mask = M31_to_obj_y .ne. 0.) - 1, &
689             MAXVAL(M31_to_obj_x, mask = M31_to_obj_y .ne. 0.) + 1, &
690              0., 1.1*MAXVAL(M31_to_obj_y), 0, 0)
691  CALL pgbin (4001, M31_to_obj_x, M31_to_obj_y, .false.)
692  CALL pglab('Proposed Distance from M31 (kpc)', 'Probability', '')
693
694  CALL pgend
695
696  WRITE (command2,*) 'convert -rotate 90 ' // TRIM(ADJUSTL(string)) // &
697                     '/bw_M31dist_PPD.ps ' // TRIM(ADJUSTL(string)) // &
698                     '/bw_M31dist_PPD.jpg'
699
700  call system(command2)
```

```
701
702 END SUBROUTINE Dist_Error
703
704 !-----------------------------------------------------------------------------------------
705
706 SUBROUTINE DistancePrior  !Multiplies Distance Posterior
707 USE Global                !Distribution by the distance prior -
708 IMPLICIT NONE             !e.g. the density function of the halo
709
710 DistPrior = 0.e0
711 prior_type = 2
712
713 IF (prior_type .eq. 1) Then   !For a Uniform Prior
714 WRITE (2,*) " "
715 WRITE (2,*) "Prior Type: Uniform"
716 DistPrior = 1.e0
717 END IF
718
719 IF (prior_type .eq. 2) Then          !For actual integrated density along line of sight
720 alpha = 1.e0                         !Slope of power law
721 WRITE (2,*) " "
722 WRITE (2,*) "Prior Type: Integrated density function, alpha =", alpha, "theta (deg) =", (theta * 180.e0/ acos(-1.e0))
723
724 DO i = 1, DistBins
725    DistPrior(i) = (Dist_PPDx(i) ** 2.e0) / &
726         (((Dist_PPDx(i) ** 2.e0) + (779.e0 ** 2.e0) - (2.e0 * 779.e0) * Dist_PPDx(i) * cos(theta)) ** (0.5e0 * alpha))
727 END DO
728 END IF
729
730 IF (prior_type .eq. 3) Then   !For a power law prior
731 alpha = 0.2e0                 !Slope of power law
732 WRITE (2,*) " "
733 WRITE (2,*) "Prior Type: Power Law, alpha =", alpha
734
735 DO i = 1, DistBins
736    IF (Dist_PPDx(i) .ne. 779.e0) THEN
737       DistPrior(i) = (ABS(779.e0 - Dist_PPDx(i))) ** (-1.e0 * alpha)
738    END IF
739    IF (Dist_PPDx(i) .eq. 779.e0) THEN
740       DistPrior(i) = 1.e0
741    END IF
742 END DO
743 END IF
744
745 IF (prior_type .eq. 4) Then   !For a linear decreasing prior
746 slope = 2.e0                  !Gradiant of diminishing probability
747 WRITE (2,*) " "
748 WRITE (2,*) "Prior Type: Linear Decreasing, slope =", slope
749
750 DO i = 1, DistBins
751    DistPrior(i) = 779.e0 - abs(slope * (Dist_PPDx(i) - 779.e0))
752 END DO
753 END IF
754
755 IF (prior_type .eq. 5) Then  !For a Gaussian Prior
756 flat = 1.e0                  !Gaussian Flattening Factor
757 hwhm = 150.e0                !Gaussian Half Width Half Maximum
758 WRITE (2,*) " "
759 WRITE (2,*) "Prior Type: Gaussian, flattening =", flat, "; hwhm =", hwhm, "kpc."
760
761 DO i = 1, DistBins
```

```
762     DistPrior(i) = exp(-((Dist_PPDx(i) - 779.e0) ** (2.e0 * flat)) / (2.e0 * hwhm ** (2.e0 * flat)))
763  END DO
764  END IF
765
766  DistPrior = DistPrior/SUM(DistPrior)
767
768  !-----------------------------------------------Plots Distance Prior
769  string2 = TRIM(ADJUSTL(string)) // '/dist_prior.ps/CPS'
770  CALL pgbegin(0,TRIM(ADJUSTL(string2)),1,1)
771
772  CALL pgenv(MINVAL(DIST_PPDx, mask = DIST_PPDx .ne. 0.) - 1, MAXVAL(DIST_PPDx) + 1, 0., 1.1*MAXVAL(DistPrior), 0, 0)
773  CALL pgbin (DistBins, Dist_PPDx, DistPrior, .false.)
774
775  CALL pglab('Proposed Distance (kpc)', 'Probability', '')
776
777  CALL pgend
778
779  WRITE (command2,*) 'convert -rotate 90 ' // TRIM(ADJUSTL(string)) // &
780                     '/dist_prior.ps ' // TRIM(ADJUSTL(string)) // &
781                     '/dist_prior.jpg'
782
783  call system(command2)
784
785  END SUBROUTINE DistancePrior
786
787  !-------------------------------------------------------------------------------------------
788
789  SUBROUTINE Confidence2 !Identifies the best parameter values and
790  USE Global             !their associated 1 sigma errors from the
791  IMPLICIT NONE          !respective posterior plots.
792
793  PPD_peak = 0.d0                          !
794  DO i = 1, DistBins                       !
795     IF (Dist_PPDy(i) .gt. PPD_peak) THEN  !
796        PPD_peak = Dist_PPDy(i)            !Find best fit TRGB value
797        dist_rec = Dist_PPDx(i)            !
798     END IF                                !
799  END DO                                   !
800
801  dist_counts = 0.d0 ; mcounts = 0.d0                          !
802  DO i = MAXLOC(Dist_PPDy, DIM = 1), 1, -1                     !
803     mcounts = mcounts + Dist_PPDy(i)                          !
804  END DO                                                       !
805  DO i = MAXLOC(Dist_PPDy, DIM = 1), 1, -1                     !
806     dist_counts = dist_counts + Dist_PPDy(i)                  !Finds negative one sigma
807     IF (dist_counts .ge. 0.682*mcounts) THEN                  !error in distance
808        dist_msigma = dist_rec - Dist_PPDx(i)                  !
809        exit                                                   !
810     END IF                                                    !
811  END DO                                                       !
812
813  dist_counts = 0.d0 ; pcounts = 0.d0                          !
814  DO i = MAXLOC(Dist_PPDy, DIM = 1), DistBins                  !
815     pcounts = pcounts + Dist_PPDy(i)                          !
816  END DO                                                       !
817  DO i = MAXLOC(Dist_PPDy, DIM = 1), DistBins                  !
818     dist_counts = dist_counts + Dist_PPDy(i)                  !Finds positive one sigma
819     IF (dist_counts .ge. 0.682*pcounts) THEN                  !error in distance
820        dist_psigma = Dist_PPDx(i) - dist_rec                  !
821        exit                                                   !
822     END IF                                                    !
```

```
823  END DO                                                    !
824
825  dist_counts = 0.d0 ; mcounts = 0.d0                       !
826  DO i = MAXLOC(Dist_PPDy, DIM = 1), 1, -1                  !
827     mcounts = mcounts + Dist_PPDy(i)                       !
828  END DO                                                    !
829  DO i = MAXLOC(Dist_PPDy, DIM = 1), 1, -1                  !
830     dist_counts = dist_counts + Dist_PPDy(i)               !Finds negative 90 credibility
831     IF (dist_counts .ge. 0.9*mcounts) THEN                 !interval in distance
832        dist_m90 = dist_rec - Dist_PPDx(i)                  !
833        exit                                                !
834     END IF                                                 !
835  END DO                                                    !
836
837  dist_counts = 0.d0 ; pcounts = 0.d0                       !
838  DO i = MAXLOC(Dist_PPDy, DIM = 1), DistBins               !
839     pcounts = pcounts + Dist_PPDy(i)                       !
840  END DO                                                    !
841  DO i = MAXLOC(Dist_PPDy, DIM = 1), DistBins               !
842     dist_counts = dist_counts + Dist_PPDy(i)               !Finds positive 90 credibility
843     IF (dist_counts .ge. 0.9*pcounts) THEN                 !interval in distance
844        dist_p90 = Dist_PPDx(i) - dist_rec                  !
845        exit                                                !
846     END IF                                                 !
847  END DO
848
849  dist_counts = 0.d0 ; mcounts = 0.d0                       !
850  DO i = MAXLOC(Dist_PPDy, DIM = 1), 1, -1                  !
851     mcounts = mcounts + Dist_PPDy(i)                       !
852  END DO                                                    !
853  DO i = MAXLOC(Dist_PPDy, DIM = 1), 1, -1                  !
854     dist_counts = dist_counts + Dist_PPDy(i)               !Finds negative 99 credibility
855     IF (dist_counts .ge. 0.99*mcounts) THEN                !interval in distance
856        dist_m99 = dist_rec - Dist_PPDx(i)                  !
857        exit                                                !
858     END IF                                                 !
859  END DO                                                    !
860
861  dist_counts = 0.d0 ; pcounts = 0.d0                       !
862  DO i = MAXLOC(Dist_PPDy, DIM = 1), DistBins               !
863     pcounts = pcounts + Dist_PPDy(i)                       !
864  END DO                                                    !
865  DO i = MAXLOC(Dist_PPDy, DIM = 1), DistBins               !
866     dist_counts = dist_counts + Dist_PPDy(i)               !Finds positive 99 credibility
867     IF (dist_counts .ge. 0.99*pcounts) THEN                !interval in distance
868        dist_p99 = Dist_PPDx(i) - dist_rec                  !
869        exit                                                !
870     END IF                                                 !
871  END DO                                                    !
872
873  WRITE (2,*) "␣"
874  WRITE (2,*) "Most␣Likely␣Distance:", dist_rec
875  WRITE (2,*) "+sigma␣-sigma␣dist+sigma␣dist-sigma:", dist_psigma, dist_msigma, dist_rec + dist_psigma, dist_rec - dist_msigma
876  WRITE (2,*) "+90␣-90␣dist+90␣dist-90:", dist_p90, dist_m90, dist_rec + dist_p90, dist_rec - dist_m90
877  WRITE (2,*) "+99␣-99␣dist+99␣dist-99:", dist_p99, dist_m99, dist_rec + dist_p99, dist_rec - dist_m99
878
879  !|| Convert distance profile mode and intervals
880  !\/ back into the equivalent in magnitudes
881  dist2mag_rec = 5.e0 * LOG10(dist_rec * 100.e0) - 3.44e0
882  d2m_psigma = (5.e0 * LOG10((dist_rec+dist_psigma) * 100.e0) - 3.44e0) - (5.e0 * LOG10(dist_rec * 100.e0) - 3.44e0)
883  d2m_msigma = (5.e0 * LOG10(dist_rec * 100.e0) - 3.44e0) - (5.e0 * LOG10((dist_rec-dist_msigma) * 100.e0) - 3.44e0)
```

```
884  d2m_p90 = (5.e0 * LOG10((dist_rec+dist_p90) * 100.e0) - 3.44e0) - (5.e0 * LOG10(dist_rec * 100.e0) - 3.44e0)
885  d2m_m90 = (5.e0 * LOG10(dist_rec * 100.e0) - 3.44e0) - (5.e0 * LOG10((dist_rec-dist_m90) * 100.e0) - 3.44e0)
886  d2m_p99 = (5.e0 * LOG10((dist_rec+dist_p99) * 100.e0) - 3.44e0) - (5.e0 * LOG10(dist_rec * 100.e0) - 3.44e0)
887  d2m_m99 = (5.e0 * LOG10(dist_rec * 100.e0) - 3.44e0) - (5.e0 * LOG10((dist_rec-dist_m99) * 100.e0) - 3.44e0)
888  !/\
889  !||
890
891  WRITE (2,*) "␣"
892  WRITE (2,*) "Distance␣back␣to␣magnitude:", dist2mag_rec
893  WRITE (2,*) "Hence,␣distance␣modulus␣after␣applying␣prior:", dist2mag_rec + 3.44e0
894  WRITE (2,*) "+sigma␣-sigma:", d2m_psigma, d2m_msigma
895  WRITE (2,*) "+90␣␣␣␣-90␣␣␣:", d2m_p90, d2m_m90
896  WRITE (2,*) "+99␣␣␣␣-99␣␣␣:", d2m_p99, d2m_m99
897
898  END SUBROUTINE Confidence2
899
900  !-----------------------------------------------------------------------------------------------------------------------
901
902  SUBROUTINE Confidence3
903  USE Global
904  IMPLICIT NONE
905
906  PPD_peak = 0.d0                              !
907  DO i = 1, 3001                               !
908     IF (M31_to_obj_y(i) .gt. PPD_peak) THEN   !
909        PPD_peak = M31_to_obj_y(i)             !Find best fit TRGB value
910        M31_dist_rec = M31_to_obj_x(i)         !
911     END IF                                    !
912  END DO                                       !
913
914  dist_counts = 0.d0 ; mcounts = 0.d0                          !
915  DO i = MAXLOC(M31_to_obj_y, DIM = 1), 1, -1                  !
916     mcounts = mcounts + M31_to_obj_y(i)                       !
917  END DO                                                       !
918  DO i = MAXLOC(M31_to_obj_y, DIM = 1), 1, -1                  !
919     dist_counts = dist_counts + M31_to_obj_y(i)               !Finds negative one sigma
920     IF (dist_counts .ge. 0.682*mcounts) THEN                  !error in M31 to object distance
921        M31_dist_msigma = M31_dist_rec - M31_to_obj_x(i)       !
922        exit                                                   !
923     END IF                                                    !
924  END DO                                                       !
925
926  dist_counts = 0.d0 ; pcounts = 0.d0                          !
927  DO i = MAXLOC(M31_to_obj_y, DIM = 1), 3001                   !
928     pcounts = pcounts + M31_to_obj_y(i)                       !
929  END DO                                                       !
930  DO i = MAXLOC(M31_to_obj_y, DIM = 1), 3001                   !
931     dist_counts = dist_counts + M31_to_obj_y(i)               !Finds positive one sigma
932     IF (dist_counts .ge. 0.682*pcounts) THEN                  !error in M31 to object distance
933        M31_dist_psigma = M31_to_obj_x(i) - M31_dist_rec       !
934        exit                                                   !
935     END IF                                                    !
936  END DO                                                       !
937
938  WRITE (2,*) "␣"
939  WRITE (2,*) "Most␣Likely␣Distance␣from␣M31:", M31_dist_rec
940  WRITE (2,*) "+sigma␣-sigma␣dist+sigma␣dist-sigma:", M31_dist_psigma, M31_dist_msigma, &
941                                                      M31_dist_rec + M31_dist_psigma, M31_dist_rec - M31_dist_msigma
942  END SUBROUTINE Confidence3
```

Program: SatPlot.f95

Creation Date: 17 February 2012 (first version 22 Sep 2011)

Relevant Section: Fig. 10 of Paper II (Ch. 4)

Notes: I wrote this program with the sole objective of producing a three dimensional plot of the satellite system. The distances and positions on the sky of each satellite are read in and used to generate a set of M31-centric cartesian coordinates for each satellite. Rotation matrices are then used to spin the view angle. The use of rotation matrices here is actually not quite correct as the order of application is not given its due importance. This means that the operation is a little clumsy but the plots themselves are unaffected. The actual PAndAS survey area footprint visible in Fig. 10 (c) of Paper II was generated (painstakingly!) for use in 'SatDensity_SampCont.f95,' but I added it to this figure for illustration.

```
1  MODULE Global !Define all
2  IMPLICIT NONE !Variables
3
4  INTEGER :: i, j, k, ios, ndata, SAP_ndata, ICP_ndata, v_angle     !SAP = Survey Area Point
5  PARAMETER (ndata = 28)
6  PARAMETER (SAP_ndata = 135)
7  REAL :: x(ndata), y(ndata), z(ndata), MWy_to_Obj(ndata), M31_to_Obj(ndata), m31_dist, m31_psig, m31_msig, x_pro(ndata), y_pro(ndata)
8  REAL :: SAP_xi(SAP_ndata), SAP_eta(SAP_ndata), SAP_x(SAP_ndata), SAP_y(SAP_ndata), SAP_theta, MWy_to_SAP(SAP_ndata)
9  REAL :: SAPn_x(SAP_ndata), SAPn_y(SAP_ndata), SAPf_x(SAP_ndata), SAPf_y(SAP_ndata), MWy_to_SAPn(SAP_ndata), MWy_to_SAPf(SAP_ndata)
10 REAL :: ICP_xi(300), ICP_eta(300), ICP_x(300), ICP_y(300), ICP_theta, MWy_to_ICP(300)
11 PARAMETER (m31_dist = 779.e0)
12 PARAMETER (m31_psig = 0.e0)
13 PARAMETER (m31_msig = 0.e0)
14 REAL :: xi(ndata), eta(ndata), theta(ndata), p_sig(ndata), m_sig(ndata), pi, dummy
15 PARAMETER (pi = ACOS(-1.e0))
16 REAL :: M31_to_Obj_psig(ndata), M31_to_Obj_msig(ndata), temp1, temp2
17
18 CHARACTER :: name(ndata)*20, string*200
19
20
21 !-----For Rotate subroutine-----
22
23 REAL :: obj_rot(ndata,3), obj_pro_rot(ndata,3), obj(ndata,5), obj_pro(ndata,3)
24 REAL :: obj_rotp(ndata,3), obj_rotm(ndata,3)
25 REAL :: SAP_rot(SAP_ndata,3), SAP(SAP_ndata,3), SAPn_rot(SAP_ndata,3), SAPn(SAP_ndata,3), SAPf_rot(SAP_ndata,3), SAPf(SAP_ndata,3)
26 REAL :: ICP_rot(300,3), ICP(300,3)
27 REAL :: x_rot(3,3), y_rot(3,3), z_rot(3,3), rot_mat(3,3), x_axis(3), y_axis(3), z_axis(3)
28 REAL :: marker_x(3), marker_y(3), marker_z(3)
29 REAL :: alpha, beta, gamma
30
31 END MODULE Global
32
33 !-----------------------------------------------------------------------------------------------------------------
34
35 PROGRAM SatPlot     !Master program
36 USE Global
37 IMPLICIT NONE
38
39 CALL GetData
```

```
40
41  string = 'M31_neighborhood_xy.ps/CPS' !
42  alpha = 0.e0 * (pi/180.e0)            !
43  beta = 0.e0 * (pi/180.e0)             !Plots satellite
44  gamma = 0.e0 * (pi/180.e0)            !positions on
45  v_angle = 1                           !xy plane
46  CALL Rotate                           !
47  CALL Plot                             !
48
49  string = 'M31_neighborhood_xz.ps/CPS' !
50  alpha = 90.e0 * (pi/180.e0)           !
51  beta = 0.e0 * (pi/180.e0)             !Plots satellite
52  gamma = 0.e0 * (pi/180.e0)            !positions on
53  v_angle = 2                           !xz plane
54  CALL Rotate                           !
55  !CALL Plot                            !
56
57  string = 'M31_neighborhood_yz.ps/CPS' !
58  alpha = 0.e0 * (pi/180.e0)            !
59  beta = 270.e0 * (pi/180.e0)           !Plots satellite
60  gamma = 0.e0 * (pi/180.e0)            !positions on
61  v_angle = 3                           !yz plane
62  CALL Rotate                           !
63  CALL Plot                             !
64
65  string = 'M31_neighborhood_xyz.ps/CPS'!
66  alpha = 5.e0 * (pi/180.e0)            !
67  beta = 5.e0 * (pi/180.e0)             !Plots satellite positions
68  gamma = 0.e0 * (pi/180.e0)            !in 3D.
69  v_angle = 4                           !
70  CALL DistancePerspective  !Remove Effects of distance on x/y positions of satellites
71  CALL Rotate
72  CALL Plot
73
74  END PROGRAM SatPlot
75
76  !-------------------------------------------------------------------------------------------------------------
77
78  SUBROUTINE GetData     !Get data :)
79  USE Global
80  IMPLICIT NONE
81
82  OPEN (unit = 1, file = './SatStats.dat', status = 'old')
83
84  i = 0 ; ios = 0
85  DO WHILE (.TRUE.)  !Reads data until end of input file and puts it into arrays
86     i = i+1
87     READ (1, *, IOSTAT = ios) xi(i), eta(i), theta(i), Mwy_to_Obj(i), p_sig(i), m_sig(i), name(i)
88
89     if (ios == 0) then ;
90     else if (ios == -1) then ;
91        i=i-1
92        exit ;
93     else if (ios > 0) then ;
94        i=i-1
95        cycle
96     end if
97
98  END DO
99
100 DO i = 1, ndata-1
```

```
101     xi(i) = xi(i) * (pi/180.e0)              !
102     eta(i) = eta(i) * (pi/180.e0)            !Convert angles from degrees to radians
103     theta(i) = theta(i) * (pi/180.e0)        !
104
105     x(i) = ABS(MWy_to_Obj(i) * cos(theta(i)) * tan(xi(i)))  !Determine length of x vector for each satellite
106     IF (xi(i) .lt. 0.e0) THEN                               !
107        x(i) = -1.e0 * x(i)                                  !Determine if x is positive or negative
108     END IF                                                  !
109
110     y(i) = ABS(MWy_to_Obj(i) * sin(eta(i)))                 !Determine length of y vector for each satellite
111     IF (eta(i) .lt. 0.e0) THEN                              !
112        y(i) = -1.e0 * y(i)                                  !Determine if y is positive or negative
113     END IF                                                  !
114
115     z(i) = MWy_to_Obj(i) * cos(theta(i)) - m31_dist         !Determine length and sign of z vector
116
117
118     M31_to_Obj(i) = (((x(i)) ** 2.e0) + ((y(i)) ** 2.e0) + ((z(i)) ** 2.e0)) ** 0.5e0   !Determine distance between M31 and satellite
119
120     temp1 = ((2.e0 * MWy_to_obj(i)) - 2.e0 * m31_dist * cos(theta(i))) ** 2.e0 / &
121             ( (MWy_to_Obj(i) ** 2.e0) + (m31_dist ** 2.e0) - (2.e0 * MWy_to_Obj(i) * m31_dist * cos(theta(i))) )
122     temp2 = ((2.e0 * m31_dist) - 2.e0 * MWy_to_obj(i) * cos(theta(i))) ** 2.e0 / &
123             ( (MWy_to_Obj(i) ** 2.e0) + (m31_dist ** 2.e0) - (2.e0 * MWy_to_Obj(i) * m31_dist * cos(theta(i))) )
124
125     M31_to_Obj_psig(i) = SQRT((temp1) * (p_sig(i) ** 2.e0) + (temp2) * (12.e0 ** 2.e0))
126     M31_to_Obj_msig(i) = SQRT((temp1) * (m_sig(i) ** 2.e0) + (temp2) * (11.e0 ** 2.e0))
127
128     WRITE (*,*) name(i)                          !Write x,y and z components of M31-to-satellite separation vector along with vector
129     WRITE (*, '(6F16.5)') x(i), y(i), z(i), M31_to_Obj(i), M31_to_Obj_psig(i), M31_to_Obj_msig(i)  !magnitude and uncertainties
130
131  END DO
132
133  !-----------------------For Survey Border----------------------------
134
135  OPEN (unit = 2, file = '../SurveyArea/Border_Coords_XiEta.dat', status = 'old')
136
137  i = 0 ; ios = 0
138  DO WHILE (.TRUE.)  !Reads data until end of input file and puts it into arrays
139     i = i+1
140     READ (2, *, IOSTAT = ios) SAP_xi(i), SAP_eta(i)              !SAP = Survey Area Point
141
142     if (ios == 0) then ;
143     else if (ios == -1) then ;
144        i=i-1
145        exit ;
146     else if (ios > 0) then ;
147        i=i-1
148        cycle
149     end if
150
151  END DO
152
153  DO i = 1, SAP_ndata
154     SAP_xi(i) = SAP_xi(i) * (pi/180.e0)                          !
155     SAP_eta(i) = SAP_eta(i) * (pi/180.e0)                        !Convert angles from degrees to radians
156     SAP_theta = SQRT(SAP_xi(i)**2.e0 + SAP_eta(i)**2.e0) * (pi/180.e0) !
157     MWy_to_SAP(i) = m31_dist / cos(SAP_theta)      !Determine Survey Area Point Distance assuming plane at distance of M31
158
159     SAP_x(i) = ABS(MWy_to_SAP(i) * cos(SAP_theta) * tan(SAP_xi(i)))  !Determine length of x vector for each Survey
160     IF (SAP_xi(i) .lt. 0.e0) THEN                                    !Area Point, assuming plane at distance of M31
161        SAP_x(i) = -1.e0 * SAP_x(i)                                   !Determine if x is positive or negative
```

```
162     END IF                                                                  !
163
164     SAP_y(i) = ABS(MWy_to_SAP(i) * sin(SAP_eta(i)))                         !Determine length of y vector for each Survey
165     IF (SAP_eta(i) .lt. 0.e0) THEN                                          !Area Point, assuming plane at distance of M31
166        SAP_y(i) = -1.e0 * SAP_y(i)                                          !Determine if y is positive or negative
167     END IF                                                                  !
168
169     MWy_to_SAPn(i) = (m31_dist + z(12)) / cos(SAP_theta)  !Determine Survey Area Point Distance assuming plane at distance of ANDXVI
170
171     SAPn_x(i) = ABS(MWy_to_SAPn(i) * cos(SAP_theta) * tan(SAP_xi(i)))  !Determine length of x vector for each Survey
172     IF (SAP_xi(i) .lt. 0.e0) THEN                                       !Area Point, assuming plane at distance of ANDXVI
173        SAPn_x(i) = -1.e0 * SAPn_x(i)                                    !Determine if x is positive or negative
174     END IF                                                              !
175
176     SAPn_y(i) = ABS(MWy_to_SAPn(i) * sin(SAP_eta(i)))                   !Determine length of y vector for each Survey
177     IF (SAP_eta(i) .lt. 0.e0) THEN                                      !Area Point, assuming plane at distance of ANDXVI
178        SAPn_y(i) = -1.e0 * SAPn_y(i)                                    !Determine if y is positive or negative
179     END IF                                                              !
180
181     MWy_to_SAPf(i) = (m31_dist + z(23)) / cos(SAP_theta)  !Determine Survey Area Point Distance assuming plane at distance of ANDXVIII
182
183     SAPf_x(i) = ABS(MWy_to_SAPf(i) * cos(SAP_theta) * tan(SAP_xi(i)))  !Determine length of x vector for each Survey
184     IF (SAP_xi(i) .lt. 0.e0) THEN                                       !Area Point, assuming plane at distance of ANDXVIII
185        SAPf_x(i) = -1.e0 * SAPf_x(i)                                    !Determine if x is positive or negative
186     END IF                                                              !
187
188     SAPf_y(i) = ABS(MWy_to_SAPf(i) * sin(SAP_eta(i)))                   !Determine length of y vector for each Survey
189     IF (SAP_eta(i) .lt. 0.e0) THEN                                      !Area Point, assuming plane at distance of ANDXVIII
190        SAPf_y(i) = -1.e0 * SAPf_y(i)                                    !Determine if y is positive or negative
191     END IF                                                              !
192
193  END DO
194
195  !----------------------For Inner CutOff Ellipse----------------------------
196
197  OPEN (unit = 3, file = '../SurveyArea/M31CutOffEllipse.dat', status = 'old')
198
199  i = 0 ; ios = 0
200  DO WHILE (.TRUE.)  !Reads data until end of input file and puts it into arrays
201     i = i+1
202     READ (3, *, IOSTAT = ios) ICP_xi(i), ICP_eta(i)              !ICP = Inner Cut-Off Point
203
204     if (ios == 0) then ;
205     else if (ios == -1) then ;
206        i=i-1
207        exit ;
208     else if (ios > 0) then ;
209        i=i-1
210        cycle
211     end if
212
213  END DO
214
215  ICP_ndata = i
216
217  DO i = 1, ICP_ndata
218     ICP_xi(i) = ICP_xi(i) * (pi/180.e0)                                  !
219     ICP_eta(i) = ICP_eta(i) * (pi/180.e0)                                !Convert angles from degrees to radians
220     ICP_theta = SQRT(ICP_xi(i)**2.e0 + ICP_eta(i)**2.e0) * (pi/180.e0) !
221     MWy_to_ICP(i) = m31_dist / cos(ICP_theta)      !Determine Inner Cut-Off Point Distance assuming plane at distance of M31
222
```

```
223     ICP_x(i) = ABS(MWy_to_ICP(i) * cos(ICP_theta) * tan(ICP_xi(i)))  !Determine length of x vector for each Inner
224     IF (ICP_xi(i) .lt. 0.e0) THEN                                    !Cut-Off Point, assuming plane at distance of M31
225        ICP_x(i) = -1.e0 * ICP_x(i)                                   !Determine if x is positive or negative
226     END IF                                                           !
227
228     ICP_y(i) = ABS(MWy_to_ICP(i) * sin(ICP_eta(i)))                  !Determine length of y vector for each Inner
229     IF (ICP_eta(i) .lt. 0.e0) THEN                                   !Cut-Off Point, assuming plane at distance of M31
230        ICP_y(i) = -1.e0 * ICP_y(i)                                   !Determine if y is positive or negative
231     END IF                                                           !
232  END DO
233
234  CLOSE(1)
235  CLOSE(2)
236  CLOSE(3)
237
238  END SUBROUTINE GetData
239
240  !-------------------------------------------------------------------------------------------------------------------
241
242  SUBROUTINE DistancePerspective !This subroutine is for removing the effects of perspective on the 3D view.
243  USE Global                     !It scales x and y positions of satellites relative to the closest satellite
244  IMPLICIT NONE                  !(And XVI) to preserve the on sky view of the satellites.
245
246  DO i = 1, ndata
247     x(i) = x(i) * ((m31_dist + z(12))/ (m31_dist + z(i)))
248     y(i) = y(i) * ((m31_dist + z(12))/ (m31_dist + z(i)))
249  END DO
250
251  DO i = 1, SAP_ndata
252     SAP_x(i) = SAP_x(i) * ((m31_dist + z(12))/ m31_dist)   !PAndAS survey border at distance of M31
253     SAP_y(i) = SAP_y(i) * ((m31_dist + z(12))/ m31_dist)   !
254     SAPn_x(i) = SAPn_x(i) * ((m31_dist + z(12))/ (m31_dist + z(12)))  !PAndAS survey border at distance of And XVI
255     SAPn_y(i) = SAPn_y(i) * ((m31_dist + z(12))/ (m31_dist + z(12)))  !(And XVI is the nearest satellite)
256     SAPf_x(i) = SAPf_x(i) * ((m31_dist + z(12))/ (m31_dist + z(23))) !PAndAS survey border at distance of And XXVII
257     SAPf_y(i) = SAPf_y(i) * ((m31_dist + z(12))/ (m31_dist + z(23))) !(And XXVII is the furthest satellite)
258  END DO
259
260  DO i = 1, ICP_ndata
261     ICP_x(i) = ICP_x(i) * ((m31_dist + z(12))/ m31_dist) !Inner cutoff ellipse at
262     ICP_y(i) = ICP_y(i) * ((m31_dist + z(12))/ m31_dist) !distance to M31.
263  END DO
264
265  END SUBROUTINE DistancePerspective
266
267  !-------------------------------------------------------------------------------------------------------------------
268
269  SUBROUTINE Rotate     !Uses rotation matrices to shift
270  USE Global            !the position on screen of the
271  IMPLICIT NONE         !satellites based on the viewing angle
272
273  x_axis(1) = MAXVAL(abs(x)) ; x_axis(2) = 0.e0 ; x_axis(3) = 0.e0  !Generate coordinates of
274  y_axis(1) = 0.e0 ; y_axis(2) = MAXVAL(abs(y)) ; y_axis(3) = 0.e0  !the positive ends of the
275  z_axis(1) = 0.e0 ; z_axis(2) = 0.e0 ; z_axis(3) = MAXVAL(abs(z))  !x, y and z axes
276
277  marker_x(1) = 100.e0 ; marker_x(2) = 0.e0 ; marker_x(3) = 0.e0    !Generate coordinates of
278  marker_y(1) = 0.e0 ; marker_y(2) = 100.e0 ; marker_y(3) = 0.e0    !the positive 100 kpc
279  marker_z(1) = 0.e0 ; marker_z(2) = 0.e0 ; marker_z(3) = 100.e0    !axis markers
280
281  x_rot(1,1) = 1.e0                  !
282  x_rot(2,1) = 0.e0                  !
283  x_rot(3,1) = 0.e0                  !
```

