Budget d'erreur en optique adaptative : Simulation numérique haute performance et modélisation dans la perspective des ELT

Florian Moura Ferreira

Université Sorbonne Paris Cité
Université Paris Diderot

Ecole doctorale 127
Astronomie & Astrophysique d'Île-de-France

Laboratoire d'Etudes Spatiales et d'Instrumentation en Astrophysique

Thèse de doctorat

en Instrumentation pour l'astronomie et l'astropysique
de l'Université Sorbonne Paris Cité
préparée à l'Université Paris Diderot

Florian Moura Ferreira

Budget d'erreur en optique adaptative : Simulation numérique haute performance et modélisation dans la perspective des ELT

Thèse dirigée par Gérard Rousset & Eric Gendron

présentée et soutenue publiquement à Meudon le 11 octobre 2018

Président du jury :	Jacques Le Bourlot	*Professeur, LERMA*
Rapporteurs :	Jean-Pierre Véran	*Senior Scientist, NRC-CNRC*
	Marcel Carbillet	*Maître de conférence, Université de Nice*
Examinateurs :	Caroline Kulcsár	*Professeure, Institut d'optique Graduate School*
	Cyril Petit	*Docteur - Ingénieur de recherche, ONERA*
Invité :	Damien Gratadour	*Maître de conférence, LESIA*
Directeur :	Gérard Rousset	*Professeur, LESIA*
Co-directeur :	Éric Gendron	*Astronome adjoint, LESIA*

Résumé

D'ici quelques années, une nouvelle classe de télescopes verra le jour : celle des télescopes géants. Ceux-ci se caractériseront par un diamètre supérieur à 20m, et jusqu'à 39m pour le représentant européen, l'Extremely Large Telescope (ELT). Seulement, l'atmosphère terrestre vient dégrader sévèrement les images obtenues lors d'observations au sol : la résolution de ces télescopes est alors réduite à celle d'un télescope amateur de quelques dizaines de centimètres de diamètre.

L'optique adaptative (OA) devient alors essentielle. Cette dernière permet de corriger en temps-réel les perturbations induites par l'atmosphère et de retrouver la résolution théorique du télescope. Néanmoins, les systèmes d'OA ne sont pas exempts de tout défaut, et une erreur résiduelle persiste sur le front d'onde (FO) et impacte la qualité des images obtenues. Cette dernière est dépendante de la Fonction d'Étalement de Point (FEP) de l'instrument utilisé, et la FEP d'un système d'OA dépend elle-même de l'erreur résiduelle de FO. L'identification et la compréhension des sources d'erreurs est alors primordiale.

Dans la perspective de ces télescopes géants, le dimensionnement des systèmes d'OA nécessaires devient tel que ces derniers représentent un challenge technologique et technique. L'un des aspects à considérer est la complexité numérique de ces systèmes. Dès lors, les techniques de calcul de haute performance deviennent nécessaires, comme la parallélisation massive. Le General Purpose Graphical Processing Unit (GPGPU) permet d'utiliser un processeur graphique à cette fin, celui-ci possédant plusieurs milliers de cœurs de calcul utilisables, contre quelques dizaines pour un processeur classique.

Dans ce contexte, cette thèse s'articule autour de trois parties. La première présente le développement de COMPASS, un outil de simulation haute performance bout-en-bout dédié à l'OA, notamment à l'échelle des ELT. Tirant pleinement parti des capacités de calcul des GPU, COMPASS permet alors de simuler une OA ELT en quelques minutes. La seconde partie fait état du développement de ROKET : un estimateur complet du budget d'erreur d'un système d'OA intégré à COMPASS, permettant ainsi d'étudier statistiquement les différentes sources d'erreurs et leurs éventuels liens. Enfin, des modèles analytiques des différentes sources d'erreur sont dérivés et permettent de proposer un algorithme d'estimation de la FEP. Les possibilités d'applications sur le ciel de cet algorithme sont également discutées.

Mots-clés : Optique adaptative ; ELT ; Calcul haute performance ; Simulation numérique ; Budget d'erreur ; FEP

Abstract

In a few years, a new class of giants telescopes will appear. The diameter of those telescope will be larger than 20m, up to 39m for the european Extremely Large Telescope (ELT). However, images obtained from ground-based observations are severely impacted by the atmosphere. Then, the resolution of those giants telescopes is equivalent to the one obtained with an amateur telescope of a few tens of centimeters of diameter.

Therefore, adaptive optics (AO) becomes essential as it aims to correct in real-time the disturbance due to the atmospherical turbulence and to retrieve the theoretical resolution of the telescope. Nevertheless, AO systems are not perfect : a wavefront residual error remains and still impacts the image quality. The latter is measured by the point spread function (PSF) of the system, and this PSF depends on the wavefront residual error. Hence, identifying and understanding the various contributors of the AO residual error is primordial.

For those extremely large telescopes, the dimensioning of their AO systems is challenging. In particular, the numerical complexity impacts the numerical simulation tools useful for the AO design. High performance computing techniques are needed, as such relying on massive parallelization. General Purpose Graphical Processing Unit (GPGPU) enables the use of GPU for this purpose. This architecture is suitable for massive parallelization as it leverages GPU's several thousand of cores, instead of a few tens for classical CPU.

In this context, this PhD thesis is composed of three parts. In the first one, it presents the development of COMPASS : a GPU-based high performance end-to-end simulation tool for AO systems that is suitable for ELT scale. The performance of the latter allows simulating AO systems for the ELT in a few minutes. In a second part, an error breakdown estimation tool, ROKET, is added to the end-to-end simulation in order to study the various contributors of the AO residual error. Finally, an analytical model is proposed for those error contributors, leading to a new way to estimate the PSF. Possible on-sky applications are also discussed.

Keywords : Adaptive optics ; ELT ; High performance computing ; Numerical simulation ; Error breakdown ; PSF

Remerciements

Je viens de terminer la rédaction de ce mémoire, et vient le moment (libérateur) d'écrire ces derniers paragraphes. Bien que plus informels que le mémoire, ce n'est pas nécessairement plus simple d'écrire ces remerciements que d'écrire un paragraphe technique. Bon, je me lance, et j'essaie de ne rien oublier.

Tout d'abord, comme le veut la tradition, mes premiers remerciements iront à mon directeur de thèse, Gérard, ainsi qu'à mes encadrants Éric et Damien. Si j'écris ces lignes aujourd'hui, c'est bien grâce à eux et je les en remercie chaleureusement. Gérard et Damien m'ont connu alors que je n'étais encore qu'un étudiant en école d'ingénieur, et ils m'ont mis le pied à l'étrier dès l'obtention de mon diplôme en m'offrant ma première expérience professionnelle dans la recherche, en tant qu'ingénieur. Expérience qui s'est par la suite transformée en cette thèse que je vous présente. Au-delà de ces considérations, leur disponibilité (malgré un agenda très chargé) et leurs conseils m'ont été précieux.

Si Gérard et Damien m'ont donné les bases en optique adaptative, Éric restera pour moi celui auprès duquel j'ai le plus appris. "Rico" (ou "riiiiiiiiiiiico" pour les intimes) a toujours su se rendre disponible pour répondre à la moindre de mes interrogations, et ce malgré sa masse de travail colossale. Il s'en est passé des après-midi entiers à discuter autour de ce budget d'erreur et des équations qui vont avec. Et toujours avec cette bonne humeur et son humour qui le caractérisent parfaitement. Bref, sans lui, cette thèse n'aurait sûrement pas été la même.

Mes prochains remerciements iront à Arnaud. Si les trois personnes précédentes m'ont beaucoup appris sur l'optique adaptative, "Nono" a été mon gourou de l'informatique, celui que je pouvais appeler au moindre message d'erreur de compilation incompréhensible. Là aussi nous avons passé quelques moment savoureux à s'arracher les cheveux sur des erreurs de segmentation... Mais bien avant ses qualités professionnelles indéniables, c'est bien l'homme, l'ami, que je tiens à saluer. Les discussions ciné, séries, voiture, jeux vidéo... Tout cela amène un environnement où il fait bon travailler (et mener une thèse). Et puis il a bien voulu partager son bureau lorsque le mien devint la demeure d'une fouine plutôt odorante...

Bien sûr, cette ambiance de travail est le fruit de toute l'équipe du bâtiment 12A (et anciennement bâtiment Lyot) : Fabrice (que j'ai fait beaucoup crié à force de modifier COMPASS), Tristan, Marie, Yann, Zoltan, Mathieu, Roderick... C'est réellement un plaisir de travailler avec vous, et vous faites en sorte que je ne rechigne pas à me lever le matin. J'ai toujours pensé que le travail qui nous convient le mieux est celui pour

lequel on sort du lit sans regrets, et vous participez tous à ce que cela soit vrai dans mon cas, alors merci.

Une pensée évidemment à l'équipe de doctorants qui s'est formée dans cette équipe : Lisa, Vincent, Nicolas, Pierre et Lucas. Si ce dernier est d'ores et déjà docteur (félicitations), Lisa en est au même stade que moi et les derniers ont encore quelques temps devant eux avant de devoir entamer la rédaction de ce mémoire. Je vous souhaite bien du courage, et merci pour ces moments de détente sur le canapé (dont il est bien difficile de sortir d'ailleurs) autour d'un café.

Enfin, un grand merci à ma famille pour leur soutien sans faille, et plus particulièrement à la femme qui partage ma vie depuis 12 ans maintenant. Je sais que ces derniers mois ont été difficiles pour elle, entre Enzo qui demande beaucoup d'attention, Noah qui lui fait le ventre bien rond et moi qui rentre tard du bureau pour pouvoir avancer sur la rédaction, elle est le pilier central sur lequel je peux me reposer en toute confiance. C'est simple : Enzo, Noah et toi, je vous aime.

Table des matières

Liste des figures

Liste des tableaux

Liste des abbréviations

API	Application Programming Interface
ASO	Analyseur de Surface d'Onde
COMPASS	COMputing Platform for Adaptive opticS Systems
CDG	Centre De Gravité
CPU	Central Processing Unit
ELT	Extremely Large Telescope
ESO	European Southern Observatory
FEP	Fonction d'Étalement de Point
FPGA	Field Programmable Gate Array
FTM	Fonction de Transfert de Modulation
FTO	Fonction de Transfert Optique
FTP	Fonction de Transfert de Phase
GAMORA	Gpu Accelerated Module fOr psf Reconstruction Algorithms
GLAO	Ground Layer Adaptive Optics
GMT	Giant Magellan Telescope
GPU	Graphics Processing Unit
GPGPU	General Purpose Graphics Processing Unit
GROOT	Gpu-base Residual errOr cOvariance maTrix
KL	Karhunen-Loève
LGS	Laser Guide Star
LTAO	Laser Tomography Adaptive Optics
MAC	Multiply-ACumulate
MCAO	Multi Conjugate Adaptive Optics
MOAO	Multi Object Adaptive Optics
MVM	Matrix Vector Multiply
NCPA	Non Common Path Aberration

OA	Optique Adaptative
RAM	Random Access Memory
ROKET	erROr breaKdown Estimation Tool
RTC	Real Time Computer
SCAO	Single Conjugate Adaptive Optics
SM	Streaming Multiprocessor
TMT	Thirty Meter Telescope
UAL	Unité Arithmétique et Logique
VLT	Very Large Telescope

À Enzo et Noah

Introduction

Nous sommes début mars, le printemps approche et il est grand temps : le froid et l'humidité sont encore bien présents, et j'ai hâte qu'ils cèdent la place aux beaux jours. Cette période est également synonyme de début de rédaction du mémoire de thèse pour la plupart des doctorants, un exercice difficile à réaliser et surtout à commencer. En quête d'inspiration, je contemple le parc de l'Observatoire à Meudon à travers la fenêtre de mon bureau. Conditions météorologiques obligent, le radiateur situé sous la fenêtre tourne pour maintenir une température acceptable dans le bureau. En regardant à travers la fenêtre juste au-dessus du radiateur, le tronc de l'arbre au loin semble se mouvoir, comme pris d'une envie folle de se trémousser au rythme d'une musique imaginaire. Nous avons tous déjà observé ce phénomène, autour d'un feu ou sur la route par une chaleur d'été. Il est dû à une variation de l'indice de réfraction de l'air induit par la source de chaleur, causant ainsi des déviations des rayons lumineux et cet effet de "mirage".

En quoi cela nous intéresse-t-il, me direz-vous ? Et bien, ce phénomène est une illustration des effets de la turbulence atmosphérique sur les images obtenues lors d'observations au sol. En effet, l'atmosphère terrestre n'est pas un long fleuve tranquille. Le Soleil tient le rôle de mon radiateur à l'échelle de la Terre, causant des fluctuations de température au sein même de l'atmosphère. Ainsi, en plus de nous secouer plus ou moins violemment dans les avions, les turbulences atmosphériques impactent grandement la qualité des images obtenues à l'aide de grands télescopes. Cette qualité dépend évidemment de la résolution de l'instrument utilisé. Celle-ci peut s'exprimer comme la distance angulaire minimale entre deux objets à laquelle l'instrument est capable de les distinguer. Dans le cas des télescopes, la résolution dépend de la longueur d'onde d'observation et du diamètre du télescope. Ainsi, depuis son invention au XVII^{e} siècle, le diamètre des télescopes n'a cessé d'augmenter. Je citerai notamment le télescope d'Herschel, construit en 1789 avec un miroir primaire de 1,22 mètres de diamètre, qui pourrait presque paraître ridicule aujourd'hui à côté du Very Large Telescope (VLT) et de ses quatres télescopes de 8,2 mètres de diamètre. Et cette évolution continue toujours, avec la future catégorie des télescopes de diamètre extrême au-delà de 20 mètres et notamment l'ELT (Extremely Large Telescope) européen de 39 mètres, auquel je m'intéresserai tout particulièrement ici.

Seulement voilà, ces télescopes prometteurs de performances fabuleuses sont ramenés à l'état de télescope amateur par la turbulence atmosphérique. En effet, sans compensation, la résolution de ces télescopes est équivalente, dans le visible, à celle d'un télescope amateur de quelques dizaines de centimètres. Il y aurait de quoi être déçu du rapport qualité prix... Heureusement, des solutions existent. Une première est de faire fi de l'atmosphère terrestre en envoyant notre télescope dans l'espace. Je pense bien évidemment au télescope spatial Hubble, mais également au futur télescope spatial James Webb. Dans ce cas, plus de problème, la résolution de notre instrument est

bien directement donnée par son diamètre. Revers de la médaille, les contraintes liées aux projets spatiaux sont fortes : coût, logistique, maintenance... Une autre solution consiste alors à corriger les effets de la turbulence atmosphérique au sein même de l'instrument. Et c'est précisément le but de l'optique adaptative (OA).

Dès 1953, H. Babcock propose l'utilisation d'un élément optique actif associé à un système de mesure du front d'onde pour corriger les effets de l'atmosphère sur les observations au sol (Babcock, 1953). Il faudra cependant attendre les années 1970 et le Département de la Défense américain pour qu'un tel système voit le jour (Hardy et al., 1977). Les premières applications à l'astronomie arrivèrent à la fin des années 1980 avec en ligne de mire le développement du VLT pour lequel un tel système est essentiel (Beckers et al., 1985; Rousset et al., 1990; Roddier, 1999).

Les principaux constituants d'un système d'optique adaptative sont :

- un Analyseur de Surface d'Onde (ASO), permettant comme son nom l'indique de mesurer le front d'onde
- un Miroir Déformable (MD) visant à corriger le front d'onde
- un calculateur temps-réel afin de calculer la forme optimale à donner au miroir déformable

Ceux-ci forment ainsi le plus simple des systèmes d'optique adaptative, appelé SCAO pour Single Conjugate Adaptive Optics. S'il est très efficace dans l'axe de l'ASO, ce système n'offre qu'un faible champ de correction, de l'ordre de quelques secondes d'arc. Cette limitation implique alors une faible couverture du ciel, puisqu' il est assez rare de pouvoir trouver une étoile assez brillante pour l'ASO à proximité de l'objet que l'on souhaite étudier. Pour pallier ce problème, de multiples variantes de systèmes d'optique adaptative ont été imaginées, utilisant des étoiles laser artificielles ainsi que de multiples ASO et miroirs déformables. Malgré tout, les systèmes d'optique adaptative ne peuvent être parfaits, et une erreur résiduelle est toujours présente dans le front d'onde après correction.

La performance d'une OA est généralement caractérisée par sa Fonction d'Étalement de Point (FEP), en la comparant à la FEP théorique du télescope en l'absence de turbulences atmosphériques. Cette réponse impulsionnelle du système conditionne directement la résolution effectivement atteinte par l'instrument. Sa caractérisation est importante, notamment pour l'utilisation de techniques a posteriori, comme la déconvolution permettant un gain de contraste dans l'image finale (Mugnier et al., 2004). Si elle peut être mesurée directement sur le ciel en observant une étoile brillante, il est préférable de plutôt identifier et quantifier les sources d'erreurs résiduelles, à partir desquelles il est possible de reconstruire la FEP a posteriori (Véran et al., 1997; Exposito et al., 2014). Ainsi, il n'est pas nécessaire d'effectuer de longues poses ne servant qu'à mesurer la FEP sur une étoile de référence, et le temps d'observation est pleinement utilisé pour l'objet d'intérêt. L'estimation du budget d'erreur ainsi nécessaire à la reconstruction de la FEP se fait à partir des données de télémétrie de l'OA, et il se base généralement sur des hypothèses statistiques difficilement vérifiables. Le développement et l'utilisation d'outils de simulations numériques peuvent alors permettre d'étudier pleinement les différents contributeurs au budget d'erreur.

La simulation numérique tient une place importante dans le développement des instruments. En effet, en permettant une estimation rapide des performances que l'on peut attendre d'un système donné, elle permet par exemple d'étudier différents designs et de faire les compromis optimaux lors de la conception de l'instrument. En OA, il existe ainsi des modèles pseudo-analytiques des sources d'erreurs permettant d'estimer la FEP d'un instrument (Jolissaint et al., 2006). Si ces modèles sont rapides, leurs approches analytiques n'est rendu possible qu'au prix de certaines hypothèses simplificatrices. Pour une étude plus complète, les approches de type Monte Carlo permettent des simulations bout-en-bout des systèmes d'OA, incluant ainsi des effets de second ordre inaccessibles aux modèles pseudo-analytiques (Rigaut & Van Dam, 2013; Wang & Ellerbroek, 2011). La complexité numérique des modèles physiques inclus dans ce type de simulation est évidemment liée à la taille du système à simuler. Ainsi, le développement des instruments de premières lumière du futur ELT représente également un défi numérique important, aussi bien pour le dimensionnement de l'ordinateur temps-réel de l'OA que pour les moyens à mettre en œuvre pour simuler un système d'OA à cette échelle.

Cette thèse s'articule ainsi autour de deux thématiques : le développement d'outils de simulations numériques efficaces à l'échelle de l'ELT, et leur utilisation dans le cadre de l'étude du budget d'erreur des systèmes d'OA. Dans la première partie de cette thèse, je reviendrai sur les principes fondamentaux de la formation d'images au foyer d'un télescope, puis j'introduirai les principes fondateurs de l'optique adaptative ainsi que ses évolutions. Dans une seconde partie, je présenterai le calcul haute performance sur carte graphique et son intérêt dans la perspective de l'ELT. Je présenterai ensuite le développement de l'outil de simulation bout-en-bout pour système d'optique adaptative, COMPASS, qui s'appuie sur l'utilisation de cartes graphiques comme accélérateurs matériels. Dans la dernière partie de cette thèse, je décrirai l'outil d'estimation du budget d'erreur en optique adaptative que j'ai développé au sein de la plateforme COMPASS. Enfin, j'exposerai certains modèles analytiques de ce budget d'erreur, permettant ainsi d'obtenir une estimation rapide de la FEP et pouvant également être utiles à la reconstruction de FEP sur le ciel.

Première partie

Optique adaptative : principes et évolution

Chapitre 1

Observer avec un télescope

Ce chapitre vise à rappeler les principes de la formation d'images au foyer d'un télescope. Après un rappel du phénomène de diffraction, j'introduirai la turbulence atmosphérique, ses propriétés statistiques et son impact sur l'image obtenue avec un télescope.

Sommaire

1.1 Résolution limitée par la diffraction

1.1.1 La fonction d'étalement de point

En astronomie, les objets que l'on souhaite observer, tels que les étoiles et les galaxies, sont suffisamment éloignés de la Terre pour être considérés comme étant situés à l'infini. Ces objets sont des sources de lumière : celle-ci est émise comme une onde sphérique, puis se propage librement dans l'espace. Ainsi, lorsqu'elle atteint la Terre, cette même onde peut être considérée comme plane. Un télescope permet alors d'intercepter la lumière à l'aide d'un miroir primaire qui, d'après les lois de l'optique géométrique, focalise l'onde sur son plan focal.

Néanmoins, de part la nature ondulatoire de la lumière, l'image d'un objet ponctuel au foyer d'un télescope ne sera pas un point infiniment petit, mais plutôt une tache dont la forme et la taille seront définies par la pupille du système optique. Cette

tache est la réponse impulsionnelle du système, encore appelée Fonction d'Étalement de Point (FEP). Dès lors, l'image obtenue peut s'exprimer comme le produit de convolution entre l'objet et la réponse impulsionnelle du système (Goodman, 1995) :

$$I(\boldsymbol{\rho}) = O(\boldsymbol{\rho}) \circledast FEP(\boldsymbol{\rho}) \tag{1.1}$$

Il apparaît alors évident que la connaissance de la FEP est essentielle à la bonne exploitation des images obtenues.

En écrivant l'onde lumineuse sous la forme :

$$\Phi(\boldsymbol{r}) = A(\boldsymbol{r}) \exp(i\phi(\boldsymbol{r})) \tag{1.2}$$

avec $A(\boldsymbol{r})$ son amplitude et $\phi(\boldsymbol{r})$ sa phase, les lois de l'optique ondulatoire nous permettent cette fois d'écrire la FEP comme le module carré de la transformée de Fourier :

$$FEP(\boldsymbol{\rho}) = \|\mathcal{F}(P(\boldsymbol{r})\,\Phi(\boldsymbol{r}))\|^2 \tag{1.3}$$

où $\boldsymbol{\rho}$ est la variable du plan focal, $\boldsymbol{r}$ la variable du plan pupille ,$\mathcal{F}$ l'opérateur de transformation de Fourier et $P(\boldsymbol{r})$ désigne la fonction pupille du télescope. En considérant un télescope circulaire de diamètre D avec une obstruction centrale également circulaire et de diamètre c, cette fonction pupille peut alors s'écrire :

$$P(\boldsymbol{r}) = \begin{cases} 1 & \text{si } c/2 < \|\boldsymbol{r}\| < D/2 \\ 0 & \text{sinon} \end{cases} \tag{1.4}$$

Ainsi, dans un cas idéal sans perturbation où $\Phi(\boldsymbol{r}) = 1$ (onde plane), la FEP est alors une tache d'Airy, à savoir une fonction de Bessel du premier ordre constituée d'un cœur central circulaire entouré d'anneaux concentriques (Figure 1.1). La largeur à mi-hauteur du pic central, environ égal à $\frac{\lambda}{D}$ où λ est la longueur d'onde d'observation, donne la capacité de résolution du télescope. Cette propriété est l'une des origines de

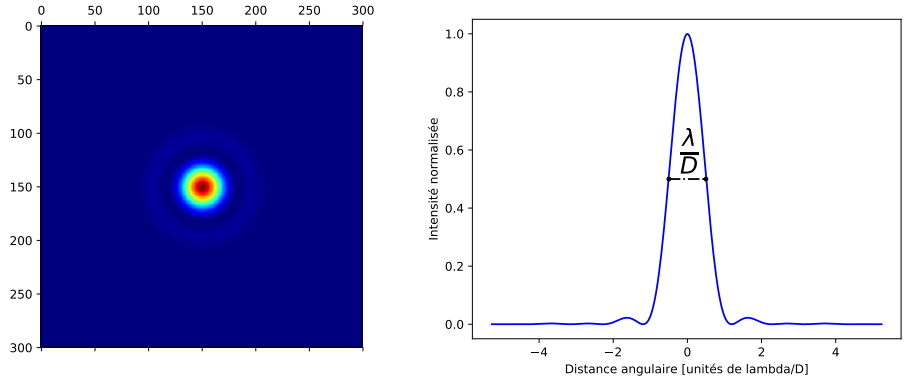

FIGURE 1.1 – Figure de diffraction d'une pupille circulaire, ou tache d'Airy. La largeur à mi-hauteur, visible sur la coupe, est environ égale à $\frac{\lambda}{D}$ et définit la résolution

l'envie des astronomes de disposer de télescopes avec le plus grand diamètre possible, afin d'accroître le pouvoir de résolution.

1.1.2 Fonction de transfert optique

La Fonction de Transfert Optique (FTO) est définie comme la transformée de Fourier de la FEP :

$$FTO(\boldsymbol{f}) = \mathcal{F}(FEP(\boldsymbol{r})) \tag{1.5}$$

où $\boldsymbol{f}$ désigne la fréquence spatiale. Cette fonction complexe peut être décomposée classiquement en un terme d'amplitude, appelé Fonction de Transfert de Modulation (FTM), et un terme de phase appelé Fonction de Transfert de Phase (FTP).

$$FTO(\boldsymbol{f}) = FTM(\boldsymbol{f})\ \exp(i\, FTP(\boldsymbol{f})) \tag{1.6}$$

La Figure 1.2 donne une représentation de la FTM, toujours dans le cas d'une pupille circulaire. Comme nous pouvons le constater, un télescope se comporte donc comme un filtre passe-bas avec une fréquence de coupure à $\frac{D}{\lambda}$.

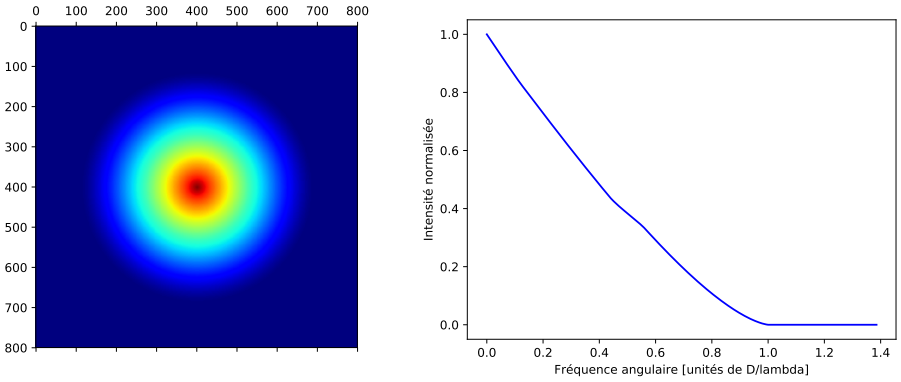

FIGURE 1.2 – Fonction de transfert de modulation dans le cas d'une pupille circulaire

En appliquant le théorème de Wiener-Khintchine à l'Eq. 1.5, la FTO peut également s'écrire comme l'auto-corrélation de la fonction pupille :

$$FTO(\frac{\boldsymbol{\rho}}{\boldsymbol{\lambda}}) = \int_{-\infty}^{\infty} P(\boldsymbol{r})P^*(\boldsymbol{r}-\boldsymbol{\rho})\, \boldsymbol{d^2 r} \tag{1.7}$$

Ainsi, il existe un lien mathématique entre le plan pupille du télescope, sa FEP et sa FTO. Ce lien est schématisé sur la Figure 1.3.

1.2 La turbulence atmosphérique

Après avoir rappelé brièvement les bases de la formation d'image au foyer d'un télescope dans le cas idéal, il est temps de considérer ce qu'il se passe dans la réalité du terrain. Comme expliqué en introduction, la FEP effectivement obtenue sur le ciel peut être éloignée de ce que l'on attend de la théorie de la diffraction, et ce à cause de la turbulence atmosphérique. Pour bien comprendre l'effet de ces dernières sur la formation d'images, je commencerai par décrire ses propriétés statistiques qui permettent ainsi de modéliser ses effets.

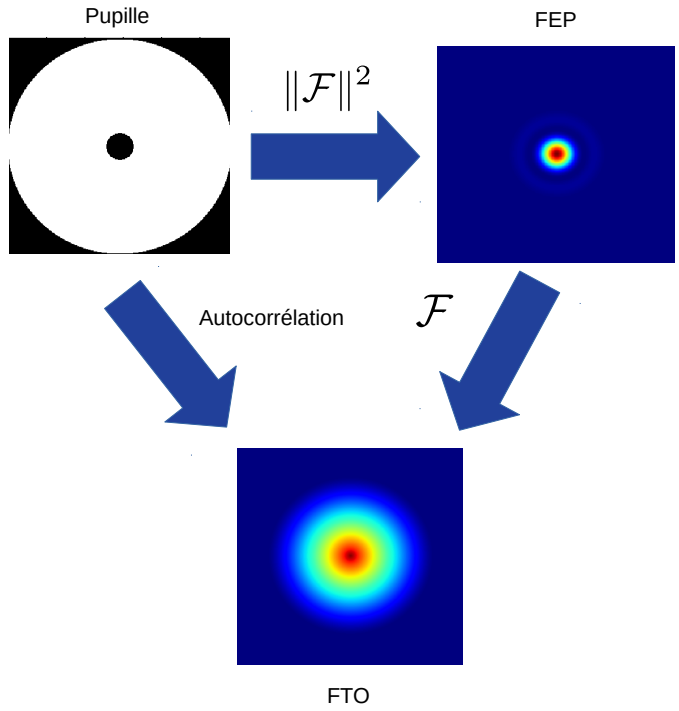

FIGURE 1.3 – Liens entre le plan pupille, la FEP et la FTO

1.2.1 Origine

Contrairement à ce que l'on pourrait penser de prime abord, l'atmosphère n'est pas un milieu homogène. Le réchauffement des masses d'air par le Soleil entraîne des différences de température provoquant des mouvements d'air à grande échelle. Le tout est un phénomène dynamique spatialement, les masses d'air se déplaçant selon un écoulement turbulent.

La loi de Gladstone lie l'indice de réfraction n d'un gaz à sa masse volumique ρ :

$$n - 1 = \kappa\rho \tag{1.8}$$

avec κ une constante. La masse volumique d'un gaz étant également dépendante de sa température, on comprend que les variations de température de l'atmosphère le long du chemin optique parcouru par un rayon lumineux entraîne des fluctuations de l'indice de réfraction, et donc une déformation du front d'onde.

1.2.2 Propagation d'une onde plane dans l'atmosphère

Lors de sa propagation dans l'atmosphère, la perturbation subie par le front d'onde peut être écrite comme la somme des perturbations engendrées par chaque couche de

l'atmosphère (Roddier, 1981) :

$$\phi_{turbu}(\boldsymbol{r},\theta) = \sum_i \phi_{couche_i}(\boldsymbol{r},\theta,h_i) \tag{1.9}$$

où $\boldsymbol{r}$ est le vecteur position dans la pupille, θ est la direction d'observation et h_i représente l'altitude de chacune des couches.

Le caractère turbulent de ce phénomène rend difficile sa caractérisation, si ce n'est par un aspect statistique. Kolmogorov émet l'hypothèse que l'énergie des tourbillons turbulents produits dans l'atmosphère est transmise jusqu'à une petite échelle l_0 où l'énergie est dissipée par frottements visqueux (Kolmogorov, 1941a,b). Dans le cadre de ce modèle, les travaux d'Obukhov (M. Obukhov, 1968) et de Tatarskii (Tatarskii, 1971) permettent d'aboutir à une expression de la densité spectrale de puissance des fluctuations de l'indice de réfraction induites par la turbulence atmosphérique. En reliant ces fluctuations à celles de la différence de marche engendrée sur le front d'onde, puis à la différence de phase, on aboutit à l'expression du spectre de Kolmogorov de la phase turbulente :

$$W_{kolmo}(f) = 0,023 r_0^{-5/3} f^{-11/3} \tag{1.10}$$

où r_0 est le paramètre de Fried, sur lequel je reviendrai dans la Section 1.2.3.

Il est important de noter que ce modèle diverge lorsque la fréquence spatiale tend vers 0. Pour considérer des échelles externes de taille finie L_0, le spectre de Von Karman permet d'écrire (Borgnino et al., 1992) :

$$W_{vk}(f) = 0,023 r_0^{-5/3} \left[f^2 + \frac{1}{L_0^2} \right]^{-11/6} \tag{1.11}$$

La Figure 1.4 représente ce spectre pour différentes valeurs de L_0.