```
284   x_rot(1,2) = 0.e0                  !
285   x_rot(2,2) = cos(alpha)            !Rotation matrix for adjusting yaw - angle alpha
286   x_rot(3,2) = -1.e0 * sin(alpha)    !
287   x_rot(1,3) = 0.e0                  !
288   x_rot(2,3) = sin(alpha)            !
289   x_rot(3,3) = cos(alpha)            !
290
291   y_rot(1,1) = cos(beta)             !
292   y_rot(2,1) = 0.e0                  !
293   y_rot(3,1) = sin(beta)             !
294   y_rot(1,2) = 0.e0                  !
295   y_rot(2,2) = 1.e0                  !Rotation matrix for adjusting pitch - angle beta
296   y_rot(3,2) = 0.e0                  !
297   y_rot(1,3) = -1.e0 * sin(beta)     !
298   y_rot(2,3) = 0.e0                  !
299   y_rot(3,3) = cos(beta)             !
300
301   z_rot(1,1) = cos(gamma)            !
302   z_rot(2,1) = -1.e0 * sin(gamma)    !
303   z_rot(3,1) = 0.e0                  !
304   z_rot(1,2) = sin(gamma)            !
305   z_rot(2,2) = cos(gamma)            !Rotation matrix for adjusting roll - angle gamma
306   z_rot(3,2) = 0.e0                  !
307   z_rot(1,3) = 0.e0                  !
308   z_rot(2,3) = 0.e0                  !
309   z_rot(3,3) = 1.e0                  !
310
311   rot_mat = MATMUL(x_rot, y_rot)     !Generate rotation matrix to adjust object coordinates
312   rot_mat = MATMUL(rot_mat, z_rot)   !for the chosen combination of yaw, pitch and roll
313
314   x_axis = MATMUL(rot_mat, x_axis)   !
315   y_axis = MATMUL(rot_mat, y_axis)   !Convert coordinates of positive ends of axes for new view angle
316   z_axis = MATMUL(rot_mat, z_axis)   !
317
318   marker_x = MATMUL(rot_mat, marker_x)
319   marker_y = MATMUL(rot_mat, marker_y)
320   marker_z = MATMUL(rot_mat, marker_z)
321
322   DO i = 1, ndata
323      obj(i,1) = x(i)                                  !
324      obj(i,2) = y(i)                                  !Convert object coordinates
325      obj(i,3) = z(i)                                  !for new viewing angle
326      obj(i,4) = z(i) + p_sig(i)                         !For error
327      obj(i,5) = z(i) - m_sig(i)                         !bars
328      obj_rot(i,:) = MATMUL(rot_mat, obj(i,(/ 1,2,3 /)))  !
329      obj_rotp(i,:) = MATMUL(rot_mat, obj(i,(/ 1,2,4 /))) !
330      obj_rotm(i,:) = MATMUL(rot_mat, obj(i,(/ 1,2,5 /))) !
331
332      obj_pro(i,1) = x(i)                               !
333      obj_pro(i,2) = y(i)                               !Convert z = 0 projection of object coordinates
334      obj_pro(i,3) = 0.e0                               !for new viewing angle to form other end of plotted z vector
335      obj_pro_rot(i,:) = MATMUL(rot_mat, obj_pro(i,:))  !
336   END DO
337
338   DO i = 1, SAP_ndata
339      SAP(i,1) = SAP_x(i)                                 !
340      SAP(i,2) = SAP_y(i)                                 !Convert SAP coordinates
341      SAP(i,3) = 0.e0                                     !for new viewing angle
342      SAP_rot(i,:) = MATMUL(rot_mat, SAP(i,:))            !
343
344      SAPn(i,1) = SAPn_x(i)                                !
```

```
345     SAPn(i,2) = SAPn_y(i)                                   !Convert SAPn coordinates
346     SAPn(i,3) = z(12)                                       !for new viewing angle
347     SAPn_rot(i,:) = MATMUL(rot_mat, SAPn(i,:))              !
348
349     SAPf(i,1) = SAPf_x(i)                                   !
350     SAPf(i,2) = SAPf_y(i)                                   !Convert SAPf coordinates
351     SAPf(i,3) = z(23)                                       !for new viewing angle
352     SAPf_rot(i,:) = MATMUL(rot_mat, SAPf(i,:))              !
353 END DO
354
355 DO i = 1, ICP_ndata
356     ICP(i,1) = ICP_x(i)                                !
357     ICP(i,2) = ICP_y(i)                                !Convert ICP coordinates
358     ICP(i,3) = 0.e0                                    !for new viewing angle
359     ICP_rot(i,:) = MATMUL(rot_mat, ICP(i,:))           !
360 END DO
361
362 END SUBROUTINE Rotate
363
364 !-----------------------------------------------------------------------------------------------------------------------
365
366 SUBROUTINE Plot     !Plot the satellites for chosen view angle
367 USE Global
368 IMPLICIT NONE
369
370 CALL pgbegin(0,TRIM(ADJUSTL(string)),1,1)
371
372 IF (v_angle .eq. 1 .or. v_angle .eq. 2) THEN                                    !
373 CALL pgenv(1.1*(MAXVAL(ABS(obj_rot(:,1)))), -1.1*(MAXVAL(ABS(obj_rot(:,1)))), &       !Set frame limits
374             -1.1*(MAXVAL(ABS(obj_rot(:,2)))), 1.1*(MAXVAL(ABS(obj_rot(:,2)))), 1, 0)   !
375 ELSE IF (v_angle .eq. 3) THEN                                                   !
376 CALL pgenv(1.1*(MINVAL(obj_rot(:,1))), 1.1*(MAXVAL(ABS(obj_rot(:,1)))), &       !and parameters for
377             1.1*(MINVAL(SAPf_rot(:,2))), 1.1*(MAXVAL(SAPf_rot(:,2))), 1, 0)       !
378 ELSE                                                                            !
379 CALL pgenv(1.1*(MAXVAL(ABS(SAP_rot(:,1)))), -1.1*(MAXVAL(ABS(SAP_rot(:,1)))), &       !various viewing angles
380             -1.1*(MAXVAL(ABS(SAP_rot(:,2)))), 1.1*(MAXVAL(ABS(SAP_rot(:,2)))), 1, -1)  !
381 END IF                                                                          !
382
383 CALL pgline(2, (/x_axis(1), -1.e0 * x_axis(1)/), (/x_axis(2), -1.e0 * x_axis(2)/))      !Draw lines from positive
384 CALL pgline(2, (/y_axis(1), -1.e0 * y_axis(1)/), (/y_axis(2), -1.e0 * y_axis(2)/))      !end to negative end of
385 CALL pgline(2, (/z_axis(1), -1.e0 * z_axis(1)/), (/z_axis(2), -1.e0 * z_axis(2)/))      !x, y and z axes
386
387 IF (v_angle .eq. 1) THEN                                !
388 CALL pgptxt(x_axis(1) - 5., x_axis(2) - 10., 0., 0.5, 'x')   !
389 CALL pgptxt(y_axis(1) - 10., y_axis(2) - 5., 0., 0.5, 'y')   !
390 END IF                                                  !
391 IF (v_angle .eq. 3) THEN                                !Print x, y and z
392 CALL pgptxt(y_axis(1) - 10., y_axis(2) - 10., 0., 0.5, 'y')  !
393 CALL pgptxt(z_axis(1) - 10., z_axis(2) - 15., 0., 0.5, 'z')  !at the positive ends of
394 END IF                                                  !
395 IF (v_angle .eq. 4) THEN                                !their respective axes
396 CALL pgptxt(x_axis(1) - 5., x_axis(2) - 5., 0., 0.5, 'x')    !
397 CALL pgptxt(y_axis(1) - 5., y_axis(2) - 5., 0., 0.5, 'y')    !
398 CALL pgptxt(z_axis(1) - 5., z_axis(2) - 5., 0., 0.5, 'z')    !
399 END IF                                                  !
400
401 IF (v_angle .ne. 4) THEN
402 CALL pgpt(1,marker_x(1), marker_x(2), 0612)                !
403 CALL pgpt(1,-1.e0 * marker_x(1), -1.e0 * marker_x(2), 0612)   !
404 CALL pgpt(1,marker_y(1), marker_y(2), 0590)                !
405 CALL pgpt(1,-1.e0 * marker_y(1), -1.e0 * marker_y(2), 0590)   !
```

```
406 END IF                                                                    !
407 IF (v_angle .eq. 2) THEN                                                  !Plot
408 CALL pgpt(1,marker_z(1), marker_z(2), 0590)                               !
409 CALL pgpt(1,-1.e0 * marker_z(1), -1.e0 * marker_z(2), 0590)               !Markers
410 ELSE                                                                      !
411 CALL pgpt(1,marker_z(1), marker_z(2), 0612)                               !
412 CALL pgpt(1,-1.e0 * marker_z(1), -1.e0 * marker_z(2), 0612)               !
413 END IF                                                                    !
414
415 CALL PGSCH(0.75)
416
417 DO i = 1, 24            !For satellites with prefix 'And'
418    CALL pgslw(2)
419    CALL pgsci(2)
420    CALL pgline(2, (/obj_pro_rot(i,1), obj_rot(i,1)/), (/obj_pro_rot(i,2), obj_rot(i,2)/)) !Draw z vector of object for
421    CALL pgslw(15)                                                                          !chosen view angle
422    CALL pgpt(1, obj_rot(i,1), obj_rot(i,2), -1)      !Draw large dot at the end of the z vector
423    CALL pgslw(2)
424    CALL pgsci(2)
425 END DO
426
427 CALL pgslw(8)
428 CALL pgsci(3)
429
430 DO i = 25, 26          !For NGC147 and NGC185
431    CALL pgslw(2)
432    CALL pgsci(3)
433    CALL pgline(2, (/obj_pro_rot(i,1), obj_rot(i,1)/), (/obj_pro_rot(i,2), obj_rot(i,2)/)) !Draw z vector of object for
434    CALL pgslw(20)                                                                          !chosen view angle
435    CALL pgpt(1, obj_rot(i,1), obj_rot(i,2), -1)     !Draw large dot at the end of the z vector
436    CALL pgslw(2)
437    CALL pgsci(3)
438 END DO
439
440 DO i = 27, 27          !For M33
441    CALL pgslw(2)
442    CALL pgsci(4)
443    CALL pgline(2, (/obj_pro_rot(i,1), obj_rot(i,1)/), (/obj_pro_rot(i,2), obj_rot(i,2)/)) !Draw z vector of object for
444    CALL pgslw(30)                                                                          !chosen view angle
445    CALL pgpt(1, obj_rot(i,1), obj_rot(i,2), -1)     !Draw large dot at the end of the z vector
446    CALL pgslw(2)
447    CALL pgsci(4)
448 END DO
449
450 IF (v_angle .eq. 4) THEN                                  !
451 DO i = 1, 27                                              !
452    CALL pgslw(2)                                          !
453    CALL pgsci(1)                                          !Plot square on survey plane
454    CALL pgpt(1, obj_pro_rot(i,1), obj_pro_rot(i,2), 0254) !
455 END DO                                                    !
456 END IF                                                   !
457
458 CALL pgslw(30)
459 CALL pgsci(4)
460 CALL pgpoint (1, 0., 0., -1)          !Plot large dot at origin for M31
461 CALL pgslw(2)
462 CALL pgsci(4)
463
464 IF (v_angle .eq. 4) THEN     !Plot Survey Area and Inner Cut-Off Ellipse on plane at M31 distance
465
466 CALL pgsci(1)
```

```
467
468 DO i = 2, SAP_ndata - 1
469     CALL pgline(2, (/SAP_rot(i-1,1), SAP_rot(i,1)/), (/SAP_rot(i-1,2), SAP_rot(i,2)/))
470 END DO
471 CALL pgline(2, (/SAP_rot(SAP_ndata-1,1), SAP_rot(1,1)/), (/SAP_rot(SAP_ndata-1,2), SAP_rot(1,2)/))
472
473 DO i = 2, ICP_ndata - 1
474     CALL pgline(2, (/ICP_rot(i-1,1), ICP_rot(i,1)/), (/ICP_rot(i-1,2), ICP_rot(i,2)/))
475 END DO
476 CALL pgline(2, (/ICP_rot(ICP_ndata-1,1), ICP_rot(1,1)/), (/ICP_rot(ICP_ndata-1,2), ICP_rot(1,2)/))
477
478 END IF
479
480 IF (v_angle .eq. 3) THEN     !Plot Survey Area on planes at And XVI and And XXVII distances
481
482 CALL pgsci(1)
483
484 DO i = 2, SAP_ndata - 1
485     CALL pgline(2, (/SAPn_rot(i-1,1), SAPn_rot(i,1)/), (/SAPn_rot(i-1,2), SAPn_rot(i,2)/))
486 END DO
487 CALL pgline(2, (/SAPn_rot(SAP_ndata-1,1), SAPn_rot(1,1)/), (/SAPn_rot(SAP_ndata-1,2), SAPn_rot(1,2)/))
488
489 DO i = 2, SAP_ndata - 1
490     CALL pgline(2, (/SAPf_rot(i-1,1), SAPf_rot(i,1)/), (/SAPf_rot(i-1,2), SAPf_rot(i,2)/))
491 END DO
492 CALL pgline(2, (/SAPf_rot(SAP_ndata-1,1), SAPf_rot(1,1)/), (/SAPf_rot(SAP_ndata-1,2), SAPf_rot(1,2)/))
493
494 !|| Plot lines from the top of the two survey areas
495 !\/ to the bottom of the two survey areas.
496 CALL pgline(2, (/MAXVAL(SAPn_rot(:,1)), MAXVAL(SAPf_rot(:,1))/), (/MAXVAL(SAPn_rot(:,2)), MAXVAL(SAPf_rot(:,2))/))
497 CALL pgline(2, (/MINVAL(SAPn_rot(:,1)), MINVAL(SAPf_rot(:,1))/), (/MINVAL(SAPn_rot(:,2)), MINVAL(SAPf_rot(:,2))/))
498
499 END IF
500
501 !-----------------------------------------Print Satellite Labels-----------------------------------------
502
503 IF (v_angle .eq. 1) THEN !Prints satellite labels for xy plane view
504 CALL pgsci(1)
505 CALL pgptxt(obj_pro_rot(1,1) + 7., obj_pro_rot(1,2), 0., 0.5, 'I')
506 CALL pgptxt(obj_pro_rot(2,1) + 7., obj_pro_rot(2,2), 0., 0.5, 'II')
507 CALL pgptxt(obj_pro_rot(3,1) - 9., obj_pro_rot(3,2), 0., 0.5, 'III')
508 CALL pgptxt(obj_pro_rot(4,1) - 8., obj_pro_rot(4,2) -4., 0., 0.5, 'V')
509 CALL pgptxt(obj_pro_rot(5,1) + 8., obj_pro_rot(5,2), 0., 0.5, 'IX')
510 CALL pgptxt(obj_pro_rot(6,1) + 8., obj_pro_rot(6,2), 0., 0.5, 'X')
511 CALL pgptxt(obj_pro_rot(7,1) + 8., obj_pro_rot(7,2), 0., 0.5, 'XI')
512 CALL pgptxt(obj_pro_rot(8,1) - 6., obj_pro_rot(8,2) - 12., 0., 0.5, 'XII')
513 CALL pgptxt(obj_pro_rot(9,1) + 11., obj_pro_rot(9,2), 0., 0.5, 'XIII')
514 CALL pgptxt(obj_pro_rot(10,1) + 11., obj_pro_rot(10,2), 0., 0.5, 'XIV')
515 CALL pgptxt(obj_pro_rot(11,1) - 9., obj_pro_rot(11,2), 0., 0.5, 'XV')
516 CALL pgptxt(obj_pro_rot(12,1) + 12., obj_pro_rot(12,2), 0., 0.5, 'XVI')
517 CALL pgptxt(obj_pro_rot(13,1) - 12., obj_pro_rot(13,2) - 3., 0., 0.5, 'XVII')
518 CALL pgptxt(obj_pro_rot(14,1), obj_pro_rot(14,2) - 11., 0., 0.5, 'XVIII')
519 CALL pgptxt(obj_pro_rot(15,1) + 12., obj_pro_rot(15,2), 0., 0.5, 'XIX')
520 CALL pgptxt(obj_pro_rot(16,1), obj_pro_rot(16,2) + 6., 0., 0.5, 'XX')
521 CALL pgptxt(obj_pro_rot(17,1) + 14., obj_pro_rot(17,2) - 5., 0., 0.5, 'XXI')
522 CALL pgptxt(obj_pro_rot(18,1) - 13., obj_pro_rot(18,2) - 2., 0., 0.5, 'XXII')
523 CALL pgptxt(obj_pro_rot(19,1) - 14., obj_pro_rot(19,2), 0., 0.5, 'XXIII')
524 CALL pgptxt(obj_pro_rot(20,1) + 5., obj_pro_rot(20,2) - 13., 0., 0.5, 'XXIV')
525 CALL pgptxt(obj_pro_rot(21,1) - 14., obj_pro_rot(21,2) - 2., 0., 0.5, 'XXV')
526 CALL pgptxt(obj_pro_rot(22,1) - 15., obj_pro_rot(22,2) - 3., 0., 0.5, 'XXVI')
527 !CALL pgptxt(obj_pro_rot(23,1) - 16., obj_pro_rot(23,2) - 10., 0., 0.5, 'XXVII') !Hidden behind NGC147
```

```
528  CALL pgptxt(obj_pro_rot(24,1) - 0., obj_pro_rot(24,2) + 5., 0., 0.5, 'XXX')
529  CALL pgptxt(obj_pro_rot(25,1) - 23., obj_pro_rot(25,2) + 7., 0., 0.5, 'NGC147')
530  CALL pgptxt(obj_pro_rot(26,1) + 28., obj_pro_rot(26,2) - 4., 0., 0.5, 'NGC185')
531  CALL pgptxt(obj_pro_rot(27,1) - 7., obj_pro_rot(27,2) - 18., 0., 0.5, 'M33')
532  CALL pgptxt(obj_pro_rot(28,1) - 15., obj_pro_rot(28,2) - 15., 0., 0.5, 'M31')
533  END IF
534
535  IF (v_angle .eq. 3) THEN !Prints satellite labels for yz plane view
536  CALL pgsci(1)
537  CALL pgptxt(1.2 * (obj_rot(1,1)), obj_rot(1,2) - 5., 0., 0.5, 'I')
538  CALL pgptxt(0.8 * (obj_rot(2,1)), obj_rot(2,2) - 15., 0., 0.5, 'II')
539  CALL pgptxt(0.5 * (obj_rot(3,1)), obj_rot(3,2) + 3., 0., 0.5, 'III')
540  CALL pgptxt(1.23 * (obj_rot(4,1)), obj_rot(4,2) - 4., 0., 0.5, 'V')
541  CALL pgptxt(1.08 * (obj_rot(5,1)), obj_rot(5,2) - 3., 0., 0.5, 'IX')
542  CALL pgptxt(1.1 * (obj_rot(6,1)), obj_rot(6,2) - 2., 0., 0.5, 'X')
543  CALL pgptxt(0.5 * (obj_rot(7,1)), obj_rot(7,2) + 2., 0., 0.5, 'XI')
544  CALL pgptxt(0.5 * (obj_rot(8,1)), obj_rot(8,2) - 15., 0., 0.5, 'XII')
545  CALL pgptxt(0.5 * (obj_rot(9,1)), obj_rot(9,2) - 15., 0., 0.5, 'XIII')
546  CALL pgptxt(7.0 * (obj_rot(10,1)), obj_rot(10,2) - 3., 0., 0.5, 'XIV')
547  CALL pgptxt(0.75 * (obj_rot(11,1)), obj_rot(11,2) + 2., 0., 0.5, 'XV')
548  CALL pgptxt(0.85 * (obj_rot(12,1)), obj_rot(12,2) + 2., 0., 0.5, 'XVI')
549  CALL pgptxt(1.35 * (obj_rot(13,1)), obj_rot(13,2) - 10., 0., 0.5, 'XVII')
550  CALL pgptxt(0.95 * (obj_rot(14,1)), obj_rot(14,2) - 12., 0., 0.5, 'XVIII')
551  CALL pgptxt(1.6 * (obj_rot(15,1)), obj_rot(15,2), 0., 0.5, 'XIX')
552  CALL pgptxt(0.5 * (obj_rot(16,1)), obj_rot(16,2), 0., 0.5, 'XX')
553  CALL pgptxt(1.4 * (obj_rot(17,1)), obj_rot(17,2) - 3., 0., 0.5, 'XXI')
554  CALL pgptxt(0.5 * (obj_rot(18,1)), obj_rot(18,2) - 15., 0., 0.5, 'XXII')
555  CALL pgptxt(0.5 * (obj_rot(19,1)), obj_rot(19,2) + 2., 0., 0.5, 'XXIII')
556  CALL pgptxt(0.75 * (obj_rot(20,1)), obj_rot(20,2) - 15., 0., 0.5, 'XXIV')
557  CALL pgptxt(0.45 * (obj_rot(21,1)), obj_rot(21,2) - 12., 0., 0.5, 'XXV')
558  CALL pgptxt(0.75 * (obj_rot(22,1)), obj_rot(22,2) + 2., 0., 0.5, 'XXVI')
559  CALL pgptxt(0.95 * (obj_rot(23,1)), obj_rot(23,2) + 3., 0., 0.5, 'XXVII')
560  CALL pgptxt(0.5 * (obj_rot(24,1)), obj_rot(24,2) + 4., 0., 0.5, 'XXX')
561  CALL pgptxt(1.5 * (obj_rot(25,1)), obj_rot(25,2) - 4., 0., 0.5, 'NGC_147')
562  CALL pgptxt(0.75 * (obj_rot(26,1)), obj_rot(26,2) - 15., 0., 0.5, 'NGC185')
563  CALL pgptxt(3.0 * (obj_rot(27,1)), obj_rot(27,2) - 2., 0., 0.5, 'M33')
564  CALL pgptxt(0.5 * (obj_rot(28,1)) + 30., obj_rot(28,2) - 20., 0., 0.5, 'M31')
565  END IF
566
567  IF (v_angle .eq. 4) THEN !Prints satellite labels for 3D view
568  CALL pgsci(1)
569  CALL pgptxt(obj_pro_rot(1,1) + 5., obj_pro_rot(1,2), 0., 0.5, 'I')
570  CALL pgptxt(obj_pro_rot(2,1) + 5., obj_pro_rot(2,2), 0., 0.5, 'II')
571  CALL pgptxt(obj_pro_rot(3,1) - 7., obj_pro_rot(3,2), 0., 0.5, 'III')
572  CALL pgptxt(obj_pro_rot(4,1) - 6., obj_pro_rot(4,2) -2., 0., 0.5, 'V')
573  CALL pgptxt(obj_pro_rot(5,1) + 6., obj_pro_rot(5,2), 0., 0.5, 'IX')
574  CALL pgptxt(obj_pro_rot(6,1) + 6., obj_pro_rot(6,2), 0., 0.5, 'X')
575  CALL pgptxt(obj_pro_rot(7,1), obj_pro_rot(7,2) - 8., 0., 0.5, 'XI')
576  CALL pgptxt(obj_pro_rot(8,1) + 6., obj_pro_rot(8,2), 0., 0.5, 'XII')
577  CALL pgptxt(obj_pro_rot(9,1) + 8., obj_pro_rot(9,2), 0., 0.5, 'XIII')
578  CALL pgptxt(obj_pro_rot(10,1) - 6., obj_pro_rot(10,2), 0., 0.5, 'XIV')
579  CALL pgptxt(obj_pro_rot(11,1) - 7., obj_pro_rot(11,2), 0., 0.5, 'XV')
580  CALL pgptxt(obj_pro_rot(12,1) + 8., obj_pro_rot(12,2), 0., 0.5, 'XVI')
581  CALL pgptxt(obj_pro_rot(13,1) - 9., obj_pro_rot(13,2), 0., 0.5, 'XVII')
582  CALL pgptxt(obj_pro_rot(14,1), obj_pro_rot(14,2) - 8., 0., 0.5, 'XVIII')
583  CALL pgptxt(obj_pro_rot(15,1) + 8., obj_pro_rot(15,2), 0., 0.5, 'XIX')
584  CALL pgptxt(obj_pro_rot(16,1), obj_pro_rot(16,2) + 4., 0., 0.5, 'XX')
585  CALL pgptxt(obj_pro_rot(17,1) + 9., obj_pro_rot(17,2) - 2., 0., 0.5, 'XXI')
586  CALL pgptxt(obj_pro_rot(18,1) + 9., obj_pro_rot(18,2) +1., 0., 0.5, 'XXII')
587  CALL pgptxt(obj_pro_rot(19,1) - 9., obj_pro_rot(19,2), 0., 0.5, 'XXIII')
588  CALL pgptxt(obj_pro_rot(20,1) + 6., obj_pro_rot(20,2) -8., 0., 0.5, 'XXIV')
```

```
589  CALL pgptxt(obj_pro_rot(21,1) - 9., obj_pro_rot(21,2) - 2., 0., 0.5, 'XXV')
590  CALL pgptxt(obj_pro_rot(22,1) - 10., obj_pro_rot(22,2) - 2., 0., 0.5, 'XXVI')
591  CALL pgptxt(obj_pro_rot(23,1) - 11., obj_pro_rot(23,2) - 2., 0., 0.5, 'XXVII')
592  CALL pgptxt(obj_pro_rot(24,1) - 7., obj_pro_rot(24,2) + 2., 0., 0.5, 'XXX')
593  CALL pgptxt(obj_pro_rot(25,1) - 8., obj_pro_rot(25,2) + 4., 0., 0.5, 'NGC147')
594  CALL pgptxt(obj_pro_rot(26,1) + 18., obj_pro_rot(26,2) - 4., 0., 0.5, 'NGC185')
595  CALL pgptxt(obj_pro_rot(27,1), obj_pro_rot(27,2) - 8., 0., 0.5, 'M33')
596  CALL pgptxt(obj_pro_rot(28,1) - 7., obj_pro_rot(28,2) - 8., 0., 0.5, 'M31')
597  END IF
598
599  !-----------------------------------END Satellite Labels-----------------------------------------
600
601  CALL pgslw(2)
602  CALL pgsci(1)
603
604  CALL PGSCH(1.0)
605  IF (v_angle .eq. 1 .or. v_angle .eq. 2 .or. v_angle .eq. 3) THEN
606  CALL pglab('kpc', 'kpc', '')          !axis labels
607  END IF
608
609  CALL pgend
610
611  END SUBROUTINE PLOT
```

Program: SatDensity_SampCont.f95

Creation Date: 20 Feb 2012

Relevant Section: §4.3 of Paper II (Ch. 4)

Notes: The analysis presented in §4.3 of Paper II concerning the density profile of the M31 halo was carried out using this program. It fits a spherically-symmetric unbroken power law to the satellite density profile, taking into account the uneven coverage of the PAndAS survey area. One of the most difficult tasks here was to actually generate the PAndAS survey polygon from the field centers specified by Mike Irwin (Institute of Astronomy, University of Cambridge) and this took me two days to complete. Using this polygon as the outer boundary, and using the inner cutoff ellipse around the M31 disk, it is possible to determine how much volume at each radius (i.e. distance from M31) would fall inside the utilized survey region. The resulting function can then be used to weight the density profile so that we can obtain an unbiased measure of the slope of the power law for the desired sample of satellites. Note that the program can perform the analysis using either the best-fit distances alone or the full distance distributions for each satellite. The code presented here is as applied to the whole satellite sample but desired satellites can be omitted from the sample by skipping over them in the 'MaxLike' subroutine.

```
1  MODULE Global  !Define all
2  IMPLICIT NONE  !variables
3
4  INTEGER :: i, j, h, n, nn, k_ndata
5  REAL :: Sat_Rad(27), alpha, alpha_hold(600), ML_logL(600), Rel_ML_logL(600), norm_fac
6  REAL :: k_vs_alpha(601,2), k(600)
7  REAL :: alpha_counts, alpha_psigma, alpha_msigma, pcounts, mcounts
8  DOUBLE PRECISION :: av_ML_logL
9
10 !|| For sampled distributions
11 !\/ with MCMC
12 INTEGER :: nit, it, nsamples, ndata_max, ios, idum = -9999
13 PARAMETER (nit = 3000000)
14 PARAMETER (nsamples = 10000)
15 PARAMETER (ndata_max = 3000000)
16 INTEGER :: time(ndata_max)
17 REAL :: Sat_Radii(27,500001), M33_dist, in_cut, out_cut, LikeA, LikeB, logL, randnum, r
18 REAL :: Radius(27,10000)
19 REAL :: x(ndata_max, 3), p(3)
20 REAL :: post_y1(600), post_y2(100), post_y3(300), Best_Combo(6)
21 REAL :: post_x1(600), post_x2(100), post_x3(300), pi
22 PARAMETER (pi = 3.141592)
23 CHARACTER :: folder*100, string*200, string2*200, command*200, sample
24 PARAMETER (sample = 'y')
25
26 END MODULE Global
27
```

```
28  !-----------------------------------------------------------------------------------------------
29
30  PROGRAM SatDensity_SampCont !Master program
31  USE Global
32  IMPLICIT NONE
33
34  WRITE (folder,*) 'Sat_Density'
35  WRITE (string,*) './' // TRIM(ADJUSTL(folder))
36
37  WRITE (command,*) 'mkdir ' // TRIM(ADJUSTL(folder))
38
39  CALL system(command)
40
41  CALL random_seed
42  CALL NonSampledRadii
43  IF (sample .eq. 'y') THEN !If using many samples of possible
44     CALL SampledRadii      !radii for each object as opposed
45     CALL SampleSelect      !to just the best fit radius
46  END IF
47  CALL k_verse_alpha
48  CALL MaxLike
49
50  string2 = TRIM(ADJUSTL(folder)) // '/results.dat'
51  OPEN(3, file=TRIM(ADJUSTL(string2)), status = 'unknown')
52  WRITE (3,*) "Results for Maximum Likelihood test"
53  WRITE (3,*) "Most Likely alpha (Max Likelihood):", alpha_hold(MAXLOC(ML_logL))
54  WRITE (3,*) "plus 1 sigma error:", alpha_psigma, "; minus 1 sigma error:", alpha_msigma
55
56  END PROGRAM SatDensity_SampCont
57
58  !-----------------------------------------------------------------------------------------------
59
60  SUBROUTINE NonSampledRadii     !Use this subroutine if using the directly
61  USE Global                     !calculated values of the Satellite to M31
62  IMPLICIT NONE                  !distance rather than samples from the PPD
63
64  Sat_Rad(1) = 68.e0     !And I
65  Sat_Rad(2) = 196.e0    !And II
66  Sat_Rad(3) = 86.e0     !And III
67  Sat_Rad(4) = 113.e0    !And V
68  Sat_Rad(5) = 182.e0    !And IX
69  Sat_Rad(6) = 130.e0    !And X
70  Sat_Rad(7) = 103.e0    !And XI
71  Sat_Rad(8) = 182.e0    !And XII
72  Sat_Rad(9) = 116.e0    !And XIII
73  Sat_Rad(10) = 163.e0   !And XIV
74  Sat_Rad(11) = 174.e0   !And XV
75  Sat_Rad(12) = 320.e0   !And XVI
76  Sat_Rad(13) = 67.e0    !And XVII
77  Sat_Rad(14) = 457.e0   !And XVIII
78  Sat_Rad(15) = 116.e0   !And XIX
79  Sat_Rad(16) = 129.e0   !And XX
80  Sat_Rad(17) = 136.e0   !And XXI
81  Sat_Rad(18) = 280.e0   !And XXII
82  Sat_Rad(19) = 128.e0   !And XXIII
83  Sat_Rad(20) = 169.e0   !And XXIV
84  Sat_Rad(21) = 91.e0    !And XXV
85  Sat_Rad(22) = 103.e0   !And XXVI
86  Sat_Rad(23) = 482.e0   !And XXVII
87  Sat_Rad(24) = 146.e0   !And XXX
88  Sat_Rad(25) = 118.e0   !NGC147
```