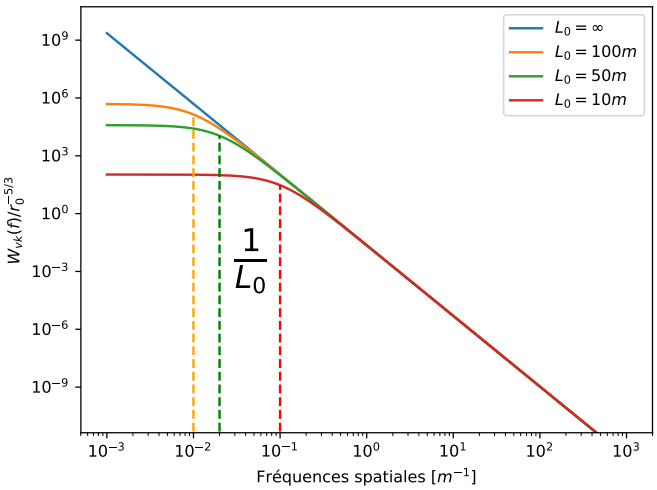

FIGURE 1.4 – Spectre de Von Karman pour différents L_0

Il est possible de définir une fonction de structure de phase caractérisant la variance de la différence de la phase turbulente entre deux point distants de $\boldsymbol{\rho}$ sur une couche d'épaisseur δh :

$$D_\phi(\boldsymbol{\rho}, h) = \langle(\phi(\boldsymbol{r}+\boldsymbol{\rho}) - \phi(\boldsymbol{r}))^2\rangle \quad (1.12)$$

$$= 2.91\left(\frac{2\pi}{\lambda}\right)^2 C_n^2(h)\delta h\|\boldsymbol{\rho}\|^{5/3} \quad (1.13)$$

où $C_n^2(h)$ est la constante de structure de l'indice n. Cette constante permet de caractériser la force de la turbulence à une altitude h donnée. En sommant cette expression sur toutes les couches, on aboutit à (Roddier, 1981) :

$$D_\phi(\boldsymbol{\rho}) = 6,88\left(\frac{\|\boldsymbol{\rho}\|}{r_0}\right)^{5/3} \quad (1.14)$$

Notons que cette expression considère que l'échelle externe est infiniment grande (spectre de Kolmogorov).

1.2.3 Le paramètre de Fried

Le paramètre de Fried r_0, introduit par Fried (1966) comme son nom l'indique, peut être défini comme le diamètre équivalent d'un télescope limité par la diffraction qui aurait la même résolution qu'un télescope de diamètre infini limité par la turbulence. En d'autres termes, le rapport $\frac{\lambda}{r_0}$ définit la résolution d'un télescope infiniment grand en présence de turbulence.

En un sens, r_0 permet donc de caractériser la force totale de la turbulence. Son expression fait d'ailleurs intervenir la constante de structure de l'indice n introduite plus tôt :

$$r_0 = \left[0,423\left(\frac{2\pi}{\lambda}\right)^2 \frac{1}{\cos\gamma}\int_0^\infty C_n^2(h)dh\right]^{-3/5} \quad (1.15)$$

avec γ l'angle zénithal du télescope lors de l'observation. Il est important de noter la dépendance du paramètre de Fried en $\lambda^{6/5}$. Ainsi, r_0 est d'autant plus grand que la longueur d'onde est grande : il est donc plus facile d'observer dans l'infrarouge avec une résolution proche de la diffraction.

Pour avoir un ordre de grandeur, le paramètre de Fried dans le visible à $0,5\mu m$ est typiquement compris entre 10 et 20 cm. En utilisant sa dépendance en $\lambda^{6/5}$, il est possible de calculer sa valeur en bande K :

$$r_0(\lambda_1 = 2,2\mu m) = r_0(\lambda_2 = 0,5\mu m)\left(\frac{\lambda_1}{\lambda_2}\right)^{6/5} \quad (1.16)$$

On obtient alors un r_0 entre 60 et 120 cm, soit un facteur 6 par rapport au visible.

1.2.4 Propriétés temporelles

En astronomie, une hypothèse répandue est de considérer que, du point de vue de l'observateur, le déplacement des couches d'atmosphère est prépondérant devant l'évolution propre de la turbulence au sein des couches (Gendron & Léna, 1996). Ainsi, la structure turbulente d'une couche est fixe et défile à une vitesse $\boldsymbol{v}$: on parle alors d'écoulement gelé [1]. Dans ces conditions, l'hypothèse de Taylor relie les propriétés spatiales et temporelles de l'atmosphère(Taylor, 1938) :

$$\frac{\partial \phi(\mathbf{x}, t)}{\partial t} = \mathbf{v} \frac{\partial \phi(\mathbf{x}, t)}{\partial \mathbf{x}} \tag{1.17}$$

En particulier, Greenwood a établi la fréquence à laquelle l'optique adaptative devrait fonctionner pour obtenir une correction optimale (Greenwood, 1977). De cette fréquence, il est possible de définir le temps de cohérence spatiale de la turbulence (Fried, 1990) :

$$\tau_0 \approx 0,314 \frac{r_0}{\bar{v}} \tag{1.18}$$

où $\bar{v}$ est la moyenne pondérée des vitesses de vent de chaque couche turbulente :

$$\bar{v} = \left[\frac{\int C_n^2(h) v(h)^{5/3} dh}{\int C_n^2(h) dh} \right]^{3/5} \tag{1.19}$$

τ_0 définit ainsi le temps durant lequel les perturbations dues à l'atmosphère peuvent être considérées comme ayant faiblement évoluées. Le système d'OA doit alors être capable de suivre la dynamique de ces distorsions.

1.3 Impact de la turbulence sur la FEP

En présence d'atmosphère, le front d'onde incident n'est plus plat ce qui affecte directement la FEP, comme indiqué par l'Équation 1.3. La Figure 1.5 montre les effets de la turbulence sur la FEP pour un rapport $\frac{D}{r_0}$ de 12.

Sur une image courte pose, c'est à dire avec un temps de pose inférieur au temps de cohérence défini plus haut, la FEP se compose de tavelures [2] dont la taille caractéristique est $\frac{\lambda}{D}$. Ces tavelures s'étalent sur une région de taille $\frac{\lambda}{r_0}$. Elles sont dues à la superposition aléatoire de réseaux de franges du fait de la phase turbulente aléatoire dans la pupille.

En les moyennant sur une longue pose, on retrouve une FEP avec un pic central, mais celui-ci est beaucoup plus large que dans le cas idéal (et si on considère D > r_0), puisque sa largeur à mi-hauteur est égale à $\frac{\lambda}{r_0}$. La capacité de résolution du télescope est alors limitée par le paramètre de Fried r_0.

1. *frozen flow* en anglais
2. *speckles* en anglais

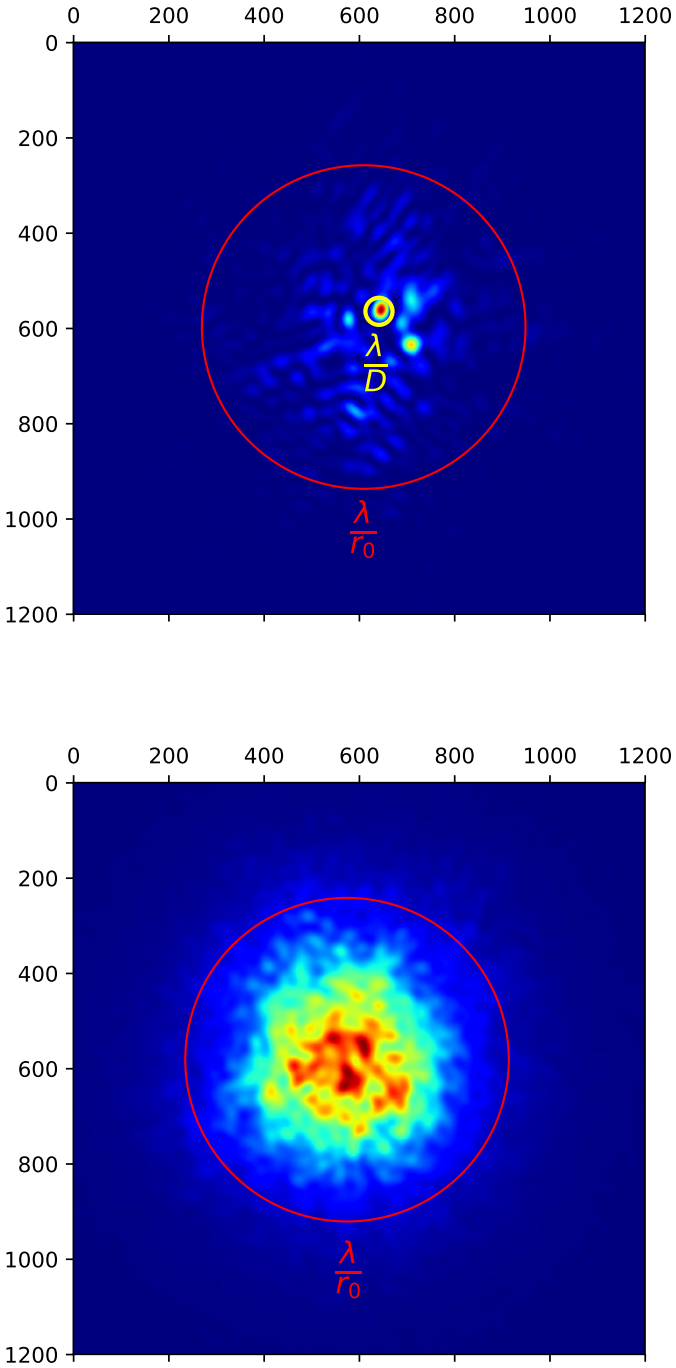

FIGURE 1.5 – FEP simulée impactée par la turbulence atmosphérique. En haut : FEP courte pose (1 trame, équivalent à 2 ms de pose). En bas : FEP "longue" pose (10 000 trames, équivalent à 20 s de pose)

1.4 Représentation modale de la phase

Afin de traiter analytiquement (et numériquement) les déformations subies par le front d'onde, il est en général pratique de l'exprimer sur une base $\mathcal{B}$ de modes orthogonaux telle que :

$$\phi(\boldsymbol{r}) = \sum_{i=1}^{\infty} a_i \mathcal{B}_i(\boldsymbol{r}) \tag{1.20}$$

où $\mathcal{B}_i(\boldsymbol{r})$ est le $i^{ème}$ mode de la base $\mathcal{B}$ et a_i le coefficient de décomposition de $\phi(\boldsymbol{r})$ sur ce mode. Ainsi, cette écriture permet de se ramener à un traitement unidimensionnel sur le vecteur $\boldsymbol{a}$ des coefficients de décomposition sur la base modale $\mathcal{B}$. Autrement dit, la base $\mathcal{B}$ étant donnée, le vecteur $\boldsymbol{a}$ permet de caractériser entièrement $\phi(\boldsymbol{r})$. Reste à choisir ou à construire correctement la base modale à utiliser. Bien qu'il en existe une infinité, deux bases particulières sont très utilisées en optique adaptative : la base des Zernike et celle des Karhunen-Loève.

1.4.1 Les polynômes de Zernike

Les polynômes de Zernike se prêtent bien à la description de l'effet de la turbulence atmosphérique sur le front d'onde (Noll, 1976). Leur propriété d'orthogonalité sur le disque unité permet d'exprimer la phase du front d'onde sur une pupille non obstruée comme la somme de déformations simples et orthogonales entre elles, comme exprimé dans l'équation précédente.

De plus, ces polynômes présentent l'avantage d'être décrits par une expression analytique :

$$Z_j(r,\theta) = \begin{cases} \sqrt{2(n+1)} R_n^m(r)\cos(m\theta) & \text{si j pair et m non nul} \\ \sqrt{2(n+1)} R_n^m(r)\sin(m\theta) & \text{si j impair et m non nul} \\ \sqrt{n+1} R_n^0(r) & \text{si m nul} \end{cases} \tag{1.21}$$

avec :

$$R_n^m(r) = \sum_{k=0}^{(n-m)/2} \frac{(-1)^k (n-k)!}{k!((n+m)/2-k)!((n-m)/2-k)!}\, r^{n-2k} \tag{1.22}$$

où n désigne le degré radial, m le degré azimuthal avec $0 \leq m \leq n$ et $n - |m|$ pair.

L'orthonormalité de la base des modes de Zernike permet d'exprimer la variance spatiale de la phase en moyenne statistique sur la turbulence comme la variance de la somme quadratique des coefficients de Zernike, i.e. les coefficients de la décomposition de la phase sur la base des modes de Zernike. Cette variance permet de s'affranchir de la divergence du mode piston :

$$\sigma_\phi^2 = \sum_{i=2}^{\infty} \langle a_i^2 \rangle \tag{1.23}$$

Noll (1976) montre alors que la variance de la phase sur la pupille peut être évaluée à :

$$\sigma_\phi^2 = 1,03 \left(\frac{D}{r_0}\right)^{5/3} \tag{1.24}$$

Plus généralement, il donne une expression de la variance de la phase résiduelle après correction par l'optique adaptative sur N modes :

$$\sigma_\varepsilon^2 = \sum_{i=N+2}^{\infty} c_{ii} \left(\frac{D}{r_0}\right)^{5/3} \tag{1.25}$$

où la somme des coefficients c_{ii} est donnée par (Noll, 1976) et décroît en $N^{-\sqrt{3}/2}$. Une décroissance plus rapide peut être obtenue si les coefficients de décomposition sur la base modale sont statistiquement indépendants, ce qui n'est pas le cas pour la base des Zernike. Il est en effet possible de montrer que les modes de Zernike possédant le même degré azimuthal m sont corrélés.

1.4.2 Les modes de Karhunen-Loève

La base formée par les modes dits de Karhunen-Loève vérifie justement la propriété d'indépendance statistique de ces modes. La décomposition de la phase sur cette base de fonctions orthogonales a notamment la propriété de minimiser, au sens des moindres carré, l'erreur résiduelle après correction par N modes.

Malheureusement, et contrairement aux polynômes de Zernike, il n'existe pas d'expression analytique permettant de calculer un mode de Karhunen-Loève (KL). Une manière de les calculer est alors de diagonaliser la matrice de covariance statistique des coefficients de Zernike (Roddier, 1990). Les Figures 1.6 et 1.7 donnent une représentation des premiers modes de Zernike et KL. J'ai arbitrairement organisé les modes KL de la même façon que les modes de Zernike afin de les faire correspondre. Je précise ainsi que l'ordre radial et le degré azimuthal qui apparaissent sur la Figure 1.7 sont donnés simplement à titre de comparaison.

1.5 Une solution : l'optique adaptative

Nous avons vu dans ce chapitre que la turbulence atmosphérique avait un impact important sur la résolution des images que l'on obtient par des observations au sol avec un télescope. En effet, cette dernière est du même ordre de grandeur que celle que l'on obtiendrait avec un simple télescope amateur. Dans ce cas, quiconque pourrait légitimement douter de l'utilité des grands télescopes existants, et encore plus des futurs télescopes géants en projet.

Si le lecteur a bien lu l'introduction de cette thèse (ou simplement l'intitulé de cette section), il sait déjà qu'une solution technique existe afin de pallier aux effets de la turbulence atmosphérique sur la qualité des images. Je consacrerai le prochain chapitre à cette dernière, à savoir l'optique adaptative.

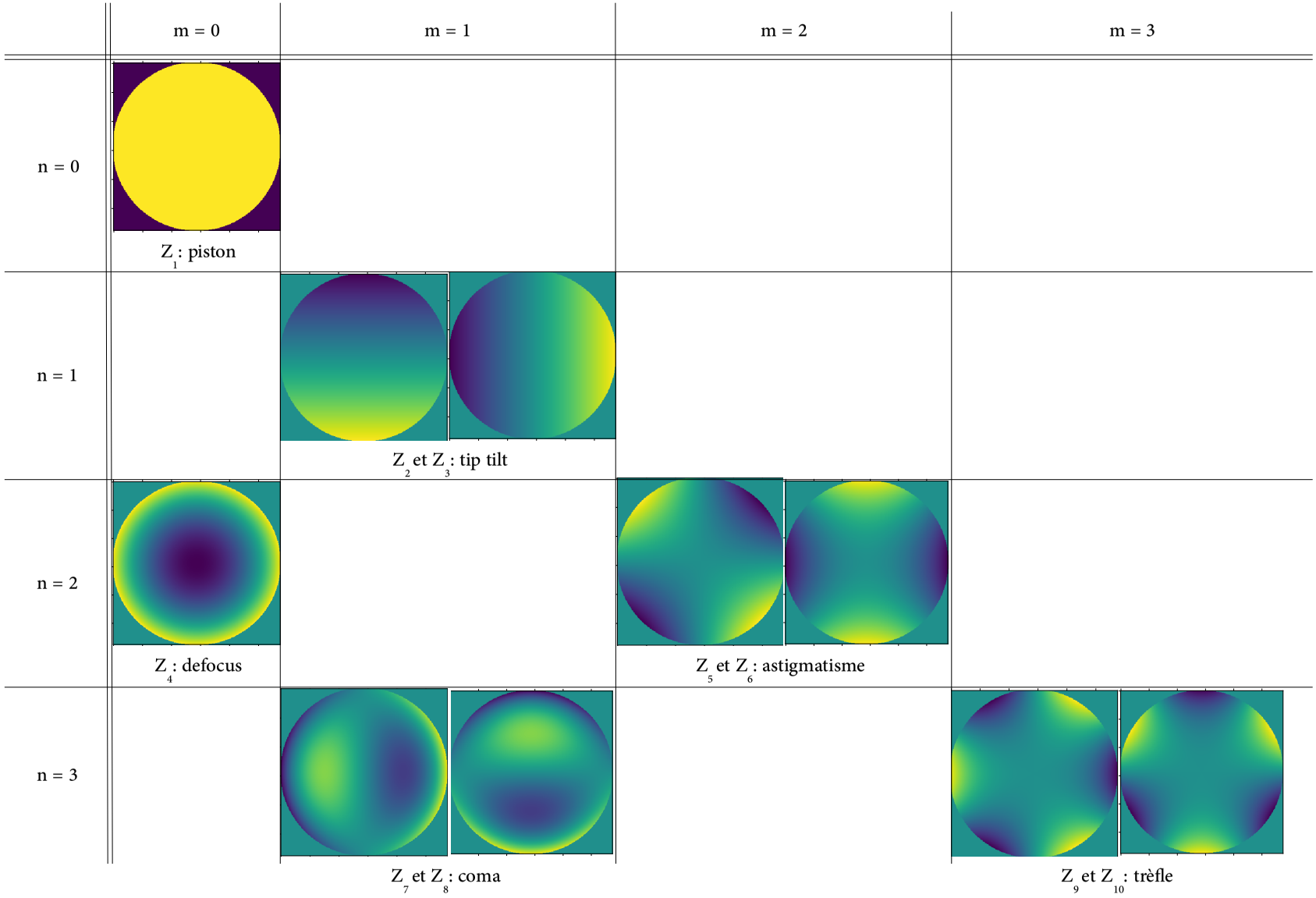

FIGURE 1.6 – Représentation 2D des premiers Zernikes

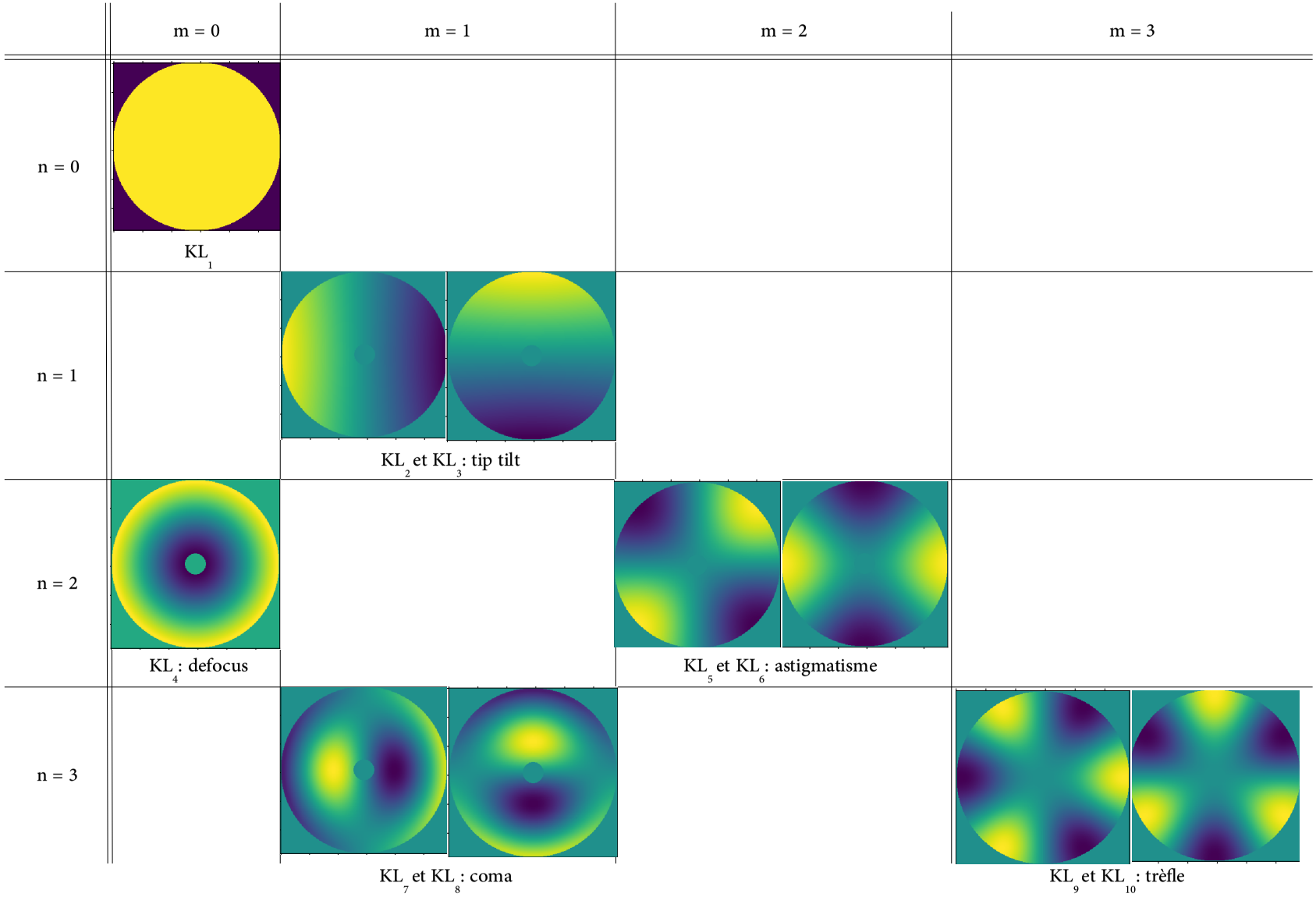

FIGURE 1.7 – Représentation 2D des premiers KL à la manière des modes de Zernike, avec une obstruction centrale

Chapitre 2

L'optique adaptative

Ce chapitre a pour but de présenter les principes fondamentaux de l'optique adaptative. J'y présente son principe de fonctionnement, les différents types d'OA existants, ainsi que ses limitations. Je m'intéresserai ensuite au défi que représente les systèmes d'OA appliqués à l'ELT.

Sommaire

2.1 Principe de fonctionnement

Après la lecture du chapitre précédent, le lecteur aura compris le problème que représente l'atmosphère pour les observations faites au sol. Les grands télescopes existants et futurs ont donc besoin d'un système permettant de corriger les effets de la turbulence pour pouvoir retrouver leur résolution théorique, fixée par leur diamètre et la longueur d'onde d'observation.

L'optique adaptative vise justement à mesurer, puis corriger les déformations subies par le front d'onde le long de son trajet à travers l'atmosphère (Roddier, 1999). La Figure 2.1 schématise le fonctionnement d'un système d'optique adaptative en boucle fermée.

Dans ce cas de figure, le front d'onde déformé intercepté par le miroir primaire du télescope est dirigé par une optique de relai vers un miroir déformable qui se trouve, dans la plupart des cas, conjugué à la pupille, puis vers la caméra scientifique. Une lame séparatrice est placée entre ces deux éléments afin de permettre à un analyseur de surface d'onde de mesurer la déformation du front d'onde. À partir de ces mesures, les commandes à appliquer au miroir déformable sont calculées en temps-réel afin de corriger le front d'onde.

Ici, le miroir déformable est donc le premier élément du système d'OA rencontré par la lumière. Par conséquent, l'analyseur de surface d'onde ne mesure plus que l'erreur résiduelle laissée sur le front d'onde après correction par le miroir déformable. On parle alors de système en *boucle fermée* pour qualifier la boucle de rétroaction ainsi formée. Dans cette configuration, les éventuelles erreurs commises par le miroir déformable sont vues par l'analyseur de surface d'onde, et sont donc corrigeables.

La Figure 2.2 donne deux images de la même étoile double observée avec ou sans optique adaptative. Il apparaît clairement que l'étoile double ne peut être distinguée sans le système d'OA.

Il existe des systèmes fonctionnant en boucle ouverte où l'analyseur mesure la déformation du front d'onde avant correction par le miroir déformable. Je citerai notamment la MOAO (Multi Object Adaptive Optics) dont le fonctionnement nécessite d'être en boucle ouverte (Vidal et al., 2010; Gendron et al., 2011).

Afin de fournir une présentation détaillée de chacun des composants d'un système d'OA, je leur consacre les sections qui suivent.

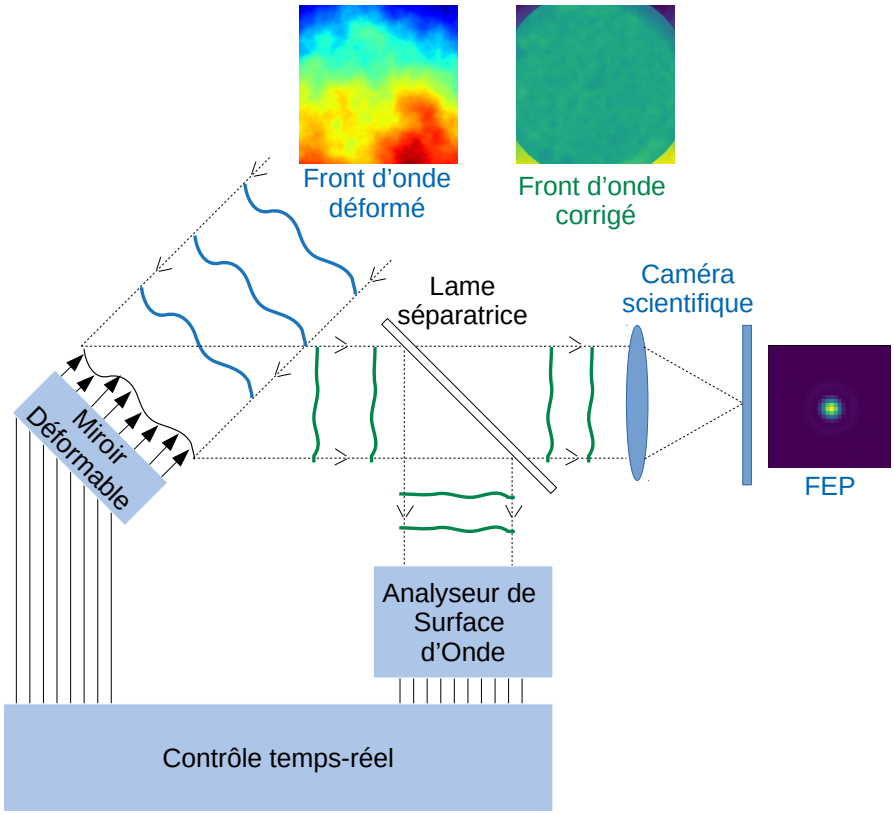

FIGURE 2.1 – Schéma de principe d'un système d'OA

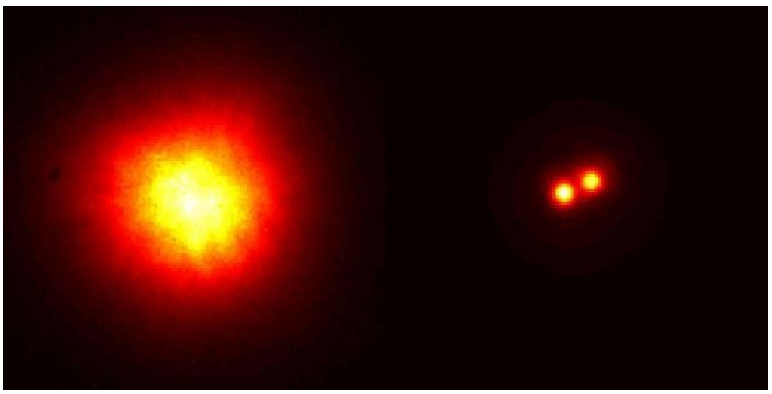

FIGURE 2.2 – Images d'une étoile double sans optique adaptative (à gauche) et avec optique adaptative (à droite). *Crédit : ESO*

2.2 Le miroir déformable

En suivant le trajet parcouru par la lumière dans le schéma présenté en Figure 2.1, le miroir déformable est le premier élément rencontré. Cet élément permet de modifier la forme du front d'onde incident en introduisant des différences de marche sur toute la surface d'onde. Le but de la boucle d'optique adaptative est alors de commander le miroir déformable afin d'aplanir le front d'onde. La correction du front d'onde par un système d'OA peut donc s'écrire :

$$\phi_\varepsilon(\boldsymbol{r},\theta) = \phi(\boldsymbol{r},\theta) + \phi_{DM}(\boldsymbol{r}) \tag{2.1}$$

avec $\phi_\varepsilon(\boldsymbol{r},\theta)$ la phase résiduelle obtenue après correction par le miroir déformable, $\phi(\boldsymbol{r},\theta)$ la phase incidente et $\phi_{DM}(\boldsymbol{r})$ la phase introduite par le miroir déformable. Notons que si le miroir déformable est conjugué à la pupille, $\phi_{DM}(\boldsymbol{r})$ ne dépend pas de l'angle d'observation θ.

Il existe différentes technologies de miroir déformable. Si leur point commun est bien évidemment de déformer une surface réfléchissante, les moyens utilisés pour effectuer cette déformation varient selon la technologie utilisée (Séchaud, 1999; Madec, 2012).

Ainsi, parmi les miroirs déformables les plus utilisés en optique adaptative, on compte les miroirs dits " à empilement de piézoélectriques" [1]. Le schéma de principe de ce type de miroir est présenté sur la Figure 2.3. Des actionneurs piézoélectriques

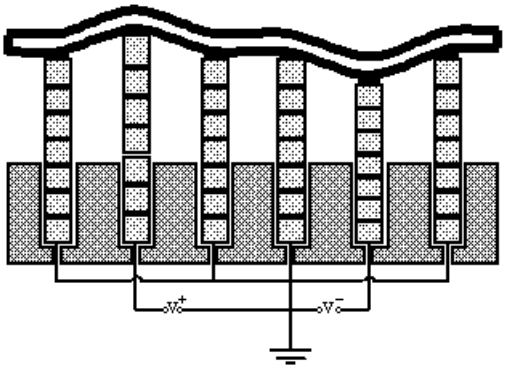

FIGURE 2.3 – Schéma de principe d'un miroir déformable piézoélectrique. *(Crédit : ONERA)*

sont collés sous la surface réfléchissante du miroir, et l'application d'une tension électrique permet alors de rétracter ou d'allonger l'actionneur, et donc de tirer ou de pousser sur la surface réfléchissante, aboutissant ainsi à une déformation locale de la surface du miroir appelée fonction d'influence. Ces actionneurs pavent ainsi toute la surface du miroir déformable, généralement selon un maillage carré ou hexagonal, engendrant un certain échantillonnage spatial de la surface de ce dernier. Les déformations engendrées sont petites, si bien que l'on reste dans le domaine d'élasticité des

1. *piezo-stack* en anglais

matériaux utilisés, et donc dans un domaine de linéarité des équations mécaniques associées. Dans ce contexte, pour un vecteur de tension $\mathbf{v}$ appliqué aux N actionneurs du miroir déformable, la différence de marche introduite par ce dernier peut s'écrire comme la somme pondérée des fonctions d'influence $\mathcal{IF}_i$:

$$\phi_{DM} = \sum_{i=1}^{N} v_i \mathcal{IF}_i \tag{2.2}$$

La Figure 2.4 illustre une fonction d'influence simulée et la surface d'un miroir déformable obtenue à partir de la somme de ces fonctions d'influence.

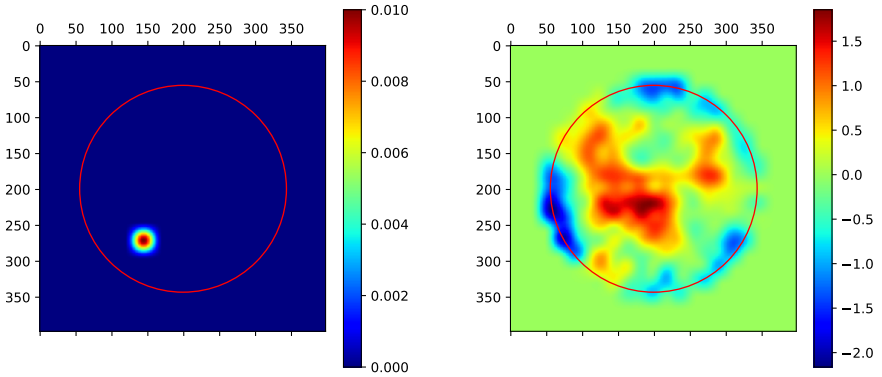

FIGURE 2.4 – Fonction d'influence simulée d'un miroir déformable piézo-stack (gauche) et la surface du miroir obtenue à partir de la somme de ses fonctions d'influence (droite)

D'autres technologies reposent sur des actionneurs discrets sans utiliser l'effet piézoélectrique. Je citerai notamment les miroirs déformables magnétiques ou électrostatiques : si le principe physique permettant de tirer ou pousser sur la surface réfléchissante diffère, le fonctionnement reste le même que celui d'un miroir piézoélectrique.