```
89  Sat_Rad(26) = 181.e0    !NGC185
90  Sat_Rad(27) = 214.e0    !M33
91
92  END SUBROUTINE NonSampledRadii
93
94  !-----------------------------------------------------------------------------------------
95
96  SUBROUTINE k_verse_alpha !This subroutine finds values of the power law normalization factor 'k' for
97  USE Global               !each power law (i.e. exponent 'alpha'). It simply interpolates based on values
98  IMPLICIT NONE            !derived by Geraint (provided in 'k_vs_alpha.dat')
99
100 OPEN(40, file='./k_vs_alpha.dat', status = 'unknown')
101
102 i = 0
103
104 DO WHILE (.TRUE.)
105
106 i = i + 1
107
108 READ (40, *, IOSTAT = ios) k_vs_alpha(i,1), k_vs_alpha(i,2)
109
110 IF (ios == -1) THEN
111    i = i - 1
112    exit
113 ELSE IF (ios .gt. 0) THEN
114    WRITE (*,*) i
115    i=i-1
116    cycle
117 END IF
118
119 END DO
120
121 CLOSE(36)
122
123 k_ndata = i
124
125 DO i = 1, 599              !k as determined for alpha = 0.01, 0.02, ..., 5.99
126    alpha = REAL(i)/100.e0
127    nn = INT((alpha - k_vs_alpha(1,1))/(k_vs_alpha(241,1) - k_vs_alpha(1,1)) * 241.e0) + 1
128    k(i) = (alpha - k_vs_alpha(nn,1))/(k_vs_alpha(nn+1, 1) - k_vs_alpha(nn,1)) * (k_vs_alpha(nn+1,2) - k_vs_alpha(nn,2)) + k_vs_alpha(
           nn,2)
129
130    k(i) = 10.e0 ** k(i)
131
132 END DO
133
134 DO i = 600, 600            !k for alpha = 6.00
135    alpha = REAL(i)/100.e0
136    nn = 241
137    k(i) = (alpha - k_vs_alpha(nn,1))/(k_vs_alpha(nn+1, 1) - k_vs_alpha(nn,1)) * (k_vs_alpha(nn+1,2) - k_vs_alpha(nn,2)) + k_vs_alpha(
           nn,2)
138
139    k(i) = 10.e0 ** k(i)
140
141 END DO
142
143
144 END SUBROUTINE k_verse_alpha
145
146 !-----------------------------------------------------------------------------------------
147
```

```
148 SUBROUTINE MaxLike  !Direct calculation of likelihoods for alpha
149 USE Global          !assuming fixed values of inner and outer
150 IMPLICIT NONE       !cutoff radii. See Eq. 16 of Paper II for operation.
151
152 string2 = TRIM(ADJUSTL(folder)) // '/alpha_ML_dist.dat'
153 OPEN(9, file=TRIM(ADJUSTL(string2)), status = 'unknown')
154
155 ML_logL = 0.e0
156 DO i = 1, 600
157 print *, i
158    alpha = REAL(i)/100.e0
159    alpha_hold(i) = alpha
160
161    IF (sample .eq. 'y') THEN
162
163    DO j = 1, 27
164       av_ML_logL = 0.d0
165       DO h = 1, 500000 !nsamples
166          av_ML_logL = av_ML_logL + ((k(i) * Sat_Radii(j, h) ** (2.d0 - alpha)))
167       END DO
168       av_ML_logL = LOG10(av_ML_logL)
169       ML_logL(i) = ML_logL(i) + av_ML_logL
170    END DO
171
172    ELSE
173
174       DO j = 1, 27
175          ML_logL(i) = ML_logL(i) + LOG10((k(i) * Sat_Rad(j) ** (2.e0 - alpha)))  !Determine likelihood of given alpha
176       END DO
177
178    END IF
179
180 END DO
181
182 !|| Plot
183 !\/ Distribution
184 string2 = TRIM(ADJUSTL(string)) // '/ml_alpha_loglike.ps/CPS'
185 CALL pgbegin(0,TRIM(ADJUSTL(string2)),1,1)
186
187 CALL pgenv(0., 6., 0.9 * MINVAL(ML_logL), 1.1 * MAXVAL(ML_logL), 0, 0)
188
189 CALL pgbin(600, alpha_hold, ML_logL, .false.)
190
191 CALL pglab('\(0627)', 'Log_Likelihood', '')
192
193 CALL pgend
194
195 WRITE (command,*) 'convert_-rotate_90_' // TRIM(ADJUSTL(string)) // &
196                   '/ml_alpha_loglike.ps_' // TRIM(ADJUSTL(string)) // &
197                   '/ml_alpha_loglike.jpg'
198
199 call system(command)
200 !/\
201 !||
202
203 !|| Plot
204 !\/ Distribution
205 string2 = TRIM(ADJUSTL(string)) // '/ml_alpha_PPD.ps/CPS'
206 CALL pgbegin(0,TRIM(ADJUSTL(string2)),1,1)
207
208 DO i = 1, 600
```

```
209     Rel_ML_logL(i) = 10.** (ML_logL(i) - MAXVAL(ML_logL))
210 END DO
211
212 Rel_ML_logL = Rel_ML_logL/ SUM(Rel_ML_logL)
213
214 CALL pgenv(0., 3.5, 0., 1.1 * MAXVAL(Rel_ML_logL), 0, 0)
215
216 DO i = 1, 600
217     WRITE (9, '(3F16.5)') alpha_hold(i), Rel_ML_logL(i), ML_logL(i)
218 END DO
219
220 CALL pgline(600, alpha_hold, Rel_ML_logL)
221
222 CALL pglab('\(0627)', 'Probability', '')
223
224 CALL pgend
225
226 WRITE (command,*) 'convert -rotate 90 ' // TRIM(ADJUSTL(string)) // &
227                         '/ml_alpha_PPD.ps ' // TRIM(ADJUSTL(string)) // &
228                         '/ml_alpha_PPD.jpg'
229
230 call system(command)
231 !/\
232 !||
233
234 alpha_counts = 0.d0 ; mcounts = 0.d0                                    !
235 DO i = MAXLOC(REL_ML_logL, DIM = 1), 1, -1                              !
236     mcounts = mcounts + REL_ML_logL(i)                                  !
237 END DO                                                                  !
238 DO i = MAXLOC(REL_ML_logL, DIM = 1), 1, -1                              !
239     alpha_counts = alpha_counts + REL_ML_logL(i)                        !Finds negative
240     IF (alpha_counts .ge. 0.682*mcounts) THEN                           !one sigma error
241         alpha_msigma = alpha_hold(MAXLOC(REL_ML_logL, DIM = 1)) - alpha_hold(i)  !
242         exit                                                            !
243     END IF                                                              !
244 END DO                                                                  !
245
246 alpha_counts = 0.d0 ; pcounts = 0.d0                                    !
247 DO i = MAXLOC(REL_ML_logL, DIM = 1), 600                                !
248     pcounts = pcounts + REL_ML_logL(i)                                  !
249 END DO                                                                  !
250 DO i = MAXLOC(REL_ML_logL, DIM = 1), 600                                !
251     alpha_counts = alpha_counts + REL_ML_logL(i)                        !Finds positive
252     IF (alpha_counts .ge. 0.682*pcounts) THEN                           !one sigma error
253         alpha_psigma = alpha_hold(i) - alpha_hold(MAXLOC(REL_ML_logL, DIM = 1))  !
254         exit                                                            !
255     END IF                                                              !
256 END DO                                                                  !
257
258
259 END SUBROUTINE
260
261 !-------------------------------------------------------------------------------------------
262 !MCMC subroutine omitted
263 !-------------------------------------------------------------------------------------------
264 !Plots subroutine omitted
265 !-----------------------------------------------------------------------
266 !LogLike subroutine omitted
267 !-------------------------------------------------------------------------------------------
268
269 SUBROUTINE SampledRadii !Read in sampled radii probability
```

```
270  USE Global              !Distributions for each satellite
271  IMPLICIT NONE
272
273  OPEN (unit = 11, file = './AndromedaIe/other_plots2/Sampled_M31_Distances.dat', status = 'old')
274  OPEN (unit = 12, file = './AndromedaIIe/other_plots2/Sampled_M31_Distances.dat', status = 'old')
275  OPEN (unit = 13, file = './AndromedaIIIe/other_plots2/Sampled_M31_Distances.dat', status = 'old')
276  OPEN (unit = 14, file = './AndromedaVe/other_plots2/Sampled_M31_Distances.dat', status = 'old')
277  OPEN (unit = 15, file = './AndromedaIXe/other_plots2/Sampled_M31_Distances.dat', status = 'old')
278  OPEN (unit = 16, file = './AndromedaXe/other_plots2/Sampled_M31_Distances.dat', status = 'old')
279  OPEN (unit = 17, file = './AndromedaXIe/other_plots2/Sampled_M31_Distances.dat', status = 'old')
280  OPEN (unit = 18, file = './AndromedaXIIe/other_plots2/Sampled_M31_Distances.dat', status = 'old')
281  OPEN (unit = 19, file = './AndromedaXIIIe/other_plots2/Sampled_M31_Distances.dat', status = 'old')
282  OPEN (unit = 20, file = './AndromedaXIVe/other_plots2/Sampled_M31_Distances.dat', status = 'old')
283  OPEN (unit = 21, file = './AndromedaXVe/other_plots2/Sampled_M31_Distances.dat', status = 'old')
284  OPEN (unit = 22, file = './AndromedaXVIe/other_plots2/Sampled_M31_Distances.dat', status = 'old')
285  OPEN (unit = 23, file = './AndromedaXVIIe/other_plots2/Sampled_M31_Distances.dat', status = 'old')
286  OPEN (unit = 24, file = './AndromedaXVIIIe/other_plots2/Sampled_M31_Distances.dat', status = 'old')
287  OPEN (unit = 25, file = './AndromedaXIXe/other_plots2/Sampled_M31_Distances.dat', status = 'old')
288  OPEN (unit = 26, file = './AndromedaXXe/other_plots2/Sampled_M31_Distances.dat', status = 'old')
289  OPEN (unit = 27, file = './AndromedaXXIe/other_plots2/Sampled_M31_Distances.dat', status = 'old')
290  OPEN (unit = 28, file = './AndromedaXXIIe/other_plots2/Sampled_M31_Distances.dat', status = 'old')
291  OPEN (unit = 29, file = './AndromedaXXIIIe/other_plots2/Sampled_M31_Distances.dat', status = 'old')
292  OPEN (unit = 30, file = './AndromedaXXIVe/other_plots2/Sampled_M31_Distances.dat', status = 'old')
293  OPEN (unit = 31, file = './AndromedaXXVe/other_plots2/Sampled_M31_Distances.dat', status = 'old')
294  OPEN (unit = 32, file = './AndromedaXXVIe/other_plots2/Sampled_M31_Distances.dat', status = 'old')
295  OPEN (unit = 33, file = './AndromedaXXVIIe/other_plots2/Sampled_M31_Distances.dat', status = 'old')
296  OPEN (unit = 34, file = './AndromedaXXXe/other_plots2/Sampled_M31_Distances.dat', status = 'old')
297  OPEN (unit = 35, file = './NGC147e_outer/other_plots2/Sampled_M31_Distances.dat', status = 'old')
298  OPEN (unit = 36, file = './NGC185e_outer/other_plots2/Sampled_M31_Distances.dat', status = 'old')
299  OPEN (unit = 37, file = './M33e/other_plots2/Sampled_M31_Distances.dat', status = 'old')
300
301  !-----------------------------------------
302
303  !|| Read in the distribution of                    **
304  !\/ distances of Andromeda I from M31
305  i = 0
306
307  DO WHILE (.TRUE.)
308
309  i = i + 1
310
311  READ (11, *, IOSTAT = ios) Sat_Radii(1,i)
312
313  IF (ios == -1) THEN
314     i = i - 1
315     exit
316  ELSE IF (ios .gt. 0) THEN
317     WRITE (*,*) i
318     i=i-1
319     cycle
320  END IF
321
322  END DO
323
324  !-----------------------------------------
325  !Repeat ** to read in the distance distribution
326  !of each satellite with respect to M31
327
328  CLOSE(11) ; CLOSE(12) ; CLOSE(13) ; CLOSE(14) ; CLOSE(15) ; CLOSE(16) ; CLOSE(17) ; CLOSE(18) ; CLOSE(19) ; CLOSE(20)
329  CLOSE(21) ; CLOSE(22) ; CLOSE(23) ; CLOSE(24) ; CLOSE(25) ; CLOSE(26) ; CLOSE(27) ; CLOSE(28) ; CLOSE(29) ; CLOSE(30)
330  CLOSE(31) ; CLOSE(32) ; CLOSE(33) ; CLOSE(34) ; CLOSE(35) ; CLOSE(36) ; CLOSE(37)
```

```
331
332 END SUBROUTINE SampledRadii
333
334 !-----------------------------------------------------------------------------------------------
335
336 SUBROUTINE SampleSelect
337 USE Global
338 IMPLICIT NONE
339
340 REAL :: gasdev
341
342 DO i = 1, 27
343    DO j = 1, nsamples
344   1   CALL random_number(randnum)
345       randnum = (randnum * 500000) + 1
346       IF (Sat_Radii(i, NINT(randnum)) .ge. 50.e0 .and. Sat_Radii(i, NINT(randnum)) .le. 600.e0) THEN
347           Radius(i,j) = Sat_Radii(i, NINT(randnum))
348           WRITE (*,*) i, j, Radius(i,j)
349       ELSE
350           goto 1
351       END IF
352    END DO
353 END DO
354
355 END SUBROUTINE SampleSelect
356
357 !-----------------------------------------------------------------------------------------------
```

D

Chapter Five Programs

Program: PlaneSigRMS.f95

Creation Date: 7 June 2012 (first version Feb 2012) Many modifications.

Relevant Section: Ch. 5

Notes: This program is representative of the many versions used to implement plane fitting on the satellite distribution. The satellite distance distributions (along with the best fit distances) are read in and stored for subsequent sampling in the 'SampledDist' subroutine. The 'Significance' subroutine then samples distances for each satellite (and M31 itself), converts these distances into 3D positions and calls 'MaxSigFind' to do the actual plane fitting (see the preface for Paper III (p. 90) and Fig. 5.1 for details). The program is currently set to repeat this process for 200,000 realizations of possible positions of the satellites as well as the particular realization where each satellite is in the position defined by its best-fit distance (mainly for plotting). The program is also set up to build 200,000 random realizations (generated in the 'RandomPoints' subroutine) of the (27) satellites and perform equivalent plane fitting on each. The version of 'RandomPoints' included in this program includes only one possible position for each satellite, and is used only in §3.3 of Paper III. The version of this subroutine used in all other sections can be seen in the program '*PlaneSigSubSets_RandReal4_noGroup.f95*' (p. 268). This version represents each satellite by a distance distribution containing 1,000 possible positions along the line of sight from Earth (i.e. an accurate representation of the real data). Note also that this program is designed to perform plane-fitting on the whole sample. The modified code segments designed to handle each satellite combination of a given size can be seen in *Subroutines for Processing Satellite Subsets* (p. 260). The 'goodness of fit statistic' used for the plane fitting by this program is the RMS. The code for alternative measures are given in *Alternative Plane Fitting Code Segments* (p. 258).

```
1  MODULE Global !Defines all variables used by BayesianTRGB
2  IMPLICIT NONE
3
4  INTEGER :: i, j, k, s, mm, ios, idum = -9999, it, nit, err_samp
5  INTEGER :: ndata_max, nsats
6  PARAMETER (ndata_max = 10000000)
7  PARAMETER (nit = 200000)
8  PARAMETER (err_samp = 200000)
9  PARAMETER (nsats = 27)
10 REAL*8 :: pi
11 PARAMETER (pi = ACOS(-1.e0))
12 REAL :: randnum, sig(nit), err_sig(nit), norm(3), best_fit_vect(3), best_fit_sigma
```

```
13  REAL :: pos(3,ndata_max), temp_pos(3)
14  REAL :: a(ndata_max), b(ndata_max), c(ndata_max), d(ndata_max)
15  REAL :: a_hist(201,2), b_hist(201,2), c_hist(201,2), d_hist(2001,2)
16  REAL :: logL, LikeA, LikeB, r, p(4), p_temp(4), min_sigma, max_sigma
17  REAL*8 :: sigma, planeDist, like, plane_sig, rms
18  REAL :: max_plane_sig, min_signif, Actual_sig, Actual_bfv(3), Actual_bfs
19  INTEGER :: dummy, sat_pick
20  REAL :: Sat_Dist(500000,27), Sat_Dist_Drawn(27), Sat_Pos(27,2), xi(27), eta(27), theta(27), M31_Dist_PPD(3000000)
21  REAL*8 :: RA, DEC, xi_dble, eta_dble
22  REAL :: xi_test, eta_test, theta_test, SAP_xi(134), SAP_eta(134), spotR
23  REAL :: Best_Sat_Dist(27)
24  REAL :: m31_dist
25  REAL :: alpha_set, beta_set, gamma_set, pole_alpha, pole_beta
26  REAL :: x_rot(3,3), y_rot(3,3), z_rot(3,3)
27  REAL :: par_like(180,6)
28  REAL :: theta_coord, phi_coord
29  CHARACTER :: argv*30, folder*100, string*200, string2*200, command*200, subsize*3
30
31  END MODULE Global
32
33  !-------------------------------------------------------------------------
34
35  PROGRAM PlaneSignificance     !Master program
36  USE Global
37  IMPLICIT NONE
38
39  WRITE (subsize,*) nsats
40
41  WRITE (folder,*) 'RMS_Plane_Stats_' // TRIM(ADJUSTL(subsize)) // '_sats' !Primary output
42  WRITE (string,*) './' // TRIM(ADJUSTL(folder))                           !directory
43
44  WRITE (command,*) 'mkdir ' // TRIM(ADJUSTL(folder))
45
46  CALL system(command)
47
48  CALL random_seed !Insure random seed for random numbers
49
50  CALL SampledDist !Get sampled satellite distances
51
52  CALL BorderGet !Get PAndAS survey boundary points
53
54  string2 = TRIM(ADJUSTL(folder)) // '/results.dat'          !File for result summary e.g. plane orient-
55  OPEN(11, file=TRIM(ADJUSTL(string2)), status = 'unknown')  !ation and RMS for real satellite distribution
56
57  string2 = TRIM(ADJUSTL(folder)) // '/sat_pos.dat'          !Positions of satellites in
58  OPEN(12, file=TRIM(ADJUSTL(string2)), status = 'unknown')  !Random Realizations
59
60  string2 = TRIM(ADJUSTL(folder)) // '/significance.dat'     !RMS distribution and poles from
61  OPEN(13, file=TRIM(ADJUSTL(string2)), status = 'unknown')  !Random Realizations
62
63  string2 = TRIM(ADJUSTL(folder)) // '/real_sig_wth_err.dat' !RMS distribution and poles for realizations
64  OPEN(14, file=TRIM(ADJUSTL(string2)), status = 'unknown')  !of the real satellite distribution
65
66  CALL Significance !The main subroutine which in turn calls the plane fitting subroutine
67
68  CALL Theta_Phi(Actual_bfv(1), Actual_bfv(2), Actual_bfv(3))
69
70  WRITE(11,*) "Best fit vector: (", Actual_bfv(1), Actual_bfv(2), Actual_bfv(3), ")"
71  WRITE(11,*) "Theta =", theta_coord, "; Phi =", phi_coord
72  WRITE(11,*) "LOG10(RMS) of best fit:", Actual_sig
73  WRITE(11,*) "Minimum RMS from random samples:", min_signif
```

```
74
75  CLOSE(11) ; CLOSE(12) ; CLOSE(13) ; CLOSE(14)
76
77  END PROGRAM PlaneSignificance
78
79  !-------------------------------------------------------------------------
80
81  SUBROUTINE RandomPoints !Generates Random Realizations of Satellites
82  USE Global
83  IMPLICIT NONE
84
85  LOGICAL :: in_poly
86
87  CALL random_number(randnum)
88  randnum = randnum * 2999999.e0 + 1.e0
89  m31_dist = M31_Dist_PPD(NINT(randnum))
90
91  DO i = 1, nsats
92
93   2 CALL random_number(randnum)
94     sat_pick = 1 + NINT(randnum*REAL(nsats - 1))   !Draw a random satellite
95
96     CALL random_number(randnum)
97     randnum = randnum * 499999.e0 + 1.e0
98     pos(1,i) = ABS(Sat_Dist(NINT(randnum),sat_pick) * cos(theta(sat_pick)) * tan(xi(sat_pick))) !Determine length of x vector
99     IF (xi(sat_pick) .lt. 0.e0) THEN                                    !                  !for each satellite
100       pos(1,i) = -1.e0 * pos(1,i)                                       !Determine if x is positive or negative
101    END IF                                                              !
102
103    pos(2,i) = ABS(Sat_Dist(NINT(randnum),sat_pick) * sin(eta(sat_pick)))   !Determine length of y vector for each satellite
104    IF (eta(sat_pick) .lt. 0.e0) THEN                                   !
105       pos(2,i) = -1.e0 * pos(2,i)                                      !Determine if y is positive or negative
106    END IF                                                              !
107
108    pos(3,i) = Sat_Dist(NINT(randnum),sat_pick) * cos(theta(sat_pick)) - m31_dist         !Determine length and sign of z vector
109
110    pos(3,i) = SQRT((pos(1,i)**2.e0) + (pos(2,i)**2.e0) + (pos(3,i)**2.e0)) !Rotate position vector to point
111    pos(1,i) = 0.e0 ; pos(2,i) = 0.e0                                   !along z-axis
112
113    CALL random_number(randnum)                    !
114    alpha_set = randnum * 360.e0 * (pi/180.e0)     !Pick random longitude
115    CALL random_number(randnum)             !Pick random latitude between 0 and 90 weighted
116    beta_set = ASIN(randnum)                !by area of a sphere as a function of latitude
117    CALL random_number(randnum)             !
118    IF (randnum .lt. 0.5e0) THEN            !Re-assign latitude as
119       beta_set = beta_set                  !
120    ELSE                                    !-1 * latitude in
121       beta_set = -beta_set                 !
122    END IF                                  !50% 0f cases
123
124    CALL Rotate
125    pos(:,i) = MATMUL(y_rot,pos(:,i))   !Rotate to the chosen
126    pos(:,i) = MATMUL(x_rot,pos(:,i))   !random angle
127
128    xi_test = ATAN(abs(pos(1,i))/(m31_dist + pos(3,i)))                 !Convert
129    IF (pos(1,i) .lt. 0.e0) THEN                                        !new random
130       xi_test = -xi_test                                               !position
131    END IF                                                              !vector
132    eta_test = ATAN(abs(pos(2,i))/SQRT(pos(1,i)**2 + (m31_dist + pos(3,i))**2))!into
133    IF (pos(2,i) .lt. 0.e0) THEN                                        !non t.p.
134       eta_test = -eta_test                                             !eta and
```

```
135     END IF                                                                  !xi
136
137     RA = xi_test                                                !
138     DEC = eta_test                                              !Use sla_DS2TP
139                                                                 !to convert true
140     CALL sla_DS2TP (RA, DEC, 0.d0, 0.d0, xi_dble, eta_dble, j)  !eta amd xi to
141                                                                 !their tangent
142     xi_test = xi_dble * (180.e0/pi)                             !plane projections
143     eta_test = eta_dble * (180.e0/pi)                           !
144
145     IF (in_poly(xi_test,eta_test,134,SAP_xi,SAP_eta)) THEN                  !Re-generate
146                                                                             !the new
147     ELSE                                                                    !randomized
148         goto 2                                                              !satellite
149     END IF                                                                  !position if
150                                                                             !the current
151     spotR = ((xi_test*cos(51.9d0*pi/180.d0) + eta_test*sin(51.9d0*pi/180.d0))**2 / 6.25d0) + & !choice doesn't
152     ((xi_test*sin(51.9d0*pi/180.d0) - eta_test*cos(51.9d0*pi/180.d0))**2 / 1.d0)            !fall within
153                                                                             !the PAndAS
154     IF (spotR .le. 1.e0) THEN                                               !footprint
155         goto 2                                                              !as viewed
156     END IF                                                                  !from Earth
157 END DO
158
159 END SUBROUTINE RandomPoints
160
161 !-------------------------------------------------------------------------------
162
163 SUBROUTINE Rotate  !Rotation Matrices for rotations about x,y and z axes. 'alpha_set'
164 USE Global         !is the desired rotation angle about the x axis, 'beta_set' about
165 IMPLICIT NONE      !the y axis and 'gamma_set' about the z axis
166
167 x_rot(1,1) = 1.e0                      !
168 x_rot(2,1) = 0.e0                      !
169 x_rot(3,1) = 0.e0                      !
170 x_rot(1,2) = 0.e0                      !
171 x_rot(2,2) = cos(alpha_set)            !Rotation about x-axis - angle alpha
172 x_rot(3,2) = -1.e0 * sin(alpha_set)    !
173 x_rot(1,3) = 0.e0                      !
174 x_rot(2,3) = sin(alpha_set)            !
175 x_rot(3,3) = cos(alpha_set)            !
176
177 y_rot(1,1) = cos(beta_set)             !
178 y_rot(2,1) = 0.e0                      !
179 y_rot(3,1) = sin(beta_set)             !
180 y_rot(1,2) = 0.e0                      !
181 y_rot(2,2) = 1.e0                      !Rotation about y-axis - angle beta
182 y_rot(3,2) = 0.e0                      !
183 y_rot(1,3) = -1.e0 * sin(beta_set)     !
184 y_rot(2,3) = 0.e0                      !
185 y_rot(3,3) = cos(beta_set)             !
186
187 z_rot(1,1) = cos(gamma_set)            !
188 z_rot(2,1) = -1.e0 * sin(gamma_set)    !
189 z_rot(3,1) = 0.e0                      !
190 z_rot(1,2) = sin(gamma_set)            !
191 z_rot(2,2) = cos(gamma_set)            !Rotation about z-axis - angle gamma
192 z_rot(3,2) = 0.e0                      !
193 z_rot(1,3) = 0.e0                      !
194 z_rot(2,3) = 0.e0                      !
195 z_rot(3,3) = 1.e0                      !
```

```
196
197  END SUBROUTINE Rotate
198
199  !-------------------------------------------------------------------------
200
201  SUBROUTINE Theta_Phi(x,y,z) !Converts from cartesian x,y,z into spherical coordinates theta
202  USE Global                  !and phi (r is not required) for obtaining positions of objects
203  IMPLICIT NONE               !(and plane normal vector pointings) in M31 galactic coordinates
204
205  REAL :: x, y, z
206
207  theta_coord = acos(z/(SQRT(x**2.e0 + y**2.e0 + z**2.e0))) - (pi/2.e0)
208
209  theta_coord = -theta_coord * (180.e0/pi)
210
211  IF (x .gt. 0.e0) THEN
212
213     phi_coord = atan(y/x)
214
215  ELSE IF (x .lt. 0.e0 .and. y .ge. 0.e0) THEN
216
217     phi_coord = atan(y/x) + pi
218
219  ELSE IF (x .lt. 0.e0 .and. y .lt. 0.e0) THEN
220
221     phi_coord = atan(y/x) - pi
222
223  ELSE IF (x .eq. 0.e0 .and. y .gt. 0.e0) THEN
224
225     phi_coord = pi/2.e0
226
227  ELSE IF (x .eq. 0.e0 .and. y .lt. 0.e0) THEN
228
229     phi_coord = -pi/2.e0
230
231  ELSE IF (x .eq. 0.e0 .and. y .eq. 0.e0) THEN
232
233     phi_coord = 0.e0
234
235  END IF
236
237  phi_coord = -phi_coord * (180.e0/pi)
238
239  END SUBROUTINE Theta_Phi
240
241  !-------------------------------------------------------------------------
242
243  SUBROUTINE MaxSigFind  !Finds best fit plane for a satellite distribution by testing goodness of fit of each
244  USE Global             !tested plane. The poles of the tested planes are all approximately equi-distant, taking
245  IMPLICIT NONE          !into account the surface area of a shere as a function of latitude.
246                         !A low resolution run finds the approximate location of the best fit plane's pole and then
247                         !poles around this point are searched at higher resolution.
248  par_like = 0.e0
249  max_plane_sig = 9999999.e0
250
251  !|| Low resolution
252  !\/ plane tests
253  DO i = 1, 30
254
255     beta_set = REAL(i*3) * (pi/180.e0)
256
```

```
257     DO j = 1, NINT(120.e0 * cos(beta_set)) !The higher the latitude, the smaller the
258                                            !number of points
259         alpha_set = (REAL(j)/NINT(120.e0 * cos(beta_set))) * 360.e0 * (pi/180.e0)
260
261         norm = (/ 0.e0, 0.e0, 1.e0 /)
262
263         CALL Rotate
264         norm = MATMUL(y_rot,norm)
265         norm = MATMUL(x_rot,norm)
266
267         plane_sig = 0.d0
268         rms = 0.d0
269         DO k = 1, nsats    !RMS Calculation
270             planeDist = norm(1)*pos(1,k) + norm(2)*pos(2,k) + norm(3)*pos(3,k)
271             rms = rms + (planeDist)**2
272         END DO
273         rms = SQRT(rms/nsats)
274         plane_sig = LOG10(rms)
275         IF (plane_sig .lt. max_plane_sig) THEN   !Most significant plane has lowest rms
276             max_plane_sig = plane_sig    !Store approx, low resolution values
277             best_fit_vect = norm         !of best fit pole and significance
278             pole_alpha = alpha_set    !Store best fit pole for
279             pole_beta = beta_set      !high resolution search
280         END IF
281     END DO
282 END DO
283
284 norm = (/ -1.e0, 0.e0, 0.e0 /) !Test at the actual pole (not included in above loop)
285
286 plane_sig = 0.d0
287 rms = 0.d0
288 DO k = 1, nsats    !RMS Calculation
289     planeDist = norm(1)*pos(1,k) + norm(2)*pos(2,k) + norm(3)*pos(3,k)
290     rms = rms + (planeDist)**2
291 END DO
292 rms = SQRT(rms/nsats)
293 plane_sig = LOG10(rms)
294
295 !|| High resolution search
296 !\/ around best fit pole
297 IF (plane_sig .lt. max_plane_sig) THEN  !Condition not met unless the RMS at the actual pole
298                                         !was better than anywhere else in the low res search
299
300 max_plane_sig = plane_sig
301 best_fit_vect = norm
302 DO i = 1, 15
303
304     beta_set = (88.5e0 + (REAL(i)/10.e0)) * (pi/180.e0)
305
306     DO j = 1, NINT(1200.e0 * cos(beta_set)) !The higher the latitude, the smaller the
307                                             !number of points
308         alpha_set = (REAL(j)/NINT(1200.e0 * cos(beta_set))) * 360.e0 * (pi/180.e0)
309
310         norm = (/ 0.e0, 0.e0, 1.e0 /)
311
312         CALL Rotate
313         norm = MATMUL(y_rot,norm)
314         norm = MATMUL(x_rot,norm)
315
316         plane_sig = 0.d0
317         rms = 0.d0
```

```
318         DO k = 1, nsats    !RMS Calculation
319             planeDist = norm(1)*pos(1,k) + norm(2)*pos(2,k) + norm(3)*pos(3,k)
320             rms = rms + (planeDist)**2
321         END DO
322         rms = SQRT(rms/nsats)
323         plane_sig = LOG10(rms)
324         IF (plane_sig .lt. max_plane_sig) THEN    !Most significant plane has lowest rms
325             max_plane_sig = plane_sig    !Store final, high resolution values
326             best_fit_vect = norm         !of best fit pole and significance
327         END IF
328     END DO
329 END DO
330
331 ELSE
332
333 DO i = 1, 11
334     DO j = 1, 11
335
336         beta_set = pole_beta + 2.e0 * REAL(j-6) * (0.15e0) * (pi/180.e0)
337         alpha_set = pole_alpha + 2.e0 * REAL(i-6) * (0.15e0) * (pi/180.e0) * (1.e0/cos(beta_set))
338
339         norm = (/ 0.e0, 0.e0, 1.e0 /)
340
341         CALL Rotate
342         norm = MATMUL(y_rot,norm)
343         norm = MATMUL(x_rot,norm)
344
345         plane_sig = 0.d0
346         rms = 0.d0
347         DO k = 1, nsats    !RMS Calculation
348             planeDist = norm(1)*pos(1,k) + norm(2)*pos(2,k) + norm(3)*pos(3,k)
349             rms = rms + (planeDist)**2
350         END DO
351         rms = SQRT(rms/nsats)
352         plane_sig = LOG10(rms)
353         IF (plane_sig .lt. max_plane_sig) THEN    !Most significant plane has lowest rms
354             max_plane_sig = plane_sig    !Store final, high resolution values
355             best_fit_vect = norm         !of best fit pole and significance
356         END IF
357     END DO
358 END DO
359
360 END IF
361
362 END SUBROUTINE MaxSigFind
363
364 !-------------------------------------------------------------------------------
365
366 SUBROUTINE Significance !Principal subroutine which generates distributions of the
367 USE Global              !plane fitting statistic (RMS in this case). The plane fitting
368 IMPLICIT NONE           !subroutine 'MaxSigFind' is called from this subroutine
369
370 !!||Determine best fit plane and significance for satellite
371 !!\/positions generated from best fit distances
372
373 m31_dist = 779.e0            !M31
374
375 Best_Sat_Dist(1) = 727.e0    !And I
376 Best_Sat_Dist(2) = 630.e0    !And II
377 Best_Sat_Dist(3) = 723.e0    !And III
378 Best_Sat_Dist(4) = 742.e0    !And V
```