Les miroirs dits "bimorphes" sont quelque peu différent. Ici, les actionneurs discrets sont remplacés par des électrodes placées entre deux feuilles piézoélectriques, elles-mêmes placées sous la surface réfléchissante du miroir. L'application d'une tension aux bornes des électrodes aura pour effet de contracter une des deux feuilles, et de dilater l'autre. La géométrie des actionneurs de ces miroirs déformables peut être adaptée au besoin par le dépôt des électrodes selon le maillage souhaité.

2.3 L'analyseur de surface d'onde

Continuons de suivre le trajet de la lumière sur la Figure 2.1. Après avoir rencontré le miroir déformable, la lumière est scindée en deux par une lame semi-réfléchissante, généralement dichroïque : une partie part vers la caméra scientifique, tandis que l'autre

est dirigée vers un analyseur de surface d'onde. L'objectif ici est d'être capable de mesurer les déformations résiduelles restantes sur le front d'onde. Comme pour le miroir déformable, il existe différents types d'analyseur de surface d'onde. Mon but n'étant pas d'être exhaustif, je me contenterai donc de présenter le fonctionnement de deux ASO [2] parmi les plus utilisés aujourd'hui en optique adaptative et qui sont ceux que j'ai été amené à modéliser numériquement (cf. Chapitre 4) : le Shack-Hartmann (Shack & Platt, 1971) et la pyramide (Ragazzoni, 1996).

2.3.1 L'analyseur Shack-Hartmann

Le principe de l'analyseur Shack-Hartmann, présenté sur la Figure 2.5, est d'échantillonner spatialement le front d'onde à l'aide d'une matrice de micro-lentilles (Rousset, 1999).

Chaque micro-lentille, également appelée sous-pupille, focalise une partie du front d'onde sur une matrice CCD. En présence d'un front d'onde plan, les spots lumineux formés par chacune des sous-pupilles sont situés non loin du centre des sous-pupilles : ils sont sur leurs positions de référence.

Lorsqu'une déformation impacte le front d'onde, la position des spots lumineux varie par rapport à la position de référence, et ce proportionnellement à la moyenne du gradient local du front d'onde :

$$\begin{aligned} \alpha_x &= \frac{c_x}{f_{ssp}} = \frac{\lambda}{2\pi S_{ssp}} \int_{ssp} \frac{d\phi}{dx} dx\, dy \\ \alpha_y &= \frac{c_y}{f_{ssp}} = \frac{\lambda}{2\pi S_{ssp}} \int_{ssp} \frac{d\phi}{dy} dx\, dy \end{aligned} \tag{2.3}$$

où α_x et α_y désignent la pente locale moyenne du front d'onde en x et y respectivement, c_x et c_y la position en x et en y des spots lumineux, f_{ssp} la distance focale des micro-lentilles et S_{ssp} leur surface.

La position des spots lumineux (c_x, c_y) permet alors de remonter à la pente moyenne du front d'onde. Cette position peut être déterminée à l'aide de divers algorithmes de calcul du centre de gravité appliqués aux spots lumineux. Ces algorithmes vont du simple centre de gravité au centre de gravité appliqué aux N pixels les plus brillants (Basden et al., 2012), en passant par le centre de gravité seuillé, ou encore pondéré (Nicolle et al., 2004). Ces différentes manières de procéder visent à limiter l'impact du bruit sur les mesures, les sources de bruit étant le bruit de photons et le bruit de lecture du CCD. L'ensemble des positions (c_x, c_y) ainsi calculées pour tous les spots lumineux forment le vecteur de mesures (ou de pentes) $\boldsymbol{m}$ de l'ASO, et les positions de référence des spots lumineux forment les pentes de références. La Figure 2.6 représente en deux dimensions les mesures faites par un Shack-Hartmann d'un écran de phase turbulent. Les flèches indiquent le déplacement de chaque spot lumineux par rapport à sa position de référence.

2. Analyseur de Surface d'Onde

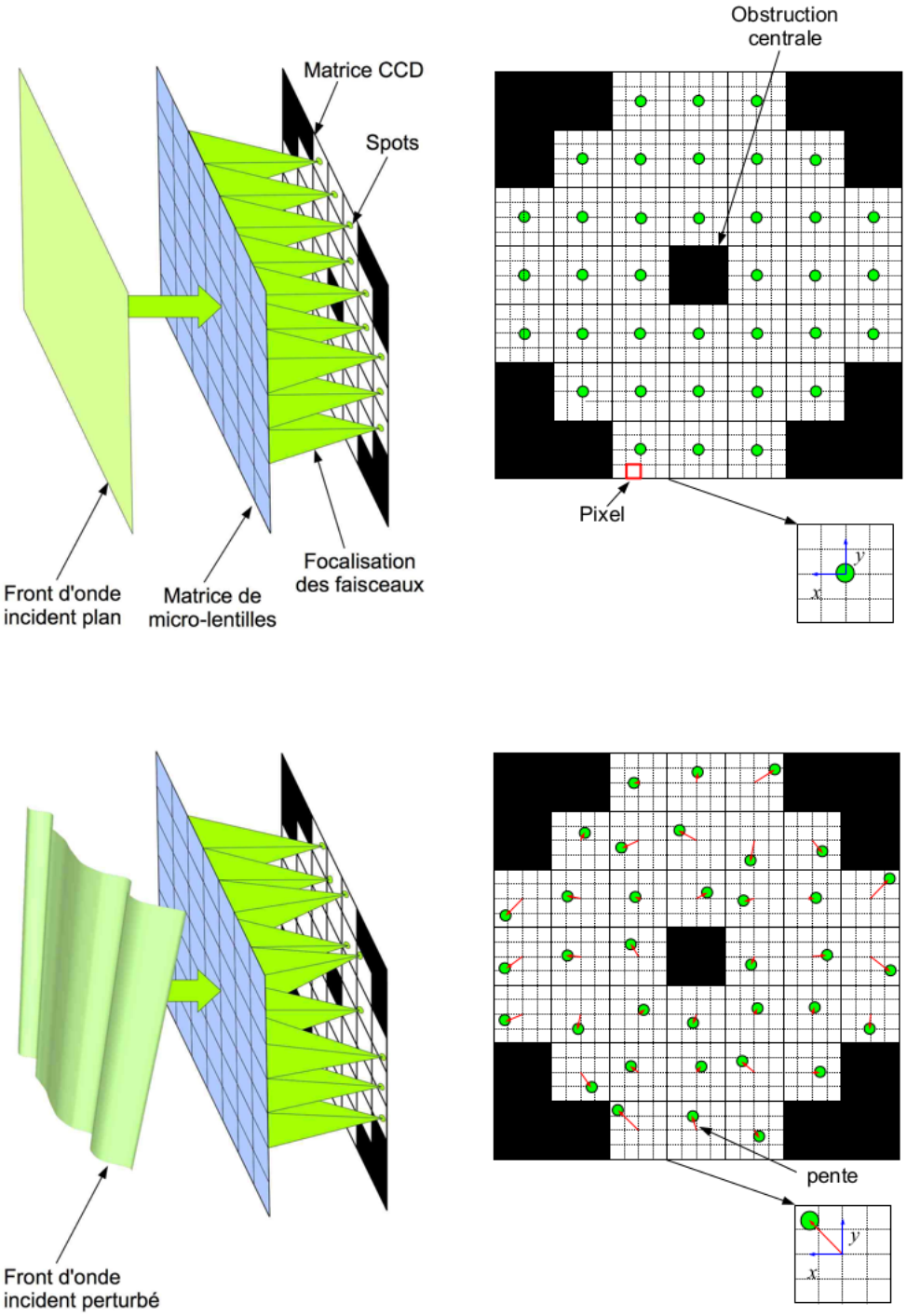

FIGURE 2.5 – Schéma de principe d'un analyseur de surface d'onde Shack-Hartmann. *(Crédit : O. Martin)*

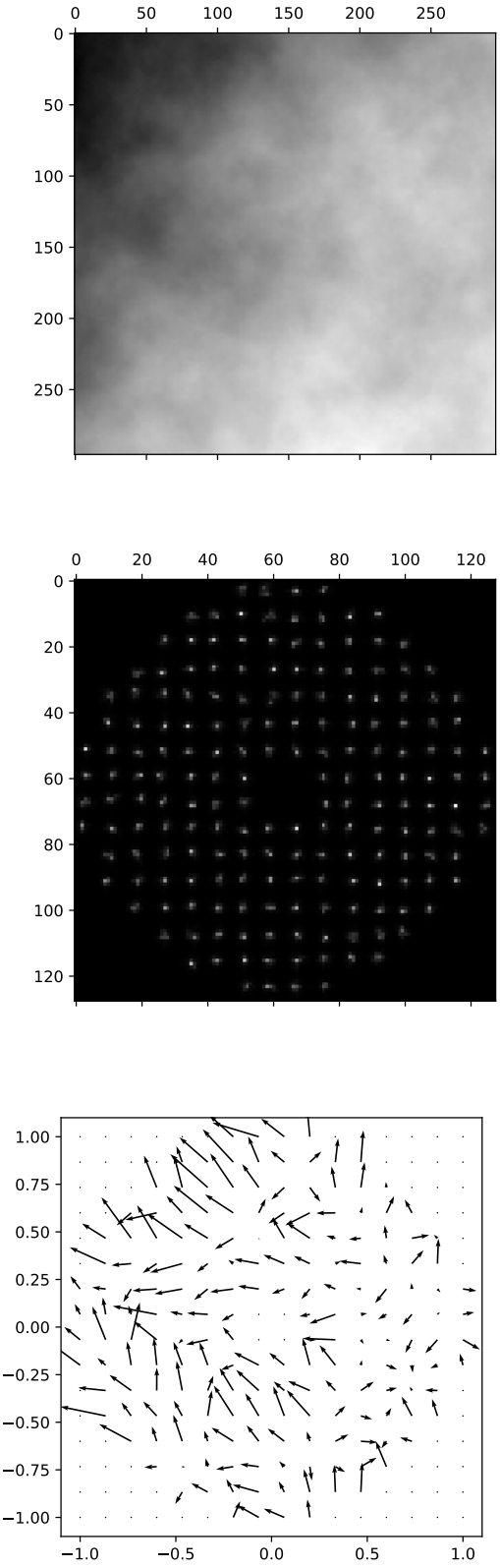

FIGURE 2.6 – Représentation en deux dimensions des déplacements mesurés des spots lumineux d'un Shack-Hartmann par rapport à leur position de référence en présence d'une phase turbulente. En haut : la phase turbulente ; au milieu : les spots ; en bas : les déplacement mesurés

2.3.2 L'analyseur pyramide

Ragazzoni (1996) propose un autre type d'analyseur de surface d'onde constitué d'un prisme pyramidal et d'une optique de relai. Le schéma de principe de cet analyseur est présenté sur la Figure 2.7. L'idée est de focaliser l'onde incidente sur le sommet

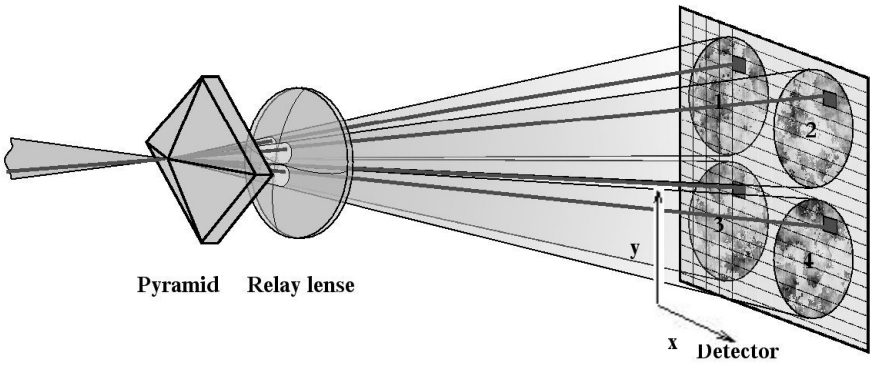

FIGURE 2.7 – Schéma de principe de l'analyseur pyramide *(Crédit : S. Egner)*

du prisme pyramidal, donc placé dans un plan focal, qui, à l'aide du relai optique, forme alors 4 images de la pupille sur le détecteur placé dans un plan pupille. Chaque image est donc formée par les rayons lumineux passant par la face correspondante de la pyramide. En régime linéaire, l'intensité lumineuse relative entre chaque pixels de chacune des 4 pupilles images est alors liée au gradient du front d'onde (Riccardi et al., 1998; Vérinaud, 2004) :

$$\begin{aligned} \frac{(I_1(x,y)+I_3(x,y))-(I_2(x,y)+I_4(x,y))}{I_1(x,y)+I_2(x,y)+I_3(x,y)+I_4(x,y)} &\approx \frac{\lambda}{\pi^2\theta_m}\frac{\partial\phi}{\partial x} \approx m_x \\ \frac{(I_1(x,y)+I_2(x,y))-(I_3(x,y)+I_4(x,y))}{I_1(x,y)+I_2(x,y)+I_3(x,y)+I_4(x,y)} &\approx \frac{\lambda}{\pi^2\theta_m}\frac{\partial\phi}{\partial y} \approx m_y \end{aligned} \tag{2.4}$$

où $I_n(x,y)$ désigne l'intensité du pixel (x,y) de l'image n de la pupille numérotée comme sur la Figure 2.7, θ_m est l'angle de modulation de la pyramide et (m_x, m_y) sont les composantes en X et en Y du vecteur de mesures $\mathbf{m}$ de l'ASO. Un exemple d'image simulée d'analyseur pyramide est visible sur la Figure 2.8.

La modulation consiste à déplacer le point de focalisation autour du sommet de la pyramide, et ce afin d'ajuster sa sensibilité et sa linéarité. En effet, l'Eq. 2.4 n'est valable que dans le domaine de linéarité de la pyramide (Ragazzoni & Farinato, 1999),

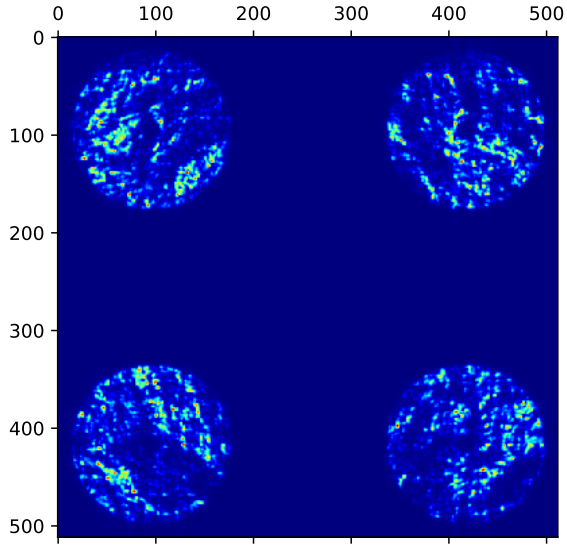

FIGURE 2.8 – Image haute-définition simulée d'un analyseur pyramide

c'est à dire si :

$$\theta_m > \frac{\lambda}{2\pi} max \left(|\frac{\partial \phi}{\partial x}| \right)_{pup} \tag{2.5}$$

La Figure 2.9 illustre la réponse de l'analyseur pyramide à un mode de basculement pour une modulation circulaire de rayon $\theta_m = 2\frac{\lambda}{D}$. La zone de linéarité est comprise entre $\pm\theta_m$, au-delà un régime de saturation est observable. Dès lors, la pyramide ne permet pas de mesurer correctement des amplitudes supérieures à $\pm\theta_m$. Augmenter l'angle de modulation permet alors d'accroître la zone de linéarité, au détriment de la sensibilité.

2.4 Le contrôle temps-réel

Le calculateur temps-réel[3] d'un système d'optique adaptative est en charge de calculer le vecteur de mesures **m** de l'ASO tel que décrit dans la section précédente. Ensuite, et c'est ce sur quoi je vais me concentrer dans cette section, vient une opération de reconstruction du front d'onde à partir de ce vecteur de mesures, et ce afin de commander le miroir déformable et corriger le front d'onde. Pour la suite, je suppose toujours que le système est linéaire.

3. *Real Time Computer (RTC)* en anglais

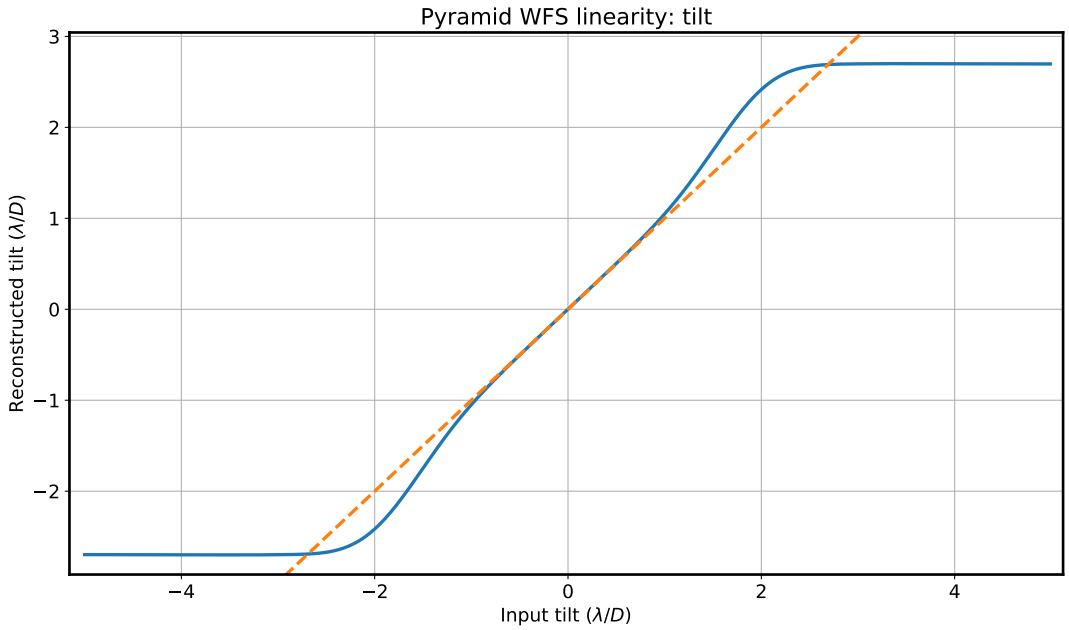

FIGURE 2.9 – Réponse de l'analyseur pyramide à un basculement pour un angle de modulation de $2\frac{\lambda}{D}$. *Crédit : V. Deo*

2.4.1 Un problème inverse

Comme expliqué dans la section précédente, l'analyseur de surface d'onde produit des images à partir desquelles il est possible d'obtenir un vecteur de mesure $\mathbf{m}$ des déformations du front d'onde. En décomposant la phase ϕ sur la base $\mathcal{IF}$ des fonctions d'influences du miroir (cf. Éq. 2.2), on obtient alors une relation entre les mesures faites par l'analyseur et les commandes $\boldsymbol{c}$ à appliquer au miroir :

$$\boldsymbol{m} = D\,\boldsymbol{c} + \boldsymbol{n} \tag{2.6}$$

où D désigne la matrice d'interaction du système d'OA que je décrirai dans la section suivante.

Le calcul des commandes est donc un problème inverse où l'on cherche une matrice de commande R telle que :

$$\boldsymbol{c} = R\,\boldsymbol{m} \tag{2.7}$$

Toute la problématique réside donc dans la détermination de la matrice de commande R.

2.4.2 La matrice d'interaction

La matrice d'interaction d'un système d'OA définit la relation linéaire qui relie les mesures de l'analyseur aux commandes du miroir déformable. Son étalonnage est donc primordial afin de prendre en compte les spécificités du miroir déformable ainsi que du montage optique du système d'OA (Boyer et al., 1990).

L'étalonnage de la matrice d'interaction consiste à envoyer une série de commande sur les actionneurs du miroir et d'enregistrer les vecteurs de mesures de l'ASO correspondants. Ainsi, une des possibilités consiste à appliquer un vecteur de commande unitaire sur le miroir déformable : l'ASO mesure alors la déformation produite par l'actionneur (i.e. la fonction d'influence) et le vecteur de mesures obtenu est stocké dans une colonne de la matrice d'interaction. Cette opération est répétée pour tous les actionneurs du miroir déformable. Tout ce processus est schématisé dans la Figure 2.10.

La matrice d'interaction est donc une matrice de taille $N_{mes} \times N_{actus}$ avec N_{mes} est le nombre de mesures de l'analyseur, lui-même égal au double du nombre de sous-pupille (une mesure en x et en y pour chacune d'entre elles), et N_{actus} est le nombre total d'actionneurs du système.

Cette procédure n'est pas la seule permettant d'étalonner la matrice d'interaction : je citerai notamment l'utilisation de matrices de Hadamard (Kasper et al., 2004; Meimon et al., 2015), de bases modales ou encore de sinusoïdes (Kellerer et al., 2012).

2.4.3 La matrice de commande

Maintenant que la matrice d'interaction D du système est connue, il est possible de calculer la matrice de commande R de l'Eq. 2.7.

Au sens des moindres carrés, la commande optimale $\boldsymbol{c}$ à appliquer au miroir déformable est celle qui minimise la quantité :

$$\varepsilon^2 = \|\boldsymbol{m} - D\boldsymbol{c}\|^2 \tag{2.8}$$

La solution est alors donnée par l'inversion généralisée de D (Boyer et al., 1990) :

$$R = (D^t D)^{-1} D^t \tag{2.9}$$

Néanmoins, une attention particulière doit être portée lors de l'inversion de la matrice $D^t D$. En effet, dans la pratique, certains modes mal vus par l'ASO, tels que les modes piston ou gaufre, conduisent à une forte propagation de bruit. Ces modes se caractérisent par des valeurs propres très basses par rapport à celles des autres modes, comme illustré sur la Figure 2.11. Pour contrer cet effet, ces modes propres sont filtrées lors de l'inversion de la matrice. Celle-ci s'effectue donc via une décomposition en valeurs propres dans laquelle les inverses qui correspondent aux modes non désirés sont mises à zéro.

Avec cette méthode, notons tout de même que l'on se restreint à un espace de modes commandés plus petit que celui des modes mesurés, et ce afin d'assurer la stabilité du système d'OA.

2.4.4 Loi de commande en boucle fermée

Un point important à garder à l'esprit est la partie "temps-réel" du contrôleur. En effet, la turbulence atmosphérique est un phénomène dynamique, et la séquence

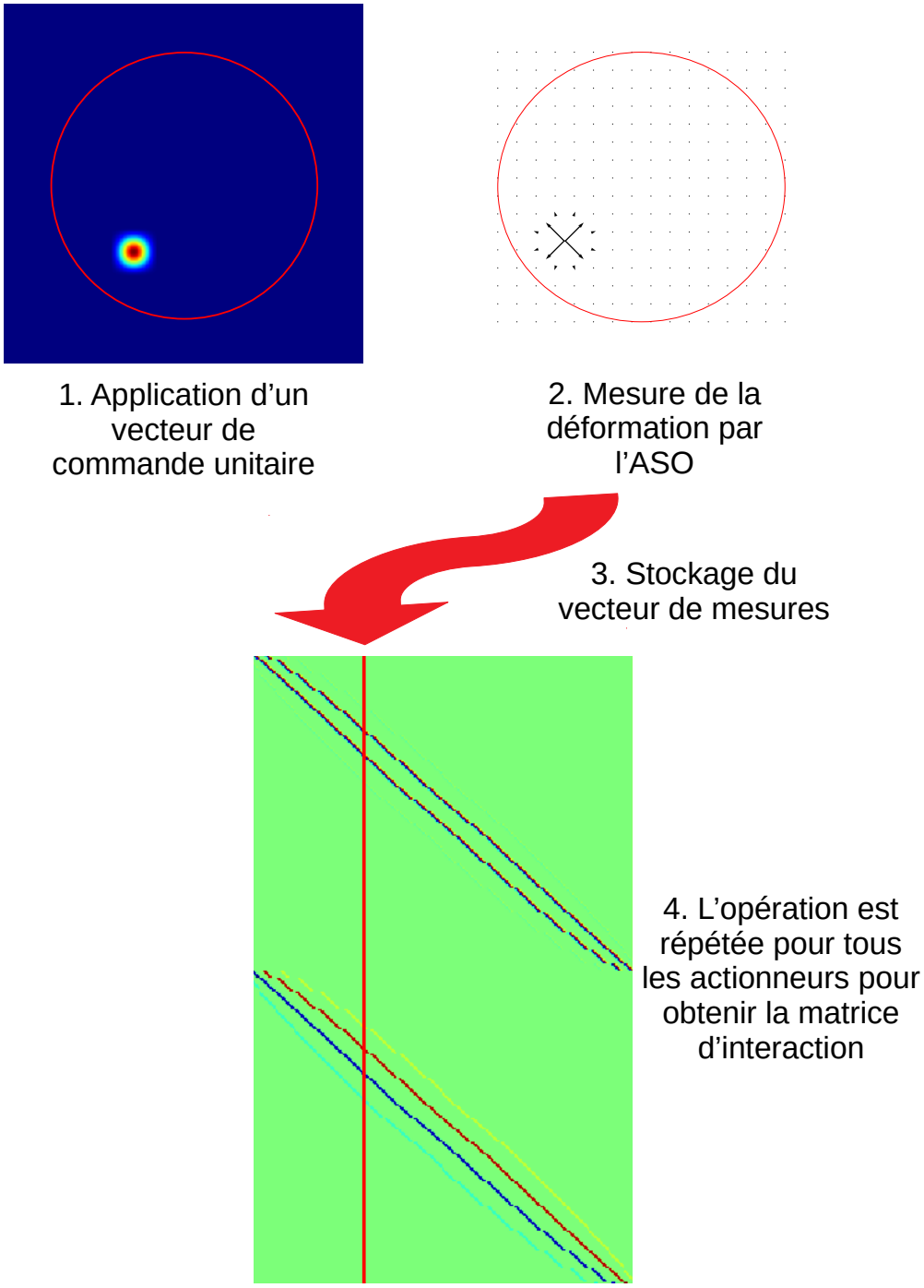

FIGURE 2.10 – Opérations nécessaires à l'étalonnage de la matrice d'interaction dans la base canonique du miroir déformable

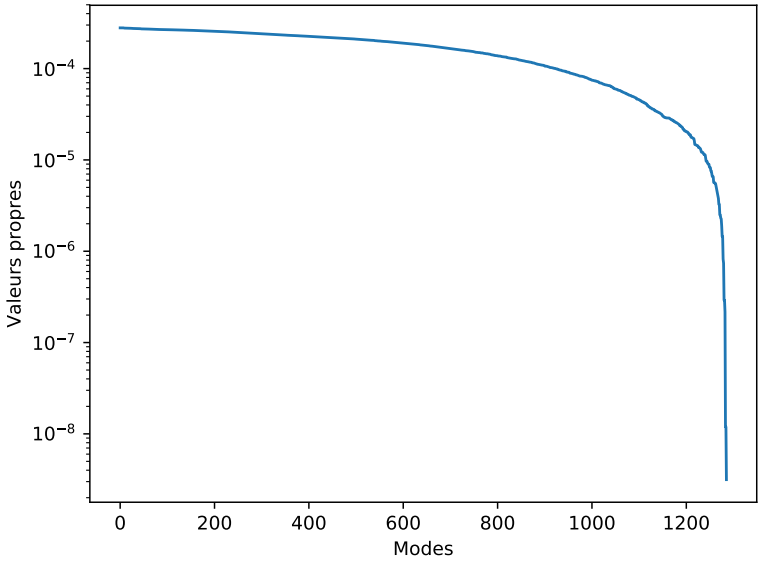

FIGURE 2.11 – Exemple de valeurs propres d'une matrice d'interaction

d'opérations du système d'OA (mesure, correction, etc.) n'est pas instantanée. De fait, le temps que le système applique la correction nécessaire, la déformation du front d'onde aura déjà évolué par rapport à celle qui aura été mesurée. Ce délai intrinsèque au fonctionnement d'un système d'OA a un impact important sur la stabilité du système qui nécessite alors l'introduction d'un gain inférieur à 1 dans la boucle, et les performances de l'OA s'en trouvent dégradées.

La Figure 2.12 schématise un système d'OA fonctionnant en boucle fermée, c'est-à-dire que l'ASO mesure la phase corrigée par le miroir déformable (Madec, 1999). Si une erreur est commise, notamment à cause du délai, le système sera capable de se corriger lui-même.

La loi de commande définit comment les commandes à appliquer au miroir déformable sont calculées. L'une des plus simples et des plus répandues est l'intégrateur dans laquelle la commande du miroir à l'instant précédent est incrémentée à partir de la mesure courante (Demerle et al., 1994) :

$$\boldsymbol{c}_k = \boldsymbol{c}_{k-1} - g\, R\, \boldsymbol{m}_k \tag{2.10}$$

où g est appelé le gain de boucle. Ce paramètre important permet d'ajuster la loi de commande : un gain faible permet d'assurer la stabilité de la boucle, mais induit une erreur temporelle plus importante. Inversement, un gain élevé tend à minimiser cette erreur temporelle, mais augmente la propagation du bruit, ce qui nuit à la stabilité du système.

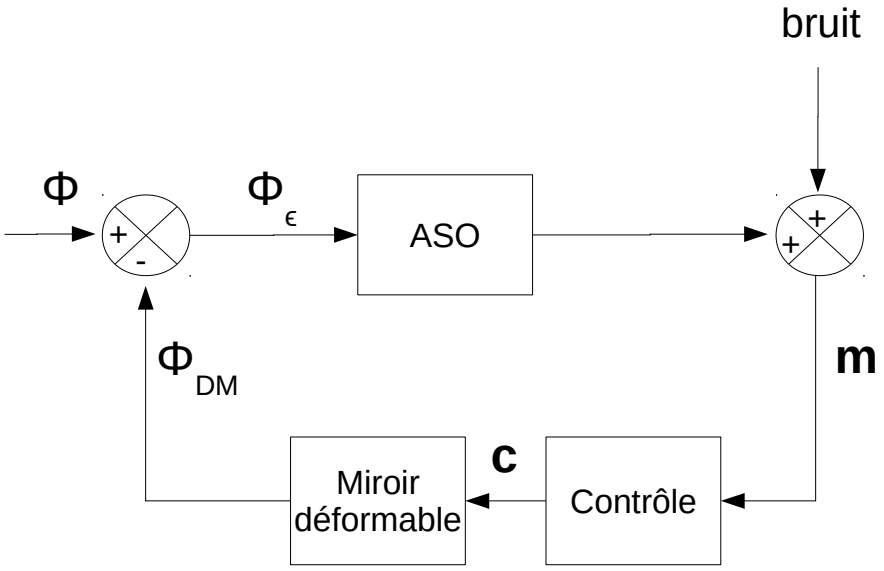

FIGURE 2.12 – Schéma d'un système d'OA en boucle fermée

2.4.5 D'autres façons de faire

Ce que j'ai décrit depuis le début de cette section est une approche possible du problème du contrôle du système d'OA et que je qualifierai de "classique" dans la mesure où les premiers systèmes d'OA utilisaient cette méthode (Rousset et al., 1990) et que cette dernière est toujours la plus couramment utilisée aujourd'hui comme implémentation de base des systèmes d'OA tel que décrit jusqu'ici.

Il est important de signaler qu'il existe bien d'autres façons de procéder. Si je devais en citer quelques unes :

- **la variance minimale** [4] **:** semblable à l'approche des moindres carrés décrite ci-dessus, mais le critère de minimisation prend en compte les connaissances que l'on peut avoir sur les propriétés statistiques de la turbulence (Wallner, 1983).
- **la commande optimale Linéaire Quadratique Gaussienne (LQG) :** issue de la théorie de la commande optimale, elle se base sur une représentation d'état du système d'OA et de la turbulence.(Le Roux et al., 2004; Kulcsár et al., 2006; Petit et al., 2009)
- **le reconstructeur cumulatif avec décomposition en domaines** [5] **:** l'algorithme CuReD permet une reconstruction du front d'onde rapide tout en limitant la propagation de bruit induite par l'algorithme CuRe original, sans décomposition en domaines (Rosensteiner, 2012).

2.4.6 Fonctions de transfert du système et gain optimal

Le comportement temporel de la boucle d'OA peut être formalisé par sa fonction de transfert. Je me place dans le formalisme de Laplace avec $p = 2i\pi f$: la Figure 2.13 reprend la Figure 2.12 dans ce contexte, en plaçant le miroir déformable et le contrôleur dans la fonction de transfert $h_{sys}(p)$.

Temporellement, l'ASO intègre le front d'onde incident pendant une période d'échantillonnage T_e : il peut donc être vu comme une fonction porte de largeur T_e pendant laquelle la phase est moyennée. Sa fonction de transfert $h_{aso}(p)$ peut alors s'écrire Rigaut et al. (1998) :

$$h_{aso}(p) = \frac{1 - \exp(-pT_e)}{pT_e} \tag{2.11}$$

Le reste du système inclut la loi de commande et le miroir déformable. L'ensemble peut être décomposé en quatre éléments simples :

- le gain de la boucle g
- l'intégrateur
- un délai τ entre le moment de la mesure et celui de la correction
- le convertisseur numérique-analogique du miroir déformable, aussi appelé bloqueur

4. *Minimum variance* en anglais
5. *Cumulative Reconstructor with Domain decomposition (CuReD)* en anglais

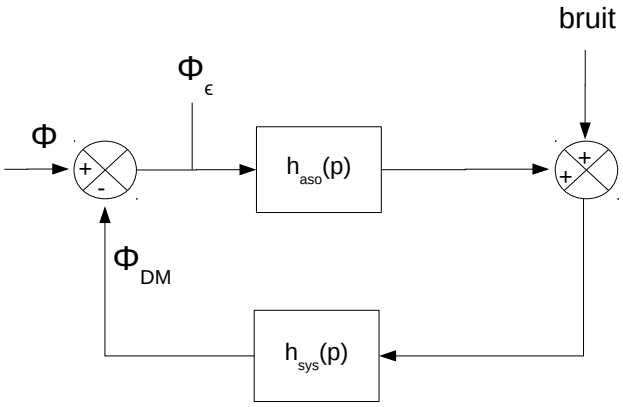

FIGURE 2.13 – Schéma de la boucle d'AO en terme de fonctions de transfert

Ainsi, on en déduit une expression de la fonction de transfert correspondante (Madec, 1999) :

$$h_{sys}(p) = \frac{g\ \exp(-p\tau)}{p\,T_e} \tag{2.12}$$

La fonction de transfert en boucle ouverte du système s'écrit alors :

$$h_{bo}(p) = \frac{\phi_\varepsilon}{\phi_{DM}} = h_{aso}(p).h_{sys}(p) \tag{2.13}$$

Enfin, on en déduit la fonction de transfert en boucle fermée h_{bf} ainsi que la fonction de transfert de correction h_{cor}, également appelée fonction de transfert de réjection :

$$h_{bf}(p) = \frac{h_{bo}(p)}{1 + h_{bo}(p)} \tag{2.14}$$

$$h_{cor}(p) = \frac{\phi_\varepsilon}{\phi} = \frac{1}{1 + h_{bo}(p)} \tag{2.15}$$

Il est également possible de définir la fonction de transfert du bruit $h_n(p)$:

$$h_n(p) = \frac{h_{sys}(p)}{1 + h_{bo}(p)} \tag{2.16}$$

La Figure 2.14 donne une représentation des modules carrés des fonctions de transfert de réjection et de bruit pour différentes valeurs du gain de boucle.

À partir de ces propriétés temporelles, Gendron (1995) montre qu'il existe des gains modaux optimaux pour chaque modes commandés par le système d'OA et pour lesquelles l'erreur résiduelle est minimale.

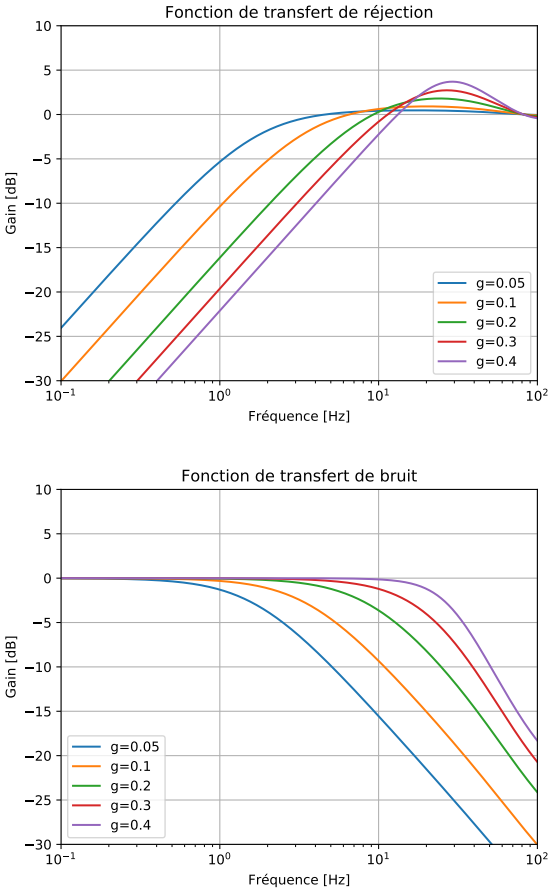

FIGURE 2.14 – Diagrammes de Bode en gain des fonctions de transfert de réjection (en haut) et de bruit (en bas) pour différentes valeurs du gain de boucle. La période d'échantillonnage T_e est de 5 ms pour un délai τ de 3,5 ms. (*Crédit : E. Gendron*)

2.5 Mesure de la performance d'une OA

Afin de caractériser la qualité de correction d'un système d'OA, on a pour habitude de comparer la FEP obtenue après correction par l'OA à la FEP théorique du télescope à la limite de diffraction. Notamment, le rapport des intensités des pics de diffraction de ces deux FEP est appelé rapport de Strehl :

$$SR = \frac{FEP(\mathbf{0})}{Airy(\mathbf{0})} \tag{2.17}$$

Cette grandeur, comprise entre 0 et 1, permet d'évaluer rapidement la correction apportée par l'OA, un rapport de Strehl de 1 signifiant que la correction est parfaite.

Comme exprimé dans l'Eq. 1.3, la FEP dépend de la phase incidente. Par conséquent, il est possible d'approximer le rapport de Strehl en connaissant l'erreur résiduelle sur la phase après correction de l'OA (Conan, 1994) :

$$SR \approx \exp(-\sigma^2_{\phi_\varepsilon}) \tag{2.18}$$

Une autre grandeur intéressante est l'énergie encadrée définit comme :

$$EE = \frac{\int_S FEP(\boldsymbol{r})\boldsymbol{dr}}{\int FEP(\boldsymbol{r})\boldsymbol{dr}} \tag{2.19}$$

L'énergie encadrée permet de quantifier l'énergie contenue dans une certaine surface S relativement à l'énergie totale contenue dans la FEP.

2.6 Limitations et budget d'erreur de l'OA

Comme tout système physique, un système d'OA ne peut pas être parfait. J'ai déjà cité plus haut l'erreur temporelle intrinsèque au fonctionnement du système, mais cette erreur n'est pas la seule. L'erreur résiduelle totale $\sigma^2_{\phi_\varepsilon}$ peut être décomposée selon plusieurs termes formant le budget d'erreur du système d'OA. Ainsi, en supposant que ces termes sont statistiquement indépendants (nous verrons que ce n'est pas toujours le cas), on écrit :

$$\sigma^2_{\phi_\varepsilon} \approx \sigma^2_{temp} + \sigma^2_{fit} + \sigma^2_{alias} + \sigma^2_{bruit} + \sigma^2_{aniso} + \sigma^2_{dev} + \sigma^2_{filt} + \sigma^2_{sci} + \sigma^2_{ncpa} \tag{2.20}$$

Je décrit ici les différents termes composant ce budget d'erreur.

2.6.1 L'erreur temporelle σ^2_{temp}

L'erreur temporelle, ou erreur de bande passante, est donc due au retard dans la boucle du système d'OA. Admettons que l'ASO mesure une déformation à l'instant t, il faudra un délai τ au contrôleur temps-réel avant de pouvoir appliquer la correction correspondante sur le miroir déformable, et cette dernière sera effective à l'instant $t + \tau$. Cependant, la turbulence atmosphérique aura eu le temps d'évoluer pendant ce

délai, et la déformation effectivement corrigée par le miroir ne sera pas exactement celle mesurée à l'instant t par l'ASO. À cela s'ajoute l'impact du gain de boucle introduit pour assurer la stabilité du système. La performance s'en trouve réduite puisque la commande calculée par le système est multipliée par ce gain inférieur à 1 avant d'être appliquée au miroir déformable. Comme nous pouvons le voir notamment sur la Figure 2.14, le gain de la boucle impacte également la bande passante du système.

L'erreur temporelle σ^2_{temp} peut s'exprimer sous la forme (Greenwood, 1977) :

$$\sigma^2_{temp} = 0,243 \left(\frac{\bar{v}}{r_0}\right)^{5/3} f_G^{-5/3} \tag{2.21}$$

où $\bar{v}$ est tel que définit par l'Eq. 1.19 et f_G est la fréquence de coupure à -3 dB de la fonction de transfert en boucle fermée du système d'OA.

2.6.2 L'erreur de sous-modélisation σ^2_{fit}

Le miroir déformable d'un système d'OA n'est constitué que d'un nombre fini d'actionneurs. Ainsi, la phase turbulente peut être décomposée en deux parties distinctes :

$$\phi(\boldsymbol{r}) = \phi_{\parallel}(\boldsymbol{r}) + \phi_{\perp}(\boldsymbol{r}) \tag{2.22}$$

où $\phi_{\parallel}(\boldsymbol{r})$ est la composante colinéaire à l'espace des modes accessibles au miroir déformable, tandis que $\Phi_{\perp}(\boldsymbol{r})$ est la composante orthogonale à cet espace.

Par définition, le miroir déformable est donc incapable de corriger $\Phi_{\perp}(\boldsymbol{r})$, qui se retrouvera donc nécessairement dans la phase résiduelle de la voie scientifique. Cette erreur de sous-modélisation [6], peut s'approximer par (Hudgin, 1977) :

$$\sigma^2_{fit} \approx 0,23 \left(\frac{d}{r_0}\right)^{5/3} \tag{2.23}$$

avec d la distance inter-actionneur du miroir déformable. On peut alors définir la fréquence de coupure f_c du miroir, au-delà de laquelle les fréquences spatiales ne pourront plus être corrigées :

$$f_c = \frac{1}{2d} \tag{2.24}$$

2.6.3 L'erreur de repliement σ^2_{alias}

Le pas des sous-pupilles d'un analyseur Shack-Hartmann (ou celui des pixels pour un analyseur pyramide) définit les fréquences spatiales auxquelles a accès l'ASO. Les fréquences spatiales supérieures à la fréquence de coupure de l'ASO sont repliées sur la mesure et faussement interprétées par l'ASO comme des basses fréquences. Pour un analyseur Shack-Hartmann, cette erreur peut s'écrire (Rigaut et al., 1998) :

$$\sigma^2_{alias} = 0.08 \left(\frac{d}{r_0}\right)^{5/3} \tag{2.25}$$

6. *fitting error* en anglais

soit environ le tiers de l'erreur de sous-modélisation.

Cette expression ne tient pas compte de la fonction de transfert du bruit (cf. Éq. 2.16).

2.6.4 L'erreur de bruit σ^2_{bruit}

Les images fournies par l'ASO à partir desquelles sont mesurées les déformations du front d'onde sont perturbées par du bruit. Ce bruit a deux origines : le bruit de photons et le bruit électronique du détecteur de l'ASO.

Pour un analyseur Shack-Hartmann, Rousset et al. (1987) donne une expression de la variance de l'erreur due au bruit de photons σ^2_{ph} et de celle due au bruit électronique σ^2_{lec} :

$$\sigma^2_{ph} = \frac{\pi^2}{2}\frac{1}{n_{ph}}\left(\frac{N_T}{N_D}\right)^2 \tag{2.26}$$

$$\sigma^2_{lec} = \frac{\pi^2}{3}\frac{\sigma^2_e N^2_s}{n^2_{ph}}\left(\frac{N_S}{N_D}\right)^2 \tag{2.27}$$

avec :

- n_{ph} le nombre de photons par sous-pupille
- N_T la largeur à mi-hauteur de la tache image en pixels
- N_D la largeur à mi-hauteur de la tache de diffraction en pixels
- σ^2_e la variance du bruit électronique par pixel et par trame en électrons^2
- N_S le nombre de pixels dans la zone de calcul du centre de gravité

Finalement, l'erreur de bruit s'écrit alors :

$$\sigma^2_{bruit} = \sigma^2_{ph} + \sigma^2_{lec} \tag{2.28}$$

Cette expression ne tient pas compte de la fonction de transfert du bruit (cf. Éq. 2.16).

2.6.5 L'erreur d'anisoplanétisme σ^2_{aniso}

Étant donnée la problématique du bruit décrite juste au-dessus, il est aisément compréhensible que l'ASO doit être asservi sur une étoile assez brillante pour amener un nombre suffisant de photons par sous-pupille. Or, il est relativement rare que l'objet d'intérêt scientifique que l'on cherche à observer soit suffisamment brillant.

Dès lors que l'ASO est asservi sur une étoile qui n'est pas l'objet d'intérêt, on parle de système d'optique adaptative hors axe. La Figure 2.15 schématise la situation. L'ASO n'analyse alors pas la même portion de turbulence que celle traversée par le front d'onde issu de l'objet d'intérêt, causant ainsi une erreur supplémentaire sur la correction de l'OA : c'est l'effet d'anisoplanétisme.

Fried (1982) définit l'angle d'anisoplanétisme θ_0 comme l'écart angulaire entre l'objet d'intérêt et l'étoile guide de l'ASO pour lequel la variance de l'erreur d'aniso-

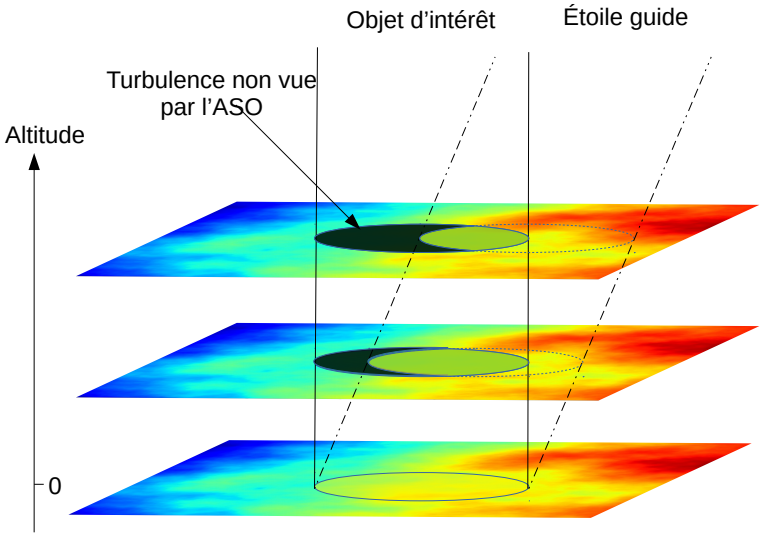

FIGURE 2.15 – Schématisation de l'effet d'anisoplanétisme

planétisme est égale à 1 rad^2 :

$$\theta_0 = 0,314 \cos(\gamma) \left(\frac{r_0}{\bar{h}}\right) \tag{2.29}$$

où γ est l'angle zénithal et $\bar{h}$ l'altitude moyenne pondérée des couches turbulentes. On en déduit également la variance de l'erreur d'anisoplanétisme pour un angle θ :

$$\sigma^2_{aniso} = \left(\frac{\theta}{\theta_0}\right)^{5/3} \tag{2.30}$$

L'anisoplanétisme est l'une des limitations fondamentales de l'optique adaptative. Dans la pratique, l'angle d'anisoplanétisme n'excède pas la dizaine de secondes d'arc dans l'infrarouge, ce qui limite grandement le champ de correction et la couverture du ciel d'un système d'OA. En effet, il devient vite ardu de trouver une étoile guide assez brillante pour l'ASO dans un rayon aussi petit autour d'un point d'intérêt donné. On estime ainsi qu'un système d'OA classique tel que décrit jusqu'ici ne peut offrir une couverture du ciel que de l'ordre de 0,01%.

2.6.6 Les déviations de la mesure de l'ASO σ^2_{dev}

Je parle ici d'un type d'erreur un peu plus difficile à définir, ou tout du moins à nommer correctement. Initialement, je décrivai cette erreur comme les "non-linéarités" de l'ASO. En effet, cette erreur inclut notamment les effets de sous-échantillonnage de l'ASO et de troncatures des spots, pour un ASO de type Shack-Hartmann par exemple, qui impactent la sensibilité de l'ASO.

Néanmoins, au premier ordre, cela se traduit par l'introduction d'un gain optique γ sur la mesure (Véran & Herriot, 2000) dans l'Éq. 2.6 :

$$\boldsymbol{m'} = \gamma\, R\mathbf{c} + \boldsymbol{n} \tag{2.31}$$

Or il s'avère que cette équation est linéaire. L'appellation de "non-linéarité" devient alors beaucoup moins pertinente. J'ai donc opté pour le nom de "déviation de la mesure" puisque cette erreur inclut tout ce qui fait dévier la mesure obtenue de la mesure théorique telle que décrite par l'Éq. 2.3 pour le Shack-Hartmann, en dehors du bruit (cf. Section 2.6.4) et du repliement (cf. Section 2.6.3).

2.6.7 L'erreur de filtrage σ^2_{filt}

Comme expliqué dans la Section 2.4.3, l'inversion de la matrice d'interaction nécessite un reconditionnement préalable. La conséquence de ceci est que certains modes de hauts ordres appartenant pourtant à l'espace du miroir déformable ne sont plus commandés, et donc non corrigés par l'OA. Ce procédé permet d'assurer la stabilité de la boucle, mais induit une erreur supplémentaire. Néanmoins, cette erreur est souvent négligeable si la matrice a été correctement conditionnée.

Il peut être intéressant de noter que puisque ces modes filtrés ne sont plus commandés par l'OA, ils sont alors orthogonaux aux modes effectivement commandés (en supposant l'utilisation d'une base orthonormée) et cette erreur peut être incluse dans l'erreur de sous-modélisation décrite dans la Section 2.6.2.

2.6.8 L'erreur de scintillation σ^2_{sci}

J'ai jusqu'ici considéré que la turbulence atmosphérique n'introduisait que du déphasage sur l'onde qui la traverse. Ce n'est pas complètement vrai, dans la mesure où l'amplitude de l'onde fluctue également par propagation de Fresnel, induisant de la scintillation dans le plan pupille. Un système d'OA ne pouvant agir que sur la phase, ce phénomène ne peut être corrigé par ce biais. Cette erreur peut cependant être négligée dans la plupart des cas d'application de l'OA (Roddier, 1981).

2.6.9 Les aberrations non communes σ^2_{ncpa}

Le chemin optique de la voie scientifique menant à la caméra n'est pas parfait : il introduit des aberrations optiques qui sont hors du système d'OA, et donc non corrigées par celui-ci. De même, le chemin optique de la voie d'analyse présente des aberrations qui ne sont pas dans la voie scientifique. Ces aberrations non communes [7] se retrouvent donc directement sur la caméra scientifique et impactent la qualité de l'image.

Évidemment, on cherchera toujours à avoir le moins possible d'aberrations afin de minimiser cette erreur. Néanmoins, il est possible de quantifier ces aberrations, par

7. *Non-Common Path Aberrations* (NCPA) en anglais

exemple grâce à des algorithmes de diversité de phase (Gonsalves, 1982). Une fois mesurées, les pentes de référence de l'ASO sont modifiées pour les prendre en compte.

2.7 Des lasers pour illuminer le ciel

Les systèmes d'OA tels que décrit jusqu'ici, i.e. avec un unique ASO et un unique miroir déformable, sont qualifiés de SCAO [8]. Comme nous venons de le voir, une des principales limitations de ces systèmes d'OA est l'effet d'anisoplanétisme. L'OA ayant besoin d'une étoile brillante pour être asservie, la couverture du ciel s'en trouve très limitée.

Pour pallier cette importante limitation, un laser peut être utilisé afin de créer une "étoile artificielle" [9] (Foy & Labeyrie, 1985).

2.7.1 Étoile laser de type sodium

L'atmosphère est notamment composée d'une couche de sodium d'une épaisseur variable de l'ordre de la dizaine de kilomètres et située entre 80 et 110 kms environ. L'idée est alors de venir exciter avec un faisceau laser les atomes de sodium de cette couche afin qu'ils émettent des photons en retour par émission spontanée (Gardner et al., 1989). La longueur d'onde du laser utilisé pour l'excitation doit correspondre à la raie d'absorption du sodium, soit 589 nm. On reconnaît ainsi facilement ce type de laser à leur couleur jaune orangé (Figure 2.16).

Ce type d'étoiles artificielles est utilisé sur divers instrument au Keck (Wizinowich et al., 2004), à Gemini (Rigaut et al., 2014) ou encore au VLT (Kolb et al., 2017).

2.7.2 Étoile laser de type Rayleigh

Cet autre type d'étoile artificielle est créé grâce à la diffusion Rayleigh des molécules de l'atmosphère (Foy & Labeyrie, 1985). La lumière, en traversant l'atmosphère, interagit avec les molécules sur son passage en y créant des moments dipolaires. Ces dipôles rayonnent alors à la même fréquence que la fréquence d'excitation.

La diffusion Rayleigh ayant une dépendance en λ^{-4}, les lasers utilisés sont plutôt de longueurs d'onde courtes (bleu ou vert). À noter également que la diffusion Rayleigh dépend également de la densité moléculaire du milieu. Ainsi, les molécules d'air se raréfiant avec l'altitude, une étoile laser de type Rayleigh est généralement formée à une vingtaine de kilomètres d'altitude au maximum. De plus, la diffusion Rayleigh ayant lieu tout le long du trajet du faisceau laser, on utilise généralement un laser pulsé et une caméra dont l'ouverture est synchronisée avec les pulses du laser (Thompson & Gardner, 1989). Contrairement à l'étoile artificielle sodium, la distance qui sépare l'étoile laser du télescope est donc fixe, et ce quelque soit le pointage du télescope.

8. *Single Conjugate Adaptive Optics* en anglais
9. Laser Guide Star (LGS) en anglais

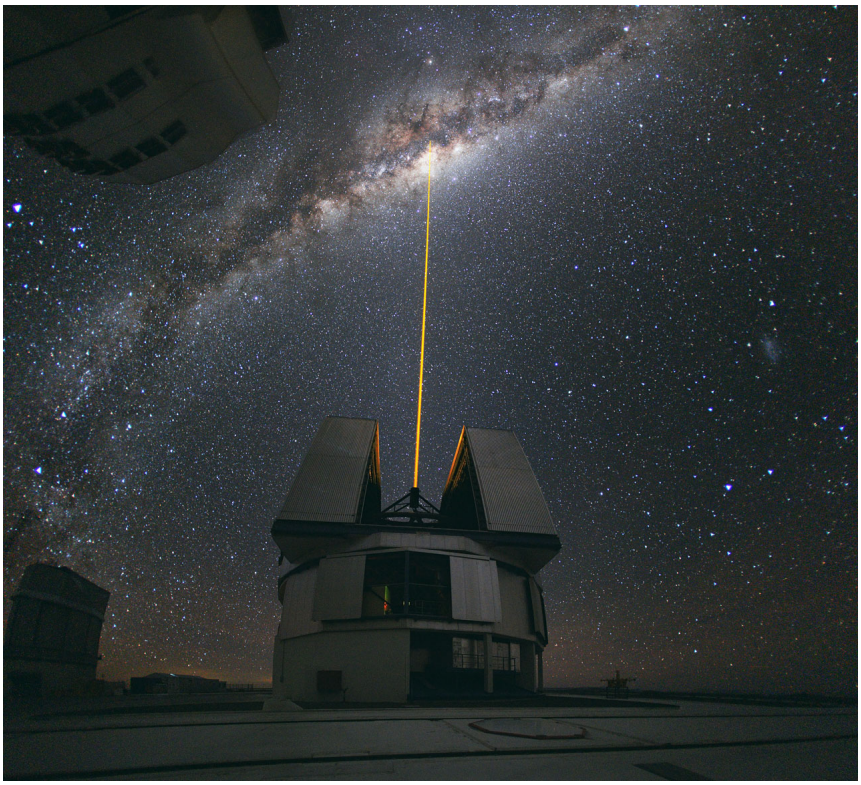

FIGURE 2.16 – Etoile laser de type sodium tirée depuis l'un des télescopes du VLT. *Crédit : ESO*

À titre d'exemple, l'instrument ARGOS (Rabien et al., 2010) utilise ce type d'étoile artificielle.

2.7.3 Une couverture du ciel plus importante, mais à quel prix ?

L'étoile artificielle apporte en théorie une couverture du ciel parfaite, le laser pouvant être pointé librement. Cependant, cette solution n'est pas sans conséquences sur les performances du système d'OA.

L'effet de cône

Les méthodes décrites ci-dessus permettent de créer une étoile artificielle dans l'atmosphère. Cette étoile ne peut pas être considérée comme étant située à l'infini comme on le ferait avec une étoile naturelle. La Figure 2.17 illustre les conséquences de ce fait. La lumière émise par l'étoile artificielle qui est captée par le télescope est

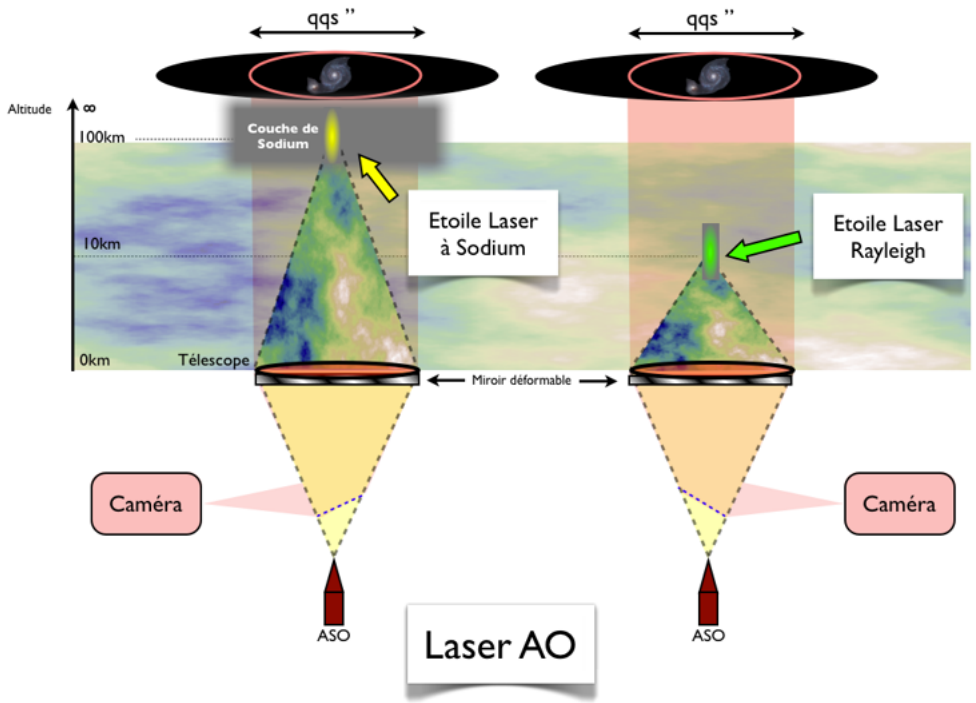

FIGURE 2.17 – Illustration de l'effet de cône pour une étoile laser de type sodium (à gauche) et de type Rayleigh (à droite) *Crédit : F. Vidal*

comprise dans un cône. S'en suit qu'une partie de la turbulence atmosphérique située en altitude et traversée par l'onde émise par l'objet observé ne peut être vue par le système d'OA (Fried & Belsher, 1994; Foy, 2000). Ce phénomène est d'autant plus important dans le cas d'une étoile laser de type Rayleigh dont l'altitude est plus basse que pour une étoile sodium (Molodij & Rousset, 1997).

Indétermination du tip-tilt

Une autre limitation due à l'utilisation d'une étoile artificielle est l'indétermination du tip-tilt (Rigaut & Gendron, 1992). En effet, la lumière émise par le laser pour créer l'étoile laser est également impactée par la turbulence atmosphérique sur le trajet montant, et plus particulièrement par les modes de basculement tip-tilt. Ainsi, la perturbation subie lors de ce trajet ne peut être séparée de celle subie au retour par le système d'OA, ce qui rend impossible la mesure du tip-tilt avec une étoile laser : une étoile naturelle doit alors être utilisée en sus pour lever cette indétermination.

Il est cependant intéressant de noter que, dans ce cas, l'étoile naturelle n'est utilisée que pour la mesure de ces modes de bas ordre, ce qui réduit drastiquement la contrainte sur la magnitude de l'étoile guide naturelle nécessaire à la mesure. La couverture du ciel offerte par le laser est alors en grande partie conservée.

Les modes de basculements ne sont pas les seuls modes indéterminés par l'étoile laser. Notamment, le focus est difficile à mesurer avec une étoile laser sodium puisque l'altitude moyenne et l'épaisseur de la couche de sodium fluctuent dans le temps. Pour palier à ce problème, il est possible d'ajuster le focus du système d'OA en cours d'opération (Summers et al., 2004).

L'élongation

Une étoile laser n'est pas ponctuelle, mais possède une forme allongée. En effet, dans le cas de l'étoile laser de type sodium par exemple, la couche de sodium est épaisse d'une dizaine de kilomètres. De fait, l'étoile apparaît comme ponctuelle (ou du moins de la taille de l'image du spot laser dans l'atmosphère) si on la regarde dans la pupille depuis un point proche du point de lancement du laser, puis de plus en plus allongée par effet de perspective à mesure que l'on s'écarte de ce point. L'élongation est donc dépendante de la position de chaque sous-pupille par rapport à l'origine d'émission du laser, comme illustré dans la Figure 2.18, où les spots diamétralement opposés au laser d'émission sont beaucoup plus allongés. Les fluctuations de densité de la couche sodium ont également un impact sur l'image du spot laser allongé, créant des variations d'intensité en son sein (Pfrommer & Hickson, 2010).

La détermination du centre de gravité de ces spots allongés devient alors problématique (Gilles & Ellerbroek, 2006; Gratadour et al., 2010). Le dimensionnement même de l'ASO s'en trouve impacté, l'élongation pouvant provoquer une troncation du spot au sein de la sous-pupille si son allongement est plus grand que la taille de cette dernière (Muller et al., 2010).

2.8 Des OA plus complexes

L'étoile laser permet donc de remédier à l'une des principales limitations de l'OA classique, à savoir sa faible couverture du ciel. Cependant, cela coûte quelques erreurs supplémentaires qui dégradent les performances d'un système d'OA fonctionnant avec une étoile artificielle par rapport à celles d'une OA classique (Bardou et al., 2016).

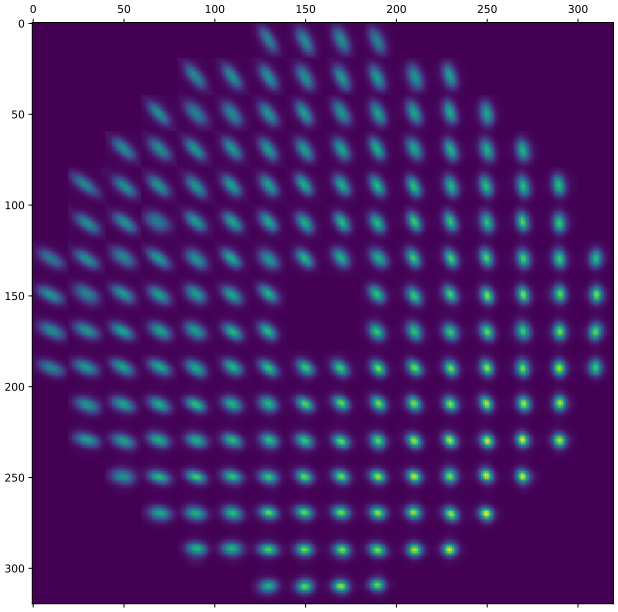

FIGURE 2.18 – Image simulée d'un Shack-Hartmann asservi sur une étoile laser lancée depuis le bord du télescope (en bas à droite de l'image).

Une autre limitation de l'OA classique, également liée à l'anisoplanétisme, est son faible rayon d'action dans le champ. En effet, la correction est efficace sur un champ de quelques secondes d'arc typiquement, bien loin d'être suffisant pour corriger l'intégralité du champ de vue d'un grand télescope. L'observation d'un amas d'étoiles devient alors plus difficile à mettre en place, obligeant différents pointages pour faire une mosaïque du champ complet.