```
379  Best_Sat_Dist(5) = 600.e0     !And IX
380  Best_Sat_Dist(6) = 670.e0     !And X
381  Best_Sat_Dist(7) = 763.e0     !And XI
382  Best_Sat_Dist(8) = 928.e0     !And XII
383  Best_Sat_Dist(9) = 760.e0     !And XIII
384  Best_Sat_Dist(10) = 793.e0    !And XIV
385  Best_Sat_Dist(11) = 626.e0    !And XV
386  Best_Sat_Dist(12) = 476.e0    !And XVI
387  Best_Sat_Dist(13) = 727.e0    !And XVII
388  Best_Sat_Dist(14) = 1214.e0   !And XVIII
389  Best_Sat_Dist(15) = 821.e0    !And XIX
390  Best_Sat_Dist(16) = 741.e0    !And XX
391  Best_Sat_Dist(17) = 827.e0    !And XXI
392  Best_Sat_Dist(18) = 920.e0    !And XXII
393  Best_Sat_Dist(19) = 748.e0    !And XXIII
394  Best_Sat_Dist(20) = 898.e0    !And XXIV
395  Best_Sat_Dist(21) = 736.e0    !And XXV
396  Best_Sat_Dist(22) = 754.e0    !And XXVI
397  Best_Sat_Dist(23) = 1255.e0   !And XXVII
398  Best_Sat_Dist(24) = 681.e0    !And XXX
399  Best_Sat_Dist(25) = 712.e0    !NGC147
400  Best_Sat_Dist(26) = 620.e0    !NGC185
401  Best_Sat_Dist(27) = 820.e0    !M33
402
403  DO i = 1, nsats
404     pos(1,i) = ABS(Best_Sat_Dist(i) * cos(theta(i)) * tan(xi(i)))  !Determine length of x vector for each satellite
405     IF (xi(i) .lt. 0.e0) THEN                                      !
406        pos(1,i) = -1.e0 * pos(1,i)                                 !Determine if x is positive or negative
407     END IF                                                         !
408
409     pos(2,i) = ABS(Best_Sat_Dist(i) * sin(eta(i)))                 !Determine length of y vector for each satellite
410     IF (eta(i) .lt. 0.e0) THEN                                     !
411        pos(2,i) = -1.e0 * pos(2,i)                                 !Determine if y is positive or negative
412     END IF                                                         !
413
414     pos(3,i) = Best_Sat_Dist(i) * cos(theta(i)) - m31_dist         !Determine length and sign of z vector
415  END DO
416
417  CALL MaxSigFind
418
419  Actual_sig = max_plane_sig
420  Actual_bfv = best_fit_vect
421
422  alpha_set =  - (90.e0 - 12.5e0) * (pi/180.e0) !Rotate to bring back out of M31's inclination       !
423  gamma_set =  + (90.e0 - 39.8e0) * (pi/180.e0) !angle and PA (i.e. to view from above the M31 pole) !
424                                                                                                     !Change
425  CALL Rotate                                                                                        !
426                                                                                                     !to
427  Actual_bfv = MATMUL(z_rot, Actual_bfv)  !Convert vectors back to how they would appear             !
428  Actual_bfv = MATMUL(x_rot, Actual_bfv)  !in M31 reference frame                                    !M31
429                                                                                                     !
430  gamma_set = 90.e0 * (pi/180.e0)         !                                                          !coordinate
431                                          !                                                          !
432  CALL Rotate                             !Additional rotation in M31 galactic longitude             !system
433                                          !                                                          !
434  Actual_bfv = MATMUL(z_rot, Actual_bfv)  !                                                          !
435
436  !!/\Determine best fit plane and significance for satellite
437  !!||positions generated from best fit distances
438
439  !!||Determine best fit plane and significance for "err_samp" samples of
```

```
!!\/possible satellite positions generated from sampled distances and plot

DO it = 1, err_samp

   CALL random_number(randnum)                          !Read in err_samps M31
   randnum = randnum * 2999999.e0 + 1.e0                !possible distances
   m31_dist = M31_Dist_PPD(NINT(randnum))               !to generate err_samps
                                                        !posible x,y,z coords

   DO i = 1, 27                                         !Read in err_samps
      CALL random_number(randnum)                       !possible distances
      randnum = randnum * 499999.e0 + 1.e0              !for each of the
      Sat_Dist_Drawn(i) = Sat_Dist(NINT(randnum),i)     !satellites to
   END DO                                               !generate err_samps
                                                        !posible x,y,z coords

   DO i = 1, 27
      pos(1,i) = ABS(Sat_Dist_Drawn(i) * cos(theta(i)) * tan(xi(i))) !Determine length of x vector for each satellite
      IF (xi(i) .lt. 0.e0) THEN                                      !
         pos(1,i) = -1.e0 * pos(1,i)                                 !Determine if x is positive or negative
      END IF                                                         !

      pos(2,i) = ABS(Sat_Dist_Drawn(i) * sin(eta(i)))                !Determine length of y vector for each satellite
      IF (eta(i) .lt. 0.e0) THEN                                     !
         pos(2,i) = -1.e0 * pos(2,i)                                 !Determine if y is positive or negative
      END IF                                                         !

      pos(3,i) = Sat_Dist_Drawn(i) * cos(theta(i)) - m31_dist        !Determine length and sign of z vector
   END DO

   CALL MaxSigFind

   alpha_set =  - (90.e0 - 12.5e0) * (pi/180.e0) !Rotate to bring back out of M31's inclination       !
   gamma_set =  + (90.e0 - 39.8e0) * (pi/180.e0) !angle and PA (i.e. to view from above the M31 pole) !
                                                                                                      !Change
   CALL Rotate                                                                                        !
                                                                                                      !to
   best_fit_vect = MATMUL(z_rot, best_fit_vect)    !Convert vectors back to how they would appear     !
   best_fit_vect = MATMUL(x_rot, best_fit_vect)    !in M31 reference frame                            !M31
                                                                                                      !
   gamma_set = 90.e0 * (pi/180.e0)                 !                                                 !coordinate
                                                   !                                                 !
   CALL Rotate                                     !Additional rotation in M31 galactic longitude    !system
                                                   !                                                 !
   best_fit_vect = MATMUL(z_rot, best_fit_vect)    !                                                 !

   CALL Theta_Phi(best_fit_vect(1), best_fit_vect(2), best_fit_vect(3))

   err_sig(it) = max_plane_sig
   WRITE (14, '(7F16.5)') REAL(it), err_sig(it), theta_coord, phi_coord, best_fit_vect(1), best_fit_vect(2), best_fit_vect(3)

END DO

string2 = TRIM(ADJUSTL(folder)) // '/err_samp_PPD_' // TRIM(ADJUSTL(subsize)) // '_sats.ps/CPS'

CALL HistoPlot(err_samp,101,err_sig,'LOG10(Minimum_RMS)','Probability', TRIM(ADJUSTL(string2)), .true.)

WRITE (command,*) 'convert_-rotate_90_' // TRIM(ADJUSTL(folder)) // '/err_samp_PPD_' // TRIM(ADJUSTL(subsize)) // '_sats.ps_' &
                                        // TRIM(ADJUSTL(folder)) // '/err_samp_PPD_' // TRIM(ADJUSTL(subsize)) // '_sats.jpg'
call system(command)
```

```
!!/\Determine best fit plane and significance for "err_samp" samples of
!!||possible satellite positions generated from sampled distances and plot


!!||Determine best fit plane and significance for "nit"
!!\/random realizations of satellite positions and plot

min_signif = 1000.e0

DO it = 1, nit

   CALL RandomPoints

   CALL MaxSigFind

   IF (max_plane_sig .lt. min_signif) THEN
      min_signif = max_plane_sig
   END IF

   alpha_set =  - (90.e0 - 12.5e0) * (pi/180.e0) !Rotate to bring back out of M31's inclination       !
   gamma_set =  + (90.e0 - 39.8e0) * (pi/180.e0) !angle and PA (i.e. to view from above the M31 pole) !
                                                                                                      !Change
   CALL Rotate                                                                                        !
                                                                                                      !to
   best_fit_vect = MATMUL(z_rot, best_fit_vect)    !Convert vectors back to how they would appear     !
   best_fit_vect = MATMUL(x_rot, best_fit_vect)    !in M31 reference frame                            !M31
                                                                                                      !
   gamma_set = 90.e0 * (pi/180.e0)                 !                                                 !coordinate
                                                   !                                                 !
   CALL Rotate                                     !Additional rotation in M31 galactic longitude    !system
                                                   !                                                 !
   best_fit_vect = MATMUL(z_rot, best_fit_vect)    !                                                 !

   CALL Theta_Phi(best_fit_vect(1), best_fit_vect(2), best_fit_vect(3))

   sig(it) = max_plane_sig
   WRITE (13, '(7F16.5)') REAL(it), sig(it), theta_coord, phi_coord, best_fit_vect(1), best_fit_vect(2), best_fit_vect(3)

END DO

string2 = TRIM(ADJUSTL(folder)) // '/sig_PPD_' // TRIM(ADJUSTL(subsize)) // '_sats.ps/CPS'

CALL HistoPlot(nit,101,sig,'LOG10(Minimum_RMS)','Probability', TRIM(ADJUSTL(string2)), .true.)

WRITE (command,*) 'convert_-rotate_90_' // TRIM(ADJUSTL(folder)) // '/sig_PPD_' // TRIM(ADJUSTL(subsize)) // '_sats.ps_' &
                                        // TRIM(ADJUSTL(folder)) // '/sig_PPD_' // TRIM(ADJUSTL(subsize)) // '_sats.jpg'
call system(command)

!!/\Determine best fit plane and significance for "nit"
!!||random realizations of satellite positions and plot

END SUBROUTINE Significance

!-----------------------------------------------------------------------

SUBROUTINE HistoPlot(nval, data_hist_bins, data, xlabel, ylabel, device, normalize)
IMPLICIT NONE
!
!*******Created 24 Feb 2012********
!
!INTEGER nval = number of data points in histogram
```

```
562 !INTEGER data_hist_bins = number of bins in histogram
563 !REAL data(nval) = The array containing the data
564 !CHARACTER xlabel = Label of x-axis of histogram
565 !CHARACTER ylabel = Label of y-axis of histogram
566 !CHARACTER device = The plotting device ('?' if unsure)
567 !LOGICAL normalize = .true. if histogram is to be
568 !normalized, else set to .false.
569 !
570 !Uses PGPLOT
571 !
572
573 INTEGER :: data_hist_bins, nval, it_num
574 REAL :: bw, data(nval), data_hist(data_hist_bins,2), data_min, data_max
575 CHARACTER(LEN=*) :: xlabel, ylabel, device
576 LOGICAL :: normalize
577
578 data_hist = 0.e0
579
580 data_min = MINVAL(data) ; data_max = MAXVAL(data)
581
582 bw = (data_max - data_min)/(REAL(data_hist_bins) - 1.e0)
583
584 DO it_num = 1, data_hist_bins
585    data_hist(it_num,1) = data_min + REAL(it_num-1) * bw
586 END DO
587
588 DO it_num = 1, nval
589    data_hist(NINT((data(it_num) - data_min)/bw) + 1,2) = &
590    data_hist(NINT((data(it_num) - data_min)/bw) + 1,2) + 1.e0
591 END DO
592
593 IF (normalize) THEN
594 data_hist(:,2) = data_hist(:,2) / SUM(data_hist(:,2))
595 END IF
596
597 CALL pgbegin(0,TRIM(ADJUSTL(device)),1,1)
598
599 CALL pgenv(MINVAL(data, mask = data .ne. 0.), &
600      MAXVAL(data, mask = data .ne. 0.), &
601       0., 1.1*MAXVAL(data_hist(:,2)), 0, 0)
602
603 CALL pgbin (data_hist_bins, data_hist(:,1), data_hist(:,2), .true.)
604 CALL pglab(TRIM(ADJUSTL(xlabel)), TRIM(ADJUSTL(ylabel)), '')
605
606 CALL pgend
607
608 END SUBROUTINE HistoPlot
609
610 !-----------------------------------------------------------------------
611
612 !logical function in_poly(x,y,np,xp,yp) omitted - see MF_TRGB.f95 in preceding appendix
613 !real function fimag(x0,xs,xe,y0,ys,ye) omitted - see MF_TRGB.f95 in preceding appendix
614
615 !-----------------------------------------------------------------------
616
617 SUBROUTINE SampledDist !Read in samples from the distance distributions of
618 USE Global             !M31 and its satellites
619 IMPLICIT NONE
620
621 DOUBLE PRECISION :: sla_DSEP
622
```

```
623  OPEN (unit = 11, file = './AndromedaIe/other_plots2/Sampled_MWy_Distances.dat', status = 'old')
624  OPEN (unit = 12, file = './AndromedaIIe/other_plots2/Sampled_MWy_Distances.dat', status = 'old')
625  OPEN (unit = 13, file = './AndromedaIIIe/other_plots2/Sampled_MWy_Distances.dat', status = 'old')
626  OPEN (unit = 14, file = './AndromedaVe/other_plots2/Sampled_MWy_Distances.dat', status = 'old')
627  OPEN (unit = 15, file = './AndromedaIXe/other_plots2/Sampled_MWy_Distances.dat', status = 'old')
628  OPEN (unit = 16, file = './AndromedaXe/other_plots2/Sampled_MWy_Distances.dat', status = 'old')
629  OPEN (unit = 17, file = './AndromedaXIe/other_plots2/Sampled_MWy_Distances.dat', status = 'old')
630  OPEN (unit = 18, file = './AndromedaXIIe/other_plots2/Sampled_MWy_Distances.dat', status = 'old')
631  OPEN (unit = 19, file = './AndromedaXIIIe/other_plots2/Sampled_MWy_Distances.dat', status = 'old')
632  OPEN (unit = 20, file = './AndromedaXIVe/other_plots2/Sampled_MWy_Distances.dat', status = 'old')
633  OPEN (unit = 21, file = './AndromedaXVe/other_plots2/Sampled_MWy_Distances.dat', status = 'old')
634  OPEN (unit = 22, file = './AndromedaXVIe/other_plots2/Sampled_MWy_Distances.dat', status = 'old')
635  OPEN (unit = 23, file = './AndromedaXVIIe/other_plots2/Sampled_MWy_Distances.dat', status = 'old')
636  OPEN (unit = 24, file = './AndromedaXVIIIe/other_plots2/Sampled_MWy_Distances.dat', status = 'old')
637  OPEN (unit = 25, file = './AndromedaXIXe/other_plots2/Sampled_MWy_Distances.dat', status = 'old')
638  OPEN (unit = 26, file = './AndromedaXXe/other_plots2/Sampled_MWy_Distances.dat', status = 'old')
639  OPEN (unit = 27, file = './AndromedaXXIe/other_plots2/Sampled_MWy_Distances.dat', status = 'old')
640  OPEN (unit = 28, file = './AndromedaXXIIe/other_plots2/Sampled_MWy_Distances.dat', status = 'old')
641  OPEN (unit = 29, file = './AndromedaXXIIIe/other_plots2/Sampled_MWy_Distances.dat', status = 'old')
642  OPEN (unit = 30, file = './AndromedaXXIVe/other_plots2/Sampled_MWy_Distances.dat', status = 'old')
643  OPEN (unit = 31, file = './AndromedaXXVe/other_plots2/Sampled_MWy_Distances.dat', status = 'old')
644  OPEN (unit = 32, file = './AndromedaXXVIe/other_plots2/Sampled_MWy_Distances.dat', status = 'old')
645  OPEN (unit = 33, file = './AndromedaXXVIIe/other_plots2/Sampled_MWy_Distances.dat', status = 'old')
646  OPEN (unit = 34, file = './AndromedaXXXe/other_plots2/Sampled_MWy_Distances.dat', status = 'old')
647  OPEN (unit = 35, file = './NGC147e_outer/other_plots2/Sampled_MWy_Distances.dat', status = 'old')
648  OPEN (unit = 36, file = './NGC185e_outer/other_plots2/Sampled_MWy_Distances.dat', status = 'old')
649  OPEN (unit = 37, file = './M33e/other_plots2/Sampled_MWy_Distances.dat', status = 'old')
650  OPEN (unit = 38, file = './M31e/other_plots/M31_Distance_PPD.dat', status = 'old')
651
652  !-----------------------------------------
653
654  i = 0
655
656  DO WHILE (.TRUE.)
657
658  i = i + 1
659
660  IF (i .gt. 500000) THEN
661     exit
662  END IF
663
664  READ (11, *, IOSTAT = ios) Sat_Dist(i,1)
665
666  IF (ios == -1) THEN
667     i = i - 1
668     exit
669  ELSE IF (ios .gt. 0) THEN
670     WRITE (*,*) i
671     i=i-1
672     cycle
673  END IF
674
675  END DO
676
677  !-----------------------------------------
678
679  !Files 12 through 36 read in as shown for
680  !file 11 above and 37 below
681
682  !-----------------------------------------
683
```

```
684  i = 0
685
686  DO WHILE (.TRUE.)
687
688  i = i + 1
689
690  IF (i .gt. 500000) THEN
691     exit
692  END IF
693
694  READ (37, *, IOSTAT = ios) Sat_Dist(i,27)
695
696  IF (ios == -1) THEN
697     i = i - 1
698     exit
699  ELSE IF (ios .gt. 0) THEN
700     WRITE (*,*) i
701     i=i-1
702     cycle
703  END IF
704
705  END DO
706
707  !-----------------------------------------
708
709  i = 0
710
711  DO WHILE (.TRUE.)
712
713  i = i + 1
714
715  IF (i .gt. 3000000) THEN
716     exit
717  END IF
718
719  READ (38, *, IOSTAT = ios) M31_Dist_PPD(i)
720
721  IF (ios == -1) THEN
722     i = i - 1
723     exit
724  ELSE IF (ios .gt. 0) THEN
725     WRITE (*,*) i
726     i=i-1
727     cycle
728  END IF
729
730  END DO
731
732  !-----------------------------------------
733
734  !||Tangent Plane projection angles (xi,eta)
735  !\/for each satellite
736  Sat_Pos(1,:) = (/ 0.577417966865471, -3.2314283795568426 /)
737  Sat_Pos(2,:) = (/ 7.122225162668977, -7.592252808099334 /)
738  Sat_Pos(3,:) = (/ -1.457960734448612, -4.765087242682244 /)
739  Sat_Pos(4,:) = (/ 4.67571438294161, 6.595921326738737 /)
740  Sat_Pos(5,:) = (/ 1.8486898643911536, 1.9594865642747519 /)
741  Sat_Pos(6,:) = (/ 4.243395063076322, 3.7004040941268976 /)
742  Sat_Pos(7,:) = (/ 0.7517384673615299, -7.5056115753940515 /)
743  Sat_Pos(8,:) = (/ 0.979245230608749, -6.921767335891222 /)
744  Sat_Pos(9,:) = (/ 1.9303864363068866, -8.301405961956007 /)
```

```
745   Sat_Pos(10,:) = (/ 1.9616750556905713, -11.722713906650105 /)
746   Sat_Pos(11,:) = (/ 6.234635220222043, -2.8843033494675128 /)
747   Sat_Pos(12,:) = (/ 3.585309428238602, -8.89395559219651 /)
748   Sat_Pos(13,:) = (/ -1.0085393602174035, 3.0650628218938585 /)
749   Sat_Pos(14,:) = (/ -7.181147930491751, 4.280747404302224 /)
750   Sat_Pos(15,:) = (/ -4.784510400133266, -6.109702992340868 /)
751   Sat_Pos(16,:) = (/ -7.269591542188213, -5.840766998597464 /)
752   Sat_Pos(17,:) = (/ -8.888840848373246, 1.8332101560489324 /)
753   Sat_Pos(18,:) = (/ 10.246238044162409, -12.92831734742824 /)
754   Sat_Pos(19,:) = (/ 9.147466892705275, -1.961574445626079 /)
755   Sat_Pos(20,:) = (/ 6.211342279324403, 5.469930868721202 /)
756   Sat_Pos(21,:) = (/ -2.1573196501753253, 5.648080455771416 /)
757   Sat_Pos(22,:) = (/ -3.203360921263025, 6.772438490131299 /)
758   Sat_Pos(23,:) = (/ -0.9036995038298262, 4.117443311518852 /)
759   Sat_Pos(24,:) = (/ -1.0080494766702268, 8.448293881357133 /)
760   Sat_Pos(25,:) = (/ -1.5924793331852394, 7.303792725460755 /)
761   Sat_Pos(26,:) = (/ -0.6317798640615756, 7.108624424318093 /)
762   Sat_Pos(27,:) = (/ 11.273546933713943, -10.076848401834724 /)
763   !/\
764   !||
765
766   DO i = 1, nsats
767      xi(i) = Sat_Pos(i,1)
768      eta(i) = Sat_Pos(i,2)
769      xi(i) = xi(i) * (pi/180.e0)              !Convert angles from
770      eta(i) = eta(i) * (pi/180.e0)            !degrees to radians
771   END DO
772
773   DO i = 1, nsats                                                    !
774      xi_dble = xi(i) ; eta_dble = eta(i)                             !
775      CALL sla_DTP2S(xi_dble, eta_dble, 0.d0, 0.d0, RA, DEC)          !Convert tangent plane
776      IF (xi_dble .lt. 0.d0) then                                     !projection angles into
777         RA = RA - (2.e0 * pi)                                        !their true angles using
778      END IF                                                          !sla_DTP2S
779      xi(i) = RA                                                      !
780      eta(i) = DEC                                                    !
781   END DO
782
783   DO i = 1, nsats                                        !Find the true angle
784      xi_dble = xi(i)                                     !theta - the angle on
785      eta_dble = eta(i)                                   !the sky between M31
786      theta(i) = sla_DSEP(0.d0, 0.d0, xi_dble, eta_dble) !and the object
787   END DO                                                 !(uses sla_DSEP)
788
789   END SUBROUTINE SampledDist
790
791   !-------------------------------------------------------------------------
792
793   SUBROUTINE BorderGet    !Read in the 134 points in xi and eta defining the PAndAS
794   USE Global              !Survey Border. These points are used to reject satellites
795   IMPLICIT NONE           !that fall out of bounds (see RandomPoints Subroutine)
796
797   OPEN (unit = 40, file = '../SurveyArea/Border_Coords_XiEta.dat', status = 'old')
798
799   DO i = 1, 134
800      READ (40, *, IOSTAT = ios) SAP_xi(i), SAP_eta(i)
801   END DO
802
803   END SUBROUTINE BorderGet
804
805   !-------------------------------------------------------------------------
```

Program: Alternative Plane Fitting Code Segments

Creation Date: First versions Feb 2012

Relevant Section: Ch. 5; Paper III §3.1, §3.2

Notes: Presented here are four separate code segments, each one performing the calculation of the goodness of fit of a tested plane. The first uses the root-mean-square (RMS) of the perpendicular distances of the satellites from the plane. This is the one used in the 'MaxSigFind' subroutine presented in *PlaneSigRMS.f95* (p. 244). The other code segments are alternatives to this RMS code segment. The second code segment calculates the goodness of fit of a given plane by summing the absolute values of the perpendicular distances of each satellite from the plane. The third uses a maximum likelihood approach and replaces the zero-thickness plane with a Gaussian distribution of some (to be determined) thickness. The fourth and final code segment serves a different purpose to the previous three in that it finds the plane of maximum asymmetry. It seeks the plane which can divide the sample most unequally. Note that some other minor modifications to the code of *PlaneSigRMS.f95* would be required for correct operation. These segments are intended to illustrate precisely how the various forms of plane fitting utilized in Paper III are implemented.

```
1
2  !--------------------------For Plane Fitting using RMS--------------------------------
3  !As used in the included version of the 'MaxSigFind' subroutine in 'PlaneSigRMS.f95'
4  !Where an alternate plane fitting statistic is used, this segment of code should be
5  !replaced with one of the versions below for each of the four times it appears in the
6  !'MaxSigFind' subroutine. Note that other minor code variations are necessary
7  !but these are included to show the way the actual plane fitting statistic is
8  !calculated in each case.
9
10       plane_sig = 0.d0
11       rms = 0.d0
12       DO k = 1, nsats    !RMS Calculation
13          planeDist = norm(1)*pos(1,k) + norm(2)*pos(2,k) + norm(3)*pos(3,k)
14          rms = rms + (planeDist)**2
15       END DO
16       rms = SQRT(rms/nsats)
17       plane_sig = LOG10(rms)
18       IF (plane_sig .lt. max_plane_sig) THEN    !Most significant plane has lowest rms
19          max_plane_sig = plane_sig    !Store approx, low resolution values
20          best_fit_vect = norm         !of best fit pole and significance
21          pole_alpha = alpha_set    !Store best fit pole for
22          pole_beta = beta_set      !high resolution search
23       END IF
24
25 !--------For Fitting using Sum of Absolute Values of Satellite Distances from Plane--------
26
27       plane_sig = 0.d0
28       ab_val = 0.d0
29       DO k = 1, nsats    !Absolute Value of distance sum Calculation
```

```
30            planeDist = abs(norm(1)*pos(1,k) + norm(2)*pos(2,k) + norm(3)*pos(3,k))
31            ab_val = ab_val + planeDist
32          END DO
33          plane_sig = LOG10(ab_val)
34          IF (plane_sig .lt. max_plane_sig) THEN   !Most significant plane has lowest AbVal
35            max_plane_sig = plane_sig   !Store approx, low resolution values
36            best_fit_vect = norm        !of best fit pole and significance
37            pole_alpha = alpha_set   !Store best fit pole for
38            pole_beta = beta_set     !high resolution search
39          END IF
40
41  !--------------------For Maximum Likelihood Fitting of 'Gaussian Plane'-------------------
42  !Replace max_plane_sig = 9999999.e0 with max_plane_sig = -9999999.e0 as initial value
43
44          DO s = 1, 30
45          sigma = REAL(s)*5.d0
46          plane_sig = 0.d0
47          DO k = 1, nsats
48            planeDist = abs(norm(1)*pos(1,k) + norm(2)*pos(2,k) + norm(3)*pos(3,k))
49            like = exp(-(planeDist**2.d0)/(2.d0 * sigma ** 2.d0))/ (sigma * SQRT(2.d0 * pi))
50            IF (LOG10(like) .le. -9999.d0) THEN
51              plane_sig = plane_sig - 9999.d0
52            ELSE
53              plane_sig = plane_sig + LOG10(like)
54            END IF
55          END DO
56          IF (plane_sig .gt. max_plane_sig) THEN !Significance Calculation
57            max_plane_sig = plane_sig   !Store approx, low resolution values
58            best_fit_vect = norm        !of best fit pole, significance
59            best_fit_sigma = sigma      !and Gaussian one sigma
60            pole_alpha = alpha_set   !Store best fit pole for
61            pole_beta = beta_set     !high resolution search
62          END IF
63          END DO
64
65  !---------------------------For Fitting Maximum Asymmetry Plane-----------------------------
66  !Replace max_plane_sig = 9999999.e0 with max_plane_asymm = 0.e0 as initial value
67
68          pos_side = 0.e0                                                    !
69          DO k = 1, nsats                                                    !Count satellites on
70            planeDist = norm(1)*pos(1,k) + norm(2)*pos(2,k) + norm(3)*pos(3,k)  !one side of plane.
71            IF (planeDist .ge. 0.d0) THEN                                    !Satellites on the
72              pos_side = pos_side + 1.e0                                     !other side of the
73            END IF                                                           !plane is known
74          END DO                                                             !automatically from
75          neg_side = nsats - pos_side                                        !total number of sats.
76          IF (pos_side .gt. neg_side) THEN   !
77            plane_asymm = pos_side          !Calculate asymmetry, defined as
78          ELSE                              !the number of satellites on the side
79            plane_asymm = neg_side          !with the most satellites
80          END IF                            !
81          IF (plane_asymm .gt. max_plane_asymm) THEN
82            max_plane_asymm = plane_asymm !If a higher asymmetry plane is
83            max_asymm_vect = norm         !found, note normal vector of that
84            ma_pos_side = pos_side        !plane as well as the satellite
85            ma_neg_side = neg_side        !counts on each side.
86            pole_alpha = alpha_set   !Store highest asymmetry pole for
87            pole_beta = beta_set     !high resolution search
88          END IF
89
90  !-------------------------------------------------------------------------------------------
```

Program: Subroutines for Processing Satellite Subsets

Creation Date: 18 July 2012 (first version 26 Apr 2012) Many modifications.

Relevant Section: Ch. 5; Paper III §3.3

Notes: The three subroutines presented here essentially modify *PlaneSigRMS.f95* (p. 244) so that it can process every possible combination of a given number of satellites rather than just the full sample. The 'Combinations' subroutine steps through every possible combination of the specified size (sizes of 3 through 7 satellites are shown) and for each one calls the 'Significance' subroutine which samples positions for each satellite in the combination and then calls 'MaxSigFind' to perform the plane fitting. Note that each possible combination of satellites is sampled 'err_samps' (currently set to 100 as used in §3.3 of Paper III) times so as to account for the uncertainties in the satellite distances. Also, this code compresses storage file size by indexing each possible pole position and then recording the number of instances of that pole as well as the number of times each satellite contributes to that pole (information which is used by *pole_vicinity_counts_satid_w.f95* - p. 274).