2.8.1 Plus de lasers pour limiter l'effet de cône

Afin d'améliorer les performances d'une OA à base d'étoile laser, l'idée est de diminuer l'impact de l'effet de cône. Pour ce faire, l'utilisation de plusieurs étoiles laser, chacune étant couplée à un ASO différent, permet alors de couvrir une plus grande partie du volume turbulent effectivement traversé par l'onde venant de l'objet d'intérêt (Figure 2.19) (Tallon & Foy, 1990; Welsh & Gardner, 1991). Cette technique, dite de LTAO (Laser Tomography Adaptive Optics), tire parti de méthodes de tomographie pour reconstruire en trois dimensions le volume turbulent vu par le télescope à partir des mesures hors axe effectuées par chaque ASO (Ellerbroek, 1994). À partir de cette connaissance, l'opération tomographique permet de se projeter dans une direction quelconque au sein de ce volume turbulent, et donc de calculer la correction à appliquer dans la direction d'intérêt.

La couverture du ciel offerte par cette méthode est alors quasi totale et on aurait envie de placer assez d'étoiles laser dans le champ pour être capable de corriger

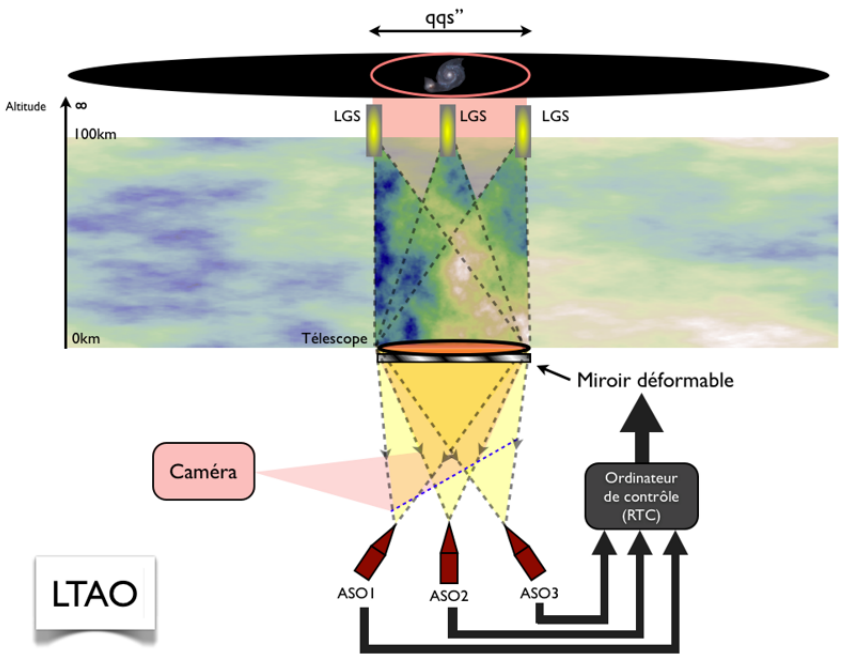

FIGURE 2.19 – Schéma de principe de la LTAO. *Crédit : F. Vidal*

efficacement une grande partie de ce dernier. Malheureusement, ce n'est pas aussi simple.

2.8.2 L'erreur de sous-modélisation généralisée

Admettons que nous disposions d'un nombre suffisant d'étoiles et que l'opération tomographique ne fasse aucune erreur et permette d'avoir une connaissance parfaite du volume turbulent dans le champ. Malgré cela, corriger l'intégralité du champ sera impossible avec un nombre fini de miroirs déformables (Ellerbroek, 1994; Rigaut et al., 2000; Gendron et al., 2014).

Supposons un miroir déformable conjugué à une altitude h_{DM} et que nous voulions corriger un champ de taille angulaire θ. Supposons également qu'une couche turbulente soit à une distance Δh du miroir déformable. La Figure 2.20 donne une représentation géométrique de la situation.

Dans ce cas de figure, le miroir déformable doit corriger localement (i.e. sur un actionneur) la phase le long d'un segment $\theta.\Delta h$ (je me place volontairement dans un cas unidimensionnel pour simplifier, mais le raisonnement est valable en deux dimensions). Il s'avère qu'aucune solution n'existe pour corriger dans toutes les directions : en pratique, le reconstructeur appliquera sur l'actionneur du miroir une correction de la phase moyennée sur $\theta.\Delta h$ et les fréquences spatiales inaccessibles seront exclues.

Cette généralisation de l'erreur de sous-modélisation montre donc que plus une couche turbulente est éloignée du miroir déformable, moins celui-ci est capable de

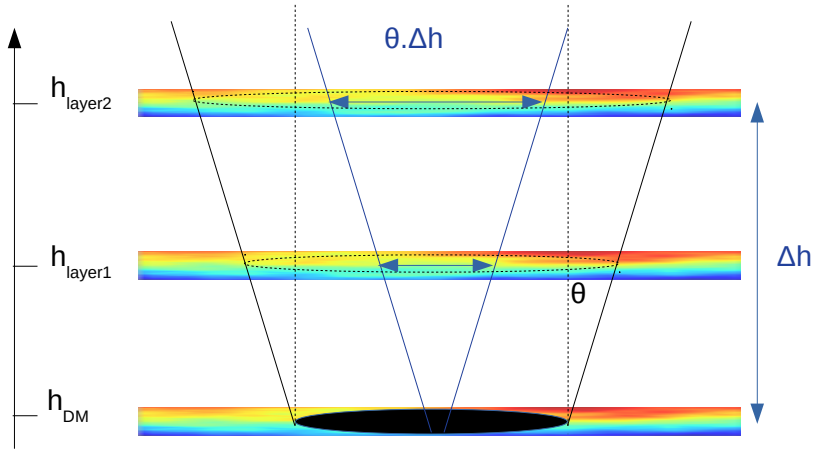

FIGURE 2.20 – Projection du miroir déformable sur les couches turbulentes pour un champ de correction θ

la corriger efficacement. Rigaut et al. (2000) donne d'ailleurs une expression de la différence maximale d'altitude entre le miroir et la couche turbulente pour laquelle l'erreur de sous-modélisation généralisée est du même ordre de grandeur que l'erreur de sous-modélisation simple telle que décrite dans la Section 2.6.2 :

$$\Delta h_{max} = \frac{1.75\, d}{\theta} \tag{2.32}$$

où d est la distance inter-actionneur du miroir déformable.

2.8.3 Systèmes d'OA grand champ

Ainsi, pour un champ de correction d'une arcminute et une distance inter-actionneur de 0.5 m, un miroir déformable est capable de compenser les couches turbulentes sur une distance de 3000 m. Pour obtenir une correction efficace sur ce champ, il faut donc utiliser plusieurs miroirs déformables conjugués à différentes altitudes : c'est le principe de l'optique adaptative multi-conjuguée [10] (Beckers, 1989; Ellerbroek, 1994; Marchetti et al., 2008).

Une autre approche consiste à n'utiliser qu'un seul miroir déformable conjugué au sol. Celui-ci ne corrigera donc que les couches turbulentes proches du sol qui s'avèrent être souvent les plus gênantes. Ce type de système est qualifié de GLAO (Ground-Layer Adaptive Optics en anglais) (Rigaut, 2002; Milton et al., 2008).

Enfin, un autre type de système d'OA grand champ vise à segmenter le champ en petites régions d'intérêts. Chaque région est alors corrigée par un miroir déformable qui lui est dédié. Le champ de correction autour de chaque région n'ayant pas besoin

10. MCAO (Multi-Conjugate Adaptive Optics) en anglais

Nom	Diamètre [m]	Site
Télescope William-Herschel	4,2	La Palma, Iles Canaries
Télescope Magellan	6	Mont Hopkins, Arizona
Gémini Sud	8,1	Cerro Pachon, Chili
Very Large Telescope	8 x 4	Cerro Paranal, Chili
Keck	10	Mauna Kea, Hawaï
Large Binocular Telescope	8 x 2	Mont Graham, Arizona
Grand Télescope des Iles Canaries	10,4	La Palma, Iles Canaries
Giant Magellan Telescope	**21**	Cerro Las Campanas, Chili
Thirty Meter Telescope	**30**	Mauna Kea, Hawaï
Extremely Large Telescope	**39**	Cerro Armazones, Chili

TABLEAU 2.1 – Liste non exhaustive des grands et futurs télescopes géants(en gras)

d'être important, un seul miroir par région est alors suffisant pour obtenir une bonne qualité de correction. On parle alors d'optique adaptative multi-objets [11] (Hammer et al., 2002; Gendron et al., 2011).

2.9 Une autre échelle : les ELT

Si les systèmes d'OA grand champ permettent de mieux tirer parti du champ de vue des grands télescopes, leur complexité n'a rien à voir avec un système d'OA classique. Et ceci est d'autant plus vrai dans la perspective des futurs très grands télescopes [12].

2.9.1 Qu'est-ce qu'un télescope géant ?

La classe des télescopes géants regroupe les futurs télescopes dont le diamètre est supérieur à 20 mètres. Les projets en cours sont au nombre de trois :

- le Giant Magellan Telescope (GMT) (McCarthy et al., 2016)
- le Thirty Meter Telescope (TMT) (Simard et al., 2016)
- l'Extremely Large Telescope (ELT) (Tamai et al., 2016)

Le Tableau 2.1 référence le diamètre de certains des plus grands télescopes actuels ainsi que ceux de ces futurs télescopes géants. Les diamètres annoncés pour ces derniers représentent des défis scientifiques et techniques dans tous les domaines (optique, mécanique, structurel, etc.).

Si le GMT et TMT sont dirigés par des consortia majoritairement américains, l'ELT quant à lui est européen, chapeauté par l'Observatoire européen austral [13]. Afin de présenter les problématiques liées à cette classe de télescope pour l'optique adaptative en particulier, je me concentrerai sur l'exemple de l'ELT.

11. Multi-Object Adaptive Optics (MOAO) en anglais
12. Extremely Large Telescope (ELT) en anglais
13. European Southern Observatory (ESO) en anglais

2.9.2 Une rapide présentation de l'ELT

Puisqu'une image vaut souvent mieux qu'un long discours, la Figure 2.21 permet de se rendre compte de l'ampleur du projet.

La Figure 2.22 représente le design optique de l'ELT. Prévu pour une première lumière en 2024 (au moment où sont écrites ces lignes), le miroir primaire de 39 mètres de diamètre sera composé de 798 segments hexagonaux de 1,4 mètres chacun. Le miroir secondaire, d'un diamètre de 4,2 mètres, renvoie la lumière au miroir tertiaire à travers un trou au centre du miroir M4.

Le miroir plan M4 est un composant critique de l'ELT. Ce dernier est en effet le miroir déformable du système d'OA embarqué par le télescope (avec le miroir plan M5 tip-tilt). Ce miroir légèrement elliptique, avec un petit diamètre de 2,4 mètres et un grand diamètre de 2,5 mètres, sera constitué de 5319 actionneurs. Autre particularité, ce miroir sera segmenté en 6 pétales afin de réduire les coûts et les risques de casse, et de faciliter la maintenance (Vernet et al., 2014). L'ELT embarque également 6 lasers pour étoiles sodium.

Après cette rapide présentation, j'en viens maintenant à ce qui nous intéresse ici : les instruments d'optique adaptative prévus en première lumière. Leur dimensionnement permet à lui seul de se rendre compte du défi à relever dans ce domaine.

2.9.3 Les OA de première lumière de l'ELT

MICADO-MAORY

MICADO (Multi-AO Imaging Camera for Deep Observations) est un spectro-imageur infrarouge (Davies et al., 2016). Le mode imageur de MICADO sera capable de fournir au choix des images grand champ (autour de 50 x 50 arscec^2) en bande H et K (mode MCAO), ou des images haute résolution sur un champ plus restreint mais permettant un échantillonnage suffisant jusqu'à la bande I (mode SCAO). Le mode spectrographe sera également capable de fournir différents niveaux de résolution. Les applications scientifiques sont alors variées, allant de la détection d'exoplanètes à l'étude des premières étapes de la formation des galaxies.

Le mode SCAO commandera ainsi les actionneurs du miroir M4 de l'ELT et se basera sur un analyseur pyramide (Clénet et al., 2016). Le dimensionnement de l'optique adaptative prévoit un détecteur de 240x240 pixels avec 96 pixels dans le diamètre des images de la pupille produites par la pyramide. Le contrôleur temps-réel est prévu pour fonctionner à hauteur de 500 Hz.

MAORY (Multi-conjugate Adaptive Optics RelaY) est le module de MCAO (Diolaiti et al., 2016). Il est constitué de 6 analyseurs de type Shack-Hartmann asservis sur étoiles laser, 3 autres ASO de même type sur étoile naturelle pour le contrôle du tip-tilt, et de 1 ou 2 miroirs déformables post-focaux, i.e. en plus du miroir M4 de l'ELT. Chaque analyseur laser possède 80 x 80 sous-pupilles, un détecteur de 880 x 840 pixels fonctionnant à une fréquence maximale de 1 kHz. Le nombre total d'actionneurs à commander se monte à près de 6000, incluant donc ceux du miroir M4

FIGURE 2.21 – Le projet de l'ELT comparé au VLT et au grand colisée. *Crédit : ESO*

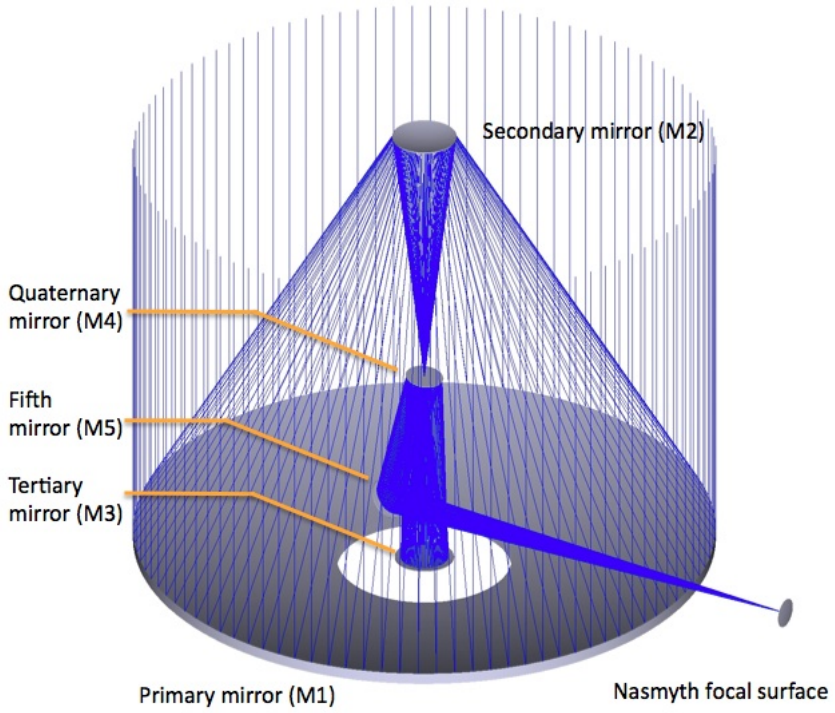

FIGURE 2.22 – Schéma du design optique de l'ELT *(Crédit : ESO)*

(Schreiber et al., 2016).

HARMONI

HARMONI (High Angular Resolution Monolithic Optical and Near-infrared Integral field spectrograph) est un spectrographe à champ intégral offrant jusqu'à 4 échelles spectrales différentes allant du visible au proche infrarouge. En terme d'optique adaptative, l'instrument sera équipé d'un module de SCAO et un module de LTAO. Ce dernier sera également composé de 6 analyseurs laser de 74 x 74 sous-pupilles, aidés par 1 analyseur naturel bas ordre. Le tout permettra de contrôler les actionneurs du miroir M4 (Neichel et al., 2016).

METIS

METIS (Mid-infrared ELT Imager and Spectrograph) est également un spectro-imageur fonctionnant quant à lui dans l'infrarouge moyen(Brandl et al., 2014). L'optique adaptative de METIS est basée sur un système SCAO avec un ASO de type pyramide fonctionnant dans l'infrarouge (bande K).

Des problématiques communes

Chacun de ces instruments doit faire face à des problématiques communes liées à la taille du télescope. À titre d'exemple, les OA grand champ induisent l'acheminement du dit grand champ jusqu'à chaque élément optique du système d'OA. En termes d'optique géométrique pure, on comprend que cela implique un design optique sans précédent devant affronter, entre autres, des défis mécaniques liés à la taille des optiques nécessaires et des défis optiques liés à la fabrication même de ces optiques. Ainsi, les dimensions physiques annoncées de l'ensemble MAORY-MICADO sont parlantes : 4m x 9m x 7m environ. Pour l'OA en elle-même, la fabrication de miroirs déformables possédant plusieurs milliers d'actionneurs devant fonctionner au kilohertz n'a également rien d'aisé.

Une autre problématique à laquelle je m'attache dans ce mémoire est la problématique numérique. Je lui dédie d'ailleurs le prochain chapitre.

Deuxième partie

Simulation numérique haute performance pour l'OA

Chapitre 3

Le calcul haute performance : un besoin pour les ELT

Dans ce dernier chapitre introductif, je me concentre sur la problématique numérique amenée par les ELT dans le cadre de l'OA. En particulier, les besoins en capacité de calcul se concentrent principalement en deux points : le contrôle temps-réel et la simulation numérique. En décrivant le besoin nécessaire dans ces deux domaines, j'introduirai la solution du calcul haute performance, et plus spécifiquement celle du calcul générique sur carte graphique.

Je préfère prévenir le lecteur que cette thèse n'étant pas une thèse en sciences de l'informatique, ce chapitre n'a pas pour but d'être exhaustif en ce qui concerne les techniques de calcul haute performance. J'y introduis simplement les notions nécessaires à la compréhension de la suite du mémoire, notamment celles que j'ai pu utiliser lors du développement des outils de simulation qui seront présentés dans les chapitres suivants. Le lecteur intéressé pourra s'appuyer sur des ouvrages tel que celui de Sanders & Kandrot (2010).

Sommaire

3.1 Les ELT : un défi numérique

3.1.1 Pour la simulation numérique

En OA, comme dans d'autres domaines de recherche et industriels, les outils de simulation numérique tiennent une place importante dans le dimensionnement et la conception d'un instrument. Lors d'une étude de compromis par exemple, l'espace des paramètres à explorer peut être très large, et le nombre de simulations à effectuer avant d'être capable de faire un choix avisé vers une solution technique plutôt qu'une autre en dépend directement. De fait, la conception de l'outil de simulation lui-même est souvent un compromis entre précision du résultat et temps de calcul.

Pour ce qui est de l'OA, la simulation numérique a souvent pour but d'estimer les performances d'un système donné. Comme nous l'avons vu dans le chapitre précédent, la qualité de correction d'une OA est donnée par sa FEP : les outils de simulation visent donc naturellement à estimer cette dernière.

J'opposerai ici deux approches pour le développement de tels outils, même si celles-ci ne sont pas orthogonales et peuvent être utilisées en complément l'une de l'autre. L'opposition vient plutôt du compromis que j'ai cité plus haut entre vitesse d'exécution et précision du résultat. Je distinguerai ainsi les approches analytiques, ou pseudo-analytiques, et les approches bout-en-bout.

Modèles analytiques

L'idée ici est de parvenir à dériver une expression analytique permettant de calculer la FEP. Plus particulièrement, la plupart des méthodes existantes aujourd'hui donnent une estimation de la fonction de transfert optique, à partir de laquelle nous pouvons facilement remonter à la FEP (cf. Éq. 1.5). Et cette estimation provient généralement d'une estimation de la fonction de structure de la phase résiduelle du système d'OA. Je reviendrai plus en détails sur la relation entre FTO et fonction de structure dans le chapitre 5. Le lecteur impatient peut se référer à l'Éq. 5.1.

Les méthodes pour estimer la fonction de structure de la phase résiduelle varient. Certains modèles utilisent la relation qui lie cette dernière au spectre $W_{\phi_\varepsilon}(f)$ de la phase :

$$D_{\phi_\varepsilon}(\rho) = 2\int_0^{2\pi}\int_0^{\infty} W_{\phi_\varepsilon}(f)(1-\cos(2\pi f\rho))k\,dk\,d\theta \tag{3.1}$$

Ces modèles visent alors à donner une expression du spectre W_{ϕ_ε} à partir des différents termes du budget d'erreur du système d'OA (Ellerbroek, 2005; Neichel et al., 2008; Jolissaint, 2010).

Gendron et al. (2014) propose une approche différente où la fonction de structure de la phase résiduelle est directement estimée à partir d'une modélisation de la matrice de covariance des termes du budget d'erreur.

Un point commun entre ces différents modèles est qu'ils se basent assez souvent sur des hypothèses simplificatrices afin de pouvoir parvenir à une expression analytique. Ces hypothèses incluent généralement l'indépendance statistique entre certains termes

du budget d'erreur, la linéarité du système ou encore la stationnarité de la phase dans la pupille. Si ces hypothèses sont en général raisonnables, elles permettent un calcul direct et donc une estimation rapide de la FEP, y compris à l'échelle de l'ELT. En contrepartie, les simplifications qu'elles impliquent font que ces modèles négligent des effets de second ordre qui peuvent malgré tout avoir un impact sur la FEP réelle. En général, les performances estimées par ces outils sont donc surévaluées.

Modèles bout-en-bout

Pour les outils de simulation dits "bout-en-bout", l'approche est différente. Puisqu'il n'est pas possible de dériver mathématiquement une expression analytique de la FEP, l'idée est plutôt de décomposer le système d'OA en sous-systèmes et de proposer des modèles physiques du comportement de chacun de ces sous-systèmes, ainsi que des entrées du système. Un tel outil est donc en général composé d'un modèle de turbulence atmosphérique, d'ASO et de miroir déformable.

Les méthodes de type Monte-Carlo permettent alors d'utiliser des tirages aléatoires afin d'estimer numériquement des grandeurs complexes dont le résultat exact ne peut être calculé directement (Metropolis & Ulam, 1949). De fait, les hypothèses nécessaires à l'établissement du modèle physique sont généralement moins fortes que dans le cas où l'on cherche à obtenir une expression analytique.

Cependant, les méthodes Monte-Carlo, puisque basées sur des tirages aléatoires, nécessitent d'être itérées suffisamment de fois pour que le résultat numérique puissent statistiquement converger vers la solution. Pour l'OA en particulier, le principe de ces méthodes est de générer aléatoirement des écrans turbulents, et de calculer la FEP obtenue après correction par l'OA pour chacune de ces réalisations. C'est pourquoi ces dernières sont en général assez lourdes à mettre en place et requièrent des temps de calcul bien plus long que pour des modèles purement analytiques.

Si de nombreux outils de simulations ont été développés pour des applications diverses, quelques uns d'entre eux sont libres de droit et publiquement disponibles en ligne :

- **OOMAO** (Object-Oriented Matlab Adaptive Optics toolbox) est développé sous licence Matlab (Conan & Correia, 2014)
- **YAO** est un plugin du langage de programmation Yorick (Rigaut & Van Dam, 2013)
- **MAOS** (Multi-threaded Adaptive Optics Simulator) se base quant à lui sur une implémentation en C++ (Wang & Ellerbroek, 2011)
- **DASP** (Durham Adaptive optics Simulations Platform) est développé en Python et en C (Basden et al., 2018)
- **CAOS** (Code for Adaptive Optics System) dont l'implémentation est basée sur le langage IDL (sous licence) (Carbillet et al., 2016)

Je citerai également **OCTOPUS**, le simulateur développé par l'ESO [1], qui, même si le code source n'est pas accessible publiquement, est disponible pour la communauté

1. https ://www.eso.org/sci/facilities/develop/ao/tecno/octopus.html

sur demande.

Tous ces simulateurs permettent de modéliser tout type de système d'OA, allant de la simple SCAO à la MCAO. La limite de ce qu'il est possible de faire avec un simulateur est la plupart du temps donnée par les capacités du matériel informatique sur lequel il est installé. Ainsi, si les outils cités ici disposent tous d'une implémentation parallélisée sur plusieurs cœurs de calcul, l'échelle des ELT implique la manipulation d'un grand nombre de données, sollicitant la bande passante mémoire de ces processeurs. Si l'espace disponible sur les mémoires peut aujourd'hui être augmenté facilement pour répondre au besoin, la bande passante reste une limitation technologique : la simulation d'un système ELT peut prendre de quelques heures à plus d'une journée selon sa complexité. Alors que les instruments de première lumière sont encore tous en phase de conception, le besoin en terme de simulation numérique performante est important pour pouvoir explorer l'intégralité de l'espace des paramètres.

Comme pour les problématiques temps-réel que j'évoquerai ci-dessous, l'utilisation de cartes graphiques (GPU [2]) permet alors de grandement accélérer les calculs, sous réserve d'un effort de programmation adaptée à l'architecture et aux spécificités d'un GPU.

3.1.2 Pour le contrôle temps-réel

La charge de calcul du contrôleur temps-réel dépend du nombre total d'actionneurs et de mesures. De ce nombre dépend directement le dimensionnement du contrôleur temps-réel qui doit être capable de traiter les images venant de l'ASO et d'envoyer les commandes au miroir déformable à une fréquence importante (de l'ordre du kHz) afin de limiter l'erreur temporelle. Or le nombre d'actionneurs est proportionnel au carré du diamètre du télescope : dans le cas de l'ELT, ce dernier est par exemple 5 fois plus grand que celui d'un seul des VLT, si bien qu'on peut s'attendre à un facteur 25 sur la charge de travail du contrôleur.

Pour se rendre compte de ce que cela représente, je vous propose une application numérique sur l'instrument MAORY visant à se faire une idée des spécifications nécessaires du contrôleur temps-réel.

Dans le chapitre précédent, je donnais quelques unes des caractéristiques de l'instrument MAORY (Section 2.9.3) qui peuvent être utilisées pour obtenir rapidement un premier dimensionnement du contrôleur.

Ainsi, côté traitement des images venant de l'ASO, le contrôleur devra être capable de traiter 6 images de 1760 x 1680 pixels pour les ASO laser, et 3 images 500 x 500 pixels pour les ASO naturels. En considérant des opérations réalisées à une fréquence d'un kiloHertz, la bande passante minimale nécessaire est donc d'environ 300 Giga bits par seconde (Gb/s).

Côté calcul des commandes pour le miroir déformable, en considérant une loi de commande de type intégrateur, la principale opération de ce calcul est une multiplica-

2. *Graphics Processing Unit* en anglais

tion matrice-vecteur [3] de taille $N_{mes} \times N_{actus}$ (cf. Éq. 2.7). À cela se rajoute 2 MVM supplémentaires pour la reconstruction des mesures boucles ouvertes nécessaires à la MCAO. Pour MAORY, effectuer cette opération à un kiloHertz sur environ 60000 mesures et 6000 actionneurs mène à une puissance de calcul nécessaire de 1 TMAC/s (la définition du MAC est donnée en bas de page [4]). À titre de comparaison, le module de LTAO de l'instrument GALACSI installé sur le VLT, composé de 4 analyseurs lasers et un miroir déformable à environ 1000 actionneurs, nécessite une puissance de calcul de 12 GMAC/s (Fedrigo et al., 2010).

La première idée permettant de franchir le pas nous séparant d'un dimensionnement ELT serait de simplement multiplier les moyens de calculs existants. De fait, en augmentant le nombre d'unité de calcul, la puissance de calcul théorique augmente également. Cependant, la bande passante et la puissance de calcul, bien que primordiales, ne sont pas les seules contraintes qui s'appliquent au contrôleur temps-réel. Ainsi, plusieurs arguments peuvent s'opposer à cette logique :

- La puissance de calcul théorique d'un serveur n'est que très rarement atteinte, car notamment limitée par la bande passante disponible au sein du serveur (ou entre plusieurs serveurs) pour permettre la communication entre les différents cœur de calcul. En pratique, la puissance effective est plutôt comprise entre 50 et 80 % avec des algorithmes optimisés.
- Certaines tâches nécessaires à une application ne peuvent être parallélisées, et ces dernières ne bénéficient donc pas de l'augmentation des ressources matérielles et fixent une limite théorique à l'accélération accessible (Amdahl, 1967)
- Multiplier les unités de calcul a également pour effet de multiplier le coût financier ainsi que l'énergie électrique nécessaire pour alimenter et refroidir le serveur de calcul ainsi obtenu. Cette dernière contrainte est d'autant plus importante que nous parlons d'une application sur un télescope qui sera situé en haut d'une montagne isolée : acheminer l'énergie suffisante peut s'avérer difficile, et dissiper cette énergie dans l'atmosphère n'est pas recommandé.

3.1.3 Le projet GREENFLASH

Le projet européen GreenFlash a pour but d'étudier les possibilités technologiques permettant de réaliser un prototype de calculateur temps-réel pour l'ELT (Gratadour et al., 2016). La Figure 3.1 illustre ainsi le schéma du prototype visé par le projet GreenFlash. À gauche, on trouve les miroirs déformables et les ASO du système d'OA qui fixent la fréquence de fonctionnement du système. Au centre se trouve le calculateur chargé de calculer les commandes à envoyer aux miroirs déformables à partir des images fournies par les ASO. La problématique ici est de minimiser la latence du calcul afin de limiter au maximum l'erreur temporelle (Bernard et al., 2017). Je définis la latence (issue de la terminologie anglaise) comme le temps nécessaire au calculateur temps-réel pour réaliser les opérations qui lui sont assignées, à savoir le calcul des

3. *Matrix Vector Multiply* (MVM) en anglais
4. *Multiply-ACcumulate* (MAC) en anglais, soit une multiplication et une addition élémentaire

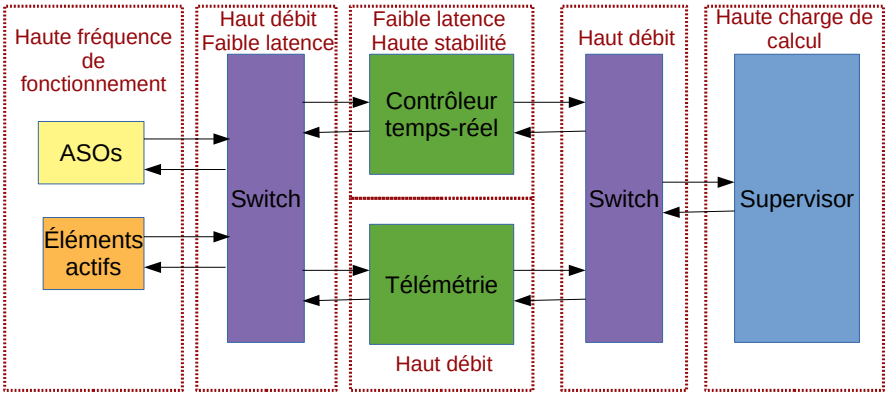

FIGURE 3.1 – Schéma blocs du prototype de calculateur temps-réel visé par le projet GreenFlash *(Crédit : D. Gratadour)*

mesures et des commandes. La latence est à différencier du retard de la boucle d'OA, qui inclut quant à lui la latence du calculateur, le temps d'intégration et de lecture de l'ASO, ainsi que le temps de réponse du miroir déformable.

À droite, le module de supervision est en charge du calcul de la matrice de commande du système. Ce sous-système doit être capable d'effectuer une charge de travail conséquente, notamment pour le calcul de reconstructeur tomographique (Doucet et al., 2017).

Une façon efficace et originale de traiter le problème est d'utiliser une architecture hétérogène CPU [5]/GPU (Gratadour et al., 2016). En effet, comme nous le verrons dans la prochaine section, les cartes graphiques présentent des caractéristiques intéressantes pour traiter une importante charge de calcul efficacement, que ce soit en termes de vitesse d'exécution ou d'énergie consommée.

Enfin, les sous-systèmes qui apparaissent en violet sur la Figure 3.1 sont chargés de la transmission des données entre les sous-systèmes cités précédemment. Ici, la problématique est celle de la haute bande passante nécessaire. Les puces électroniques programmables [6] sont alors une solution technologique adaptée. Le développement d'une méthode de communication directe entre une carte FPGA et la mémoire d'un GPU permet également de minimiser la latence et de pouvoir utiliser les GPUs pour réaliser du calcul temps-réel (Perret et al., 2016; Bernard et al., 2017).

Mis bout à bout, ces différents sous-systèmes permettent ainsi de répondre au besoin des systèmes d'OA pour l'ELT en termes de calcul temps-réel.

5. *Central Processing Unit* en anglais
6. Field-Programmable Gate Array (FPGA) en anglais

3.2 GPGPU : General-Purpose Computing on Graphics Processing Units

Le GPGPU (comprenez le calcul scientifique sur processeurs graphiques) est en essor depuis le début des années 2000. L'objectif est d'utiliser les processeurs graphiques pour réaliser des calculs classiquement effectués par les processeurs conventionnels. L'intérêt vient de l'architecture même d'un processeur graphique : ce dernier ayant été initialement conçu pour le rendu vidéo, il possède beaucoup plus de cœurs de calcul et une bien meilleure bande passante mémoire qu'un processeur conventionnel ce qui le rend très performant pour effectuer des calculs parallèles.

3.2.1 Architecture d'un processeur graphique

Un processeur peut être décomposé en trois constituants principaux :

- **L'unité arithmétique et logique (UAL)** [7] qui est en charge d'effectuer les calculs
- **La mémoire cache** qui permet d'enregistrer des données localement pour un accès plus rapide à ces dernières lors des calculs
- **L'unité de contrôle** qui permet de commander le processeur

À cela s'ajoute la mémoire vive [8] qui permet un stockage plus important que sur le cache, mais avec un accès moins rapide. Lors d'opérations, l'unité de contrôle commande la communication entre le cache et la mémoire vive afin de fournir aux UAL les données nécessaires aux calculs.

La Figure 3.2 schématise les architectures matérielles d'un processeur conventionnel (CPU) et d'un processeur graphique (GPU). Un CPU ayant pour vocation d'être multi-taches, ses unités de calculs sont indépendantes et partagent la même mémoire cache. L'unité de contrôle joue alors le rôle de chef d'orchestre afin que les différentes taches en cours d'exécution soient traitées par les unités de calcul.

L'architecture d'un GPU est bien différente. Ici, les unités de calculs sont plus nombreuses et regroupées au sein de multiprocesseur de flux (SM) [9] géré par une unité de contrôle dédiée. La mémoire cache se trouve également au niveau de ces SM et ne peut être accessible qu'aux unités de calcul du SM en question. Les données à traiter doivent donc être distribuées entre les différents SM pour effectuer une opération. En contrepartie, la taille de la mémoire cache est moindre que sur un CPU, et ce pour physiquement permettre la présence du grand nombre d'UAL.

Ainsi, le GPU est idéal pour des applications massivement parallèles où la latence peut être élevée : la petite taille du cache oblige à des accès à la mémoire globale plus fréquents et donc à une latence plus élevée que sur un CPU. Le GPU est donc qualifié de processeur à haute latence et haut débit, alors que le CPU est à faible latence et faible débit. Cette spécificité implique que la charge de travail donnée à

7. Arithmetic-Logic Unit (ALU) en anglais
8. Random Access Memory (RAM) en anglais
9. *Streaming Multiprocessor* (SM) en anglais

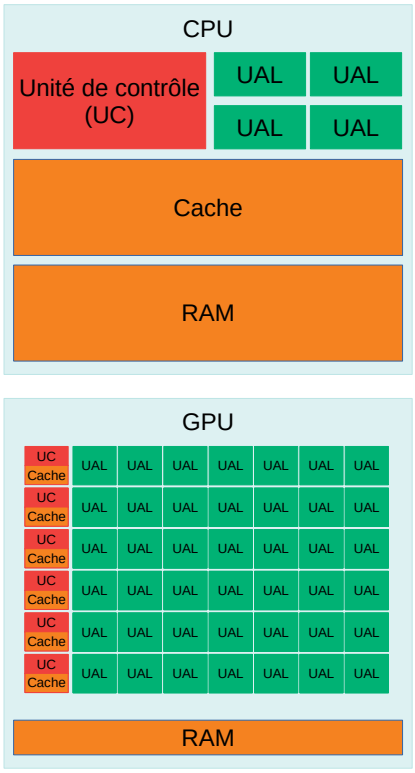

FIGURE 3.2 – Architectures de CPU et GPU. *Crédit : Nvidia*

un GPU doit être conséquente pour que ce dernier soit efficace par rapport au CPU. La Figure 3.3 illustre cela en donnant les temps d'exécution en secondes d'un produit matrice-matrice en fonction du nombre de lignes ou de colonnes de matrices carrées. On constate effectivement que le GPU ne devient plus efficace que le CPU qu'à partir d'une certaine taille de matrice. Ces temps ont été obtenues en simple précision sur un

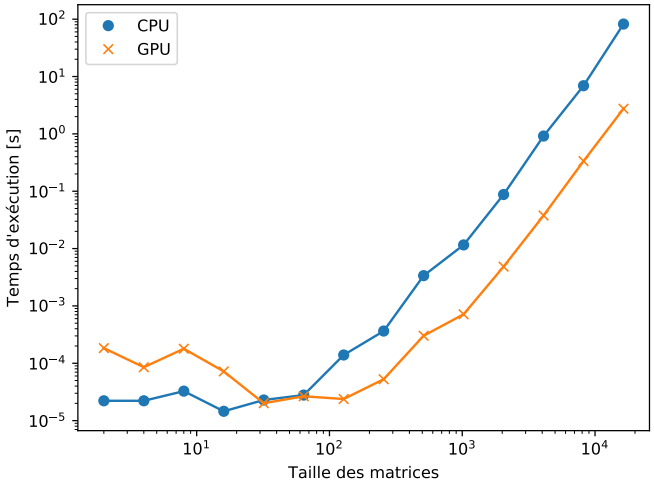

FIGURE 3.3 – Temps d'exécution d'un produit matrice-matrice de taille $n \times n$ sur un CPU et sur un GPU

ordinateur portable équipé d'un processeur multi-cœur Intel Core i7-7700HQ cadencé à 2,8 GHz, et d'un GPU Nvidia GTX 1060 composé de 1280 cœurs à 1,67 GHz. Côté CPU, le calcul a été réalisé sur les 8 cœurs de ce dernier avec la libraire Numpy [10] de Python. Pour le GPU, la librairie cuBLAS [11] de Nvidia fournit des algorithmes optimisés pour les opérations d'algèbre linéaire telles que le produit matrice-matrice.

3.2.2 Programmation sur GPU

Initialement, le GPGPU était assez difficile à mettre en œuvre de par la spécificité architecturale du GPU. Un GPU étant conçu pour le traitement d'image et le rendu vidéo, sa programmation nécessitait la manipulation de concepts spécifiques tels que les texels (plus petit élément d'une texture) par exemple (Du et al., 2012).

L'introduction de CUDA [12] (Compute Unified Device Architecture) par Nvidia au milieu des années 2000 a permis de rendre le GPGPU beaucoup plus accessible. En effet, cet outil propose une interface de programmation applicative [13] qui permet de développer des algorithmes pour le GPU dans un langage C, C++ ou Fortran

10. http ://www.numpy.org
11. https ://docs.nvidia.com/cuda/cublas
12. https ://docs.nvidia.com/cuda/cuda-c-programming-guide
13. *Application programming Interface* (API) en anglais

enrichi de quelques mots clés. Comme illustré sur la Figure 3.4, cette API se compose de librairies optimisées pour GPU, d'un environnement d'exécution [14] qui permet de faire le lien entre l'application développée et le dernier élément de cet API, à savoir le pilote [15]. Ce dernier élément a pour rôle de transmettre les instructions au GPU.

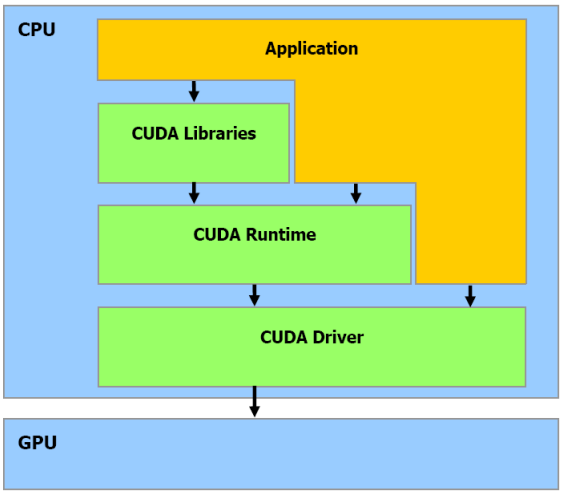

FIGURE 3.4 – Schéma de l'API CUDA *Crédit : Nvidia*

Avec CUDA, une application parallèle est une succession de noyaux de calcul [16], parties de code à exécuter sur le GPU. Ces derniers doivent être écrits en gardant à l'esprit l'architecture du GPU : un noyau de calcul doit être lancé sur une grille de blocs [17], eux-mêmes composés de tâches [18]. En simplifiant énormément, pour la compréhension du concept, une tâche peut être assimilée à l'un des cœurs de calcul du GPU, le bloc auquel elle appartient est alors un des SM du GPU, et la grille est l'ensemble des SM de la carte graphique. Ce fonctionnement est illustré sur la Figure 3.5.

Il appartient au développeur de spécifier les dimensions de la grille et des blocs sur lesquels il veut que son noyau de calcul soit exécuté. Un point important à garder en mémoire pour cette répartition est que la mémoire cache ne peut être partagée qu'au sein d'un même bloc, et est donc indépendante entre les blocs. Dès lors, pour obtenir la meilleure performance, il est important de répartir correctement le travail sur les cœurs de calcul du GPU en fonction de l'architecture du GPU utilisé et de l'algorithme considéré. La conception même de l'algorithme doit prendre en compte ces spécificités pour être efficace.

14. *runtime* en anglais
15. *driver* en anglais
16. *kernels* en anglais
17. *blocks* en anglais
18. *threads* en anglais

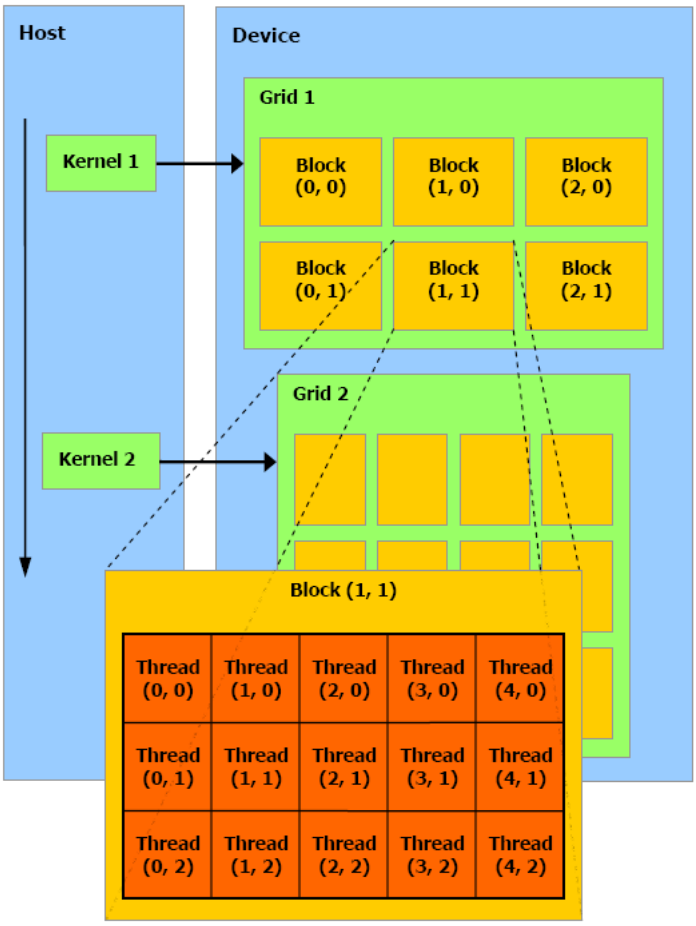

FIGURE 3.5 – Schéma d'exécution d'une application CUDA *Crédit : Nvidia*

3.2.3 Principes généraux de l'optimisation sur GPU

Il est important de noter que le GPU ne remplace pas le CPU, mais agit comme un accélérateur matériel. Les instructions sont données par le CPU qui communiquent avec la carte graphique via le bus PCI [19] Express (PCI-E). L'utilisateur d'une application n'ayant accès qu'à la mémoire vive du CPU, les données à traiter doivent être transférées jusqu'à la mémoire globale du GPU, ce dernier effectuent les opérations et écrit le résultat en mémoire. Pour pouvoir être exploité par l'utilisateur, les résultats doivent finalement être recopiés de la mémoire globale du GPU vers la mémoire vive du CPU. Ces transferts de mémoire sont coûteux en terme de performance étant donnée la faible bande passante du bus PCI-E et doivent donc être minimisés.

Il est possible de distinguer deux limites à la performance : la bande passante mémoire et le cadençage des cœurs de calcul. Pour une application limitée par la bande passante mémoire, le problème est que le temps de calcul est court devant les temps d'accès mémoire : les cœurs de calcul ne sont pas exploités au mieux puisqu'ils doivent attendre les données. À l'inverse, le cadençage des cœurs peut être une limite si la charge de travail est telle que les données doivent attendre avant de pouvoir être traitées.

CUDA fournit également un outil de profilage de code offrant la possibilité au développeur de visualiser l'exécution de son application sur le GPU, comme illustré sur la Figure 3.6. Le profil temporel de l'application renvoyé par cet outil affiche la séquence d'opérations réalisée sur le GPU (ligne *Compute* de la Figure 3.6). Il est également possible d'obtenir des statistiques détaillées de chacune de ces opérations en cliquant simplement sur l'une d'entre elles. Sur le bas de la Figure 3.6, un résumé des opérations effectuées sur le GPU est affiché sous la forme d'un tableau. Pour chaque opération, l'outil fournit le nombre d'appels à cette opération, sa durée moyenne, la mémoire cache utilisée, etc. Enfin, les lignes *MemCpy (HtoD)*, *MemCpy (DtoH)* et *MemCpy (DtoD)* montrent respectivement les temps de communication du CPU vers le GPU, du GPU vers le CPU et entre les différents GPU.

En fonction des résultats, le développeur peut déterminer si son application est limitée par la bande passante mémoire ou par le calcul, et orienté ses optimisations en conséquence.

3.3 Mise en pratique : le lancer de rayon

Après ces aspects théoriques, je vous propose ici une mise en situation permettant d'illustrer les principes de la programmation sur GPU avec CUDA avec un algorithme courant, à savoir le lancer de rayon.

19. *Peripheral Component Interconnect* en anglais

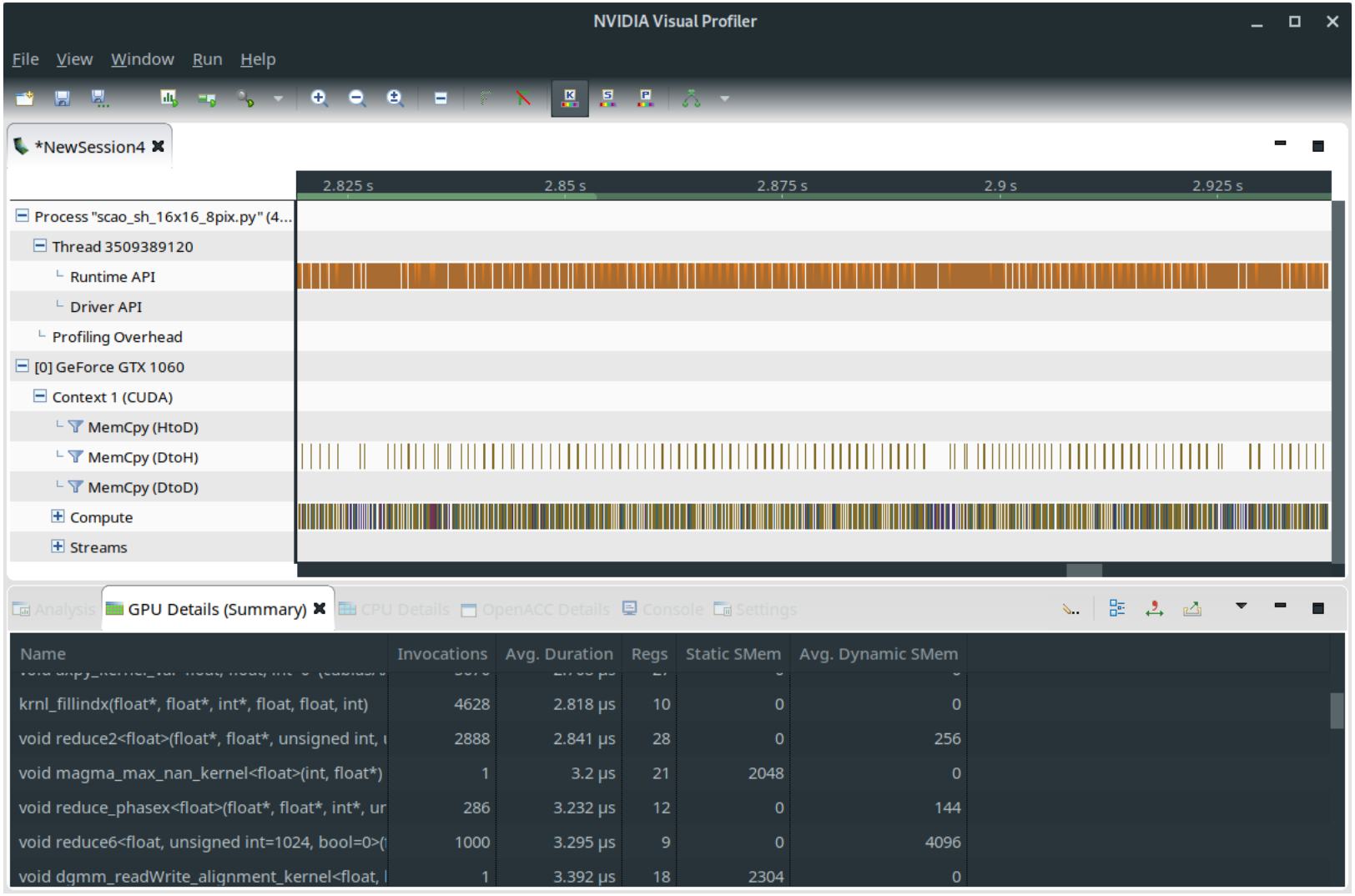

FIGURE 3.6 – Exemple de profil fournit pour l'outil de profilage CUDA

3.3.1 Principe de l'algorithme

Dans le cadre d'un simulateur d'OA bout-en-bout, ce type d'algorithme sera utilisé pour déterminer la phase vue par un analyseur par exemple à partir d'un écran de phase turbulent incident. Son principe est illustré par la Figure 3.7

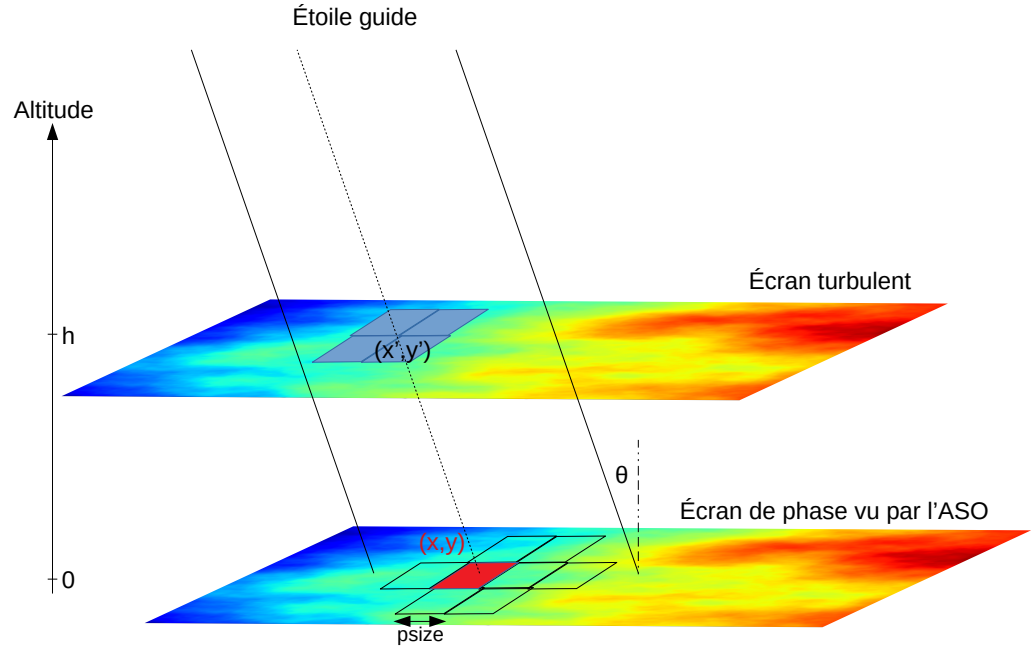

FIGURE 3.7 – Schéma du principe de lancer de rayon pour la détermination des écrans de phases de l'ASO

À chaque pixel (x, y) de l'écran de phase vue par l'ASO correspond une position (x', y') sur l'écran de phase turbulent. Cette position peut être déterminée géométriquement connaissant l'altitude h de la couche turbulente, la position angulaire (θ_x, θ_y) de l'étoile guide de l'ASO dans le champ et la taille angulaire $psize$ d'un pixel :

$$\begin{aligned} x' &= x + \frac{\theta_x h}{psize} \\ y' &= y + \frac{\theta_y h}{psize} \end{aligned} \tag{3.2}$$

Une fois cette position déterminée, la valeur du pixel en (x, y) de l'écran de phase vue par l'ASO est calculée à partir de l'interpolation linéaire des quatre pixels les plus proches de la position (x', y').

Le code séquentiel en langage C de cet algorithme s'écrit alors intuitivement :

```
void raytrace(float *odata, float *idata,
              int nx, int Nx,
              float xoff, float yoff) {
```

```
    float x, y; // Position sur idata
    float xshift, yshift;
    float  wx1, wx2, wy1, wy2; // Poids pour l'interpolation
    float iref, jref; // Pixels sur idata

    // Boucle sur tous les pixels de idata
    for(int i = 0 ; i < nx ; i++){
        x = i + xoff; // Determination de la position
        iref = (int) x; // En pixels
        xshift = x - iref;
        wx1 = 1.0f - xshift; // Determination des poids
        wx2 = xshift;
        for(int j = 0 ; j < nx ; j++){
            y = j + yoff;
            jref = (int) y;
            yshift = y - jref;
            wy1 = 1.0f - yshift;
            wy2 = yshift;

            // Interpolation lineaire
            odata[i + j * nx] = (wx1 * wy1 * idata[iref + jref * Nx]
                                 + wx2 * wy1 * idata[iref + 1 + jref * Nx]
                                 + wx1 * wy2 * idata[iref + (jref + 1) * Nx]
                                 + wx2 * wy2 * idata[iref + 1 + (jref + 1) * Nx]);
        }
    }
}
```

où *odata* est l'écran de phase vue par l'ASO de taille *(nx, nx)*, et *idata* est l'écran de phase turbulent de taille *(Nx, Nx)*. Pour simplifier, je suppose que l'écran de phase turbulent est plus grand que l'écran de phase vue par l'ASO afin de ne pas avoir à traiter séparément les pixels de bords de tableaux.

3.3.2 Programmation CUDA

Il apparaît évident que cet algorithme est hautement parallélisable : chaque pixel de *odata* se calcule indépendamment les uns des autres. Ce type de comportement est idéal pour une programmation sur GPU. Le passage en CUDA du code précédent se fait en 2 étapes : l'écriture du noyau de calcul qui réalise le calcul sur le GPU, puis l'écriture du lanceur qui répartit la charge de travail sur le GPU et lance le noyau de calcul. Je suppose également que les tableaux *odata* et *idata* sont maintenant alloués sur le GPU.

Le noyau de calcul

Un noyau de calcul décrit le code qui sera exécuté par chaque tâche lancée lors de l'exécution. L'implémentation naïve de l'algorithme précédent ressemble alors énormément au code séquentiel, la double boucle en moins :

```
__global__ void raytrace_krnl(float *odata, float *idata,
                int nx, int Nx,
                float xoff, float yoff) {

    float x, y; // Position sur idata
    float xshift, yshift;
    float  wx1, wx2, wy1, wy2; // Poids pour l'interpolation

    int i = threadIdx.x + blockIdx.x * blockDim.x;
    int j = threadIdx.y + blockIdx.y * blockDim.y;

    x = i + xoff; // Determination de la position
    iref = (int) x; // En pixels
    xshift = x - iref;
    wx1 = 1.0f - xshift; // Determination des poids
    wx2 = xshift;
```

```
    y = j + yoff;
    jref = (int) y;
    yshift = y - jref;
    wy1 = 1.0f - yshift;
    wy2 = yshift;

    // Interpolation lineaire
    odata[i + j * nx] = (wx1 * wy1 * idata[iref + jref * Nx]
                       + wx2 * wy1 * idata[iref + 1 + jref * Nx]
                       + wx1 * wy2 * idata[iref + (jref + 1) * Nx]
                       + wx2 * wy2 * idata[iref + 1 + (jref + 1) * Nx]);


}
```

Les différences avec le code séquentiel sont minimes, et c'est là l'intérêt de CUDA :

- le mot clé *___global___* devant la définition de la fonction permet d'identifier cette dernière comme étant un noyau de calcul devant être exécuté sur le GPU
- *threadIdx* désigne l'indice de la tâche au sein de son bloc
- *blockIdx* désigne l'indice du bloc au sein de la grille
- *blockDim* donne la dimension du bloc

Reste maintenant à définir cette fameuse grille de blocs sur laquelle le code sera exécuté.

Le lanceur

Le lanceur [20] a pour fonction de lancer l'exécution du noyau de calcul sur une grille de blocs et de tâches :

```
void raytrace_launcher(float *odata, float *idata,
                int nx, int Nx,
                float xoff, float yoff, int block_size) {

    dim3 blocks(nx / block_size, ny / block_size), threads(block_size,
        block_size);

    raytrace_krnl<<<blocks, threads>>>(d_odata, d_idata, nx, Nx, xoff,
        yoff);

}
```

Décomposons ces lignes de codes :

- La première instruction permet de créer la grille de blocs et de tâches. Plus précisément, *blocks(nx / block_size, ny / block_size)* crée une grille de *(nx / block_size, nx / block_size)* blocs, puis *threads(block_size,block_size)* définit la taille des blocs, i.e. le nombre de tâches par bloc, soit *(block_size,block_size)*. Je ne rentrerai pas ici dans le détail du choix de la valeur de *block_size*. Pour être totalement optimal, ce nombre va dépendre de la carte graphique utilisée et de la taille des tableaux à traiter.
- la seconde instruction lance l'exécution du noyau de calcul. Notons l'utilisation des triples chevrons <<< >>> qui permettent de communiquer au noyau de calcul la grille sur laquelle il doit s'exécuter.

Félicitations, nous venons d'écrire notre premier code GPU avec CUDA ! L'effort à fournir pour passer d'un code séquentiel à un code parallélisé sur GPU est donc

20. *launcher* en anglais

minime, tout du moins pour une implémentation naïve permettant tout de même d'accélérer le code. En effet, sur le même système que celui décrit dans la section 3.2.1, le code séquentiel s'exécute en 383 μs pour un écran de phase de taille $nx = 288$. Le code CUDA que je viens de décrire s'exécute quant à lui en 18 μs, soit un facteur 21 en terme de temps d'exécution. Néanmoins, ce code est encore optimisable.

3.3.3 Optimisons le noyau de calcul

Comme je l'ai expliqué précédemment, l'un des premiers freins à la performance sur GPU sont les accès mémoire. Dans le cas qui nous intéresse, nous pouvons constater facilement que chaque pixel du tableau *odata* est chargé 4 fois depuis la mémoire globale. Une optimisation possible est donc de passer par une première phase de chargement des données en mémoire cache qui, je le rappelle, est bien plus rapide que la mémoire globale.

La difficulté ici est de correctement gérer la mémoire partagée entre les tâches. En effet, celle-ci n'est accessible qu'aux tâches d'un même bloc. L'interpolation nécessitant d'accéder aux pixels voisins, les tâches en bord de bloc n'auront pas accès aux données nécessaires.

Une solution possible est d'augmenter la taille des blocs à *(block_size + 1 ,block_size + 1)* et d'allouer une mémoire partagée de la même taille. Ainsi, tous les tâches participeront au chargement des données, puis seuls les tâches qui ne sont pas en bord de blocs feront le calcul d'interpolation. Une dernière précaution à prendre est de s'assurer que tous les tâches ont terminé de charger les données en mémoire partagée avant de lancer le calcul de l'interpolation. Au final, le kernel optimisé se présente sous cette forme :

```
__global__ void raytrace_krnl(float *odata, float *idata,
                int nx, int Nx,
                float xoff, float yoff) {

    extern __shared__ float cache[];

    float x, y; // Position sur idata
    float xshift, yshift;
    float  wx1, wx2, wy1, wy2; // Poids pour l'interpolation

    int i = threadIdx.x + blockIdx.x * (blockDim.x - 1);
    int j = threadIdx.y + blockIdx.y * (blockDim.y - 1);

    x = i + xoff; // Determination de la position
    y = j + yoff;
    iref = (int) x; // En pixels
    jref = (int) y;

    cache[threadIdx.x + threadIdx.y * blockDim.x] = idata[iref + jref * Nx];

    __syncthreads(); // On attend que tout le monde ait fini
    if ((threadIdx.x != (blockDim.x-1)) && (threadIdx.y != (blockDim.y-1)) {
        xshift = x - iref;
        wx1 = 1.0f - xshift; // Determination des poids
        wx2 = xshift;


        yshift = y - jref;
        wy1 = 1.0f - yshift;
        wy2 = yshift;

        // Interpolation lineaire
        odata[i + j * nx] = (wx1 * wy1 * idata[iref + jref * Nx]
                           + wx2 * wy1 * idata[iref + 1 + jref * Nx]
                           + wx1 * wy2 * idata[iref + (jref + 1) * Nx]
```

```
                            + wx2 * wy2 * idata[iref + 1 + (jref + 1) * Nx]);
    }

}
```

Et la mémoire partagée est allouée dynamiquement par le lanceur :

```
void raytrace_launcher(float *odata, float *idata,
                int nx, int Nx,
                float xoff, float yoff, int block_size) {

    dim3 blocks(nx / block_size, ny / block_size), threads(block_size + 1,
        block_size + 1);

    int smemSize = (block_size + 1) * (block_size + 1) * sizeof(float);
    raytrace_krnl<<<blocks, threads, smemSize>>>(d_odata, d_idata, nx, Nx, xoff,
        yoff);
}
```

L'utilisation de la mémoire partagée permet de parvenir à un temps d'exécution de 15 μs, soit un facteur 25 par rapport au code séquentiel. Si d'autres niveaux d'optimisation sont encore possible, comme vérifier l'alignement en mémoire des données par exemple, je m'arrêterai ici en ce qui concerne cet exemple. L'idée était de présenter au lecteur non aguerri ce à quoi ressemble la programmation GPU qui a représentée une bonne partie de mon travail lors de cette thèse, notamment sur le code de simulation COMPASS que je vais décrire dans le chapitre suivant. Le type de programmation et d'optimisation que je viens de décrire est typique du travail que j'ai pu mener sur COMPASS. Ainsi, afin de permettre une lecture de ce mémoire plus fluide, je ne rentrerai plus dans le détail des lignes de codes comme je viens de le faire, sauf si cela s'avère nécessaire pour la compréhension du lecteur. Néanmoins, ce travail d'optimisation a été omniprésent au cours de la thèse, et ce afin de proposer un code de simulation le plus rapide possible, tout en assurant une précision de calcul optimale.

Chapitre 4

COMPASS : COMputing Platform for Adaptive opticS Systems

Après les trois chapitres introductifs de cette thèse, il est temps de rentrer dans le cœur du sujet qui m'a interessé durant ces dernières années, ou du moins l'un de ses aspects : la simulation haute performance pour l'OA. Le chapitre qui suit est consacré à la plateforme de simulation COMPASS. Après avoir été activement impliqué dans son développement, elle a été mon principal outil de travail pour l'étude des budgets d'erreur en OA.

Je présente ici l'architecture générale de COMPASS, l'implémentation des modèles physiques utilisés ainsi que les performances fournies par cet outil.

Sommaire

4.1 Le projet COMPASS

Le projet a vu le jour en 2013, suite à un financement de l'Agence Nationale de la Recherche. L'objectif initial du projet était de fournir à la communauté une plateforme de simulation pour le développement des optiques adaptatives de l'ELT utilisant des cartes graphiques comme accélérateurs matériels (Gratadour et al., 2014). D'autre part, le développement de la plateforme devait inclure un cœur temps réel pouvant être réutilisé sur un système d'OA réel.

Mené par le LESIA (Laboratoire d'Etudes Spatiales et d'Instrumentation en Astrophysique), le développement s'est fait en collaboration avec plusieurs laboratoires français :

- le laboratoire Galaxies - Étoiles - Physique - Instrumentation (GEPI) de l'Observatoire de Paris
- l'Institut de Planétologie et d'Astrophysique de Grenoble (IPAG)
- le Laboratoire d'Astrophysique de Marseille (LAM)
- la Maison de la Simulation
- l'Office National d'Études et de Recherches Aéronautiques

Cette collaboration a permis d'aboutir à une plateforme de simulation ouverte et activement maintenue. Elle comprend une architecture logicielle basée sur CUDA contenant un cœur de calcul en temps réel à l'échelle des ELT fournissant diverses lois de contrôle, ainsi qu'une surcouche logicielle en Python pour faciliter son utilisation comme simulateur de systèmes d'OA. COMPASS est alors capable de simuler tout type d'OA à l'échelle des ELT avec une vitesse d'exécution permettant d'effectuer les campagnes de simulations nécessaires au développement des OA de l'ELT (Vidal et al., 2014a; Clénet et al., 2016; Vidal et al., 2017; Bertram et al., 2018).