```
1  !Code Segments for testing *all* combinations of a particular number (nsatsub)
2  !of satellites possible from the total sample (total sample is 25 satellites here
3  !as NGC147/NGC185/AndXXX are treated as a single point).
4
5  !See 'PlaneSigRMS.f95' for all subroutines called that are not incuded
6  !-------------------------------------------------------------------------------
7
8  SUBROUTINE Combinations !Finds the best fit plane to every possible combination of 'nsatsub'
9  USE Global              !satellites. The pole of each combination's best fit plane is converted
10 IMPLICIT NONE           !to M31-centric lat. and long. and stored for plotting as a pole plot
11                         !map on an aitoff-hammer projection.
12 subsetcounts = 0
13 RMSmin = 9999999.e0
14
15 IF (nsatsub .eq. 3) THEN
16
17 DO s1 = 1, nsats-2
18 DO s2 = s1+1, nsats-1
19 DO s3 = s2+1, nsats
20    satholder(1) = s1
21    satholder(2) = s2
22    satholder(3) = s3
23    subsetcounts = subsetcounts + 1
24    CALL Significance
25    write(13) Actual_sig, theta_coord, phi_coord, Actual_bfv(1), Actual_bfv(2), Actual_bfv(3), &
26              s1, s2, s3
27 END DO
28 END DO
29 END DO
30
31 !-------------------------------------
32
33 ELSE IF (nsatsub .eq. 4) THEN
```

```
34
35  DO s1 = 1, nsats-3
36  DO s2 = s1+1, nsats-2
37  DO s3 = s2+1, nsats-1
38  DO s4 = s3+1, nsats
39      satholder(1) = s1
40      satholder(2) = s2
41      satholder(3) = s3
42      satholder(4) = s4
43      subsetcounts = subsetcounts + 1
44     CALL Significance
45      write(13) Actual_sig, theta_coord, phi_coord, Actual_bfv(1), Actual_bfv(2), Actual_bfv(3), &
46                  s1, s2, s3, s4
47  END DO
48  END DO
49  END DO
50  END DO
51
52  !-------------------------------------
53
54  ELSE IF (nsatsub .eq. 5) THEN
55
56  DO s1 = 1, nsats-4
57  DO s2 = s1+1, nsats-3
58  DO s3 = s2+1, nsats-2
59  DO s4 = s3+1, nsats-1
60  DO s5 = s4+1, nsats
61      satholder(1) = s1
62      satholder(2) = s2
63      satholder(3) = s3
64      satholder(4) = s4
65      satholder(5) = s5
66      subsetcounts = subsetcounts + 1
67     CALL Significance
68      write(13) Actual_sig, theta_coord, phi_coord, Actual_bfv(1), Actual_bfv(2), Actual_bfv(3), &
69                  s1, s2, s3, s4, s5
70  END DO
71  END DO
72  END DO
73  END DO
74  END DO
75
76  !-------------------------------------
77
78  ELSE IF (nsatsub .eq. 6) THEN
79
80  DO s1 = 1, nsats-5
81  DO s2 = s1+1, nsats-4
82  DO s3 = s2+1, nsats-3
83  DO s4 = s3+1, nsats-2
84  DO s5 = s4+1, nsats-1
85  DO s6 = s5+1, nsats
86      satholder(1) = s1
87      satholder(2) = s2
88      satholder(3) = s3
89      satholder(4) = s4
90      satholder(5) = s5
91      satholder(6) = s6
92      subsetcounts = subsetcounts + 1
93     CALL Significance
94      write(13) Actual_sig, theta_coord, phi_coord, Actual_bfv(1), Actual_bfv(2), Actual_bfv(3), &
```

```
95                  s1, s2, s3, s4, s5, s6
96  END DO
97  END DO
98  END DO
99  END DO
100 END DO
101 END DO
102
103 !-------------------------------------
104
105 ELSE IF (nsatsub .eq. 7) THEN
106
107 DO s1 = 1, nsats-6
108 DO s2 = s1+1, nsats-5
109 DO s3 = s2+1, nsats-4
110 DO s4 = s3+1, nsats-3
111 DO s5 = s4+1, nsats-2
112 DO s6 = s5+1, nsats-1
113 DO s7 = s6+1, nsats
114    satholder(1) = s1
115    satholder(2) = s2
116    satholder(3) = s3
117    satholder(4) = s4
118    satholder(5) = s5
119    satholder(6) = s6
120    satholder(7) = s7
121    subsetcounts = subsetcounts + 1
122    CALL Significance
123    write(13) Actual_sig, theta_coord, phi_coord, Actual_bfv(1), Actual_bfv(2), Actual_bfv(3), &
124                 s1, s2, s3, s4, s5, s6, s7
125 END DO
126 END DO
127 END DO
128 END DO
129 END DO
130 END DO
131 END DO
132
133 !-------------------------------------
134
135 END IF
136
137 write(11,*) "Results:"
138 write(11,*) "Total number of combinations of", nsatsub, "satellites:", subsetcounts
139 write(11,*) "Best satellite combination:"
140 DO i = 1, nsatsub
141    write(11, *) best_sat_combo(i)
142 END DO
143 write(11,*) "Normal vector of best fit plane for this combination:"
144 write(11,*) best_sat_bfv(1), best_sat_bfv(2), best_sat_bfv(3)
145 CALL Theta_Phi(best_sat_bfv(1), best_sat_bfv(2), best_sat_bfv(3))
146 write(11,*) "Theta and phi of normal vector of best fit plane for this combination:"
147 write(11,*) "Theta =", theta_coord, "; Phi =", phi_coord
148 write(11,*) "LOG10(RMS) of best fit plane for this combination:", RMSmin
149
150 mode_counts = 0.e0        !||Every pole position possible is given an index and the number of times
151 DO i = 1, 31              !||a pole is recorded at that position is recorded. This greatly reduces
152    DO j = 1, 120          !||file storage size. The number of times a particular satellite contributes
153       DO k = 1, 15        !\/to a pole at each possible position is also recorded.
154          DO l = 1, 30
155    IF (poles_per_pos(i,j,k,l,6) .ne. 0.e0) THEN
```

```
156         WRITE (17, '(31F11.5)') poles_per_pos(i,j,k,l,1), poles_per_pos(i,j,k,l,2), poles_per_pos(i,j,k,l,3), &
157                                 poles_per_pos(i,j,k,l,4), poles_per_pos(i,j,k,l,5), poles_per_pos(i,j,k,l,6), &
158                                 poles_per_pos(i,j,k,l,7), poles_per_pos(i,j,k,l,8), poles_per_pos(i,j,k,l,9), &
159                                 poles_per_pos(i,j,k,l,10), poles_per_pos(i,j,k,l,11), poles_per_pos(i,j,k,l,12), &
160                                 poles_per_pos(i,j,k,l,13), poles_per_pos(i,j,k,l,14), poles_per_pos(i,j,k,l,15), &
161                                 poles_per_pos(i,j,k,l,16), poles_per_pos(i,j,k,l,17), poles_per_pos(i,j,k,l,18), &
162                                 poles_per_pos(i,j,k,l,19), poles_per_pos(i,j,k,l,20), poles_per_pos(i,j,k,l,21), &
163                                 poles_per_pos(i,j,k,l,22), poles_per_pos(i,j,k,l,23), poles_per_pos(i,j,k,l,24), &
164                                 poles_per_pos(i,j,k,l,25), poles_per_pos(i,j,k,l,26), poles_per_pos(i,j,k,l,27), &
165                                 poles_per_pos(i,j,k,l,28), poles_per_pos(i,j,k,l,29), poles_per_pos(i,j,k,l,30), &
166                                 poles_per_pos(i,j,k,l,31)
167       END IF
168       IF (poles_per_pos(i,j,k,l,6) .gt. mode_counts) THEN
169          mode_counts = poles_per_pos(i,j,k,l,6)
170          pos_mpc(1) = i ; pos_mpc(2) = j ; pos_mpc(3) = k ; pos_mpc(4) = l
171       END IF
172             END DO
173          END DO
174       END DO
175    END DO
176
177    x_mode = poles_per_pos(pos_mpc(1),pos_mpc(2),pos_mpc(3),pos_mpc(4),1)       !Most freq. normal vector x
178    y_mode = poles_per_pos(pos_mpc(1),pos_mpc(2),pos_mpc(3),pos_mpc(4),2)       !Most freq. normal vector y
179    z_mode = poles_per_pos(pos_mpc(1),pos_mpc(2),pos_mpc(3),pos_mpc(4),3)       !Most freq. normal vector z
180    theta_mode = poles_per_pos(pos_mpc(1),pos_mpc(2),pos_mpc(3),pos_mpc(4),4)   !Most freq. pole theta
181    phi_mode = poles_per_pos(pos_mpc(1),pos_mpc(2),pos_mpc(3),pos_mpc(4),5)     !Most freq. pole phi
182
183    write(11,*) "Normal vector of most frequently encountered plane:"
184    write(11,*) x_mode, y_mode, z_mode
185    write(11,*) "Theta and phi of most frequently encountered pole:"
186    write(11,*) "Theta =", theta_mode, "; Phi =", phi_mode
187    write(11,*) "Number of instances of this pole:", mode_counts
188
189    END SUBROUTINE Combinations
190
191    !-------------------------------------------------------------------------
192
193    SUBROUTINE Significance    !Finds RMS and pole of best fit plane to a given satellite combination.
194    USE Global                 !Does this for 'err_samps' possible versions of the combination
195    IMPLICIT NONE              !using distances drawn from the respective satellite distance PPDs.
196
197       WRITE(16,*) "Combinations tested so far:", subsetcounts !Progress update
198
199    DO samp_it = 1, err_samps
200
201    CALL random_number(randnum)                          !Read one possible M31
202    randnum = randnum * 2999999.e0 + 1.e0                !distance to generate
203    m31_1K_dist = M31_Dist_PPD(NINT(randnum))            !one possible set of
204                                                         !x,y,z coords
205
206    DO i = 1, nsatsub                                                !Read in one possible
207       IF (satholder(i) .lt. 24) THEN                                !distance for each of the
208       CALL random_number(randnum)                                   !dwarf sph sats except
209       randnum = randnum * 499999.e0 + 1.e0                          !AND XXX to generate one
210       sat_1K_dist(satholder(i)) = Sat_Dist(NINT(randnum),satholder(i))!set of possible distances
211       ELSE IF (satholder(i) .eq. 24) THEN
212       DO j = 25, 26                                                 !Get possible distances
213          CALL random_number(randnum)                                !for NGC147 and NGC185
214          randnum = randnum * 499999.e0 + 1.e0                       !to combine into one point
215          sat_1K_dist(j) = Sat_Dist(NINT(randnum),j)                 !to represent the NGC147,
216       END DO                                                        !NGC185, AND XXX group
```

```
217     ELSE IF (satholder(i) .eq. 25) THEN
218        CALL random_number(randnum)                                  !Get a possible
219        randnum = randnum * 499999.e0 + 1.e0                         !distance for
220        sat_1K_dist(27) = Sat_Dist(NINT(randnum),27)                 !M33
221     END IF
222  END DO
223
224  DO i = 1, nsatsub          !Convert distances to 3D positions for:
225     IF (satholder(i) .lt. 24) THEN !A: All the dwarf spheroidal satellites except Andromeda XXX
226
227     pos(1,satholder(i)) = ABS(sat_1K_dist(satholder(i)) * cos(theta(satholder(i))) * tan(xi(satholder(i))))!Determine length of x
228     IF (xi(satholder(i)) .lt. 0.e0) THEN                                                                    !vector for each satellite
229        pos(1,satholder(i)) = -1.e0 * pos(1,satholder(i))                                                    !Determine if x is positive or negative
230     END IF                                                                                                  !
231
232     pos(2,satholder(i)) = ABS(sat_1K_dist(satholder(i)) * sin(eta(satholder(i)))) !Determine length of y vector for each satellite
233     IF (eta(satholder(i)) .lt. 0.e0) THEN                                         !
234        pos(2,satholder(i)) = -1.e0 * pos(2,satholder(i))                          !Determine if y is positive or negative
235     END IF                                                                        !
236
237     pos(3,satholder(i)) = sat_1K_dist(satholder(i)) * cos(theta(satholder(i))) - m31_1K_dist!Determine length and sign of z vector
238
239     ELSE IF (satholder(i) .eq. 24) THEN !B: The NGC147/NGC185/AND XXX subgroup
240     DO j = 25, 26
241
242     pos(1,j) = ABS(sat_1K_dist(j) * cos(theta(j)) * tan(xi(j)))!Determine length of x vector for each
243     IF (xi(j) .lt. 0.e0) THEN                                  !satellite vector for each satellite
244        pos(1,j) = -1.e0 * pos(1,j)                             !Determine if x is positive or negative
245     END IF                                                     !
246
247     pos(2,j) = ABS(sat_1K_dist(j) * sin(eta(j)))       !Determine length of y vector for each satellite
248     IF (eta(j) .lt. 0.e0) THEN                         !
249        pos(2,j) = -1.e0 * pos(2,j)                     !Determine if y is positive or negative
250     END IF                                             !
251
252     pos(3,j) = sat_1K_dist(j) * cos(theta(j)) - m31_1K_dist     !Determine length and sign of z vector
253
254     END DO
255     pos(:,24) = pos(:,25) + ((100.e0)**(0.2e0*0.2e0)) * pos(:,26)
256     pos(:,24) = pos(:,24)/(1.e0 + (100.e0)**(0.2e0*0.2e0))
257
258     ELSE IF (satholder(i) .eq. 25) THEN !C: M33
259
260     pos(1,25) = ABS(sat_1K_dist(27) * cos(theta(27)) * tan(xi(27)))!Determine length of x vector for each
261     IF (xi(27) .lt. 0.e0) THEN                                     !satellite vector for each satellite
262        pos(1,25) = -1.e0 * pos(1,25)                               !Determine if x is positive or negative
263     END IF                                                         !
264
265     pos(2,25) = ABS(sat_1K_dist(27) * sin(eta(27)))       !Determine length of y vector for each satellite
266     IF (eta(27) .lt. 0.e0) THEN                           !
267        pos(2,25) = -1.e0 * pos(2,25)                      !Determine if y is positive or negative
268     END IF                                                !
269
270     pos(3,25) = sat_1K_dist(27) * cos(theta(27)) - m31_1K_dist     !Determine length and sign of z vector
271
272     END IF
273
274  END DO
275
276  CALL MaxSigFind
277
```

```
278  Actual_sig = max_plane_sig
279  Actual_bfv = best_fit_vect
280
281  alpha_set =  - (90.e0 - 12.5e0) * (pi/180.e0) !Rotate to bring back out of M31's inclination       !
282  gamma_set =  + (90.e0 - 39.8e0) * (pi/180.e0) !angle and PA (i.e. to view from above the M31 pole) !
283                                                                                                     !Change
284  CALL Rotate                                                                                        !
285                                                                                                     !to
286  Actual_bfv = MATMUL(z_rot, Actual_bfv)           !Convert vectors back to how they would appear   !
287  Actual_bfv = MATMUL(x_rot, Actual_bfv)           !in M31 reference frame                          !M31
288                                                                                                     !
289  gamma_set = 90.e0 * (pi/180.e0)          !                                                       !coordinate
290                                           !                                                       !
291  CALL Rotate                              !Additional rotation in M31 galactic longitude          !system
292                                           !                                                       !
293  Actual_bfv = MATMUL(z_rot, Actual_bfv)   !                                                       !
294
295  CALL Theta_Phi(Actual_bfv(1), Actual_bfv(2), Actual_bfv(3))
296
297  !||Every pole position possible is given an index and the number of times
298  !||a pole is recorded at that position is recorded. This greatly reduces
299  !||file storage size. The number of times a particular satellite contributes
300  !\/to a pole at each possible position is also recorded.
301  poles_per_pos(best_pol_loc(1), best_pol_loc(2), best_pol_loc(3), best_pol_loc(4),1) = Actual_bfv(1)     !
302  poles_per_pos(best_pol_loc(1), best_pol_loc(2), best_pol_loc(3), best_pol_loc(4),2) = Actual_bfv(2)     !Update
303  poles_per_pos(best_pol_loc(1), best_pol_loc(2), best_pol_loc(3), best_pol_loc(4),3) = Actual_bfv(3)     !counts
304  poles_per_pos(best_pol_loc(1), best_pol_loc(2), best_pol_loc(3), best_pol_loc(4),4) = theta_coord       !at a
305  poles_per_pos(best_pol_loc(1), best_pol_loc(2), best_pol_loc(3), best_pol_loc(4),5) = phi_coord         !particular
306  poles_per_pos(best_pol_loc(1), best_pol_loc(2), best_pol_loc(3), best_pol_loc(4),6) = &                 !pole
307  poles_per_pos(best_pol_loc(1), best_pol_loc(2), best_pol_loc(3), best_pol_loc(4),6) + (1.e0/(10.e0 ** Actual_sig)) !
308  DO i = 1, nsatsub
309   poles_per_pos(best_pol_loc(1), best_pol_loc(2), best_pol_loc(3), best_pol_loc(4),6+satholder(i)) = &
310   poles_per_pos(best_pol_loc(1), best_pol_loc(2), best_pol_loc(3), best_pol_loc(4),6+satholder(i)) + (1.e0/(10.e0 ** Actual_sig))
311  END DO
312
313  IF (Actual_sig .lt. RMSmin) THEN
314     RMSmin = Actual_sig                   !
315     DO i = 1, nsatsub                     !Store best possible satellite
316        best_sat_combo(i) = satholder(i)   !combination encountered so far
317     END DO                                !(i.e. the combination within lowest
318     best_sat_bfv = Actual_bfv             !RMS of its best fit plane)
319  END IF                                   !
320
321  END DO
322
323  END SUBROUTINE Significance
324
325  !-------------------------------------------------------------------------
326
327  SUBROUTINE MaxSigFind  !Finds best fit plane for a satellite distribution by testing goodness of fit of each
328  USE Global             !tested plane. The poles of the tested planes are all approximately equi-distant, taking
329  IMPLICIT NONE          !into account the surface area of a shere as a function of latitude.
330                         !A low resolution run finds the approximate location of the best fit plane's pole and then
331                         !poles around this point are searched at higher resolution.
332  par_like = 0.e0
333  max_plane_sig = 9999999.e0
334
335  !|| Low resolution
336  !\/ plane tests
337  DO i = 1, 30
338
```

```
339      beta_set = REAL(i*3) * (pi/180.e0)
340
341      DO j = 1, NINT(120.e0 * cos(beta_set)) !The higher the latitude, the smaller the
342                                             !number of points
343          alpha_set = (REAL(j)/NINT(120.e0 * cos(beta_set))) * 360.e0 * (pi/180.e0)
344
345          norm = (/ 0.e0, 0.e0, 1.e0 /)
346
347          CALL Rotate
348          norm = MATMUL(y_rot,norm)
349          norm = MATMUL(x_rot,norm)
350
351          plane_sig = 0.d0
352          rms = 0.d0
353          DO k = 1, nsatsub    !RMS calculation
354              planeDist = norm(1)*pos(1,satholder(k)) + norm(2)*pos(2,satholder(k)) + norm(3)*pos(3,satholder(k))
355              rms = rms + (planeDist)**2
356          END DO
357          rms = SQRT(rms/nsatsub)
358          plane_sig = LOG10(rms)
359          IF (plane_sig .lt. max_plane_sig) THEN   !Most significant plane has lowest rms
360              max_plane_sig = plane_sig    !Store approx, low resolution values
361              best_fit_vect = norm         !of best fit pole and significance
362              pole_alpha = alpha_set   !Store best fit pole for
363              pole_beta = beta_set     !high resolution search
364              best_pol_loc(1) = i   !Used for cumulative
365              best_pol_loc(2) = j   !pole count
366          END IF
367      END DO
368  END DO
369
370  norm = (/ -1.e0, 0.e0, 0.e0 /) !Test at the actual pole (not included in above loop)
371
372  plane_sig = 0.d0
373  rms = 0.d0
374  DO k = 1, nsatsub    !RMS calculation
375      planeDist = norm(1)*pos(1,satholder(k)) + norm(2)*pos(2,satholder(k)) + norm(3)*pos(3,satholder(k))
376      rms = rms + (planeDist)**2
377  END DO
378  rms = SQRT(rms/nsatsub)
379  plane_sig = LOG10(rms)
380
381
382  !|| High resolution search
383  !\/ around best fit pole
384  IF (plane_sig .lt. max_plane_sig) THEN  !Condition not met unless the RMS at the actual pole
385                                          !was better than anywhere else in the low res search
386  best_pol_loc(1) = 31    !Used for cumulative
387  best_pol_loc(2) = 1     !pole count
388
389  max_plane_sig = plane_sig
390  best_fit_vect = norm
391  DO i = 1, 15
392
393      beta_set = (88.5e0 + (REAL(i)/10.e0)) * (pi/180.e0)
394
395      DO j = 1, NINT(1200.e0 * cos(beta_set)) !The higher the latitude, the smaller the
396                                              !number of points
397          alpha_set = (REAL(j)/NINT(1200.e0 * cos(beta_set))) * 360.e0 * (pi/180.e0)
398
399          norm = (/ 0.e0, 0.e0, 1.e0 /)
```

```
400
401         CALL Rotate
402         norm = MATMUL(y_rot,norm)
403         norm = MATMUL(x_rot,norm)
404
405         plane_sig = 0.d0
406         rms = 0.d0
407         DO k = 1, nsatsub    !RMS calculation
408             planeDist = norm(1)*pos(1,satholder(k)) + norm(2)*pos(2,satholder(k)) + norm(3)*pos(3,satholder(k))
409             rms = rms + (planeDist)**2
410         END DO
411         rms = SQRT(rms/nsatsub)
412         plane_sig = LOG10(rms)
413         IF (plane_sig .lt. max_plane_sig) THEN    !Most significant plane has lowest rms
414             max_plane_sig = plane_sig    !Store final, high resolution values
415             best_fit_vect = norm         !of best fit pole and significance
416             best_pol_loc(3) = i    !Used for cumulative
417             best_pol_loc(4) = j    !pole count
418         END IF
419     END DO
420 END DO
421
422 ELSE
423
424 DO i = 1, 11
425     DO j = 1, 11
426
427         beta_set = pole_beta + 2.e0 * REAL(j-6) * (0.15e0) * (pi/180.e0)
428         alpha_set = pole_alpha + 2.e0 * REAL(i-6) * (0.15e0) * (pi/180.e0) * (1.e0/cos(beta_set))
429
430         norm = (/ 0.e0, 0.e0, 1.e0 /)
431
432         CALL Rotate
433         norm = MATMUL(y_rot,norm)
434         norm = MATMUL(x_rot,norm)
435
436         plane_sig = 0.d0
437         rms = 0.d0
438         DO k = 1, nsatsub    !RMS calculation
439             planeDist = norm(1)*pos(1,satholder(k)) + norm(2)*pos(2,satholder(k)) + norm(3)*pos(3,satholder(k))
440             rms = rms + (planeDist)**2
441         END DO
442         rms = SQRT(rms/nsatsub)
443         plane_sig = LOG10(rms)
444         IF (plane_sig .lt. max_plane_sig) THEN    !Most significant plane has lowest rms
445             max_plane_sig = plane_sig    !Store final, high resolution values
446             best_fit_vect = norm         !of best fit pole and significance
447             best_pol_loc(3) = i    !Used for cumulative
448             best_pol_loc(4) = j    !pole count
449         END IF
450     END DO
451 END DO
452
453 END IF
454
455 END SUBROUTINE MaxSigFind
456
457 !---------------------------------------------------------------------------
```

Program: PlaneSigSubSets_RandReal4_noGroup.f95

Creation Date: 3 Oct 2012 (first version 26 Apr 2012) Many modifications.

Relevant Section: Ch. 5; Paper III §3.1, §3.2, §3.4

Notes: This program is designed specifically for finding the most planar combination of large subsets of satellites. It can only be used where we do not require a measurement for every possible subset (i.e. a pole distribution map). In this program, the 'MaxSigFind' subroutine is completely different to the version seen in *PlaneSigRMS.f95* (p. 244). It throws down 10,000 random planes and finds the closest 'nsatsub' (15 in this case) satellites to the tested plane out of the full sample (nsats = 27) and records the associated RMS. That combination which is fit with the lowest RMS is then taken to approximate the most planar sub set. Note that the 'RandomPoints' subroutine presented here is also substantially different to that in *PlaneSigRMS.f95* as it represents each satellite by a distance distribution containing 1,000 possible positions along the line of sight from Earth.

```
1  MODULE Global !Defines all variables used by BayesianTRGB
2  IMPLICIT NONE
3
4  INTEGER :: i, j, k, l, s, mm, ios, idum = -9999, it, nit
5  INTEGER :: ndata_max, nsats, nsatsub, subsetcounts
6  PARAMETER (ndata_max = 10000000)
7  PARAMETER (nsats = 27)
8  PARAMETER (nit = 10000)
9  PARAMETER (nsatsub = 15)
10 REAL*8 :: pi
11 PARAMETER (pi = ACOS(-1.e0))
12 REAL :: randnum, sig(nit), norm(3), best_fit_vect(3), best_fit_sigma
13 REAL :: pos(3,ndata_max), temp_pos(3)
14 REAL :: a(ndata_max), b(ndata_max), c(ndata_max), d(ndata_max)
15 REAL :: a_hist(201,2), b_hist(201,2), c_hist(201,2), d_hist(2001,2)
16 REAL :: logL, LikeA, LikeB, r, p(4), p_temp(4), min_sigma, max_sigma
17 REAL*8 :: sigma, planeDist, like, plane_sig, rms, min_rms, rms_average(nit)
18 REAL :: max_plane_sig, RMSmin, Actual_sig, Actual_bfv(3), best_sat_bfv(3), Actual_bfs
19 INTEGER :: dummy, sat_pick(27)
20 REAL :: Sat_Dist(500000,nsats), Sat_Pos(nsats,2), xi(nsats), eta(nsats), theta(nsats), M31_Dist_PPD(3000000)
21 REAL :: Sat_Dist_change(nsats), new_Earth_Dist, Sat_Dist_store
22 REAL :: art_xi(nsats), art_eta(nsats), art_theta(nsats)
23 REAL*8 :: RA, DEC, xi_dble, eta_dble
24 REAL :: xi_test, eta_test, theta_test, SAP_xi(134), SAP_eta(134), spotR
25 REAL :: Best_Sat_Dist(nsats)
26 REAL :: m31_dist
27 REAL :: alpha_set, beta_set, gamma_set, pole_alpha, pole_beta
28 REAL :: x_rot(3,3), y_rot(3,3), z_rot(3,3)
29 REAL :: par_like(180,6)
30 REAL :: theta_coord, phi_coord
31 REAL :: poles_per_pos(31,120,15,30,6) = 0.e0
32 INTEGER :: best_pol_loc(4)
33 INTEGER :: s1, s2, s3, s4, s5, s6, s7, s8, s9, s10, s11, s12, s13
34 INTEGER :: s14, s15, s16, s17, s18, s19, s20, s21, s22, s23, s24, s25, s26
35 INTEGER :: satholder(nsats), best_sat_combo(nsats), pos_mpc(4)
```

```
36  REAL :: x_mode, y_mode, z_mode, theta_mode, phi_mode, mode_counts
37  CHARACTER :: argv*30, folder*100, string*200, string2*200, command*200, subsize*3, itnum*5
38  LOGICAL :: new_sats
39  REAL :: closest_sats(nsatsub)
40  INTEGER :: closest_sats_id(nsatsub), best_sats(nsatsub), u
41
42  END MODULE Global
43
44  !-------------------------------------------------------------------------
45
46  PROGRAM PlaneSignificance     !Master program
47  USE Global
48  IMPLICIT NONE
49
50  WRITE (subsize,*) nsatsub
51
52  WRITE (folder,*) 'Plane_Stats_' // TRIM(ADJUSTL(subsize)) // '_sats_RandReal_weighted'!Create
53  WRITE (string,*) './' // TRIM(ADJUSTL(folder))                                       !primary
54                                                                                       !output
55  WRITE (command,*) 'mkdir ' // TRIM(ADJUSTL(folder))                                 !directory
56
57  CALL system(command)
58
59  CALL random_seed !Insure random seed for random numbers
60
61  CALL SampledDist !Get sampled
62  CALL FixedDist   !satellite distances
63  CALL BorderGet   !Get PAndAS survey boundary points
64
65  string2 = TRIM(ADJUSTL(folder)) // '/sat_pos.dat'          !Positions of satellites in
66  OPEN(12, file=TRIM(ADJUSTL(string2)), status = 'unknown')  !Random Realizations
67
68  string2 = TRIM(ADJUSTL(folder)) // '/RMS_' // TRIM(ADJUSTL(subsize)) // '_sats.dat' !Best Plane RMS
69  OPEN(13, file=TRIM(ADJUSTL(string2)), status = 'unknown')                           !output file
70
71  rms_average = 0.e0                                !
72                                                    !Principal loop which
73  DO it = 1, nit                                    !Generates 'nit'
74     CALL RandomPoints                              !random realizations
75     rms_average(it) = rms_average(it)/REAL(1000)   !and finds average RMS
76     WRITE (13, '(2F16.5)') REAL(it), rms_average(it) !for best fit plane of
77     CALL Flush(13) !Empty buffer                   !most planar satellite
78  END DO                                            !combination in each
79
80  string2 = TRIM(ADJUSTL(folder)) // '/sig_PPD.ps/CPS'
81
82  CALL HistoPlot(nit,101,REAL(rms_average),'RMS_(kpc)','Probability', TRIM(ADJUSTL(string2)), .true.)
83
84  WRITE (command,*) 'convert -rotate 90 ' // TRIM(ADJUSTL(folder)) // &
85                    '/sig_PPD.ps ' // TRIM(ADJUSTL(folder)) // &
86                    '/sig_PPD.jpg'
87
88  call system(command)
89
90  CLOSE(11) ; CLOSE(12) ; CLOSE(13) ; CLOSE(15) ; CLOSE(17)
91
92  END PROGRAM PlaneSignificance
93
94  !-------------------------------------------------------------------------
95
96  SUBROUTINE RandomPoints !Generates a random realization containing nsats satellites
```

```
97  USE Global              !Each satellite is represented by 1,000 samples of possible
98  IMPLICIT NONE           !positions along the line of sight from Earth.
99                          !This routine is different to that of the same name in
100                         !'PlaneSigRMS.f95' which includes only 1 possible position
101                         !for each artificial satellite
102
103 LOGICAL :: in_poly
104 DOUBLE PRECISION :: sla_DSEP
105
106 CALL random_number(randnum)
107 randnum = randnum * 2999999.e0 + 1.e0
108 m31_dist = M31_Dist_PPD(NINT(randnum))
109
110 DO i = 1, nsats
111
112  2 CALL random_number(randnum)
113    sat_pick(i) = 1 + NINT(randnum*REAL(nsats - 1))   !Draw a random satellite
114
115    CALL random_number(randnum)
116    randnum = randnum * 499999.e0 + 1.e0
117    Sat_Dist_store = Sat_Dist(NINT(randnum),sat_pick(i))
118    pos(1,i) = ABS(Sat_Dist_store * cos(theta(sat_pick(i))) * tan(xi(sat_pick(i)))) !Determine length of x vector for each satellite
119    IF (xi(sat_pick(i)) .lt. 0.e0) THEN                                             !
120       pos(1,i) = -1.e0 * pos(1,i)                                                  !Determine if x is positive or negative
121    END IF                                                                          !
122
123    pos(2,i) = ABS(Sat_Dist_store * sin(eta(sat_pick(i)))) !Determine length of y vector for each satellite
124    IF (eta(sat_pick(i)) .lt. 0.e0) THEN                                            !
125       pos(2,i) = -1.e0 * pos(2,i)                                                  !Determine if y is positive or negative
126    END IF                                                                          !
127
128
129
130    pos(3,i) = Sat_Dist_store * cos(theta(sat_pick(i))) - m31_dist                   !Determine length and sign of z vector
131
132
133    pos(3,i) = SQRT((pos(1,i)**2.e0) + (pos(2,i)**2.e0) + (pos(3,i)**2.e0)) !Rotate position vector to point
134    pos(1,i) = 0.e0 ; pos(2,i) = 0.e0                                       !along z-axis
135
136    CALL random_number(randnum)                    !
137    alpha_set = randnum * 360.e0 * (pi/180.e0)     !Pick random longitude
138    CALL random_number(randnum)            !Pick random latitude between 0 and 90 weighted
139    beta_set = ASIN(randnum)               !by area of a sphere as a function of latitude
140    CALL random_number(randnum)              !
141    IF (randnum .lt. 0.5e0) THEN             !Re-assign latitude as
142       beta_set = beta_set                   !
143    ELSE                                     !-1 * latitude in
144       beta_set = -beta_set                  !
145    END IF                                   !50% 0f cases
146
147    CALL Rotate
148    pos(:,i) = MATMUL(y_rot,pos(:,i))   !Rotate to the chosen
149    pos(:,i) = MATMUL(x_rot,pos(:,i))   !random angle
150
151    xi_test = ATAN(abs(pos(1,i))/(m31_dist + pos(3,i)))                     !Convert
152    IF (pos(1,i) .lt. 0.e0) THEN                                            !new random
153       xi_test = -xi_test                                                   !position
154    END IF                                                                  !vector
155    eta_test = ATAN(abs(pos(2,i))/SQRT(pos(1,i)**2 + (m31_dist + pos(3,i))**2))!into
156    IF (pos(2,i) .lt. 0.e0) THEN                                            !non t.p.
157       eta_test = -eta_test                                                 !eta and
```

```
158     END IF                                                                  !xi
159
160     art_xi(i) = xi_test                                     !
161     art_eta(i) = eta_test                                   !Store position on sky of
162     xi_dble = art_xi(i)                                     !randomly oriented satellite
163     eta_dble = art_eta(i)                                   !for 1000 samples.
164     art_theta(i) = sla_DSEP(0.d0, 0.d0, xi_dble, eta_dble)  !
165
166     new_Earth_Dist = abs(pos(2,i))/sin(abs(eta_test)) !Calculate new distance of sat from Earth after rotation
167
168     Sat_Dist_change(i) = new_Earth_Dist - Sat_Dist_store !Calculate the difference between the new and old
169                                                         !Earth distances for each satellite
170
171     RA = xi_test                                        !
172     DEC = eta_test                                      !Use sla_DS2TP
173                                                         !to convert true
174     CALL sla_DS2TP (RA, DEC, 0.d0, 0.d0, xi_dble, eta_dble, j)  !eta amd xi to
175                                                         !their tangent
176     xi_test = xi_dble * (180.e0/pi)                     !plane projections
177     eta_test = eta_dble * (180.e0/pi)                   !
178
179
180     IF (in_poly(xi_test,eta_test,134,SAP_xi,SAP_eta)) THEN                  !Re-generate
181                                                                             !the new
182     ELSE                                                                    !randomized
183         goto 2                                                              !satellite
184     END IF                                                                  !position if
185                                                                             !the current
186     spotR = ((xi_test*cos(51.9d0*pi/180.d0) + eta_test*sin(51.9d0*pi/180.d0))**2 / 6.25d0) + & !choice doesn't
187     ((xi_test*sin(51.9d0*pi/180.d0) - eta_test*cos(51.9d0*pi/180.d0))**2 / 1.d0)            !fall within
188                                                                             !the PAndAS
189     IF (spotR .le. 1.e0) THEN                                               !footprint
190         goto 2                                                              !as viewed
191     END IF                                                                  !from Earth
192 END DO
193 new_sats = .true.
194 CALL MaxSigFind
195 new_sats = .true.
196
197 DO j = 1, 999
198
199     CALL random_number(randnum)
200     randnum = randnum * 2999999.e0 + 1.e0
201     m31_dist = M31_Dist_PPD(NINT(randnum))
202
203     DO i = 1, nsats
204         CALL random_number(randnum)
205         randnum = randnum * 499999.e0 + 1.e0
206         Sat_Dist_store = Sat_Dist(NINT(randnum),sat_pick(i)) + Sat_Dist_change(i) !Adjust drawn Earth distance for new position
207         pos(1,i) = ABS(Sat_Dist_store * cos(art_theta(i)) * tan(art_xi(i))) !Determine length of x vector for each satellite
208         IF (art_xi(i) .lt. 0.e0) THEN                                   !
209             pos(1,i) = -1.e0 * pos(1,i)                                 !Determine if x is positive or negative
210         END IF                                                          !
211
212         pos(2,i) = ABS(Sat_Dist_store * sin(art_eta(i))) !Determine length of y vector for each satellite
213         IF (art_eta(i) .lt. 0.e0) THEN                                  !
214             pos(2,i) = -1.e0 * pos(2,i)                                 !Determine if y is positive or negative
215         END IF                                                          !
216
217         pos(3,i) = Sat_Dist_store * cos(art_theta(i)) - m31_dist         !Determine length and sign of z vector
218     END DO
```

```
    CALL MaxSigFind
END DO

END SUBROUTINE RandomPoints

!------------------------------------------------------------------------------

!'Rotate' Subroutine - See 'PlaneSigRMS.f95'

!------------------------------------------------------------------------------

SUBROUTINE MaxSigFind  !Finds the best fit plane through the *most planar* combination
USE Global             !of 'nsatsub' satellites. This is achieved by "throwing in" 10,000
IMPLICIT NONE          !random planes and determining the 'nsatsub' closest satellites to
                       !each plane and the associated RMS.
min_rms = 9999.e0

DO i = 1, 10000

   IF (new_sats) THEN
      closest_sats = 999.e0  !If finding the best fit combo each time
      closest_sats_id = 0    !reset these parameters
   ELSE
      closest_sats_id = best_sats
   END IF

   norm = (/ 0.e0, 0.e0, 1.e0 /)

   CALL random_number(randnum)                    !
   alpha_set = randnum * 360.e0 * (pi/180.e0)     !Pick random longitude
   CALL random_number(randnum)            !Pick random latitude between 0 and 90 weighted
   beta_set = ASIN(randnum)               !by area of a sphere as a function of latitude
   CALL random_number(randnum)              !
   IF (randnum .lt. 0.5e0) THEN             !Re-assign latitude as
      beta_set = beta_set                   !
   ELSE                                     !-1 * latitude in
      beta_set = -beta_set                  !
   END IF                                   !50% Of cases

   CALL Rotate
   norm(:) = MATMUL(y_rot,norm(:))   !Rotate to the chosen
   norm(:) = MATMUL(x_rot,norm(:))   !random angle

   IF (new_sats) THEN
   DO k = 1, nsats                                                              !
      planeDist = abs(norm(1)*pos(1,k) + norm(2)*pos(2,k) + norm(3)*pos(3,k)) !
      IF (planeDist .lt. MAXVAL(closest_sats)) THEN                           !Find the closest 'nsatsub'
         u = MAXLOC(closest_sats, DIM = 1)                                    !satellites to the currently
         closest_sats(u) = planeDist                                          !tested plane
         closest_sats_id(u) = k                                               !
      END IF                                                                  !
   END DO
   END IF
   rms = 0.d0                    !
   DO k = 1, nsatsub             !Measure the RMS for the plane based on the closest 'nsatsub' satellites
      planeDist = abs(norm(1)*pos(1,closest_sats_id(k)) + norm(2)*pos(2,closest_sats_id(k)) + norm(3)*pos(3,closest_sats_id(k)))
      rms = rms + (planeDist)**2 !
   END DO                        !
   rms = SQRT(rms/nsatsub)       !