4.2 Mes contributions

COMPASS étant un projet d'envergure impliquant ou ayant impliqué de nombreux contributeurs, il peut être difficile de mettre en évidence le travail que j'ai accompli sur ce projet. Si les prochaines sections viseront à décrire la plateforme et ses développements, je souhaite utiliser cette section pour lister mes contributions propres au développement de COMPASS, et je reviendrai sur certaines d'entre elles dans le reste de ce chapitre :

- Implémentation du reconstructeur minimum variance (cf. Section 4.4.9)

- Implémentation de la projection de phase sur le miroir déformable (cf. Section 4.4.9)
- Implémentation de l'optimisation modale (cf. Section 4.4.9)
- Implémentation d'une loi de commande générique (cf. Section 4.4.9)
- Implémentation du calcul du centre de gravité sur les pixels les plus brillants (cf. Section 4.4.8) (Basden et al., 2012)
- Implémentation de l'effet de cône pour les étoiles laser (cf. Section 4.4.5)
- Implémentation de la classe Télescope incluant la pupille ELT et les aberrations de phase sur les segments du primaire de l'ELT (cf. Section 4.4.2)
- Implémentation des fonctions d'influence du miroir déformable "piezo-stack" (cf. Section 4.4.10)
- Implémentation de la base de données HDF5 pour la réutilisation de données lors de l'initialisation de la simulation (cf. Section 4.6.1)
- Implémentation d'un outil d'estimation du budget d'erreur (cf. Chapitre 5)
- Implémentation et développement de l'interface graphique
- Modules "miroir déformable" et "RTC" rendus autonomes, i.e. utilisables en-dehors de la plateforme, notamment sur banc (Deo et al., 2017) (cf. Section 4.6.2)
- Optimisation générale des noyaux de calcul CUDA pour le support de l'échelle ELT
- Optimisation de l'empreinte mémoire du modèle d'ASO Shack-Hartmann (cf. Section 4.4.6)
- Développement préliminaire du support multi-GPU pour le modèle d'ASO Shack-Hartmann
- Participation à l'implémentation du modèle d'ASO pyramide (cf. Section 4.4.7)
- Participation au passage de la couche logicielle utilisateur du langage Yorick vers Python (cf. Section 4.3)
- Restructuration complète de la couche utilisateur Python et mise en place du paquet shesha (cf. Section 4.3.3)
- Support pour l'équipe MICADO pour l'utilisation de la plateforme et implémentation de fonctionnalités dédiées (Vidal et al., 2017)
- Développement et mise en ligne de la page internet via GitHub Pages, incluant manuel utilisateur et instructions pour l'installation de la plateforme (https ://anr-compass.github.io/compass/)

4.3 Architecture logicielle

4.3.1 Aperçu général

Un aperçu général de l'architecture logicielle de COMPASS est proposé dans la Figure 4.1. Celle-ci peut se décomposer en trois couches :

- une couche C++/CUDA à bas niveau qui implémente les calculs en eux-mêmes
- une couche Python à haut niveau permettant l'interaction avec la couche bas

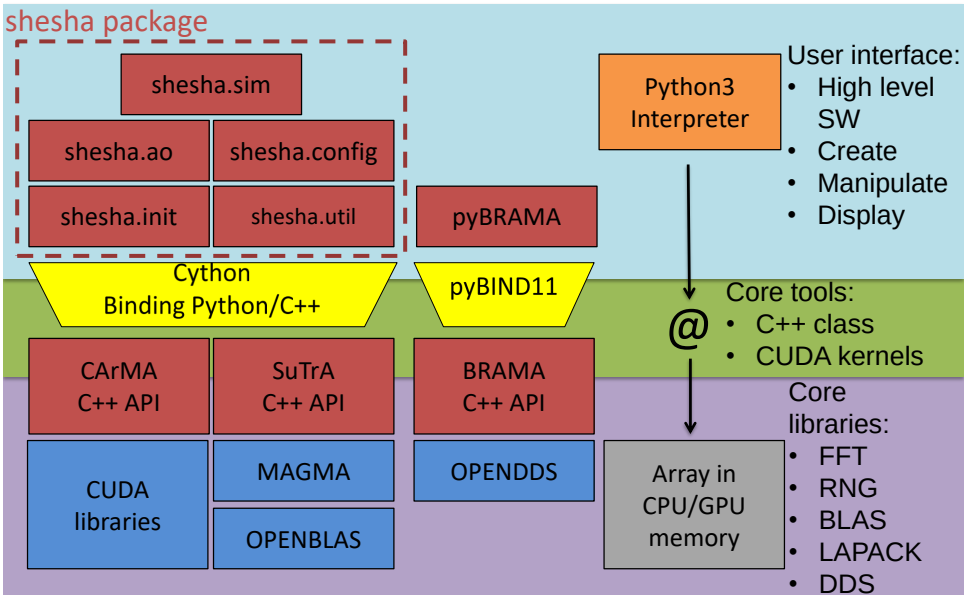

FIGURE 4.1 – Schéma de l'architecture logicielle de COMPASS

niveau

- une couche intermédiaire en Cython servant à faire le lien entre les couches haut niveau et bas niveau

Cette architecture permet d'allier le calcul haute performance apporté par la couche bas niveau à la facilité d'utilisation amenée par la couche utilisateur Python. Par dessus ces couches vient encore se superposer une interface graphique utilisant PyQt-Graph et permettant de visualiser en temps réel l'évolution de la simulation et de la contrôler. D'autres fonctionnalités ont été ajoutées par la suite et seront rapidement abordées dans la Section 4.6.

Je vais tout d'abord m'appliquer à détailler les principales couches de cette architecture citée ci-dessus, hormis la couche intermédiaire qui ne sert qu'à faire le lien entre les deux autres.

4.3.2 Le secret de la performance : CArMA et SuTrA

Comme décrit sur la Figure 4.1, la couche bas niveau se compose de deux librairies C++ : CArMA (C++ Api for Massively parallel Applications) et SuTrA (Simulation Tool for Adaptive optics).

CArMA a été développée dans le but de faciliter l'allocation et la manipulation de données sur la carte graphique. Comme illustré par la Figure 4.2, elle amène différentes classes C++ pouvant être utilisées dans des applications parallèles :

- la classe *carma_device* permet d'obtenir des informations sur la ou les cartes graphiques à disposition et de gérer ces dernières dans le cadre de l'application.

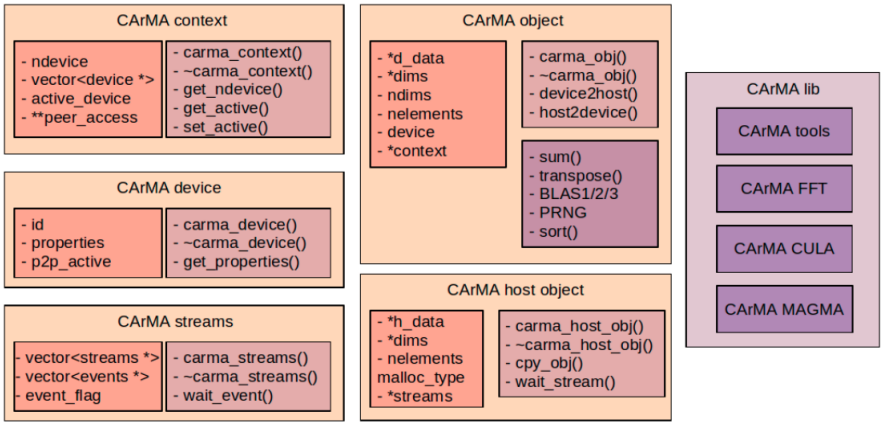

FIGURE 4.2 – Les principales classes de la librairie CArMA

- la classe *carma_obj* est dédiée à l'allocation et la manipulation de tableaux sur la carte graphique. Elle inclut, par exemple, des méthodes permettant d'obtenir les dimensions d'un objet alloué sur le GPU, de recopier le contenu de cet objet sur le CPU ou encore d'effectuer des opérations mathématiques sur l'objet telles qu'une somme, un produit matrice-vecteur ou matrice-matrice.
- la classe *carma_host_obj* est le pendant de la classe *carma_obj* pour les objets alloués sur le CPU.
- la classe *carma_streams* facilite quant à elle l'utilisation des CUDA streams, ces derniers permettant d'effectuer en parallèle des taches séquentielles.

CArMA permet également une utilisation simplifiée des librairies mathématiques apportées par CUDA :

- **cuBLAS** pour les opérations d'algèbre linéaire
- **cuSPARSE** pour l'algèbre linéaire sur matrices creuses
- **cuFFT** pour le calcul des transformées de Fourier
- **cuRAND** pour la génération de nombres aléatoires
- **MAGMA** pour l'inversion et la décomposition de matrices

SuTrA utilise donc les possibilités offertes par CArMA pour l'implémentation de modèles servant à la simulation d'optique adaptative. La librairie se compose de classes C++ modélisant chacun des sous-systèmes nécessaires à la simulation. Le Tableau 4.1 résume le contenu de la librairie ainsi que les fonctionnalités offertes. Les algorithmes utilisés reposent sur l'utilisation de la librairie CArMA ainsi que sur le développement de noyaux de calcul dédiés. Je reviendrai en détails sur l'implémentation des modèles physiques utilisés dans COMPASS dans la Section 4.4.

4.3.3 L'interface utilisateur : *shesha*

L'interface utilisateur de COMPASS, nommée *shesha*, est développée sous la forme d'un paquet Python 3. Ce paquet est lui-même composé de cinq modules :

- le module *config* définit l'ensemble des classes de paramètres nécessaires à la

Tableau 4.1 – Composants et fonctionnalités de la librairie SuTrA

Classe	Fonctionnalités
Telescope	pupille ELT
	Aberration de phase
Atmosphère	Couches indépendantes
	Spectre de Von Karman
WFS	Shack Hartmann
	Pyramide avec implémentation multi-GPU
	Étoile laser avec élongation et effet de cône
	Calcul direct de gradient
Miroir déformable	Piezo-stack avec multiples fonction d'influences
	Miroir tip-tilt
	Modes de Karhunen-Loeve
	Miroir personnalisé via fichier HDF5
Calcul des mesures	Centre De Gravité (CDG)
	CDG pondéré
	CDG seuillé
	CDG sur les pixels les plus brillants Basden et al. (2012)
	Corrélation
Méthode de reconstruction	Moindre carrés
	Maximum a posteriori
	CuReD Rosensteiner (2012)
	Projection directe

simulation. La définition de ces paramètres inclut des fonctions assurant la sûreté du typage, prévenant ainsi d'éventuelles erreurs dues à de mauvaises manipulations de l'utilisateur.

- le module *init* inclut les sous-modules servant à initialiser les différentes classes SuTrA définies au-dessus. À partir d'un fichier de paramètres définit par l'utilisateur, chaque sous-module calcule les éléments nécessaires à cette initialisation, comme par exemple les tailles de tableaux nécessaires.
- le module *ao* permet l'accès à des fonctions d'optimisation de la boucle d'optique adaptative pouvant être utilisées en cours de simulation. On y trouve ainsi les calculs de matrice d'interaction et de commande, la projection sur une base modale, etc.
- le module *util* regroupe des fonctions mathématiques génériques intégrées pour le besoin de la simulation : rotation, génération d'indices, etc.
- le module *sim* définit la classe *Simulator* qui agit comme une couche d'abstraction pour l'écriture de scripts de simulation. Les attributs de cette classe sont les objets définis lors de la phase d'initialisation et peuvent être manipulés aisément au travers de la classe.

Une interface graphique est également disponible en plus de ces modules.

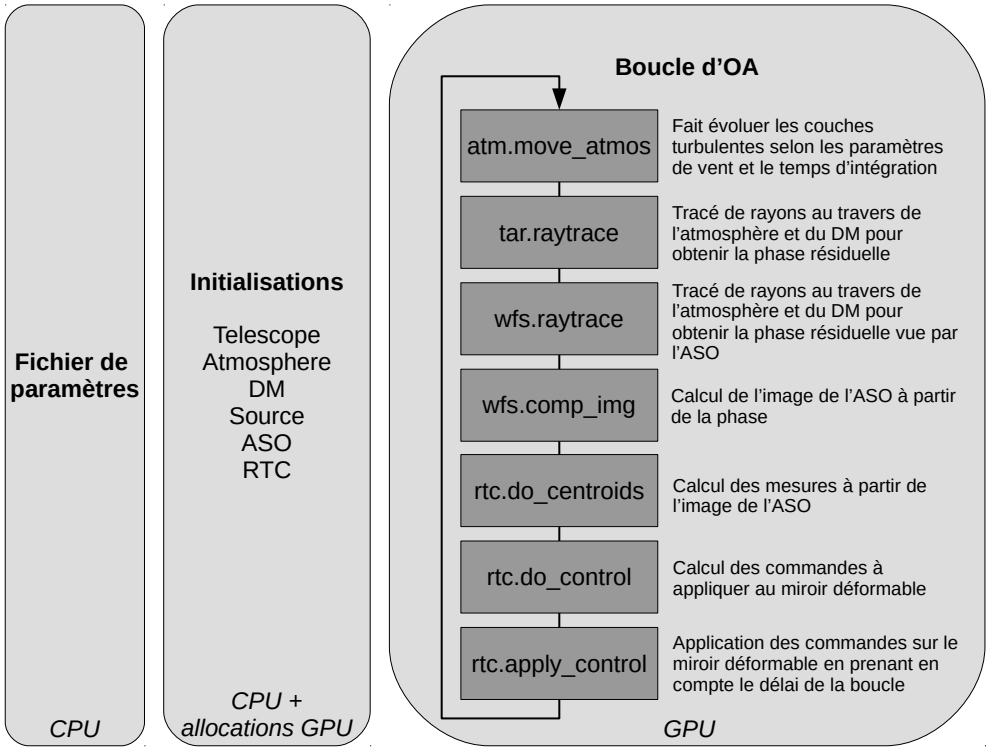

FIGURE 4.3 – Étapes d'une simulation COMPASS

Au final, une simulation COMPASS se déroule comme illustré sur la Figure 4.3 : l'utilisateur fournit un fichier de paramètres sous la forme d'un fichier Python servant à initialiser les différentes composantes de la simulation. Ces initialisations reposent sur un code Python servant à allouer l'espace mémoire nécessaire sur la carte graphique via l'instantiation de classes SuTrA. Enfin, la boucle d'OA peut être lancée et celle-ci ne fait intervenir que la carte graphique, et ce afin d'éviter les accès mémoire entre le CPU et le GPU qui limiteraient la performance.

4.4 Modèles physiques et implémentation

Les bases de COMPASS ont été posées par Gratadour et al. (2012). Je rappelle ici les modèles physiques qui y ont été décris, et je décris également de nouveaux modèles qui ont été intégrés depuis.

4.4.1 L'échantillonnage

L'un des points clés d'une simulation numérique est de correctement choisir son échantillonnage, qu'il soit spatial ou temporel. En effet, je rappelle que l'objectif de la simulation numérique est d'obtenir une FEP. L'échantillonnage choisi pour les modèles physiques qui permettent d'aboutir à cette FEP définit également l'échantillonnage de la FEP finale. Ainsi, si l'on considère que la pupille est simulée avec une taille de pixel p exprimée en mètre, alors le champ de vue de la FEP obtenue par transformée de Fourier à la longueur d'onde λ sera de $\frac{\lambda}{p}$ radians. Dès lors, en rappelant également que la taille du pixel résulte de la simulation d'une pupille, ou d'une sous-pupille, de diamètre d échantillonnée sur N_{phase} points, le champ de vue final de la FEP FoV_{FEP} s'exprime :

$$FoV_{FEP} = N_{phase}\frac{\lambda}{d} \tag{4.1}$$

L'échantillonnage dans COMPASS dépend du type d'ASO utilisé. Pour un analyseur Shack-Hartmann, le code calcule tout d'abord le nombre de points de phase N_{phase} nécessaire par sous-pupille en fonction du seeing :

$$N_{phase} = E\left(k \times \frac{d}{r_0}\right) \tag{4.2}$$

où $E(x)$ dénote la partie entière de x, d est le diamètre d'une sous-pupille en mètre, r_0 est le paramètre de Fried à la longueur d'onde de l'analyseur et k est un entier arbitraire permettant de s'assurer qu'il y aura assez de points de phase pour éviter que des tavelures ne soient repliées dans l'image. En considérant l'Éq. 4.1, cela revient à choisir un champ de vue égal à k fois le seeing :

$$FoV_{FEP} = N_{phase}\frac{\lambda}{d} = k \times \frac{\lambda}{r_0} \tag{4.3}$$

Nous avons choisit une valeur par défaut de $k = 6$ afin d'obtenir un champ par sous-pupille équivalent à 6 fois la tâche de seeing. Un tel champ permet d'assurer l'intégration de la majeure partie des tavelures dans l'image simulée. Ainsi, le dimensionnement

de la simulation dépend des conditions de turbulence simulées. Ce nombre peut ensuite être modifié pour assurer un facteur de ré-échantillonnage entier pour arriver à la taille de pixel entrée en paramètre par l'utilisateur, et également pour parvenir à des tailles de tableaux optimisant les performances de calcul des transformées de Fourier rapide.

Dans le cas d'un analyseur pyramide, l'échantillonnage choisi est un compromis entre différents paramètres. Comme pour le Shack-Hartmann, il faut s'assurer que l'échantillonnage soit suffisamment fin pour éviter le repliement des tavelures. Pour la pyramide, cela se traduit par une contrainte sur le diamètre D_p de la pupille en pixels :

$$D_p > E\left(m \times \frac{D}{r_0}\right) \tag{4.4}$$

où D est le diamètre de la pupille en mètre et m est un entier arbitraire similaire à k utilisé pour le Shack-Hartmann. Nous avons choisit $m = 3$ comme valeur par défaut. Ce choix permet de limiter la taille des tableaux à manipuler au cours de la simulation tout en assurant une taille de champ correcte pour l'intégration des tavelures dans l'image simulée. Il faut ensuite s'assurer à nouveau que le ré-échantillonnage puisse se faire correctement et que la taille des tableaux soient optimales pour le calcul des transformées de Fourier.

Dans tous les cas, le paramètre r_0 étant défini par l'utilisateur, le code s'assure d'un échantillonnage minimal indépendant de la valeur du seeing, pour des cas extrêmes où r_0 serait très grand par exemple. Ainsi, le nombre minimal de points d'échantillonnage de la phase par sous-pupille du Shack-Hartmann est de 16.

4.4.2 La pupille du télescope

Si COMPASS peut produire des pupilles circulaires, avec ou sans obstruction centrale, de manière tout à fait classique, une attention particulière a été apportée à la simulation de la pupille de l'ELT. En effet, le miroir primaire de ce dernier sera segmenté, et l'alignement imparfait des segments entraînera des aberrations supplémentaires.

COMPASS permet de prendre en compte ces effets spécifiques en introduisant diverses aberrations sur les segments du miroir, paramétrables par l'utilisateur. Ces dernières se limitent à des modes de tous premiers ordres, à savoir piston et tip-tilt. En sus, les variations de réflectivités entre les segments peuvent être prises en compte, ainsi que les segments manquants pour maintenance. La Figure 4.4 illustre ainsi une réalisation de la pupille ELT et des aberrations de phase prises en compte.

4.4.3 La turbulence atmosphérique

Les écrans de phase turbulents de COMPASS sont générés selon la méthode d'extrusion décrite par Assémat et al. (2006) puis Fried & Clark (2008) permettant d'optimiser l'empreinte mémoire de ces derniers. Cette méthode permet en effet de créer

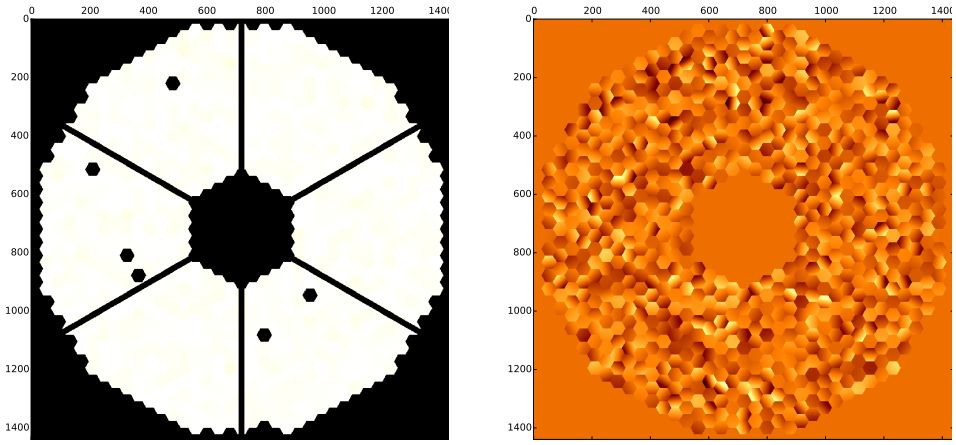

FIGURE 4.4 – Pupille ELT simulée par COMPASS avec différentes réflectivités et segments manquants (à gauche) et carte des aberrations de phase simulées pour les différents segments (à droite)

des écrans de phase infiniment grands en prolongeant l'écran d'une colonne supplémentaire à partir de quelques pixels judicieusement choisis de l'écran existant. Une fois cette colonne générée, tout le tableau de phase est décalé dans le "sens du vent", libérant la mémoire des valeurs sortantes en aval, et peuplant l'espace libre en amont par cette nouvelle colonne.

Assémat et al. (2006) pose ainsi :

$$\mathbf{X} = \mathbf{A}\mathbf{Z} + \mathbf{B}\beta \tag{4.5}$$

où $\mathbf{X}$ est le vecteur extension de l'écran de phase (i.e. celui que l'on doit rajouter à l'écran de phase après décalage), $\mathbf{A}$ et $\mathbf{B}$ deux matrices, β un vecteur de nombres aléatoires tirés selon une distribution gaussienne à moyenne nulle et $\mathbf{Z}$ un vecteur contenant certains des pixels de l'écran de phase existant. Le choix des pixels à inclure dans le vecteur $\mathbf{Z}$ a été revu par Fried & Clark (2008) : il inclut tout d'abord l'intégralité des éléments de la dernière colonne existante. Puis, en considérant un écran de phase de $2^N + 1$ pixels de côté, il faut des éléments provenant de N colonnes supplémentaires, ces colonnes étant les colonnes 2^{n-1} avec $n \in [1, N]$. Dans chacune de ces colonnes, le nombre d'éléments à considérer est égal à $2^{N-n} + 1$ à espacer uniformément sur la colonne.

Le vecteur $\mathbf{Z}$ ainsi connu, il reste à calculer les matrices $\mathbf{A}$ et $\mathbf{B}$. Assémat et al. (2006) montrent que

$$\mathbf{A} = \langle \mathbf{X}\mathbf{Z}^t \rangle \langle \mathbf{Z}\mathbf{Z}^t \rangle^{-1} \tag{4.6}$$

et que les matrices de covariance $\langle \mathbf{X}\mathbf{Z}^t \rangle$ et $\langle \mathbf{Z}\mathbf{Z}^t \rangle$ peuvent être calculées en utilisant la

fonction de structure de la phase turbulente. De même, ils montrent que

$$\mathbf{B}\mathbf{B}^t = \langle \mathbf{X}\mathbf{X}^t \rangle - \mathbf{A}\langle \mathbf{Z}\mathbf{X}^t \rangle \tag{4.7}$$

La matrice $\mathbf{B}$ peut alors être calculée en diagonalisant la matrice $\mathbf{B}\mathbf{B}^t$.

L'implémentation dans COMPASS comprend donc le calcul des matrices $\mathbf{A}$ et $\mathbf{B}$, tel que décrit au-dessus, pendant la phase d'initialisation. Ces matrices sont ensuite chargées sur la carte graphique ainsi que les indices de tableaux permettant de remplir le vecteur $\mathbf{Z}$ à partir de l'écran de phase existant. À partir des paramètres de vent définis par l'utilisateur, il est trivial de calculer de combien de colonnes Δ_x et de lignes Δ_y l'écran de phase doit se déplacer à chaque itération :

$$\begin{aligned} \Delta_x &= \frac{vdt}{p}\cos(\theta_v) \\ \Delta_y &= \frac{vdt}{p}\sin(\theta_v) \end{aligned} \tag{4.8}$$

où p est la taille d'un pixel de l'écran de phase en mètres, v est la vitesse du vent, dt est le temps d'itération de la boucle et θ_v est la direction du vent. À chaque itération de la boucle, l'Eq. 4.5 est calculée grâce aux librairies cuBLAS pour les produits matrice-vecteur et la librairie cuRAND est utilisée pour générer le vecteur aléatoire β. Il est important de noter que ses décalages n'ont lieu que sur un nombre entier de pixels : les prochaines versions de COMPASS pourront inclure une interpolation à cette opération afin de permettre un décalage fractionnaire.

La taille des écrans de phase dépend de la position des étoiles cibles et des étoiles guides des ASO définie par l'utilisateur en paramètres de la simulation. Le modèle de turbulence de COMPASS repose également sur l'hypothèse de Taylor : un écran de phase indépendant est généré par couche turbulente. À noter également que tous les écrans de phases générés grâce au modèle de COMPASS sont exprimés en terme de différence de chemin optique, en microns.

4.4.4 Le modèle de Shack-Hartmann

Le modèle d'analyseur Shack-Hartmann de COMPASS est basé sur un calcul de transformée de Fourier de la portion du front d'onde vue par chaque sous-pupille. L'écran de phase effectivement vu par l'ASO est déterminé à partir des écrans turbulents, des écrans de phase produits par le modèle de miroir déformable (cf. Section 4.4.10) et de la pupille du télescope par une opération de lancer de rayons comme décrite dans la Section 3.3.1. Cette opération est réalisée en parallèle sur la carte graphique autant de fois qu'il y a d'écrans de phase à considérer, les résultats étant sommés pour obtenir l'écran de phase final.

Le processus permettant ensuite d'aboutir à l'image de l'ASO est schématisé par la Figure 4.5. Une fois que la phase sur la pupille, dans la direction de visée de l'ASO est connue, cette dernière est découpée selon l'emplacement des sous-pupilles : la portion de phase "vue" par chaque sous-pupille est ainsi déterminée, passée sous forme

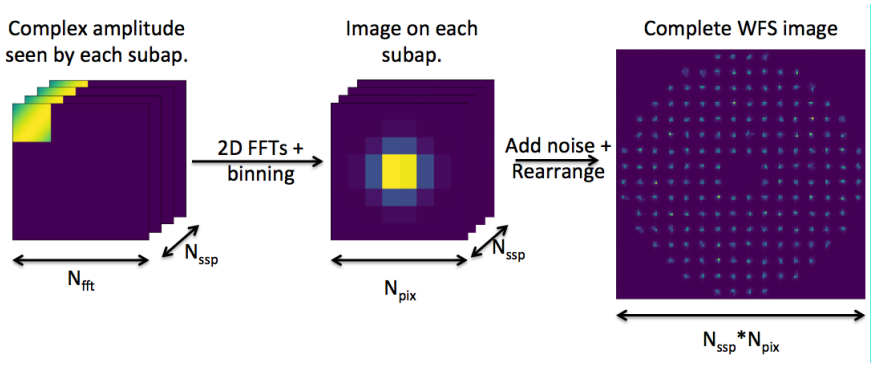

FIGURE 4.5 – Processus de calcul de l'image fournie par un ASO SH dans COMPASS

complexe et le spot lumineux qui en résulte est calculé par une transformée de Fourier en utilisant la librairie cuFFT. Une fois que tous les spots ont été calculés, ils sont ré-échantillonnés sur la taille des pixels dans une sous-pupille définie par l'utilisateur, et réarrangés en 2D pour former l'image finale de l'ASO. Notons que la réorganisation en 2D n'a lieu que lorsque l'utilisateur demande à voir l'image.

Le bruit de photons est ajouté par un tirage aléatoire selon une distribution de Poisson d'espérance N_{ph} où N_{ph} est le nombre de photons reçus par sous-pupille en une itération de boucle :

$$N_{ph} = F_0 \times 10^{-0,4m}\, dt\, T\, S \tag{4.9}$$

avec F_0 le flux par seconde et par mètre carré à magnitude nulle , m la magnitude de l'étoile guide, dt le temps d'itération de la boucle, T le coefficient de transmission optique de l'ASO et S la surface d'une sous-pupille, tous ces termes étant définis par l'utilisateur.

Le bruit de lecture est généré quant à lui selon une distribution normale d'écart type défini par l'utilisateur.

J'ai également ajouté dans COMPASS la possibilité de "désaligner" l'ASO : l'utilisateur peut paramétrer un décalage en X et en Y, un angle de rotation, ainsi qu'un facteur de grandissement qui agissent au moment du lancer de rayon pour modifier l'écran de phase vu par l'ASO. Cette fonctionnalité est utile en simulation afin, par exemple, d'établir les tolérances d'alignement lors de la conception de l'instrument.

Chaque sous-pupille du Shack-Hartmann étant indépendante des autres, ce calcul se prête bien à la parallélisation sur GPU. Après le lancer de rayon, chaque sous-pupille est traitée en parallèle des autres par un bloc d'exécution CUDA afin d'obtenir une image par sous-pupille. Les calculs font aussi bien intervenir des librairies fournies par CUDA pour les calculs de transformées de Fourier et la génération de nombres pseudo-aléatoires par exemple, mais également des noyaux de calcul dédiés, notamment pour le sous-échantillonnage et la formation de l'image finale de l'ASO à partir des images des sous-pupilles.

4.4.5 L'étoile laser

COMPASS inclus également un modèle d'étoile laser : celui-ci prend en compte l'élongation des images de l'étoile laser et l'effet de cône. L'effet d'indétermination du tip-tilt n'est quant à lui pas intégré dans le modèle, le trajet montant du laser n'étant pas simulé : l'étoile laser agit alors comme une source parfaitement fixe par rapport aux étoiles naturelles.

L'élongation est obtenue en convoluant l'image d'une sous-pupille calculée comme précédemment par un objet allongé représentant la source créée par le laser dont les caractéristiques varient pour chaque sous-pupille. Ces images allongées (cf. Figure 4.6) sont calculées selon les positions respectives de la sous-pupille et du télescope d'émission du laser, et du profil de la couche de sodium utilisé.

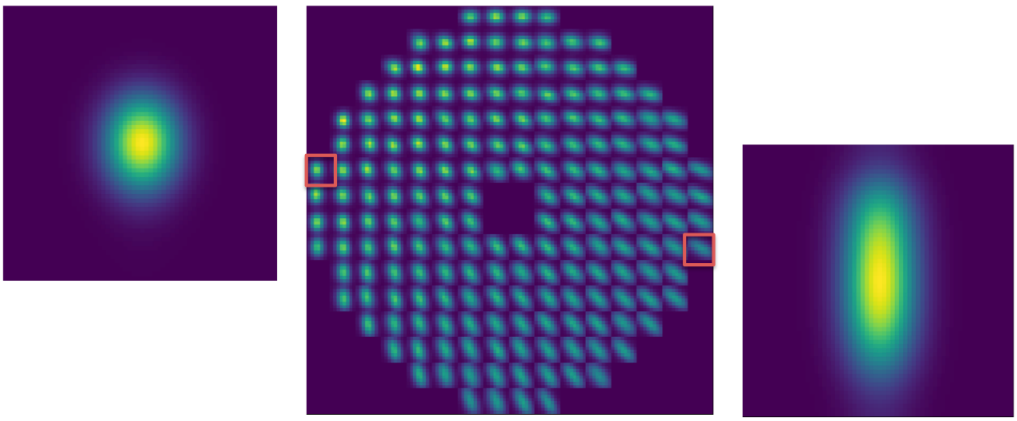

FIGURE 4.6 – Spots allongés selon la position de la sous-pupille par rapport à la position d'émission du laser utilisés pour la convolution

Le profil de la couche de sodium peut être chargé par l'utilisateur comme un fichier dans lequel figure la densité de sodium en fonction de l'altitude et du temps. Par défaut, le modèle d'étoile laser inclus un profil gaussien centré autour de 90 km d'altitude, mais des profils plus complexes, issus de mesure de profilométrie, peuvent être utilisés comme illustré dans la Figure 4.7.