   IF (rms .lt. min_rms) THEN     !
```

```
280         min_rms = rms                !If the RMS is the
281         best_sats = closest_sats_id  !lowst encountered so far
282       END IF                         !then store it
283     END DO                           !
284
285     rms_average(it) = rms_average(it) + min_rms  !min_rms is now a good approximation to the lowest possible for the tested sample
286
287     END SUBROUTINE MaxSigFind
288
289     !-------------------------------------------------------------------------------
290
291     !'Theta_Phi' Subroutine - See 'PlaneSigRMS.f95'
292
293     !-------------------------------------------------------------------------------
294
295     !'HistoPlot' Subroutine - See 'PlaneSigRMS.f95'
296
297     !-------------------------------------------------------------------------------
298
299     !logical function in_poly(x,y,np,xp,yp) omitted - see MF_TRGB.f95 in preceding appendix
300     !real function fimag(x0,xs,xe,y0,ys,ye) omitted - see MF_TRGB.f95 in preceding appendix
301
302     !-------------------------------------------------------------------------------
303
304     !'SampledDist' Subroutine - See 'PlaneSigRMS.f95'
305
306     !-------------------------------------------
307
308     !'FixedDist' Subroutine - See 'PlaneSigRMS.f95'
309
310     !The function of this subroutine is to read in the best fit satellite positions
311     !(as opposed to the positions generated from sampled satellite distances). This
312     !subroutine is not included specifically in 'PlaneSigRMS.f95' but it's functions
313     !are performed at the beginning of the 'Significance' subroutine and at the end
314     !of the 'SampledDist' subroutine
315
316     !-------------------------------------------------------------------------------
317
318     !'BorderGet' Subroutine - See 'PlaneSigRMS.f95'
319
320     !-------------------------------------------------------------------------------
```

Program: pole_vicinity_counts_satid_w.f95

Creation Date: 24 June 2012

Relevant Section: Ch. 5; Paper III §3.3

Notes: This is an analysis program for handling pole distribution maps from the real data, as produced using *Subroutines for Processing Satellite Subsets* (p. 260). A similar program was written to process the individual pole distribution maps from the many random realizations of satellites, which are then averaged. The code in this program performs two main tasks. The first is to generate a density profile for all poles falling within 15° of the most frequent pole location (e.g. Fig. 12 in Paper III). The second is to produce a histogram showing the extent to which each satellite has contributed to the most frequent pole (e.g. Fig. 13 in Paper III).

```
1  MODULE Global !Defines all variables
2  IMPLICIT NONE
3
4  INTEGER :: i, j, k, ios
5
6  REAL :: counts, angle, err_samps, rad_bins(15,2) = 0.e0, sat_counts(25,2)
7  REAL :: max_counts, pole_theta_mode, pole_phi_mode
8  PARAMETER(err_samps = 100.e0)
9
10 REAL*8 :: dummy, best_theta, best_phi, pole_theta, pole_phi, pi
11 PARAMETER(pi = acos(-1.d0))
12
13 REAL :: ncombos = 53130.e0
14 REAL :: cum_pole_count
15
16 REAL :: sat(25)
17
18 LOGICAL :: cumulative
19 PARAMETER(cumulative = .true.)
20
21 END MODULE Global
22
23 !-----------------------------------------------------------------------------------------------
24
25 PROGRAM pole_vicinity_counts !Counts number of poles within 'x' degrees of the best-fit pole
26 USE Global                   !where x is an integer such that 1 .ge. x .ge. 15
27 IMPLICIT NONE                !Counts are divided by the number of samples of each combination
28
29 DOUBLE PRECISION :: sla_DSEP
30
31 best_theta = 38.37154d0 ; best_phi = -78.7439d0 !lat and long of most freq pole
32 !best_theta = 9.9d0 ; best_phi = -87.9d0 !lat and long of blob
33
34 !||Open file with
35 !\/pole positions
36 OPEN(unit = 11, file='Sat_Combo_Planes/Plane_Stats_5_sats_err_weighted/poles_per_pos_5_sats.dat', status = 'old')
37
38 DO i = 1, 15               !degree
39    rad_bins(i,1) = REAL(i) !values
```

```
40  END DO                      !of bin
41
42  best_theta = best_theta * (pi/180.d0) !Convert to
43  best_phi = best_phi * (pi/180.d0)     !radians
44
45  !-----------------------------------------
46
47  cum_pole_count = 0.e0
48  i = 0
49
50  DO WHILE (.TRUE.)
51
52  i = i + 1
53
54  !||Read in pole positions and satellite
55  !\/contributions at that position
56  READ (11, *, IOSTAT = ios) dummy, dummy, dummy, pole_theta, pole_phi, counts, &
57                             sat(1), sat(2), sat(3), sat(4), sat(5), sat(6), sat(7), sat(8), sat(9), sat(10), &
58                             sat(11), sat(12), sat(13), sat(14), sat(15), sat(16), sat(17), sat(18), sat(19), sat(20), &
59                             sat(21), sat(22), sat(23), sat(24), sat(25)
60
61  cum_pole_count = cum_pole_count + counts
62
63  IF (counts .gt. max_counts) THEN
64     max_counts = counts
65     pole_theta_mode = pole_theta
66     pole_phi_mode = pole_phi
67  END IF
68
69  pole_theta = pole_theta * (pi/180.d0) !Convert to
70  pole_phi = pole_phi * (pi/180.d0)     !radians
71
72  !||Measure angular distance between current
73  !\/pole and best-fit pole (uses SLALIB)
74  angle = sla_DSEP(best_phi, best_theta, pole_phi, pole_theta)
75
76  angle = angle * (180.d0/pi) !Convert back to degrees
77
78  !||Find angular distance bin to put
79  !\/pole into (if it is within 15 degrees)
80  IF (angle .le. 1.e0) THEN
81     rad_bins(1,2) = rad_bins(1,2) + counts
82     DO k = 1, 25                          !Count number of
83        sat_counts(k,2) = sat_counts(k,2) + sat(k) !contributions to this
84     END DO                                !pole from each satellite
85  ELSE IF (angle .gt. 1.e0 .and. angle .le. 2.e0) THEN
86     rad_bins(2,2) = rad_bins(2,2) + counts
87     DO k = 1, 25                          !Count number of
88        sat_counts(k,2) = sat_counts(k,2) + sat(k) !contributions to this
89     END DO                                !pole from each satellite
90  ELSE IF (angle .gt. 2.e0 .and. angle .le. 3.e0) THEN
91     rad_bins(3,2) = rad_bins(3,2) + counts
92     DO k = 1, 25                          !Count number of
93        sat_counts(k,2) = sat_counts(k,2) + sat(k) !contributions to this
94     END DO                                !pole from each satellite
95  ELSE IF (angle .gt. 3.e0 .and. angle .le. 4.e0) THEN
96     rad_bins(4,2) = rad_bins(4,2) + counts
97  ELSE IF (angle .gt. 4.e0 .and. angle .le. 5.e0) THEN
98     rad_bins(5,2) = rad_bins(5,2) + counts
99  ELSE IF (angle .gt. 5.e0 .and. angle .le. 6.e0) THEN
100    rad_bins(6,2) = rad_bins(6,2) + counts
```

```
101  ELSE IF (angle .gt. 6.e0 .and. angle .le. 7.e0) THEN
102     rad_bins(7,2) = rad_bins(7,2) + counts
103  ELSE IF (angle .gt. 7.e0 .and. angle .le. 8.e0) THEN
104     rad_bins(8,2) = rad_bins(8,2) + counts
105  ELSE IF (angle .gt. 8.e0 .and. angle .le. 9.e0) THEN
106     rad_bins(9,2) = rad_bins(9,2) + counts
107  ELSE IF (angle .gt. 9.e0 .and. angle .le. 10.e0) THEN
108     rad_bins(10,2) = rad_bins(10,2) + counts
109  ELSE IF (angle .gt. 10.e0 .and. angle .le. 11.e0) THEN
110     rad_bins(11,2) = rad_bins(11,2) + counts
111  ELSE IF (angle .gt. 11.e0 .and. angle .le. 12.e0) THEN
112     rad_bins(12,2) = rad_bins(12,2) + counts
113  ELSE IF (angle .gt. 12.e0 .and. angle .le. 13.e0) THEN
114     rad_bins(13,2) = rad_bins(13,2) + counts
115  ELSE IF (angle .gt. 13.e0 .and. angle .le. 14.e0) THEN
116     rad_bins(14,2) = rad_bins(14,2) + counts
117  ELSE IF (angle .gt. 14.e0 .and. angle .le. 15.e0) THEN
118     rad_bins(15,2) = rad_bins(15,2) + counts
119  END IF
120  !/\Find angular distance bin to put
121  !||pole into (if it is within 15 degrees)
122
123  !||Chack for
124  !\/end of file
125  IF (ios == -1) THEN
126     i = i - 1
127     exit
128  ELSE IF (ios .gt. 0) THEN
129     WRITE (*,*) i
130     i=i-1
131     cycle
132  END IF
133
134  END DO
135
136  WRITE (*,*) "Most frequent pole at theta =", pole_theta_mode, "phi =", pole_phi_mode, "with", max_counts, "counts."
137
138  ncombos = cum_pole_count/ err_samps
139
140  rad_bins(:,2) = rad_bins(:,2)/ err_samps          !Divide by number of samples.
141
142  sat_counts(:,2) = sat_counts(:,2)/err_samps
143
144  DO i = 1, 25
145     sat_counts(i,1) = REAL(i)
146  END DO
147
148
149  IF (cumulative) THEN
150     DO i = 2, 15                                   !
151        rad_bins(i,2) = rad_bins(i,2) + rad_bins(i-1,2) !Convert to cumulative counts
152     END DO                                         !
153  END IF
154
155  DO i = 1, 15                                              !Print number
156     WRITE (*, '(3F16.5)') rad_bins(i,1), rad_bins(i,2), &  !of poles
157                           (rad_bins(i,2)/ncombos) * 100.e0 !between x-1 and
158  END DO                                                    !x (< 15) degrees
159
160  IF (cumulative) THEN
161     WRITE (*,*) "Total poles within 15 degrees of best-fit pole:", rad_bins(15,2), &
```

```
                "(", (rad_bins(15,2)/ncombos) * 100.e0, "%)"!
ELSE
   WRITE (*,*) "Total poles within 15 degrees of best-fit pole:", SUM(rad_bins(:,2)), &
                "(", (SUM(rad_bins(:,2))/ncombos) * 100.e0, "%)"!
END IF

WRITE (*,*) ""                                                  !
WRITE (*,*) "Contributions from each satellite to a pole within 3 degrees of most frequent pole:"
                                                                !Print number of contributions
DO i = 1, 25                                                    !from each satellite to a
   WRITE (*, '(2F16.5)') sat_counts(i,1), sat_counts(i,2)!pole within 3 degrees of the
END DO                                                          !best fit pole


!-----------------------------------------------

!||Make histogram of average pole density in 15 nested
!\/one degree wide annuli around the most frequent pole

CALL pgbegin(0,'pole_sat_prof_err_w.ps/CPS',1,1)
!CALL pgbegin(0,'pole_sat_prof_err_w_blob.ps/CPS',1,1)

CALL pgenv(0.,15.,0.,1.1*MAXVAL(rad_bins(:,2))/ncombos, 0, 0)
CALL pgbin(15, rad_bins(:,1)-0.5, rad_bins(:,2)/ncombos, .true.)
IF (cumulative) THEN
   CALL pglab('Degrees', 'Cumulative Probability','')
ELSE
   CALL pglab('Degrees', 'Probability','')
END IF
CALL pgend

!/\Make histogram of average pole density in 15 nested
!||one degree wide annuli around the most frequent pole

!-----------------------------------------------

CLOSE(11)

!||Make histogram of contributions of each satellite to a pole
!\/with in 3 degrees of the location of most frequent pole

CALL pgbegin(0,'pole_sat_cont_err_w.ps/CPS',1,1)
!CALL pgbegin(0,'pole_sat_cont_err_w_blob.ps/CPS',1,1)

CALL pgsvp(0.1,0.9,0.1,0.9)
CALL pgswin(0.5,25.5,0.,1.1*MAXVAL(sat_counts(:,2)))
CALL pgbox('BCST',0.0,0,'BCNST',0.0,0)

CALL pgbin(25, sat_counts(:,1), sat_counts(:,2), .true.)

CALL pglab('','counts','')

CALL PGPTXT (1.15, -1., 90.0, 1.0, 'I')
CALL PGPTXT (2.15, -1., 90.0, 1.0, 'II')
CALL PGPTXT (3.15, -1., 90.0, 1.0, 'III')
CALL PGPTXT (4.15, -1., 90.0, 1.0, 'V')
CALL PGPTXT (5.15, -1., 90.0, 1.0, 'IX')
CALL PGPTXT (6.15, -1., 90.0, 1.0, 'X')
CALL PGPTXT (7.15, -1., 90.0, 1.0, 'XI')
CALL PGPTXT (8.15, -1., 90.0, 1.0, 'XII')
CALL PGPTXT (9.15, -1., 90.0, 1.0, 'XIII')
```

```
223   CALL PGPTXT (10.15, -1., 90.0, 1.0, 'XIV')
224   CALL PGPTXT (11.15, -1., 90.0, 1.0, 'XV')
225   CALL PGPTXT (12.15, -1., 90.0, 1.0, 'XVI')
226   CALL PGPTXT (13.15, -1., 90.0, 1.0, 'XVII')
227   CALL PGPTXT (14.15, -1., 90.0, 1.0, 'XVIII')
228   CALL PGPTXT (15.15, -1., 90.0, 1.0, 'XIX')
229   CALL PGPTXT (16.15, -1., 90.0, 1.0, 'XX')
230   CALL PGPTXT (17.15, -1., 90.0, 1.0, 'XXI')
231   CALL PGPTXT (18.15, -1., 90.0, 1.0, 'XXII')
232   CALL PGPTXT (19.15, -1., 90.0, 1.0, 'XXIII')
233   CALL PGPTXT (20.15, -1., 90.0, 1.0, 'XXIV')
234   CALL PGPTXT (21.15, -1., 90.0, 1.0, 'XXV')
235   CALL PGPTXT (22.15, -1., 90.0, 1.0, 'XXVI')
236   CALL PGPTXT (23.15, -1., 90.0, 1.0, 'XXVII')
237   CALL PGPTXT (24.15, -1., 90.0, 1.0, 'group')
238   CALL PGPTXT (25.15, -1., 90.0, 1.0, 'M33')
239
240   CALL pgend
241
242   !/\Make histogram of contributions of each satellite to a pole
243   !||with in 3 degrees of the location of most frequent pole
244
245   END PROGRAM pole_vicinity_counts
```

Program: aitoff_hammer.f95

Creation Date: Cir. May 2012 Many versions

Relevant Section: Ch. 5 (Paper III aitoff-hammer plots)

Notes: This program illustrates the way in which the aitoff-hammer plots were produced. I wrote several versions but that presented here is the one used for the standard plots which show the plane member satellites, the great circle on the sky representing the plane, and the pole and anti-pole of the plane (see Fig. 15 of Paper III for example). The aitoff-hammer grid is produced by first making a rectangular grid of a large number of points along the desired lines of latitude and longitude and then transforming the x and y of the points via a Hammer projection. The points are then linked up to produce the final grid. All positions in Paper III (satellites and plane poles) are actually calculated first in the Cartesian coordinate system of Fig. 4.1, rotated into the M31-centric reference frame, converted to Spherical coordinates and then finally transformed into their equivalent Hammer projection locations for plotting.

```
1  MODULE Global !Defines all variables
2  IMPLICIT NONE
3
4  INTEGER :: i, j, s, idum = -9999, nit, seam_loc, color(29), nsats
5  PARAMETER(nit = 1000000)
6  PARAMETER(nsats = 27)
7
8  REAL :: pi
9  PARAMETER(pi = acos(-1.e0))
10
11 REAL :: sat_xyz(29,3)
12 REAL :: theta_coord, phi_coord
13 REAL :: theta(29), phi(29), theta_t(28), phi_t(28)
14 REAL :: theta_ah, phi_ah
15 REAL :: alpha_set, beta_set, gamma_set
16 REAL :: x_rot(3,3), y_rot(3,3), z_rot(3,3)
17 REAL :: pole(2), pole2(2), gc(361,2), gc_t(361,2)
18 REAL :: cart_hold(361,3), pole_cart(3), gc_cart(361,3)
19 REAL :: seam_val
20 REAL :: Best_Sat_Dist(27), Sat_Pos(27,2), xi(27), eta(27), m31_dist
21 REAL*8 :: RA, DEC, xi_dble, eta_dble
22
23 END MODULE Global
24
25 !-----------------------------------------------------------------------
26
27 PROGRAM aitoff_hammer_proj  !Master program
28 USE Global
29 IMPLICIT NONE
30
31 CALL sat_xyz_data
32 CALL aitoff_hammer
33
34 END PROGRAM aitoff_hammer_proj
35
```

```
!-----------------------------------------------------------------------

SUBROUTINE aitoff_hammer  !Produce the plot
USE Global
IMPLICIT NONE

INTEGER :: it_i, it_j, it_k

REAL :: lat(181,2,13), lon(181,2,13), lat_t(181,2,13), lon_t(181,2,13)

!Note lat and lon are lines of constant latitude and longitude
!respectively, each made up of 181 dots
!lat(:,1,:) is the longitude of the dot
!lat(:,2,:) is the latitude of the dot
!So lon(81,1,3) is the longitude of the 81st dot of the 3rd parallel
!It has a value of +120 (i.e. the fixed longitude of this parallel)
!lon(81,2,3) has a value of -10 i.e. the latitude of the dot along
!this line of longitude.
!
!lat_t and lon_t store the lat and lon values before their conversion
!to the aitoff hammer projection.

lon(:,1,1) = 180.e0
lon(:,1,2) = 150.e0
lon(:,1,3) = 120.e0
lon(:,1,4) = 90.e0
lon(:,1,5) = 60.e0
lon(:,1,6) = 30.e0
lon(:,1,7) = 0.e0
lon(:,1,8) = -30.e0
lon(:,1,9) = -60.e0
lon(:,1,10) = -90.e0
lon(:,1,11) = -120.e0
lon(:,1,12) = -150.e0
lon(:,1,13) = -180.e0

lat(:,2,1) = 90.e0
lat(:,2,2) = 75.e0
lat(:,2,3) = 60.e0
lat(:,2,4) = 45.e0
lat(:,2,5) = 30.e0
lat(:,2,6) = 15.e0
lat(:,2,7) = 0.e0
lat(:,2,8) = -15.e0
lat(:,2,9) = -30.e0
lat(:,2,10) = -45.e0
lat(:,2,11) = -60.e0
lat(:,2,12) = -75.e0
lat(:,2,13) = -90.e0

DO it_i = 1, 181
   lon(it_i,2,:) = REAL(it_i - 91)
   lat(it_i,1,:) = REAL((2 * it_i) - 182)
END DO

lon = lon * (pi/180.e0) ; lat = lat * (pi/180.e0)  !Convert to radians

lon_t = lon ; lat_t = lat

DO it_i = 1, 13            !The conversion to an aitoff-
   DO it_j = 1, 181        !hammer projection
```

```
97          lon(it_j,1,it_i) = 2.e0 * SQRT(2.e0) * cos(lon_t(it_j,2,it_i)) * sin(lon_t(it_j,1,it_i)/2.e0) / &
98               SQRT(1.e0 + cos(lon_t(it_j,2,it_i)) * cos(lon_t(it_j,1,it_i)/2.e0))
99          lon(it_j,2,it_i) = SQRT(2.e0) * sin(lon_t(it_j,2,it_i)) / &
100              SQRT(1.e0 + cos(lon_t(it_j,2,it_i)) * cos(lon_t(it_j,1,it_i)/2.e0))
101         lat(it_j,1,it_i) = 2.e0 * SQRT(2.e0) * cos(lat_t(it_j,2,it_i)) * sin(lat_t(it_j,1,it_i)/2.e0) / &
102              SQRT(1.e0 + cos(lat_t(it_j,2,it_i)) * cos(lat_t(it_j,1,it_i)/2.e0))
103         lat(it_j,2,it_i) = SQRT(2.e0) * sin(lat_t(it_j,2,it_i)) / &
104              SQRT(1.e0 + cos(lat_t(it_j,2,it_i)) * cos(lat_t(it_j,1,it_i)/2.e0))
105      END DO
106   END DO
107
108   !---------------------------------------
109   !|| Plotting
110   !\/ Code
111
112   CALL pgbegin(0,'sat_combo_15_sats.ps/CPS',1,1)
113
114   CALL pgenv(-1.0 * pi, 1.0 * pi, -0.5 * pi, 0.5 * pi, 1, -2)
115
116   DO it_i = 1, 13
117      DO it_j = 2, 181
118         CALL pgline(2, (/lon(it_j - 1, 1, it_i), lon(it_j, 1, it_i)/), (/lon(it_j - 1, 2, it_i), lon(it_j, 2, it_i)/))
119         CALL pgline(2, (/lat(it_j - 1, 1, it_i), lat(it_j, 1, it_i)/), (/lat(it_j - 1, 2, it_i), lat(it_j, 2, it_i)/))
120      END DO
121   END DO
122
123   !---------------------------------------
124   !This segment takes the normal vector of the best fit plane, finds the corresponding
125   !pole, anti-pole and great-circle in lat., long. and then converts all to an aitoff
126   !-hammer projection. These are then plotted on the aitoff-hammer sphere.
127
128   pole_cart = (/ 0.15819097, 0.76853156, 0.61994755 /) !x,y,z of normal vector to BFP
129
130   color(1) = 2 ;  color(8) = 2 ; color(15) = 1 ; color(22) = 2
131   color(2) = 1 ;  color(9) = 2 ; color(16) = 1 ; color(23) = 2
132   color(3) = 2 ; color(10) = 2 ; color(17) = 1 ; color(24) = 2 !8
133   color(4) = 1 ; color(11) = 1 ; color(18) = 1 ; color(25) = 2 !8
134   color(5) = 2 ; color(12) = 2 ; color(19) = 1 ; color(26) = 2 !8
135   color(6) = 1 ; color(13) = 2 ; color(20) = 1 ; color(27) = 1
136   color(7) = 2 ; color(14) = 1 ; color(21) = 2 ; color(28) = 1
137
138   color(29) = 2 !Colour for the NGC147/ NGC185/ AND XXX group midpoint icon
139
140   CALL Theta_Phi(pole_cart(1), pole_cart(2), pole_cart(3)) !
141                                                            !Convert to
142   pole(1) = theta_coord                                    !lat, long
143   pole(2) = phi_coord                                      !
144
145   pole2(1) = -1.e0 * pole(1)          !
146   pole2(2) = pole(2) + 180.e0         !Find latitude and
147   IF (pole2(2) .gt. 180.e0) THEN      !longitude of anti-pole
148      pole2(2) = pole2(2) - 360.e0     !
149   END IF                              !
150
151   DO i = 1, 361                                  !Find x,y,z of vectors perpendicular
152      gc_cart(i,1) = cos(REAL(i-181) * (pi/180.e0)) !to pole_cart - i.e. cartesian
153      gc_cart(i,2) = sin(REAL(i-181) * (pi/180.e0)) !coordinates of best fit plane!!!!!!!!!!!!!
154      gc_cart(i,3) = -1.e0 * (pole_cart(1) * gc_cart(i,1) + pole_cart(2) * gc_cart(i,2))/ & !
155                     pole_cart(3)                                                           !
156      gc_cart(i,:) = gc_cart(i,:)/ SQRT(gc_cart(i,1)**2.e0 + gc_cart(i,2) ** 2.e0 + &       !
157                                        gc_cart(i,3)**2.e0)                                 !
```

```
158 END DO                                                                    !
159
160 seam_val = pi
161 DO i = 1, 361
162
163    CALL Theta_Phi(gc_cart(i,1), gc_cart(i,2), gc_cart(i,3)) !Convert best
164                                                             !fit plane x,y,z
165    gc(i,1) = theta_coord * (pi/180.e0)                      !to their lat.
166    gc(i,2) = phi_coord * (pi/180.e0)                        !and long. values
167
168    IF (gc(i,2) .le. seam_val) THEN !Find seam
169       seam_val = gc(i,2)           !where -180
170       seam_loc = i                 !longitude
171    END IF                          !meets +180
172
173 END DO
174
175 gc_t = gc   !Set gc_t to pre-order-fix gc
176
177 IF (gc(1,2) .lt. gc(361,2)) THEN !
178    gc(1,:) = gc_t(361,:)         !fix
179    gc(361,:) = gc_t(1,:)         !order
180 END IF                           !
181
182 gc_t = gc   !Set gc_t to new gc
183
184 DO i = 1, 361                                                          !
185    gc(i,2) = 2.e0 * SQRT(2.e0) * cos(gc_t(i,1)) * sin(gc_t(i,2)/2.e0)  !Transform great-
186    gc(i,2) = gc(i,2) / SQRT(1.e0 + cos(gc_t(i,1)) * cos(gc_t(i,2)/2.e0)) !circle into
187    gc(i,1) = SQRT(2.e0) * sin(gc_t(i,1))                               !aitoff-hammer
188    gc(i,1) = gc(i,1) / SQRT(1.e0 + cos(gc_t(i,1)) * cos(gc_t(i,2)/2.e0)) !projection
189 END DO                                                                 !
190
191 CALL pgsci(4)                                                  !
192 CALL pgslw(3)                                                 !
193 CALL pgsch(2.0)                                               !
194 CALL aitoff_convert(pole(1), pole(2))                        !
195 CALL pgpt(1, pole(2), pole(1), 845)                          !Plot pole
196 CALL aitoff_convert(pole2(1), pole2(2))                      !
197 CALL pgpt(1, pole2(2), pole2(1), 846)                       !anti-pole and
198 CALL pgsch(1.0)                                             !
199 CALL pgslw(1)                                               !great circle
200                                                             !
201 DO i = 2, 361                                               !
202    CALL pgline(2, (/gc(i-1,2), gc(i,2)/), (/gc(i-1,1), gc(i,1)/)) !
203 END DO                                                      !
204
205 !-------------------------------------
206
207 !||Plot individual satellites
208 !\/and labels
209 s = 1
210 CALL pgsci(color(s))
211 CALL pgpt(1, phi(s), theta(s), 843)
212 CALL pgptxt(phi(s)-0.1, theta(s)+0.0, 0., 0., 'I')
213 !:
214 !: S = 2, ..., 26
215 !:
216 s = 27
217 CALL pgsci(color(s))
218 CALL pgpt(1, phi(s), theta(s), 768)
```

```
219   CALL pgptxt(phi(s)-0.30, theta(s)+0.05, 0., 0., 'M33')
220
221   s = 28
222   CALL pgsci(color(s))
223   CALL pgpt(1, phi(s), theta(s), 2284)
224   CALL pgptxt(phi(s)-0.3, theta(s)+0.05, 0., 0., 'MWy')
225
226   !s = 29 !Ngc147/ NGC185/ And XXX Group midpoint icon
227   !CALL pgsci(color(s))
228   !CALL pgpt(1, phi(s), theta(s), 0904)
229
230   !/\Plot individual satellites
231   !||and labels
232
233   CALL pgsci(1)
234
235   CALL pgpt(1, 0., 0., 2293)
236   CALL pgptxt(-0.3, 0.05, 0., 0., 'M31')
237
238   CALL pgsch(1.0)
239
240   !||Plots labels for lines of
241   !\/constant lat. and long.
242   CALL pgptxt(lat(1,1,1)-0.05, lat(1,2,1)+0.05, 0., 0., '90')
243   CALL pgptxt(lat(1,1,2)-0.06, lat(1,2,2)+0.05, 0., 0., '75')
244   CALL pgptxt(lat(1,1,3)-0.08, lat(1,2,3)+0.05, 0., 0., '60')
245   CALL pgptxt(lat(1,1,4)-0.14, lat(1,2,4)+0.03, 0., 0., '45')
246   CALL pgptxt(lat(1,1,5)-0.18, lat(1,2,5)+0.0, 0., 0., '30')
247   CALL pgptxt(lat(1,1,6)-0.20, lat(1,2,6)-0.02, 0., 0., '15')
248   CALL pgptxt(lat(1,1,7)-0.15, lat(1,2,7)-0.02, 0., 0., '0')
249   CALL pgptxt(lat(1,1,8)-0.33, lat(1,2,8)-0.05, 0., 0., '-15')
250   CALL pgptxt(lat(1,1,9)-0.31, lat(1,2,9)-0.07, 0., 0., '-30')
251   CALL pgptxt(lat(1,1,10)-0.24, lat(1,2,10)-0.13, 0., 0., '-45')
252   CALL pgptxt(lat(1,1,11)-0.20, lat(1,2,11)-0.13, 0., 0., '-60')
253   CALL pgptxt(lat(1,1,12)-0.21, lat(1,2,12)-0.14, 0., 0., '-75')
254   CALL pgptxt(lat(1,1,13)-0.20, lat(1,2,13)-0.12, 0., 0., '-90')
255
256   CALL pgsch(0.5)
257   CALL pgptxt(lon(91,1,11)-0.20, lon(91,2,11)-0.12, 0., 0., '-120')
258   CALL pgptxt(lon(91,1,9)-0.15, lon(91,2,9)-0.12, 0., 0., '-60')
259   CALL pgptxt(lon(91,1,5)-0.12, lon(91,2,5)-0.12, 0., 0., '60')
260   CALL pgptxt(lon(91,1,3)-0.16, lon(91,2,3)-0.12, 0., 0., '120')
261   CALL pgptxt(lon(91,1,1)-0.16, lon(91,2,1)-0.12, 0., 0., '180')
262
263   CALL pgsch(1.0)
264
265   CALL pgend
266
267   END SUBROUTINE aitoff_hammer
268
269   !----------------------------------------------------------------------
270
271   SUBROUTINE sat_xyz_data  !Get the data
272   USE Global
273   IMPLICIT NONE
274
275   DOUBLE PRECISION :: sla_DSEP
276
277   m31_dist = 779.e0              !M31
278
279   !Best_Sat_Dist(1:27) as per 'PlaneSigRMS.f95' - see 'Significance' subroutine
```

```
280
281 !Sat_Pos(1:27,:) as per 'PlaneSigRMS.f95' - see 'SampledDist' subroutine
282
283 DO i = 1, nsats
284    xi(i) = Sat_Pos(i,1)
285    eta(i) = Sat_Pos(i,2)
286    xi(i) = xi(i) * (pi/180.e0)             !Convert angles from
287    eta(i) = eta(i) * (pi/180.e0)           !degrees to radians
288 END DO
289
290 DO i = 1, nsats                                          !
291    xi_dble = xi(i) ; eta_dble = eta(i)                   !
292    CALL sla_DTP2S(xi_dble, eta_dble, 0.d0, 0.d0, RA, DEC) !Convert tangent plane
293    IF (xi_dble .lt. 0.d0) then                           !projection angles into
294       RA = RA - (2.e0 * pi)                              !their true angles using
295    END IF                                                !sla_DTP2S
296    xi(i) = RA                                            !
297    eta(i) = DEC                                          !
298 END DO
299
300 DO i = 1, nsats                                  !
301    xi_dble = xi(i)                               !Find the true angle
302    eta_dble = eta(i)                             !theta_t - the angle on
303    theta_t(i) = sla_DSEP(0.d0, 0.d0, xi_dble, eta_dble)!the sky between M31 and
304 END DO                                           !the object (uses sla_DSEP)
305
306 DO i = 1, nsats
307    sat_xyz(i,1) = ABS(Best_Sat_Dist(i) * cos(theta_t(i)) * tan(xi(i))) !Determine length of x vector for each satellite
308    IF (xi(i) .lt. 0.e0) THEN                                           !
309       sat_xyz(i,1) = -1.e0 * sat_xyz(i,1)                              !Determine if x is positive or negative
310    END IF                                                              !
311
312    sat_xyz(i,2) = ABS(Best_Sat_Dist(i) * sin(eta(i)))                  !Determine length of y vector for each satellite
313    IF (eta(i) .lt. 0.e0) THEN                                          !
314       sat_xyz(i,2) = -1.e0 * sat_xyz(i,2)                              !Determine if y is positive or negative
315    END IF                                                              !
316
317    sat_xyz(i,3) = Best_Sat_Dist(i) * cos(theta_t(i)) - m31_dist        !Determine length and sign of z vector
318 END DO
319
320 sat_xyz(28,:) = (/ 0.e0, 0.e0, -779.e0 /)                             !MWy
321
322 sat_xyz(29,:) = sat_xyz(25,:) + ((100.e0)**(0.2e0*0.2e0)) * sat_xyz(26,:) !NGC147/ NGC185/ AND XXX
323 sat_xyz(29,:) = sat_xyz(29,:)/(1.e0 + (100.e0)**(0.2e0*0.2e0))            !group mid point
324
325
326 DO i = 1, 29
327 WRITE (*,*) i, sat_xyz(i,1), sat_xyz(i,2), sat_xyz(i,3)
328 alpha_set =  - (90.e0 - 12.5e0) * (pi/180.e0) !Rotate to bring back out of M31's inclination       !
329 gamma_set =  + (90.e0 - 39.8e0) * (pi/180.e0) !angle and PA (i.e. to view from above the M31 pole) !
330                                                                                                    !Change
331 CALL Rotate                                                                                       !
332                                                                                                    !to
333 sat_xyz(i,:) = MATMUL(z_rot, sat_xyz(i,:))        !Convert vectors back to how they would appear  !
334 sat_xyz(i,:) = MATMUL(x_rot, sat_xyz(i,:))        !in M31 reference frame                         !M31
335                                                                                                    !
336 gamma_set = 90.e0 * (pi/180.e0)           !                                                      !coordinate
337                                           !                                                      !
338 CALL Rotate                               !Additional rotation in M31 galactic longitude         !system
339                                           !                                                      !
340 sat_xyz(i,:) = MATMUL(z_rot, sat_xyz(i,:))!                                                      !
```

```
341
342   WRITE (*,*) i, sat_xyz(i,1), sat_xyz(i,2), sat_xyz(i,3)
343
344   CALL Theta_Phi(sat_xyz(i,1), sat_xyz(i,2), sat_xyz(i,3))
345
346   CALL aitoff_convert(theta_coord, phi_coord)
347
348   theta(i) = theta_coord ; phi(i) = phi_coord
349
350   END DO
351
352   END SUBROUTINE sat_xyz_data
353
354   !-------------------------------------------------------------------------
355
356   SUBROUTINE aitoff_convert(theta_ah, phi_ah)  !Convert to aitoff-
357   IMPLICIT NONE                                !hammer projection
358
359   REAL :: theta_ah, phi_ah, pre_theta_ah, pre_phi_ah
360
361   REAL :: pi
362   PARAMETER(pi = acos(-1.e0))
363
364   pre_theta_ah = theta_ah * (pi/180.e0)
365   pre_phi_ah = phi_ah * (pi/180.e0)
366
367   phi_ah = 2.e0 * SQRT(2.e0) * cos(pre_theta_ah) * sin(pre_phi_ah/2.e0)
368   phi_ah = phi_ah / SQRT(1.e0 + cos(pre_theta_ah) * cos(pre_phi_ah/2.e0))
369
370   theta_ah = SQRT(2.e0) * sin(pre_theta_ah)
371   theta_ah = theta_ah / SQRT(1.e0 + cos(pre_theta_ah) * cos(pre_phi_ah/2.e0))
372
373   END SUBROUTINE aitoff_convert
374
375   !-------------------------------------------------------------------------
376
377   !'Rotate' Subroutine - See 'PlaneSigRMS.f95'
378
379   !-------------------------------------------------------------------------
380
381   !'Theta_Phi' Subroutine - See 'PlaneSigRMS.f95'
382
383   !-------------------------------------------------------------------------
```

Program: RR_Histograms.f95

Creation Date: 7 Oct 2012

Relevant Section: Ch. 5; Paper III Figs. 5 (RH column), 6, 9 (RH column), 16 (b)

Notes: This program illustrates the way in which the histograms of the goodness of fit statistic for the random realizations were generated. Throughout the entire thesis I have generated many histograms and toward the end I decided to make a stand alone subroutine 'HistoPlot' (see *PlaneSigRMS.f95* - p. 244) that automated the process. I later modified that subroutine to add the credibility interval color-coding used for the papers. It is this subroutine which is shown here: 'HistoPlotAdv.' The 'DataCall' subroutine is designed to handle the many different outputs from the various plane fitting programs. It is set up to plot the histogram of the average RMS (or other plane fitting statistic) values from the random satellite realizations and also to take the average of the histogram produced from the real data.

```
1  MODULE Global !Defines all variables used by BayesianTRGB
2  IMPLICIT NONE
3
4  INTEGER :: i, idum = -9999, ios, ndata, ndata2, counts
5  PARAMETER (ndata = 10000)
6  PARAMETER (ndata2 = 200000)
7  REAL :: signif(ndata), signif2, dummy, scale_factor, average_sig
8  CHARACTER :: folder*300, string*300, string2*300, command*300
9  LOGICAL :: Best15, RMS, AbVal, Asy, AsyFP, ML, Sigma
10 PARAMETER (Best15 = .true.)
11 PARAMETER (RMS = .false.)
12 PARAMETER (AbVal = .false.)
13 PARAMETER (Asy = .false.)
14 PARAMETER (AsyFP = .false.)
15 PARAMETER (ML = .false.)
16 PARAMETER (Sigma = .false.)
17
18 END MODULE Global
19
20 !-----------------------------------------------------------------------
21
22 PROGRAM DataCall !Reads in data to be
23 USE Global       !plotted and passes
24 IMPLICIT NONE    !to HistoPlotAdv
25
26 IF (Best15) THEN  !For Plane of best 15 satellites
27 OPEN(unit = 11, file='./Sat_Combo_planes/Plane_Stats_15_sats_RandReal_weighted/RMS_15_sats.dat', status = 'old')
28 OPEN(unit = 12, file='./Sat_Combo_planes/RMS_Plane_Stats_Best_15_sats/real_sig_wth_err.dat', status = 'old')
29 END IF
30
31 IF (RMS) THEN      !For RMS distribution
32 OPEN(unit = 11, file='./Sat_Comp_Set_Stats/Plane_Stats_27_sats_RandReal_weighted_RMS/RMS_27_sats.dat', status = 'old')
33 OPEN(unit = 12, file='./Sat_Comp_Set_Stats/RMS_Plane_Stats_27_sats/real_sig_wth_err.dat', status = 'old')
34 END IF
35
36 IF (AbVal) THEN    !For 'sum of Absolute Values' distribution
37 OPEN(unit = 11, file='./Sat_Comp_Set_Stats/Plane_Stats_27_sats_RandReal_weighted_AV/AV_27_sats.dat', status = 'old')
```

```
38  OPEN(unit = 12, file='./Sat_Comp_Set_Stats/AbVal_Plane_Stats/real_sig_wth_err.dat', status = 'old')
39  END IF
40
41  IF (Asy) THEN      !For Asymmetry distribution
42  OPEN(unit = 11, file='./Sat_Comp_Set_Stats/Plane_Stats_27_sats_RandReal_weighted_Asy/Asy_27_sats.dat', status = 'old')
43  OPEN(unit = 12, file='./Sat_Comp_Set_Stats/Asymm_Stats/real_asy_wth_err.dat', status = 'old')
44  END IF
45
46  IF (AsyFP) THEN   !For Distribution of Asymmetry about M31 tangent plane
47  OPEN(unit = 11, file='./Sat_Comp_Set_Stats/Plane_Stats_27_sats_RandReal_weighted_AsyFP/AsyFP_27_sats.dat', status = 'old')
48  OPEN(unit = 12, file='./Sat_Comp_Set_Stats/Asymm_Stats_FixedPlane/real_asy_wth_err.dat', status = 'old')
49  END IF
50
51  IF (ML .or. Sigma) THEN   !For Maximum Likelihood (and sigma) distributions
52  OPEN(unit = 11, file='./Sat_Comp_Set_Stats/Plane_Stats_27_sats_RandReal_weighted_ML/ML_27_sats.dat', status = 'old')
53  OPEN(unit = 12, file='./Sat_Comp_Set_Stats/Plane_Stats/real_sig_wth_err.dat', status = 'old')
54  END IF
55
56  !||Read in plane fitting statistic (e.g. RMS) from
57  !\/Random Realizations for generation of histogram
58
59  i = 0
60
61  DO WHILE (.TRUE.)
62
63  i = i + 1
64
65  IF (i .gt. ndata) THEN
66     i = i-1
67     exit
68  END IF
69
70  IF (Sigma) THEN
71     READ (11, *, IOSTAT = ios) dummy, dummy, signif(i)
72  ELSE
73     READ (11, *, IOSTAT = ios) dummy, signif(i)
74  END IF
75
76  IF (ios == -1) THEN
77     i = i - 1
78     exit
79  ELSE IF (ios .gt. 0) THEN
80     WRITE (*,*) i
81     i=i-1
82     cycle
83  END IF
84
85  END DO
86
87  !/\Read in plane fitting statistic (e.g. RMS) from
88  !||Random Realizations for generation of histogram
89
90  !||Read in plane fitting statistic (e.g. RMS) from Realizations
91  !\/of possible positions of the real satellites to find average
92
93  i = 0
94
95  DO WHILE (.TRUE.)
96
97  i = i + 1
98
```

```
99  IF (i .gt. ndata2) THEN
100    i = i-1
101    exit
102 END IF
103
104 IF (Sigma) THEN
105    READ (12, *, IOSTAT = ios) dummy, dummy, signif2
106 ELSE
107    READ (12, *, IOSTAT = ios) dummy, signif2
108 END IF
109
110 IF (Asy .or. AsyFP .or. ML .or. Sigma) THEN
111
112 ELSE
113    signif2 = 10.e0 ** signif2
114 END IF
115
116 average_sig = average_sig + signif2
117
118 IF (ios == -1) THEN
119    i = i - 1
120    exit
121 ELSE IF (ios .gt. 0) THEN
122    WRITE (*,*) i
123    i=i-1
124    cycle
125 END IF
126
127 END DO
128
129 !/\Read in plane fitting statistic (e.g. RMS) from Realizations
130 !||of possible positions of the real satellites to find average
131
132 average_sig = average_sig/ REAL(ndata2)
133
134 CALL pgbegin(0,'RR_Histogram.ps/CPS',1,1)
135
136 IF (Best15) THEN
137 CALL pgenv(5.,35.,0.,0.15, 0, 0)        !For Best 15 satellites
138 END IF
139 IF (RMS) THEN
140 CALL pgenv(30.,90.,0.,0.07, 0, 0)       !For RMS distribution
141 END IF
142 IF (AbVal) THEN
143 CALL pgenv(700.,1900.,0.,0.003, 0, 0) !For AbVal distribution
144 END IF
145 IF (Asy) THEN
146 CALL pgenv(13.,28.,0.,0.4, 0, 0)       !For Asymmetry distribution
147 END IF
148 IF (AsyFP) THEN
149 CALL pgenv(13.,28.,0.,0.8, 0, 0)       !For Asymmetry Fixed Plane distribution
150 END IF
151 IF (ML) THEN
152 CALL pgenv(-69.,-59.,0.,0.4, 0, 0)     !For ML distribution
153 END IF
154 IF (Sigma) THEN
155 CALL pgenv(30.,90.,0.,0.07, 0, 0)       !For Sigma distribution
156 END IF
157
158 IF (Sigma) THEN
159    CALL HistoPlotAdv(ndata, 21, signif, '', '', '?', .true.) !Make Histogram
```

```
160 ELSE
161     CALL HistoPlotAdv(ndata, 51, signif, '', '', '?', .true.) !Make Histogram
162 END IF
163
164 !||Plot dot-dash line at location of histogram average from realizations
165 !\/of possible positions of the real satellites
166 CALL pgsls(3)
167 CALL pgsci(8)
168 CALL pgslw(5)
169 IF (Best15) THEN
170 CALL pgline (2, (/ average_sig, average_sig /), (/ 0.0, 0.15 /))   !For Best 15 satellites
171 END IF
172 IF (RMS) THEN
173 CALL pgline (2, (/ average_sig, average_sig /), (/ 0.0, 0.07 /))   !For RMS distribution
174 END IF
175 IF (AbVal) THEN
176 CALL pgline (2, (/ average_sig, average_sig /), (/ 0.0, 0.003 /)) !For AbVal distribution
177 END IF
178 IF (Asy) THEN
179 CALL pgline (2, (/ average_sig, average_sig /), (/ 0.0, 0.4 /)) !For Asymmetry distribution
180 END IF
181 IF (AsyFP) THEN
182 CALL pgline (2, (/ average_sig, average_sig /), (/ 0.0, 0.8 /)) !For Asymmetry distribution
183 END IF
184 IF (ML) THEN
185 CALL pgline (2, (/ average_sig, average_sig /), (/ 0.0, 0.4 /)) !For ML distribution
186 END IF
187 IF (Sigma) THEN
188 CALL pgline (2, (/ average_sig, average_sig /), (/ 0.0, 0.07 /)) !For Sigma distribution
189 END IF
190 CALL pgslw(1)
191 CALL pgsci(1)
192 CALL pgsls(1)
193 !/\Plot dot-dash line at location of histogram average from realizations
194 !||of possible positions of the real satellites
195
196 IF (RMS .or. Best15) THEN
197 CALL pglab('Minimum␣RMS␣(kpc)', 'Probability', '')                  !For RMS distribution
198 END IF
199 IF (AbVal) THEN
200 CALL pglab('Minimum␣Absolute␣Distance␣Sum␣(kpc)', 'Probability', '') !For AbVal distribution
201 END IF
202 IF (Asy .or. AsyFP) THEN
203 CALL pglab('Maximum␣Hemisphere␣Satellite␣Count', 'Probability', '')  !For Asymmetry distribution
204 END IF
205 IF (ML) THEN
206 CALL pglab('LOG10(Maximum␣Likelihood)', 'Probability', '')          !For ML distribution
207 END IF
208 IF (Sigma) THEN
209 CALL pglab('Plane␣Sigma␣(kpc)', 'Probability', '')                  !For Sigma distribution
210 END IF
211
212 CALL pgend
213
214 WRITE (*,*) "Average␣of␣observed␣plane:", average_sig
215
216 IF (Best15) THEN
217    counts = 0
218    DO i = 1, ndata
219       IF (signif(i) .le. average_sig) THEN
220          counts = counts + 1
```

```
221         END IF
222     END DO
223     WRITE (*,*) "An RMS of", average_sig, "was equalled or exceeded", counts, "out of", ndata, "times."
224 END IF
225
226 IF (Asy .or. AsyFP) THEN
227     counts = 0
228     DO i = 1, ndata
229         IF (signif(i) .ge. average_sig) THEN
230         counts = counts + 1
231         END IF
232     END DO
233     WRITE (*,*) "An Asymmetry of", average_sig, "was equalled or exceeded", counts, "out of", ndata, "times."
234 END IF
235
236 END PROGRAM
237
238 !-----------------------------------------------------------------------
239
240 SUBROUTINE HistoPlotAdv(nval, data_hist_bins, data, xlabel, ylabel, device, normalize)
241 USE Global
242 IMPLICIT NONE
243
244 !
245 !**************Created 16 Jun 2012**************
246 !
247 !INTEGER nval = number of data points in histogram
248 !INTEGER data_hist_bins = number of bins in histogram
249 !REAL data(nval) = The array containing the data
250 !CHARACTER xlabel = Label of x-axis of histogram
251 !CHARACTER ylabel = Label of y-axis of histogram
252 !CHARACTER device = The plotting device ('?' if unsure)
253 !LOGICAL normalize = .true. if histogram is to be
254 !normalized, else set to .false.
255 !
256 !Uses PGPLOT
257 !
258
259 INTEGER :: data_hist_bins, nval, it_num, BFL
260 REAL :: bw, data(nval), data_hist(data_hist_bins,2), data_min, data_max
261 REAL :: BFV, psig, msig, max_bin_height, data_counts, pcounts, mcounts
262 REAL :: xpts(2), ypts(2), p90, m90, p99, m99
263 CHARACTER(LEN=*) :: xlabel, ylabel, device
264 LOGICAL :: normalize
265
266 !|| Builds the specified
267 !\/ histogram
268
269 data_hist = 0.e0
270
271 data_min = MINVAL(data) ; data_max = MAXVAL(data)
272
273 bw = (data_max - data_min)/(REAL(data_hist_bins) - 1.e0)
274
275 DO it_num = 1, data_hist_bins
276     data_hist(it_num,1) = data_min + REAL(it_num-1) * bw
277 END DO
278
279 DO it_num = 1, nval
280     data_hist(NINT((data(it_num) - data_min)/bw) + 1,2) = &
281     data_hist(NINT((data(it_num) - data_min)/bw) + 1,2) + 1.e0
```

```
282  END DO
283
284  IF (normalize) THEN
285      data_hist(:,2) = data_hist(:,2) / (bw * SUM(data_hist(:,2)))
286  END IF
287
288  !/\ Builds the specified
289  !|| histogram
290
291  max_bin_height = 0.d0                                  !
292  DO it_num = 1, data_hist_bins                          !
293      IF (data_hist(it_num,2) .gt. max_bin_height) THEN  !
294          max_bin_height = data_hist(it_num,2)           !Find best fit TRGB value
295          BFV = data_hist(it_num,1)                      !
296          BFL = it_num                                   !
297      END IF                                             !
298  END DO                                                 !
299  WRITE (*,*) "Best fit value is at:", BFV
300
301  data_counts = 0.d0 ; pcounts = 0.d0                !
302  DO it_num = BFL, data_hist_bins                    !
303      pcounts = pcounts + data_hist(it_num,2)        !
304  END DO                                             !
305  DO it_num = BFL, data_hist_bins                    !
306      data_counts = data_counts + data_hist(it_num,2)!Finds positive one sigma
307      IF (data_counts .ge. 0.682*pcounts) THEN       !error in distance
308          psig = data_hist(it_num,1) - BFV           !
309          exit                                       !
310      END IF                                         !
311  END DO                                             !
312  WRITE (*,*) "Plus 1 sigma: ", psig
313
314  data_counts = 0.d0 ; mcounts = 0.d0                !
315  DO it_num = BFL, 1, -1                             !
316      mcounts = mcounts + data_hist(it_num,2)        !
317  END DO                                             !
318  DO it_num = BFL, 1, -1                             !
319      data_counts = data_counts + data_hist(it_num,2)!Finds negative one sigma
320      IF (data_counts .ge. 0.682*mcounts) THEN       !error in distance
321          msig = BFV - data_hist(it_num,1)           !
322          exit                                       !
323      END IF                                         !
324  END DO                                             !
325  WRITE (*,*) "Minus 1 sigma:", msig
326
327  data_counts = 0.d0 ; pcounts = 0.d0                !
328  DO it_num = BFL, data_hist_bins                    !
329      pcounts = pcounts + data_hist(it_num,2)        !
330  END DO                                             !
331  DO it_num = BFL, data_hist_bins                    !
332      data_counts = data_counts + data_hist(it_num,2)!Finds positive 90% credibility
333      IF (data_counts .ge. 0.9*pcounts) THEN         !error in distance
334          p90 = data_hist(it_num,1) - BFV            !
335          exit                                       !
336      END IF                                         !
337  END DO                                             !
338  WRITE (*,*) "Plus 90%: ", p90
339
340  data_counts = 0.d0 ; mcounts = 0.d0                !
341  DO it_num = BFL, 1, -1                             !
342      mcounts = mcounts + data_hist(it_num,2)        !
```

```
343 END DO                                          !
344 DO it_num = BFL, 1, -1                          !
345    data_counts = data_counts + data_hist(it_num,2)!Finds negative 90% credibility
346    IF (data_counts .ge. 0.9*mcounts) THEN       !error in distance
347       m90 = BFV - data_hist(it_num,1)           !
348       exit                                      !
349    END IF                                       !
350 END DO                                          !
351 WRITE (*,*) "Minus 90%:", m90
352
353 data_counts = 0.d0 ; pcounts = 0.d0            !
354 DO it_num = BFL, data_hist_bins                 !
355    pcounts = pcounts + data_hist(it_num,2)      !
356 END DO                                          !
357 DO it_num = BFL, data_hist_bins                 !
358    data_counts = data_counts + data_hist(it_num,2)!Finds positive 99% credibility
359    IF (data_counts .ge. 0.99*pcounts) THEN      !error in distance
360       p99 = data_hist(it_num,1) - BFV           !
361       exit                                      !
362    END IF                                       !
363 END DO                                          !
364 WRITE (*,*) "Plus 99%: ", p99
365
366 data_counts = 0.d0 ; mcounts = 0.d0            !
367 DO it_num = BFL, 1, -1                          !
368    mcounts = mcounts + data_hist(it_num,2)      !
369 END DO                                          !
370 DO it_num = BFL, 1, -1                          !
371    data_counts = data_counts + data_hist(it_num,2)!Finds negative 99% credibility
372    IF (data_counts .ge. 0.99*mcounts) THEN      !error in distance
373       m99 = BFV - data_hist(it_num,1)           !
374       exit                                      !
375    END IF                                       !
376 END DO                                          !
377 WRITE (*,*) "Minus 99%:", m99
378
379 !||Plot histogram with coloured
380 !\/credibility intervals
381 DO it_num = 1, data_hist_bins-1
382    IF (data_hist(it_num,1) .ge. BFV - msig .and. data_hist(it_num,1) .lt. BFV + psig) THEN
383       CALL pgsci(2)                                                    !
384       CALL pgbin (2, data_hist(it_num,1), data_hist(it_num,2),.false.) !
385       IF (data_hist(it_num,1) .eq. BFV - msig) THEN                    !
386          xpts = data_hist(it_num,1)                                    !
387          ypts(1) = 0.e0 ; ypts(2) = data_hist(it_num,2)                !One Sigma
388          CALL pgline (2, xpts, ypts)                                   !
389       END IF                                                           !Credibility
390       IF (data_hist(it_num+1,1) .eq. BFV + psig) THEN                  !
391          xpts = data_hist(it_num+1,1)                                  !Interval
392          ypts(1) = 0.e0 ; ypts(2) = data_hist(it_num,2)                !
393          CALL pgline (2, xpts, ypts)                                   !
394       END IF                                                           !
395    ELSE IF (data_hist(it_num,1) .ge. BFV - m90 .and. data_hist(it_num,1) .lt. BFV + p90) THEN
396       CALL pgsci(3)                                                    !
397       CALL pgbin (2, data_hist(it_num,1), data_hist(it_num,2),.false.) !
398       IF (data_hist(it_num,1) .eq. BFV - m90) THEN                     !
399          xpts = data_hist(it_num,1)                                    !
400          ypts(1) = 0.e0 ; ypts(2) = data_hist(it_num,2)                !90 percent
401          CALL pgline (2, xpts, ypts)                                   !
402       END IF                                                           !Credibility
403       IF (data_hist(it_num+1,1) .eq. BFV + p90) THEN                   !
```

```
404             xpts = data_hist(it_num+1,1)                                    !Interval
405             ypts(1) = 0.e0 ; ypts(2) = data_hist(it_num,2)                  !
406             CALL pgline (2, xpts, ypts)                                     !
407          END IF                                                             !
408       ELSE IF (data_hist(it_num,1) .ge. BFV - m99 .and. data_hist(it_num,1) .lt. BFV + p99) THEN
409          CALL pgsci(4)                                                      !
410          CALL pgbin (2, data_hist(it_num,1), data_hist(it_num,2),.false.)   !
411          IF (data_hist(it_num,1) .eq. BFV - m99) THEN                       !
412             xpts = data_hist(it_num,1)                                      !
413             ypts(1) = 0.e0 ; ypts(2) = data_hist(it_num,2)                  !99 percent
414             CALL pgline (2, xpts, ypts)                                     !
415          END IF                                                             !Credibility
416          IF (data_hist(it_num+1,1) .eq. BFV + p99) THEN                     !
417             xpts = data_hist(it_num+1,1)                                    !Interval
418             ypts(1) = 0.e0 ; ypts(2) = data_hist(it_num,2)                  !
419             CALL pgline (2, xpts, ypts)                                     !
420          END IF                                                             !
421       ELSE
422          CALL pgsci(1)
423          CALL pgbin (2, data_hist(it_num,1), data_hist(it_num,2),.false.)   !
424          IF (it_num .eq. 1) THEN                                            !
425             xpts = data_hist(it_num,1)                                      !
426             ypts(1) = 0.e0 ; ypts(2) = data_hist(it_num,2)                  !
427             CALL pgline (2, xpts, ypts)                                     !Distribution
428          END IF                                                             !
429          IF (it_num .eq. data_hist_bins-1) THEN                             !outside of 99 %
430             xpts = data_hist(it_num+1,1) + bw                               !
431             ypts(1) = 0.e0 ; ypts(2) = data_hist(it_num+1,2)                !Cred. Interval
432             CALL pgline (2, xpts, ypts)                                     !
433          END IF                                                             !
434       END IF                                                                !
435    END DO
436    !/\Plot histogram with coloured
437    !||confidence intervals
438
439    END SUBROUTINE HistoPlotAdv
440
441    !-------------------------------------------------------------------------
```

List of Abbreviations

The following list is neither exhaustive nor exclusive, but may be helpful.

$2MASS$ ..... The Two-Micron All-Sky Survey

λCDM or CDM [Lambda] Cold Dark Matter (cosmological model)

AGB ..... Asymptotic Giant Branch

$CFHT$ ..... The Canada-France-Hawaii Telescope

CMD ..... Colour-Magnitude Diagram

LF ..... Luminosity Function

$PAndAS$ .... The Pan-Andromeda Archaeological Survey

RGB ..... Red Giant Branch

$SDSS$ ..... The Sloan Digital Sky Survey

$TRGB$ ..... Tip of the Red Giant Branch

References

C. Alcock, R. A. Allsman, D. R. Alves, T. S. Axelrod, A. C. Becker, D. P. Bennett, K. H. Cook, N. Dalal, A. J. Drake, K. C. Freeman, M. Geha, K. Griest, M. J. Lehner, S. L. Marshall, D. Minniti, C. A. Nelson, B. A. Peterson, P. Popowski, M. R. Pratt, P. J. Quinn, C. W. Stubbs, W. Sutherland, A. B. Tomaney, T. Vandehei, and D. Welch. The MACHO Project: Microlensing Results from 5.7 Years of Large Magellanic Cloud Observations. *Astrophysical Journal*, 542:281–307, October 2000. 17

M. Bayes and M. Price. An Essay towards Solving a Problem in the Doctrine of Chances. By the Late Rev. Mr. Bayes, F. R. S. Communicated by Mr. Price, in a Letter to John Canton, A. M. F. R. S. *Royal Society of London Philosophical Transactions Series I*, 53:370–418, 1763. 47

K. Bekki and K. C. Freeman. Formation of ω Centauri from an ancient nucleated dwarf galaxy in the young Galactic disc. *Monthly Notices of the Royal Astronomical Society*, 346:L11–L15, December 2003. 7

M. Bellazzini. The Tip of the Red Giant Branch. *Memorie della Societa Astronomica Italiana*, 79:440, 2008. 28

M. Bellazzini, F. R. Ferraro, and E. Pancino. A Step toward the Calibration of the Red Giant Branch Tip as a Standard Candle. *Astrophysical Journal*, 556:635–640, August 2001. 28

R. A. Benjamin, E. Churchwell, B. L. Babler, R. Indebetouw, M. R. Meade, B. A. Whitney, C. Watson, M. G. Wolfire, M. J. Wolff, R. Ignace, T. M. Bania, S. Bracker, D. P. Clemens,

L. Chomiuk, M. Cohen, J. M. Dickey, J. M. Jackson, H. A. Kobulnicky, E. P. Mercer, J. S. Mathis, S. R. Stolovy, and B. Uzpen. First GLIMPSE Results on the Stellar Structure of the Galaxy. *Astrophysical Journal Letters*, 630:L149–L152, September 2005. 3

J. Bland-Hawthorn and K. C. Freeman. Galactic history: formation Memorie della Societa Astronomica Italiana evolution. *Memorie della Societa Astronomica Italiana*, 77:1095–+, 2006. 13

A. Z. Bonanos, K. Z. Stanek, R. P. Kudritzki, L. M. Macri, D. D. Sasselov, J. Kaluzny, P. B. Stetson, D. Bersier, F. Bresolin, T. Matheson, B. J. Mochejska, N. Przybilla, A. H. Szentgyorgyi, J. Tonry, and G. Torres. The First DIRECT Distance Determination to a Detached Eclipsing Binary in M33. *Astrophysical Journal*, 652:313–322, November 2006.

A. Brunthaler, M. J. Reid, H. Falcke, L. J. Greenhill, and C. Henkel. The Geometric Distance and Proper Motion of the Triangulum Galaxy (M33). *Science*, 307:1440–1443, March 2005.

R. G. Carlberg, H. B. Richer, A. W. McConnachie, M. Irwin, R. A. Ibata, A. L. Dotter, S. Chapman, M. Fardal, A. M. N. Ferguson, G. F. Lewis, J. F. Navarro, T. H. Puzia, and D. Valls-Gabaud. Density Variations in the NW Star Stream of M31. *Astrophysical Journal*, 731:124, April 2011. 25

B. Chaboyer, P. Demarque, P. J. Kernan, and L. M. Krauss. The Age of Globular Clusters in Light of Hipparcos: Resolving the Age Problem? *Astrophysical Journal*, 494:96–+, February 1998. 6

B. Chaboyer, P. Demarque, and A. Sarajedini. Globular Cluster Ages and the Formation of the Galactic Halo. *Astrophysical Journal*, 459:558–+, March 1996. 6

R. Cockcroft, W. E. Harris, A. M. N. Ferguson, A. Huxor, R. Ibata, M. J. Irwin, A. W. McConnachie, K. A. Woodley, S. C. Chapman, G. F. Lewis, and T. H. Puzia. The M33 Globular Cluster System with PAndAS Data: the Last Outer Halo Cluster? *Astrophysical Journal*, 730:112, April 2011. 25

G. S. Da Costa, T. E. Armandroff, N. Caldwell, and P. Seitzer. The Dwarf Spheroidal Companions to M31: WFPC2 Observations of Andromeda I. *Astronomical Journal*, 112:2576, December 1996.

G. S. Da Costa, T. E. Armandroff, N. Caldwell, and P. Seitzer. The Dwarf Spheroidal Companions to M31: WFPC2 Observations of Andromeda II. *Astronomical Journal*, 119: 705–726, February 2000.

J. H. J. de Bruijne. Science performance of Gaia, ESA's space-astrometry mission. *Astrophysics and Space Science*, 341:31–41, September 2012. 14

G. M. De Silva, K. C. Freeman, J. Bland-Hawthorn, M. Asplund, and M. S. Bessell. Chemically Tagging the HR 1614 Moving Group. *Astronomical Journal*, 133:694–704, February 2007. 5

G. de Vaucouleurs. Photoelectric photometry of the Andromeda nebula in the UBV system. *Astrophysical Journal*, 128:465, November 1958.

G. de Vaucouleurs, A. de Vaucouleurs, H. G. Corwin, R. J. Buta, G. Paturel, and P. Fouqué. *Third reference catalogue of bright galaxies*. Third Reference Catalogue of Bright Galaxies. Springer-Verlag, 1991.

J. Diemand, M. Kuhlen, and P. Madau. Dark Matter Substructure and Gamma-Ray Annihilation in the Milky Way Halo. *Astrophysical Journal*, 657:262–270, March 2007. 18, 19

O. J. Eggen and A. R. Sandage. Stellar groups, IV. The Groombridge 1830 group of high velocity stars and its relation to the globular clusters. *Monthly Notices of the Royal Astronomical Society*, 119:255–+, 1959. 4

ESA. The Hipparcos and Tycho Catalogues (ESA 1997). *VizieR Online Data Catalog*, 1239: 0–+, February 1997. 10

A. M. N. Ferguson, J. S. Gallagher, and R. F. G. Wyse. On the Nature of Andromeda IV. *Astronomical Journal*, 120:821–832, August 2000.

A. M. N. Ferguson, M. J. Irwin, R. A. Ibata, G. F. Lewis, and N. R. Tanvir. Evidence for Substructure in the Halo of M31. In *American Astronomical Society Meeting Abstracts #200*, volume 34 of *Bulletin of the American Astronomical Society*, page 716, May 2002. 24

K. Freeman and J. Bland-Hawthorn. The New Galaxy: Signatures of Its Formation. *Annual Reviews in Astronomy and Astrophysics*, 40:487–537, 2002. 2, 3, 4, 5, 6, 7

M. Geha, R. P. van der Marel, P. Guhathakurta, K. M. Gilbert, J. Kalirai, and E. N. Kirby. Local Group Dwarf Elliptical Galaxies. II. Stellar Kinematics to Large Radii in NGC 147 and NGC 185. *Astrophysical Journal*, 711:361–373, March 2010.

Y. M. Georgelin and Y. P. Georgelin. The spiral structure of our Galaxy determined from H II regions. *Astronomy and Astrophysics*, 49:57–79, May 1976. 2

G. Gilmore and N. Reid. New light on faint stars. III - Galactic structure towards the South Pole and the Galactic thick disc. *Monthly Notices of the Royal Astronomical Society*, 202: 1025–1047, March 1983. 2

T. M. Girard, D. I. Dinescu, W. F. van Altena, I. Platais, D. G. Monet, and C. E. López. The Southern Proper Motion Program. III. A Near-Complete Catalog to V=17.5. *Astronomical Journal*, 127:3060–3071, May 2004. 12

N. Gouda, T. Yano, Y. Yamada, Y. Kobayashi, T. Tsujimoto, and The Jasmine Working Group. Japan Astrometry Satellite Mission for Infrared Exploration (JASMINE). In C. Turon, K. S. O'Flaherty, and M. A. C. Perryman, editors, *The Three-Dimensional Universe with Gaia*, volume 576 of *ESA Special Publication*, pages 77–+, January 2005. 12

D. O. Gough. Lessons Learned From Solar Oscillations. In T. von Hippel, C. Simpson, and N. Manset, editors, *Astrophysical Ages and Times Scales*, volume 245 of *Astronomical Society of the Pacific Conference Series*, pages 31–+, 2001. 5

P. C. Gregory. *Bayesian Logical Data Analysis for the Physical Sciences: A Comparative Approach with 'Mathematica' Support*. Cambridge University Press, 2005. 43

S. Gwyn. The MegaCam ugriz filter set. 2010. URL http://www2.cadc-ccda.hia-iha.nrc-cnrc.gc.ca/megapipe/docs/filters.html. 24, 26

F. Hammer, Y. B. Yang, J. L. Wang, M. Puech, H. Flores, and S. Fouquet. Does M31 Result from an Ancient Major Merger? *Astrophysical Journal*, 725:542–555, December 2010.

F. D. A. Hartwick. The Structure of the Outer Halo of the Galaxy and its Relationship to Nearby Large-Scale Structure. *Astronomical Journal*, 119:2248–2253, May 2000.

W. K. Hastings. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika*, 57(1):97–109, 1970. 43

P. W. Hodge. Distribution of luminosity and color in the galaxy NGC 185. *Astronomical Journal*, 68:691, November 1963.

E. Høg, C. Fabricius, V. V. Makarov, S. Urban, T. Corbin, G. Wycoff, U. Bastian, P. Schwekendiek, and A. Wicenec. The Tycho-2 catalogue of the 2.5 million brightest stars. *Astronomy and Astrophysics*, 355:L27–L30, March 2000. 10

E. Holmberg. A study of physical groups of galaxies. *Arkiv for Astronomi*, 5:305–343, 1969.

F. Hoyle and M. Schwarzschild. On the Evolution of Type II Stars. *Astrophysical Journal Supplement Series*, 2:1, June 1955. 28

E. P. Hubble. Cepheids in Spiral Nebulae. *Popular Astronomy*, 33:252, 1925. 27

R. Ibata, S. Chapman, A. M. N. Ferguson, M. Irwin, G. Lewis, and A. McConnachie. Taking measure of the Andromeda halo: a kinematic analysis of the giant stream surrounding M31. *Monthly Notices of the Royal Astronomical Society*, 351:117–124, June 2004. 22

R. Ibata, M. Irwin, G. Lewis, A. M. N. Ferguson, and N. Tanvir. A giant stream of metal-rich stars in the halo of the galaxy M31. *Nature*, 412:49–52, July 2001a. 24

R. Ibata, G. F. Lewis, M. Irwin, E. Totten, and T. Quinn. Great Circle Tidal Streams: Evidence for a Nearly Spherical Massive Dark Halo around the Milky Way. *Astrophysical Journal*, 551:294–311, April 2001b. 3, 20, 21, 22, 23

R. Ibata, N. F. Martin, M. Irwin, S. Chapman, A. M. N. Ferguson, G. F. Lewis, and A. W. McConnachie. The Haunted Halos of Andromeda and Triangulum: A Panorama of Galaxy Formation in Action. *Astrophysical Journal*, 671:1591–1623, December 2007. 24

R. A. Ibata, G. F. Lewis, M. J. Irwin, and L. Cambrésy. Substructure of the outer Galactic halo from the 2-Micron All-Sky Survey. *Monthly Notices of the Royal Astronomical Society*, 332:921–927, June 2002a. 9, 21

R. A. Ibata, G. F. Lewis, M. J. Irwin, and T. Quinn. Uncovering cold dark matter halo substructure with tidal streams. *Monthly Notices of the Royal Astronomical Society*, 332: 915–920, June 2002b. 20

I. Iben, Jr. and A. Renzini. Asymptotic giant branch evolution and beyond. *Annual Reviews in Astronomy and Astrophysics*, 21:271–342, 1983. xvii, 28

M. J. Irwin. Andromeda XXX/ Cassiopeia II discovery paper - In preparation. 2012. 25

M. J. Irwin, A. M. N. Ferguson, A. P. Huxor, N. R. Tanvir, R. A. Ibata, and G. F. Lewis. Andromeda XVII: A New Low-Luminosity Satellite of M31. *Astrophysical Journal Letters*, 676:L17–L20, March 2008.

K. V. Johnston, D. N. Spergel, and C. Haydn. How Lumpy Is the Milky Way's Dark Matter Halo? *Astrophysical Journal*, 570:656–664, May 2002. 20, 21

G. Jungman, M. Kamionkowski, and K. Griest. Supersymmetric dark matter. *Physics Reports*, 267:195–373, March 1996. 17

M. Jurić et al. The Milky Way Tomography with SDSS. I. Stellar Number Density Distribution. *Astrophysical Journal*, 673:864–914, February 2008. 2

S. C. Keller, B. P. Schmidt, M. S. Bessell, P. G. Conroy, P. Francis, A. Granlund, E. Kowald, A. P. Oates, T. Martin-Jones, T. Preston, P. Tisserand, A. Vaccarella, and M. F. Waterson. The SkyMapper Telescope and The Southern Sky Survey. *Publications of the Astronomical Society of Australia*, 24:1–12, May 2007. 9

A. Klypin, H. Zhao, and R. S. Somerville. ΛCDM-based Models for the Milky Way and M31. I. Dynamical Models. *Astrophysical Journal*, 573:597–613, July 2002. 22

A. Knebe, S. P. D. Gill, B. K. Gibson, G. F. Lewis, R. A. Ibata, and M. A. Dopita. Anisotropy in the Distribution of Satellite Galaxy Orbits. *Astrophysical Journal*, 603:7–11, March 2004.

A. Koch and E. K. Grebel. The Anisotropic Distribution of M31 Satellite Galaxies: A Polar Great Plane of Early-type Companions. *Astronomical Journal*, 131:1405–1415, March 2006. xix

A. V. Kravtsov, O. Y. Gnedin, and A. A. Klypin. The Tumultuous Lives of Galactic Dwarfs and the Missing Satellites Problem. *Astrophysical Journal*, 609:482–497, July 2004. 18

P. Kroupa, C. Theis, and C. M. Boily. The great disk of Milky-Way satellites and cosmological sub-structures. *Astronomy and Astrophysics*, 431:517–521, February 2005.

P.S. Laplace. *Théorie analytique des probabilités*. Courcier, 1812. URL `http://books.google.com.au/books?id=fjAVAAAAQAAJ`. 47

M. G. Lee, W. L. Freedman, and B. F. Madore. The Tip of the Red Giant Branch as a Distance Indicator for Resolved Galaxies. *Astrophysical Journal*, 417:553, November 1993. xvii, 28, 32, 34

B. Letarte, S. C. Chapman, M. Collins, R. A. Ibata, M. J. Irwin, A. M. N. Ferguson, G. F. Lewis, N. Martin, A. McConnachie, and N. Tanvir. A Keck/DEIMOS spectroscopic survey of the faint M31 satellites AndXV and AndXVI. *Monthly Notices of the Royal Astronomical Society*, 400:1472–1478, December 2009.

M. R. Lovell, V. R. Eke, C. S. Frenk, and A. Jenkins. The link between galactic satellite orbits and subhalo accretion. *Monthly Notices of the Royal Astronomical Society*, 413: 3013–3021, June 2011.

D. Lynden-Bell. Dwarf galaxies and globular clusters in high velocity hydrogen streams. *Monthly Notices of the Royal Astronomical Society*, 174:695–710, March 1976.

D. Lynden-Bell. The Fornax-Leo-Sculptor system. *The Observatory*, 102:202–208, October 1982. xix

D. Lynden-Bell and R. M. Lynden-Bell. Ghostly streams from the formation of the Galaxy's halo. *Monthly Notices of the Royal Astronomical Society*, 275:429–442, July 1995.

A. D. Mackey, A. P. Huxor, A. M. N. Ferguson, M. J. Irwin, N. R. Tanvir, A. W. McConnachie, R. A. Ibata, S. C. Chapman, and G. F. Lewis. Evidence for an Accretion Origin for the Outer Halo Globular Cluster System of M31. *Astrophysical Journal Letters*, 717:L11–L16, July 2010. 25

P. Madau and J. Silk. Population III and the near-infrared background excess. *Monthly Notices of the Royal Astronomical Society*, 359:L37–L41, May 2005. 18

B. F. Madore, V. Mager, and W. L. Freedman. Sharpening the Tip of the Red Giant Branch. *Astrophysical Journal*, 690:389–393, January 2009.

S. R. Majewski, R. L. Beaton, R. J. Patterson, J. S. Kalirai, M. C. Geha, R. R. Muñoz, M. S. Seigar, P. Guhathakurta, K. M. Gilbert, R. M. Rich, J. S. Bullock, and D. B. Reitzel. Discovery of Andromeda XIV: A Dwarf Spheroidal Dynamical Rogue in the Local Group? *Astrophysical Journal Letters*, 670:L9–L12, November 2007.

D. Makarov, L. Makarova, L. Rizzi, R. B. Tully, A. E. Dolphin, S. Sakai, and E. J. Shaya. Tip of the Red Giant Branch Distances. I. Optimization of a Maximum Likelihood Algorithm. *Astronomical Journal*, 132:2729–2742, December 2006.

N. F. Martin, R. A. Ibata, M. J. Irwin, S. Chapman, G. F. Lewis, A. M. N. Ferguson, N. Tanvir, and A. W. McConnachie. Discovery and analysis of three faint dwarf galaxies and a globular cluster in the outer halo of the Andromeda galaxy. *Monthly Notices of the Royal Astronomical Society*, 371:1983–1991, October 2006.

N. F. Martin, A. W. McConnachie, M. Irwin, L. M. Widrow, A. M. N. Ferguson, R. A. Ibata, J. Dubinski, A. Babul, S. Chapman, M. Fardal, G. F. Lewis, J. Navarro, and R. M. Rich. PAndAS' CUBS: Discovery of Two New Dwarf Galaxies in the Surroundings of the

Andromeda and Triangulum Galaxies. *Astrophysical Journal*, 705:758–765, November 2009. 25

A. W. McConnachie. Satellites and Substructure in the Local Group. *PhD Thesis; Institute of Astronomy, University of Cambridge*, 2005.

A. W. McConnachie. The Pan-Andromeda Archaeological Survey: Galaxy Formation In The Near-Field. In *American Astronomical Society Meeting Abstracts #213*, volume 41 of *Bulletin of the American Astronomical Society*, page #307.05, January 2009.

A. W. McConnachie. PAndAS: The Pan-Andromeda Archaeological Survey. In *StSci 2010 May Symposium*, Stellar Populations in the Cosmological Context, May 2010. 25

A. W. McConnachie, A. Huxor, N. F. Martin, M. J. Irwin, S. C. Chapman, G. Fahlman, A. M. N. Ferguson, R. A. Ibata, G. F. Lewis, H. Richer, and N. R. Tanvir. A Trio of New Local Group Galaxies with Extreme Properties. *Astrophysical Journal*, 688:1009–1020, December 2008. 25

A. W. McConnachie and M. J. Irwin. The satellite distribution of M31. *Monthly Notices of the Royal Astronomical Society*, 365:902–914, January 2006.

A. W. McConnachie, M. J. Irwin, A. M. N. Ferguson, R. A. Ibata, G. F. Lewis, and N. Tanvir. Determining the location of the tip of the red giant branch in old stellar populations: M33, Andromeda I and II. *Monthly Notices of the Royal Astronomical Society*, 350:243–252, May 2004.

A. W. McConnachie, M. J. Irwin, A. M. N. Ferguson, R. A. Ibata, G. F. Lewis, and N. Tanvir. Distances and metallicities for 17 Local Group galaxies. *Monthly Notices of the Royal Astronomical Society*, 356:979–997, January 2005.

A. W. McConnachie, M. J. Irwin, R. A. Ibata, J. Dubinski, L. M. Widrow, N. F. Martin, P. Côté, A. L. Dotter, J. F. Navarro, A. M. N. Ferguson, T. H. Puzia, G. F. Lewis, A. Babul, P. Barmby, O. Bienaymé, S. C. Chapman, R. Cockcroft, M. L. M. Collins, M. A. Fardal, W. E. Harris, A. Huxor, A. D. Mackey, J. Peñarrubia, R. M. Rich, H. B. Richer, A. Siebert,

N. Tanvir, D. Valls-Gabaud, and K. A. Venn. The remnants of galaxy formation from a panoramic survey of the region around M31. *Nature*, 461:66–69, September 2009. xviii, 24

C. F. McKee and J. C. Tan. Massive star formation in 100,000 years from turbulent and pressurized molecular clouds. *Nature*, 416:59–61, March 2002. 5

A. McWilliam and R. M. Rich. The first detailed abundance analysis of Galactic bulge K giants in Baade's window. *Astrophysical Journal Supplement Series*, 91:749–791, April 1994. 2

B. Méndez, M. Davis, J. Moustakas, J. Newman, B. F. Madore, and W. L. Freedman. Deviations from the Local Hubble Flow. I. The Tip of the Red Giant Branch as a Distance Indicator. *Astronomical Journal*, 124:213–233, July 2002. xviii, 28, 33

R. B. Metcalf and P. Madau. Compound Gravitational Lensing as a Probe of Dark Matter Substructure within Galaxy Halos. *Astrophysical Journal*, 563:9–20, December 2001. 19

N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of State Calculations by Fast Computing Machines. *Journal of Chemical Physics*, 21: 1087–1092, June 1953. 43

M. Metz, P. Kroupa, and H. Jerjen. The spatial distribution of the Milky Way and Andromeda satellite galaxies. *Monthly Notices of the Royal Astronomical Society*, 374:1125–1145, January 2007.

M. Metz, P. Kroupa, and H. Jerjen. Discs of satellites: the new dwarf spheroidals. *Monthly Notices of the Royal Astronomical Society*, 394:2223–2228, April 2009a.

M. Metz, P. Kroupa, and N. I. Libeskind. The Orbital Poles of Milky Way Satellite Galaxies: A Rotationally Supported Disk of Satellites. *Astrophysical Journal*, 680:287–294, June 2008.

M. Metz, P. Kroupa, C. Theis, G. Hensler, and H. Jerjen. Did the Milky Way Dwarf Satellites Enter The Halo as a Group? *Astrophysical Journal*, 697:269–274, May 2009b.

S. P. Meyn and R. L. Tweedie. *Markov chains and stochastic stability*. Communications and Control Engineering Series. Springer-Verlag London Ltd., London, 1993. ISBN 3-540-19832-6. URL http://probability.ca/MT/. 43

D. Monet, B. Canzian, H. Harris, N. Reid, A. Rhodes, and S. Sell. The PMM USNO-A1.0 Catalogue (Monet 1997). *VizieR Online Data Catalog*, 1243:0–+, July 1998. 11

D. G. Monet. The 526,280,881 Objects In The USNO-A2.0 Catalog. In *Bulletin of the American Astronomical Society*, volume 30 of *Bulletin of the American Astronomical Society*, pages 1427–+, December 1998. 11

B. Moore, J. Diemand, P. Madau, M. Zemp, and J. Stadel. Globular clusters, satellite galaxies and stellar haloes from early dark matter peaks. *Monthly Notices of the Royal Astronomical Society*, 368:563–570, May 2006. 18

J. Mould and J. Kristian. The dwarf spheroidal galaxy Andromeda I. *Astrophysical Journal*, 354:438–445, May 1990.

J. F. Navarro, M. G. Abadi, and M. Steinmetz. Tidal Torques and the Orientation of Nearby Disk Galaxies. *Astrophysical Journal Letters*, 613:L41–L44, September 2004.

B. Nordstrom, M. Mayor, J. Andersen, J. Holmberg, F. Pont, B. R. Jorgensen, E. H. Olsen, S. Udry, and N. Mowlavi. Geneva-Copenhagen Survey of Solar neighbourhood (Nordstrom+, 2004). *VizieR Online Data Catalog*, 5117:0–+, May 2004. 13

C. Palma, S. R. Majewski, and K. V. Johnston. On the Distribution of Orbital Poles of Milky Way Satellites. *Astrophysical Journal*, 564:736–761, January 2002.

M. S. Pawlowski, P. Kroupa, G. Angus, K. S. de Boer, B. Famaey, and G. Hensler. Filamentary accretion cannot explain the orbital poles of the Milky Way satellites. *Monthly Notices of the Royal Astronomical Society*, 424:80–92, July 2012a.

M. S. Pawlowski, J. Pflamm-Altenburg, and P. Kroupa. The VPOS: a vast polar structure of satellite galaxies, globular clusters and streams around the Milky Way. *Monthly Notices of the Royal Astronomical Society*, 423:1109–1126, June 2012b. xix

T. Pearson. PGPLOT: Device-independent Graphics Package for Simple Scientific Graphs. *Astrophysics Source Code Library*, page 3002, March 2011. Note: The PGPLOT Subroutine library was commenced in 1983. A list of the subroutines can be found at http://www.astro.caltech.edu/ tjp/pgplot/subroutines.html. vii

S. F. Portegies Zwart. The Lost Siblings of the Sun. *Astrophysical Journal Letters*, 696: L13–L16, May 2009. 5

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes in FORTRAN. The art of scientific computing*. 1992. viii

P. J. Quinn and J. Goodman. Sinking satellites of spiral systems. *Astrophysical Journal*, 309: 472–495, October 1986.

B. Reipurth. The Early Sun: Evolution and Dynamic Environment. In A. N. Krot, E. R. D. Scott, and B. Reipurth, editors, *Chondrites and the Protoplanetary Disk*, volume 341 of *Astronomical Society of the Pacific Conference Series*, pages 54–+, December 2005. 5

J. C. Richardson, M. J. Irwin, A. W. McConnachie, N. F. Martin, A. L. Dotter, A. M. N. Ferguson, R. A. Ibata, S. C. Chapman, G. F. Lewis, N. R. Tanvir, and R. M. Rich. PAndAS' Progeny: Extending the M31 Dwarf Galaxy Cabal. *Astrophysical Journal*, 732:76, May 2011. 25

T. Riehm, E. Zackrisson, O. Möller, E. Mörtsell, and K. Wiik. Prospects for CDM subhalo detection using high angular resolution observations. *Journal of Physics Conference Series*, 131(1):012045–+, October 2008. 19

A. G. Riess, J. Fliri, and D. Valls-Gabaud. Cepheid Period-Luminosity Relations in the Near-infrared and the Distance to M31 from the Hubble Space Telescope Wide Field Camera 3. *Astrophysical Journal*, 745:156, February 2012.

L. Rizzi, R. B. Tully, D. Makarov, L. Makarova, A. E. Dolphin, S. Sakai, and E. J. Shaya. Tip of the Red Giant Branch Distances. II. Zero-Point Calibration. *Astrophysical Journal*, 661:815–829, June 2007.

C. M. Rockosi, M. Odenkirchen, E. K. Grebel, W. Dehnen, K. M. Cudworth, J. E. Gunn, D. G. York, J. Brinkmann, G. S. Hennessy, and Ž. Ivezić. A Matched-Filter Analysis of the Tidal Tails of the Globular Cluster Palomar 5. *Astronomical Journal*, 124:349–363, July 2002. xviii

S. Ruphy, A. C. Robin, N. Epchtein, E. Copet, E. Bertin, P. Fouque, and F. Guglielmo. New determination of the disc scale length and the radial cutoff in the anticenter with DENIS data. *Astronomy and Astrophysics*, 313:L21–L24, September 1996. 2

V. Sahni. Dark Matter and Dark Energy. In E. Papantonopoulos, editor, *Lecture Notes in Physics, Berlin Springer Verlag*, volume 653 of *Lecture Notes in Physics, Berlin Springer Verlag*, pages 141–+, 2004. 15, 16, 17, 18

S. Sakai, B. F. Madore, and W. L. Freedman. Tip of the Red Giant Branch Distances to Galaxies. III. The Dwarf Galaxy Sextans A. *Astrophysical Journal*, 461:713, April 1996. 32, 34

M. Salaris and S. Cassisi. The 'tip' of the red giant branch as a distance indicator: results from evolutionary models. *Monthly Notices of the Royal Astronomical Society*, 289:406–414, August 1997. 27

M. Salaris, S. Cassisi, and A. Weiss. Red Giant Branch Stars: The Theoretical Framework. *Publications of the Astronomical Society of the Pacific*, 114:375–402, April 2002. 27

D. J. Schlegel, D. P. Finkbeiner, and M. Davis. Maps of Dust Infrared Emission for Use in Estimation of Reddening and Cosmic Microwave Background Radiation Foregrounds. *Astrophysical Journal*, 500:525, June 1998.

SDSS-III Collaboration, :, C. P. Ahn, R. Alexandroff, C. Allende Prieto, S. F. Anderson, T. Anderton, B. H. Andrews, É. A. S. Bailey, R. Barnes, and et al. The Ninth Data Release of the Sloan Digital Sky Survey: First Spectroscopic Data from the SDSS-III Baryon Oscillation Spectroscopic Survey. *ArXiv e-prints*, July 2012. 8

A. Siebert, M. E. K. Williams, A. Siviero, W. Reid, C. Boeche, M. Steinmetz, J. Fulbright, U. Munari, T. Zwitter, F. G. Watson, R. F. G. Wyse, R. S. de Jong, H. Enke, B. Anguiano,

D. Burton, C. J. P. Cass, K. Fiegert, M. Hartley, A. Ritter, K. S. Russel, M. Stupar, O. Bienaymé, K. C. Freeman, G. Gilmore, E. K. Grebel, A. Helmi, J. F. Navarro, J. Binney, J. Bland-Hawthorn, R. Campbell, B. Famaey, O. Gerhard, B. K. Gibson, G. Matijevič, Q. A. Parker, G. M. Seabroke, S. Sharma, M. C. Smith, and E. Wylie-de Boer. The RAdial Velocity Experiment (RAVE): Third Data Release. *Astronomical Journal*, 141:187, June 2011. 14

M. F. Skrutskie et al. The Two Micron All Sky Survey (2MASS). *Astronomical Journal*, 131:1163–1183, February 2006. 9

V. Springel, J. Wang, M. Vogelsberger, A. Ludlow, A. Jenkins, A. Helmi, J. F. Navarro, C. S. Frenk, and S. D. M. White. The Aquarius Project: the subhaloes of galactic haloes. *Monthly Notices of the Royal Astronomical Society*, 391:1685–1711, December 2008.

K. Z. Stanek and P. M. Garnavich. Distance to M31 with the Hubble Space Telescope and HIPPARCOS Red Clump Stars. *Astrophysical Journal Letters*, 503:L131, August 1998.

E. Starkenburg, A. Helmi, G. De Lucia, Y.-S. Li, J. F. Navarro, A. S. Font, C. S. Frenk, V. Springel, C. A. Vera-Ciro, and S. D. M. White. The satellites of the Milky Way - Insights from semi-analytic modelling in a LambdaCDM cosmology. *ArXiv e-prints*, May 2012.

M. Steinmetz et al. The Radial Velocity Experiment (RAVE): First Data Release. *Astronomical Journal*, 132:1645–1668, October 2006. 14

G. A. Tammann, A. Sandage, and B. Reindl. The expansion field: the value of H_0. *The Astronomy and Astrophysics Review*, 15:289–331, July 2008.

M. B. Taylor. TOPCAT Memorie della Societa Astronomica Italiana STIL: Starlink Table/VOTable Processing Software. In P. Shopbell, M. Britton, and R. Ebert, editors, *Astronomical Data Analysis Software and Systems XIV*, volume 347 of *Astronomical Society of the Pacific Conference Series*, page 29, December 2005. 91

E. J. Totten and M. J. Irwin. The APM survey for cool carbon stars in the Galactic halo. I. *Monthly Notices of the Royal Astronomical Society*, 294:1–+, February 1998. 21

S. E. Urban, T. E. Corbin, and G. L. Wycoff. The ACT Reference Catalog. *Astronomical Journal*, 115:2161–2166, May 1998. 11

J. P. Vallée. The Spiral Arms and Interarm Separation of the Milky Way: An Updated Statistical Study. *Astronomical Journal*, 130:569–575, August 2005. 3

S. van den Bergh. Search for Faint Companions to M31. *Astrophysical Journal Letters*, 171: L31, January 1972.

S. van den Bergh. The Dwarf Satellites of M31 and the Galaxy. *Astronomical Journal*, 132: 1571–1574, October 2006. 3, 18

P. T. Wallace. The SLALIB Library. In D. R. Crabtree, R. J. Hanisch, and J. Barnes, editors, *Astronomical Data Analysis Software and Systems III*, volume 61 of *Astronomical Society of the Pacific Conference Series*, page 481, 1994. ix

X. X. Xue, H. W. Rix, G. Zhao, P. Re Fiorentin, T. Naab, M. Steinmetz, F. C. van den Bosch, T. C. Beers, Y. S. Lee, E. F. Bell, C. Rockosi, B. Yanny, H. Newberg, R. Wilhelm, X. Kang, M. C. Smith, and D. P. Schneider. The Milky Way's Circular Velocity Curve to 60 kpc and an Estimate of the Dark Matter Halo Mass from the Kinematics of ˜2400 SDSS Blue Horizontal-Branch Stars. *Astrophysical Journal*, 684:1143–1158, September 2008. 3

S.-C. Yang and A. Sarajedini. HST/WFPC2 imaging of the dwarf satellites And XI and And XIII: horizontal branch morphology and RR Lyraes. *Monthly Notices of the Royal Astronomical Society*, 419:1362–1375, January 2012.

B. Yanny et al. SEGUE: A Spectroscopic Survey of 240,000 Stars with g = 14-20. *Astronomical Journal*, 137:4377–4399, May 2009. 13

D. G. York et al. The Sloan Digital Sky Survey: Technical Summary. *Astronomical Journal*, 120:1579–1587, September 2000. 8

N. Zacharias, S. E. Urban, M. I. Zacharias, G. L. Wycoff, D. M. Hall, D. G. Monet, and T. J. Rafferty. The Second US Naval Observatory CCD Astrograph Catalog (UCAC2). *Astronomical Journal*, 127:3043–3059, May 2004. 11

A. R. Zentner, A. V. Kravtsov, O. Y. Gnedin, and A. A. Klypin. The Anisotropic Distribution of Galactic Satellites. *Astrophysical Journal*, 629:219–232, August 2005. xix

M. Zoccali and G. Piotto. Comparison between observed and theoretical Red Giant Branch luminosity functions of galactic globular clusters. *Astronomy and Astrophysics*, 358:943–955, June 2000. 27

D. B. Zucker, A. Y. Kniazev, D. Martínez-Delgado, E. F. Bell, H.-W. Rix, E. K. Grebel, J. A. Holtzman, R. A. M. Walterbos, C. M. Rockosi, D. G. York, J. C. Barentine, H. Brewington, J. Brinkmann, M. Harvanek, S. J. Kleinman, J. Krzesinski, D. Long, E. H. Neilsen, Jr., A. Nitta, and S. A. Snedden. Andromeda X, a New Dwarf Spheroidal Satellite of M31: Photometry. *Astrophysical Journal Letters*, 659:L21–L24, April 2007.

F. Zwicky. Die Rotverschiebung von extragalaktischen Nebeln. *Helvetica Physica Acta*, 6: 110–127, 1933. 15

Update Concerning Acknowledgements and References Sections

I would like to thank the American Astronomical Society (AAS) for granting permission to reproduce the published versions of papers I and II in chapters 3 and 4 respectively.

Paper I:

A. R. Conn, G. F. Lewis, R. A. Ibata, Q. A. Parker, D. B. Zucker, A. W. McConnachie, N. F. Martin, M. J. Irwin, N. Tanvir, M. A. Fardal, and A. M. N. Ferguson. A Bayesian Approach to Locating the Red Giant Branch Tip Magnitude. I. *Astrophysical Journal*, 740:69, October 2011. (As referenced in the "List of Publications" section)

Paper II:

A. R. Conn, R. A. Ibata, G. F. Lewis, Q. A. Parker, D. B. Zucker, N. F. Martin, A. W. McConnachie, M. J. Irwin, N. Tanvir, M. A. Fardal, A. M. N. Ferguson, S. C. Chapman, and D. Valls-Gabaud. A Bayesian Approach to Locating the Red Giant Branch Tip Magnitude. II. Distances to the Satellites of M31. *Astrophysical Journal*, 758:11, October 2012. (As referenced in the "List of Publications" section)

In addition, **Paper III** (as referenced in the "List of Publications" section and presented in Chapter 5) has now been published in the *Astrophysical Journal* with the following bibliographic details:

A. R. Conn, G. F. Lewis, R. A. Ibata, Q. A. Parker, D. B. Zucker, A. W. McConnachie, N. F. Martin, D. Valls-Gabaud, N. Tanvir, M. J. Irwin, A. M. N. Ferguson, and S. C. Chapman. The Three-dimensional Structure of the M31 Satellite System; Strong Evidence for an Inhomogeneous Distribution of Satellites. *Astrophysical Journal*, 766:120, April 2013.

Note also that Ibata et al. 2012 (ILC12) referenced in Paper III has now been published in *nature*, with the following bibliographic details:

R. A. Ibata, G. F. Lewis, A. R. Conn, M. J. Irwin, A. W. McConnachie, S. C. Chapman, M. L. Collins, M. Fardal, A. M. N. Ferguson, N. G. Ibata, A. D. Mackey, N. F. Martin, J. Navarro, R. M. Rich, D. Valls-Gabaud, and L. M. Widrow. A vast, thin plane of corotating dwarf galaxies orbiting the Andromeda galaxy. *Nature*, 493:62–65, January 2013.