La prise en compte de l'effet de cône dans le modèle a été l'une de mes premières réalisations sur le code. Son implémentation consiste uniquement à modifier le modèle de lancer de rayons. Pour un pixel à la position (x, y) de l'écran de phase vu par l'ASO correspond une position (x', y') sur l'écran de phase turbulent. Pour une étoile naturelle sans effet de cône, (x', y') s'obtiennent par une simple translation fonction de l'altitude h de la couche turbulente :

$$\begin{aligned} x' &= x + \frac{\theta_x \times h}{p} \\ y' &= y + \frac{\theta_y \times h}{p} \end{aligned} \tag{4.10}$$

avec θ_x et θ_y la position angulaire en X et en Y de l'étoile guide, et p la taille d'un

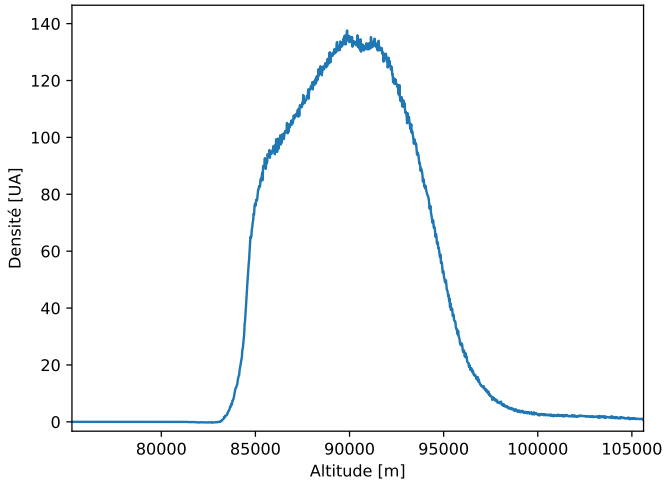

FIGURE 4.7 – Profil de sodium utilisé dans COMPASS (unité arbitraire)

pixel en mètres. Dans le cas d'une étoile laser, son altitude finie doit être prise en compte et la position (x', y') peut alors être calculée géométriquement comme :

$$\begin{aligned} x' &= x \times (1 - \Delta_h) + \frac{\theta_x \times h}{p} \\ y' &= y \times (1 - \Delta_h) + \frac{\theta_y \times h}{p} \end{aligned} \tag{4.11}$$

avec :

$$\Delta_h = \frac{h}{h_{lgs}} \tag{4.12}$$

où h_{lgs} est l'altitude de l'étoile laser. Tyler (1994) donne une expression analytique de la variance résiduelle du front d'onde due à l'effet de cône :

$$\sigma^2_{cone} = \left(\frac{D}{d_0}\right)^{5/3} \tag{4.13}$$

avec D le diamètre du télescope et d_0 un second diamètre de Fried défini pour l'effet de cône. La Figure 4.8 superpose cette expression à l'erreur d'effet de cône effectivement obtenue avec COMPASS en fonction de l'altitude de la couche turbulente avec une étoile laser à 90 km d'altitude et sert de validation à l'implémentation de l'effet de cône dans le modèle.

4.4.6 Optimisation de l'empreinte mémoire des ASO

Si l'implémentation du modèle d'analyseur Shack-Hartmann présentée au-dessus est efficace sur GPU, elle est également gourmande en terme d'espace mémoire. En effet, elle requiert l'allocation de plusieurs tableaux de tailles importantes, notamment

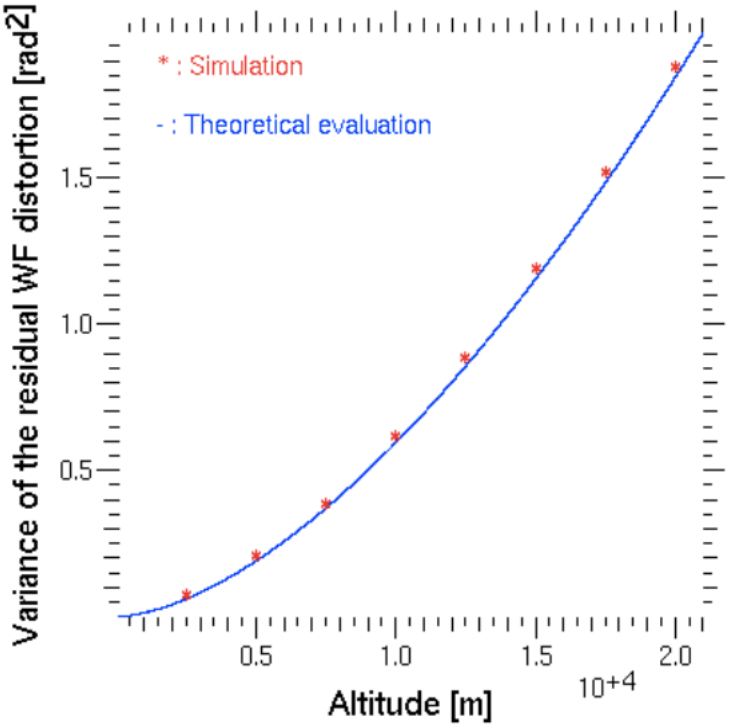

FIGURE 4.8 – Variance de l'erreur résiduelle due à l'effet de cône en fonction de l'altitude de la couche turbulente pour une étoile laser sodium à 90 km d'altitude. En bleu : l'expression analytique, en rouge : la variance obtenue avec COMPASS

pour le calcul des transformées de Fourier, et ceci est d'autant plus vrai que le système d'OA à simuler est grand. En effet, le calcul d'une transformée de Fourier avec la librairie cuFFT requiert l'allocation de 3 espaces mémoires : un espace de départ (le tableau dont on veut la transformée de Fourier), un espace d'arrivée (le tableau contenant la transformée de Fourier après calcul) et un espace de calcul intermédiaire. L'allocation de ces trois espaces mémoires est donc requis à l'initialisation du code, et 3 autres tableaux seront requis dans le cas d'une étoile laser pour réaliser la convolution avec la source allongée. Dans le cas d'un ELT, l'espace nécessaire atteint alors facilement quelques Gigaoctets de mémoire sur le GPU pour seulement modéliser l'ASO.

Or, l'espace mémoire disponible sur une carte graphique est limitée : en mai 2018, seules quelques cartes professionnelles atteignent 16 Go de mémoire, quand les cartes plus classiques se situent entre 4 et 8 Go de mémoire. Dès lors, la simulation de système de type MCAO ou MOAO nécessitant plusieurs ASO devient vite impossible, faute d'espace mémoire suffisant.

Un travail important a alors visé à optimiser l'empreinte mémoire du modèle d'ASO sur le GPU. Dans la pratique, la solution que j'ai proposée et qui est intégrée dans COMPASS actuellement est de dimensionner les différents tableaux nécessaires en fonction de l'ASO nécessitant le plus d'espace mémoire. Les tableaux sont alors alloués une seule fois, et les ASO utilisent alors séquentiellement ces tableaux. En pratique, l'implémentation se fait via une classe d'abstraction gérant l'allocation de ces tableaux et donnant les pointeurs vers ces tableaux aux différents ASO, qui sont alors traités séquentiellement pour éviter les conflits de lecture et d'écriture sur ces tableaux. Ce travail a permis de limiter l'empreinte mémoire à celle d'un unique analyseur, rendant possible les simulations de système à plusieurs analyseurs, au prix d'un calcul séquentiel pour chaque ASO.

4.4.7 L'analyseur pyramide

COMPASS inclut un modèle de pyramide "haute résolution" dans lequel l'image de la pyramide est calculée à haute résolution et le sous-échantillonnage n'intervient qu'en fin de calcul. La pyramide étant située dans le plan focal, la première étape du modèle est de calculer la FEP complexe qui se forme au sommet de la pyramide. Cette FEP est obtenue comme la transformée de Fourier du front d'onde incident auquel s'ajoute un terme de phase introduit par la modulation et la pyramide elle-même. La fonction pyramide représentant la phase introduite par sa forme est illustrée sur la Figure 4.9. Ainsi, le modèle considère une pyramide à 4 faces dont l'angle au sommet est défini par l'utilisateur.

La modulation est calculée sur un nombre discret de positions de modulation N_{pts} défini par :

$$N_{pts} = 4 \times E(1 + \frac{\pi}{2} R_{mod}) \tag{4.14}$$

où E désigne la partie entière et R_{mod} le rayon de modulation exprimé en unité de λ/D et défini par l'utilisateur. Cette formule assure d'avoir une position de modulation à

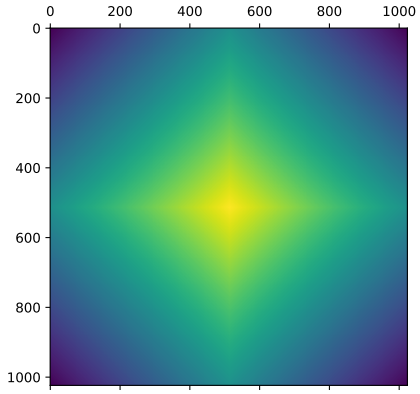

FIGURE 4.9 – Exemple de fonction de phase pyramide utilisée dans COMPASS

chaque λ/D, tout en ayant un nombre multiple de 4 assurant une répartition équitable entre les 4 faces de la pyramide. Ces points sont répartis circulairement autour du sommet de la pyramide. Chaque position de modulation introduit donc un terme de tip-tilt différent sur la phase incidente. Les images des 4 pupilles produites à chaque position de modulation sont obtenues par transformée de Fourier inverse et module carré, et sommées les unes aux autres pour obtenir l'image haute résolution des pupilles. Cette dernière est ensuite sous-échantillonnée pour satisfaire aux paramètres définis par l'utilisateur (cf. Figure 4.10).

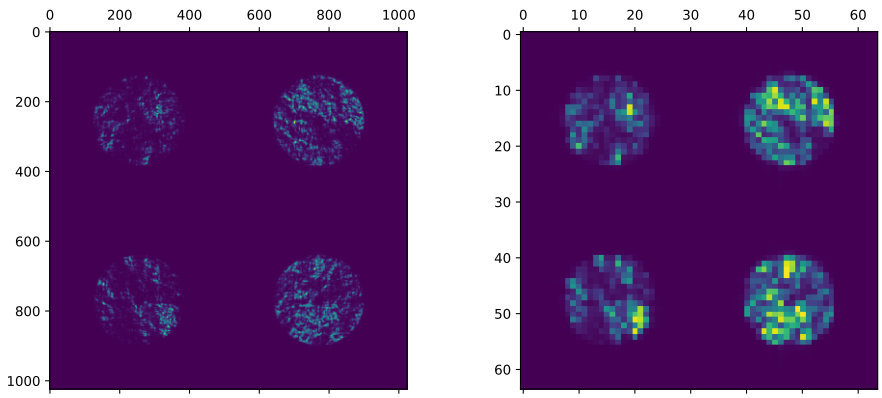

FIGURE 4.10 – Image haute résolution calculée par le modèle d'ASO pyramide (à gauche), puis sous-échantillonnée (à droite)

Ce modèle utilisant des transformées de Fourier sur de grands supports pour obtenir la haute résolution, une attention particulière lui a été porté afin d'accélérer le calcul. Une implémentation multi GPU a ainsi été mise en place, permettant de

paralléliser le calcul pour chaque position de modulation entre les GPU disponibles. Les performances obtenues grâce à cette implémentation seront discutées dans la Section 4.5.1.

4.4.8 Le calcul des mesures

Le modèle d'ASO a fourni une image, il faut maintenant en déduire les mesures de déformation du front d'onde correspondant. Pour ce faire, COMPASS propose divers algorithmes courants en optique adaptative.

Pour un analyseur Shack-Hartmann, plusieurs déclinaisons de calcul de centre de gravité sont disponibles. Les centres de gravité sont calculés parallèlement pour chaque sous-pupille, et les résultats sont stockés dans un vecteur de mesure de taille $2 \times N_{ssp}$: la première moitié contient les mesures selon l'axe X, la seconde celles selon l'autre axe.

Le calcul du centre de gravité se fait classiquement :

$$\begin{aligned} m_x &= \frac{\sum_{i=1}^{N_{pix}} x_i \, I(x_i, y_i)}{\sum_{i=1}^{N_{pix}} I(x_i, y_i)} \\ m_y &= \frac{\sum_{i=1}^{N_{pix}} y_i \, I(x_i, y_i)}{\sum_{i=1}^{N_{pix}} I(x_i, y_i)} \end{aligned} \tag{4.15}$$

avec m_x et m_y la mesure en X et en Y pour la sous pupille considérée, N_{pix} le nombre de pixels par sous-pupille et $I(x, y)$ la valeur du pixel à la position (x, y). À cette formulation classique s'ajoute plusieurs déclinaisons :

- un seuillage peut être appliqué afin de négliger les pixels dont l'intensité est inférieure au seuil défini
- une pondération selon une fonction de poids définie par l'utilisateur peut également être appliquée aux intensités des pixels (Fusco et al., 2004)
- le calcul du centre de gravité peut être effectué en ne considérant que les pixels les plus brillants (Basden et al., 2012)

Enfin, une méthode de mesure par corrélation est également disponible pour les analyseurs Shack-Hartmann. Cette dernière, surtout utile dans le cas de spots étendus comme on en trouve notamment en optique adaptive solaire ou avec l'élongation de l'étoile laser, consiste à utiliser une image de référence et à calculer la corrélation spatiale entre cette image et celle obtenue dans chaque sous-pupille. La position du maximum de corrélation donne alors la mesure (Michau et al., 1993). Pour plus de détails sur les méthodes utilisées pour le calcul des mesures avec un analyseur Shack-Hartmann, je renvoie le lecteur vers Thomas et al. (2006) et Gratadour et al. (2010) où une étude comparative entre ces différentes méthodes est menée.

Pour ce qui est de l'analyseur pyramide, le calcul des mesures est fait selon la description donnée par Ragazzoni (1996) et déjà énoncée dans l'Eq. (2.4). COMPASS intègre également une variante de cette expression où la somme des pixels placée au dénominateur est remplacée par la valeur moyenne de la totalité des pixels dans les 4 images de la pupille.

Pour terminer, COMPASS permet également d'obtenir les mesures directement à partir de l'écran de phase, en calculant le gradient. Les mesures obtenues sont alors telles que décrites par l'Éq. 2.3. Ce calcul est mené en découpant la phase selon les sous-pupilles (ou pixels) de l'ASO, et en calculant l'intégrale curviligne le long de cette sous-pupille. En supposant des sous-pupilles carrées, cela revient à soustraire la somme le long des bords verticaux pour la mesure en X, et horizontaux pour la mesure en Y.

4.4.9 Les lois de commande

COMPASS comprend divers types de lois de commande, la plus simple étant la méthode des moindres carrées avec intégrateur décrite dans la section introductive 2.4.4. J'ai par la suite été amené à intégrer d'autres méthodes que je vais décrire ici. Les performances de ces différentes approches seront discutées dans la Section 4.5.1.

Optimisation modale

Gendron & Léna (1994) propose une méthode d'optimisation qui consiste à appliquer un gain différent sur chaque mode contrôlé par la boucle d'optique adaptative. En décomposant la phase sur une base modale orthonormée, il montre que l'erreur résiduelle sur les modes peut être minimisée en calculant un gain modal g_i à appliquer sur le mode i tel que :

$$\int \frac{dH_{cor}}{dg_i} \left(\|\tilde{z}_i(f)\|^2 - p(i)\sigma_n^2 \right) df + p(i)\sigma_n^2 \int \frac{dH_n}{dg_i} df = 0 \tag{4.16}$$

où $H_{cor} = \|h_{cor}\|^2$ est le module carré de la fonction de transfert de réjection de la boucle, $\tilde{z}_i(f)$ est la transformée de Fourier de l'évolution temporelle du mode i, σ_n^2 la variance du bruit sur l'ASO, $p(i)$ le coefficient de propagation du bruit sur le mode i et $H_n = \|h_n\|^2$ est le module carré de la fonction de transfert du bruit. Les fonctions de transfert utilisées ici sont celles définies dans la Section 2.4.6.

En pratique, les gains modaux sont calculés entre une valeur minimale g_{min} et une valeur maximale g_{max} comme les gains g_i qui minimisent :

$$g_i = \arg\min \left\{ \int H_{cor}(g_i, f) \left(\|\tilde{z}_i(f)\|^2 \right) df \;\middle|\; g_i \in [g_{min}, g_{max}] \right\} \tag{4.17}$$

Le calcul des coefficients z_i est réalisé en projetant des mesures boucle ouverte sur une base de Karhunen-Loeve :

$$\mathbf{z} = S2M.\mathbf{m_{bo}} \tag{4.18}$$

avec $\mathbf{z}$ le vecteur composé des coefficients z_i, $S2M$ la matrice de passage entre l'espace des mesures et l'espace des modes, et $\mathbf{m_{bo}}$ le vecteur de mesures en boucle ouverte.

La matrice de passage $S2M$ est calculée comme :

$$S2M = \left[(D.M2V)^t D.M2V \right]^{-1} (D.M2V)^t \tag{4.19}$$

où D la matrice d'interaction du système et $M2V$ la matrice de passage de l'espace des modes vers l'espace des actionneurs du miroir déformable. Cette dernière est calculée par le processus de double diagonalisation décrit par Gendron (1995).

Les gains modaux sont recalculés régulièrement au cours de la simulation afin de rafraîchir la matrice de commande R qui est calculée comme :

$$R = M2V.\mathbf{g}.S2M \tag{4.20}$$

avec $\mathbf{g}$ le vecteur des gains modaux optimaux. En boucle fermée, le vecteur $\mathbf{m}_{bo}$ n'est pas directement accessible mais peut être reconstruit à partir des mesures en boucle fermée $\mathbf{m}_{bf}$:

$$\mathbf{m}_{bo} = \mathbf{m}_{bf} - D\left(\tau \times \mathbf{c}_{k-2} + (1-\tau) \times \mathbf{c}_{k-1}\right) \tag{4.21}$$

avec τ le retard de la boucle d'OA en nombre fractionnaire de trames et $\mathbf{c}_k$ le vecteur de commande calculé à la trame k :

$$\mathbf{c}_k = \mathbf{c}_{k-1} - R\mathbf{m}_{bf} \tag{4.22}$$

En résumé, l'algorithme d'optimisation modale intégré dans COMPASS fonctionne comme suit :

1. Calcul de la matrice $M2V$ par double diagonalisation
2. Calcul de la matrice $S2M$
3. Calcul de la fonction de transfert de réjection $H_{cor}(f)$
4. Acquisition de mesures boucle ouverte ou reconstruction à partir de mesures boucle fermée
5. Projection des mesures dans l'espace modal
6. Calcul des gains modaux selon l'Eq. 4.17
7. Calcul de la matrice de commande R selon l'Eq. 4.20

Les étapes 1 à 3 n'étant réalisées qu'une seule fois au cours de l'initialisation de la simulation, les étapes suivantes étant quant à elles répétées régulièrement au cours de la simulation pour rafraîchir la valeur des gains modaux.

Approche de type maximum de vraisemblance

L'approche du maximum de vraisemblance consiste à maximiser la probabilité que la phase reconstruite $\hat{\Phi}$ soit égale à la phase incidente Φ connaissant la mesure $\mathbf{m}$ fournies par l'analyseur. Dans le cas d'une statistique Gaussienne à moyenne nulle, cela revient à minimiser l'écart quadratique moyen entre la phase reconstruite et la phase incidente (Van Trees, 1968) :

$$\varepsilon = \|\hat{\Phi} - \Phi\|^2 \tag{4.23}$$

On montre alors que le reconstructeur optimal s'écrit (Wallner, 1983; Thiébaut & Tallon, 2010) :

$$R = C_\Phi D^t (D C_\Phi D^t + C_n)^{-1} \tag{4.24}$$

avec C_Φ la matrice de covariance de la phase turbulente et C_n la matrice de covariance du bruit. En considérant que la matrice $(DC_\Phi D^t + C_n)$ n'est autre que la matrice de covariance des mesures C_{mm} de l'analyseur en boucle ouverte, la matrice de commande R peut se réécrire (Gendron et al., 2014) :

$$R = C_{vm} C_{mm}^{-1} \tag{4.25}$$

où C_{vm} est la matrice de covariance entre les actionneurs du miroir déformable et les mesures de l'ASO. Ces matrices sont calculées par l'algorithme proposé par Gendron et al. (2014), à la différence près que la matrice C_{vm} fait intervenir une fonction de structure de phase modifiée ne prenant en compte que les fréquences spatiales accessibles au miroir déformable. Je reviendrai en détails sur le calcul de cette fonction de structure dans la Section 5.2.

Une fois le reconstructeur calculé, la loi de commande fait également intervenir une reconstruction des mesures en boucle ouverte à partir des mesures boucle fermée (cf. Eq. (4.21)). Enfin, la commande est calculée comme :

$$\mathbf{c}_k = (1-g)\mathbf{c}_{k-1} + gR\,\mathbf{m}_{bo} \tag{4.26}$$

Une loi de commande générique

J'ai également implémenté dans COMPASS une loi de contrôle "générique" dans le sens où les matrices qui interviennent dans cette loi ne sont pas calculées par le code, mais doivent être fournies par l'utilisateur :

$$\mathbf{c_k} = \mathbf{f}\,E\,\mathbf{c}_{k-1} + \mathbf{g}\,R\,\mathbf{m}_k \tag{4.27}$$

où les vecteurs $\mathbf{f}$, $\mathbf{g}$ et les matrices E et R sont définies par l'utilisateur.

Ce procédé permet ainsi à l'utilisateur de calculer lui-même la matrice de commande du système par la méthode qui lui convient, puis de l'intégrer dans une loi de commande dont le comportement dépend des vecteurs et de la matrice qu'il apporte.

Projection de la phase

Une dernière méthode que j'ai dû implémenter, car nécessaire à l'établissement du budget d'erreur tel que je le définirai dans le chapitre 5, consiste à projeter directement la phase incidente orthogonalement sur le sous-espace des fonctions d'influence du miroir déformable :

$$\mathbf{c}_k = P\Phi \tag{4.28}$$

où P est la matrice de projection vers l'espace du miroir déformable, cette projection étant entendue au sens du produit scalaire entre deux fonctions défini par l'intégrale de leur produit sur la pupille. Cette matrice peut être calculée à partir des fonctions d'influence du miroir :

$$P = (IF^t\,IF)^{-1}\,IF^t \tag{4.29}$$

avec IF la matrice des fonctions d'influence du miroir.

Cette méthode permet d'obtenir directement la meilleure correction possible fournie par le miroir déformable : l'erreur résiduelle restante est alors l'erreur de sous-modélisation uniquement. La Figure 4.11 donne un exemple du résultat fourni par cette méthode sur un télescope de 8m avec une distance inter-actionneur de 20 cm.

À noter que la matrice IF peut être grande puisqu'elle a pour dimensions $(N_\Phi \times N_{act})$ où N_Φ est le nombre de pixels contenu dans la pupille du télescope. Pour un cas SCAO ELT, on arrive rapidement à une matrice de 32 Go, bien au-delà de la capacité mémoire d'une carte graphique.

Le modèle que j'ai mis en place prend alors en compte le fait que cette matrice est creuse. CUDA propose une librairie d'algèbre linéaire creuse, cuSPARSE, qui stocke les matrices creuses sous la forme de 3 vecteurs et un scalaire :

- le scalaire nnz est le nombre d'éléments non nuls de la matrice
- le vecteur **val** qui contient la valeur de ces éléments
- le vecteur **col** de longueur nnz qui contient l'indice de la colonne où se situe chaque élément de **val**
- le vecteur **row** de longueur $m+1$ où m est le nombre de lignes de la matrice. Le premier élément vaut 0 ou 1 selon l'indexation voulue pour la matrice, et le i^e élément dénombre les éléments non nuls de la matrice jusqu'à la i^e ligne.

Parce qu'un exemple vaut mieux qu'un long discours, considérons la matrice A de dimensions (4×5) :

$$A = \begin{pmatrix} 1 & 4 & 0 & 0 & 0 \\ 0 & 2 & 3 & 0 & 0 \\ 5 & 0 & 0 & 7 & 8 \\ 0 & 0 & 9 & 0 & 6 \end{pmatrix}$$

En utilisant une indexation commençant à 0, sa description au format creux s'écrit alors :

$$\begin{aligned} \textbf{val} &= \begin{pmatrix} 1 & 4 & 2 & 3 & 5 & 7 & 8 & 9 & 6 \end{pmatrix} \\ \textbf{col} &= \begin{pmatrix} 0 & 1 & 1 & 2 & 0 & 3 & 4 & 2 & 4 \end{pmatrix} \\ \textbf{row} &= \begin{pmatrix} 0 & 2 & 4 & 7 & 9 \end{pmatrix} \end{aligned}$$

La librairie cuSPARSE propose des algorithmes permettant de passer du format dense classique à ce format creux, cependant, dans notre cas, la matrice dense IF ne peut être chargée sur la carte graphique. L'idée est alors de charger la matrice ligne par ligne : chaque ligne est transformée selon le format décrit ci-dessus. Une fois que toutes les lignes sont disponibles, elles sont utilisées pour reformer la matrice creuse complète.

4.4.10 Le miroir déformable

COMPASS intègre un modèle de miroir déformable de type "piézostack" à partir d'un ensemble de fonction d'influences. Ces dernières peuvent être calculées analytiquement à partir d'une bibliothèque de fonctions existantes, ou bien fournies par

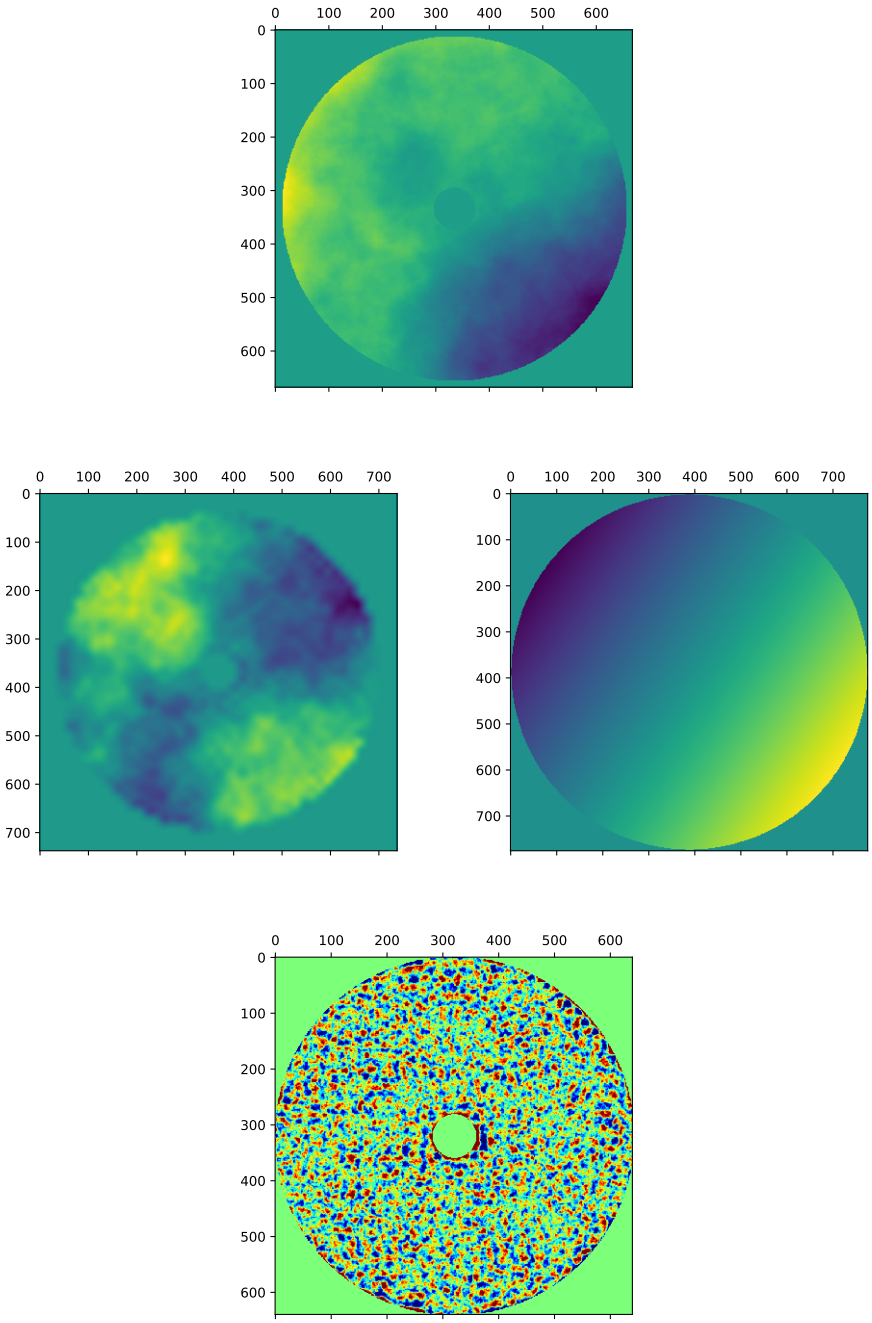

FIGURE 4.11 – En haut : phase turbulente, au milieu : résultat de la projection sur le miroir déformable (à gauche) et sur le miroir tip-tilt (à droite), en bas : phase résiduelle

l'utilisateur. Elles sont ensuite placées à l'endroit des actionneurs du miroir déformable, pondérées par la commande à appliquer aux actionneurs, et sommées.

Afin d'optimiser le temps de calcul, ces fonctions d'influence sont définies et donc calculées sur un support de petite taille devant celle de la pupille et qui dépend de la distance inter-actionneur et du facteur de couplage inter-actionneur qui définit l'étalement de la fonction d'influence. Une fois les fonctions d'influence calculées et placées dans un tableau 3D, la simulation calcule à l'initialisation, pour chaque point de la pupille, le nombre de fonctions d'influence participant à ce point et les indices du tableau 3D correspondants. Ce pré-calcul permet alors de paralléliser efficacement le calcul de la forme du miroir déformable une fois la commande calculée.

COMPASS intègre nativement différents types de fonctions d'influence analytiques $IF(x, y)$:

- une fonction d'influence identique à celle exploitée par le simulateur YAO (Rigaut & Van Dam, 2013), ajustée pour le miroir déformable du projet COME-ON (Gaffard, 1988) :

$$IF(x,y) = (1 - x^{p_1} + a \times \log(x) \times x^{p_2})(1 - y^{p_1} + cp \times \log(y) \times y^{p_2}) \quad (4.30)$$

  où les termes p_1, p_2 et a sont calculés en fonction de la valeur du couplage entre les actionneurs voulu par l'utilisateur
- une fonction d'influence Gaussienne, qui devra être définie sur un support fini et donc inévitablement tronquée, ce qui peut générer des discontinuités dans la phase (heureusement faibles) mais qui créent des répliques à la fréquence des actionneurs dans la FEP finale :

$$IF(x,y) = \exp\left(-0,5\left(\frac{x^2}{\sigma^2} + \frac{y^2}{\sigma^2}\right)\right) \quad (4.31)$$

  avec $\sigma^2 = \frac{pitch}{\sqrt{(-2\log(cp))}}$ où $pitch$ est la distance inter-actionneur en pixels et cp le facteur de couplage entre les actionneurs, compris entre 0 et 1.
- des fonctions appartenant à l'espace de Schwartz (Schwartz, 1966) (carrées ou rondes) présentant l'avantage de rapidement tendre vers 0 ainsi que toutes leurs dérivées, et ce sur un support fini, évitant ainsi les discontinuités de la phase évoquées précédemment (cf. Figure 4.12). À titre d'exemple, la fonction radiale s'écrit :

$$IF(r) = \exp\left(\frac{k\,a^2}{r - a^2} + k\right) \quad (4.32)$$

  avec $0 < r < 1$, k l'ordre de la fonction ($k = 6$ par défaut dans COMPASS) et $a = \frac{pitch}{\sqrt{\frac{k}{\log(cp)-k}+1}}$
- une fonction d'influence basée sur des fonctions de Bessel (Li & Zhang, 2014)

Ainsi, la Figure 4.13 donne une coupe le long de quelques fonctions d'influence simulée avec un "pitch" de 18 pixels et un facteur de couplage de 0,2. L'actionneur est

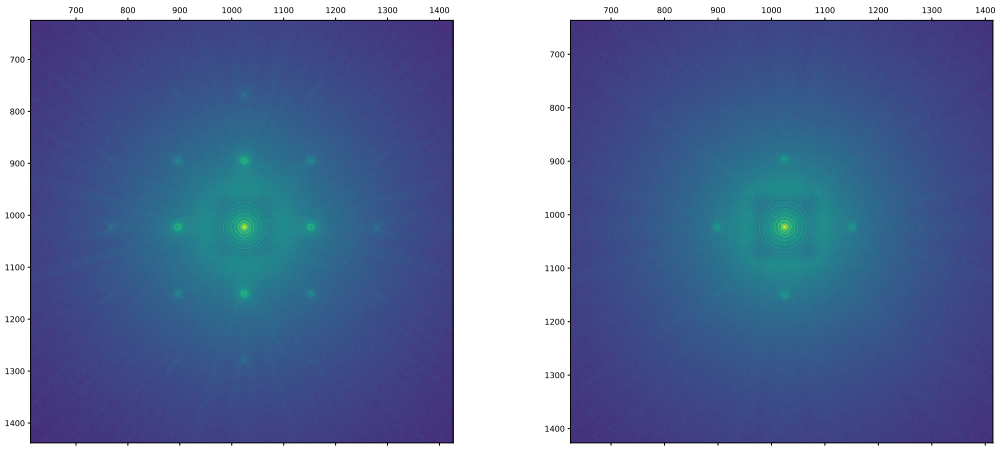

FIGURE 4.12 – FEP obtenue avec des fonctions d'influences gaussiennes (à gauche) et avec des fonctions d'influences de type Schwartz (à droite). Les pics secondaires visibles sont des artefacts dûs au support fini des fonctions d'influences. L'effet est limité par l'utilisation de fonctions tendant rapidement vers 0, telles que les fonctions de Schwartz. Ces répliques sont toutefois très petites : une bone correction et une échelle logarithmique sont requises pour les voir apparaître.

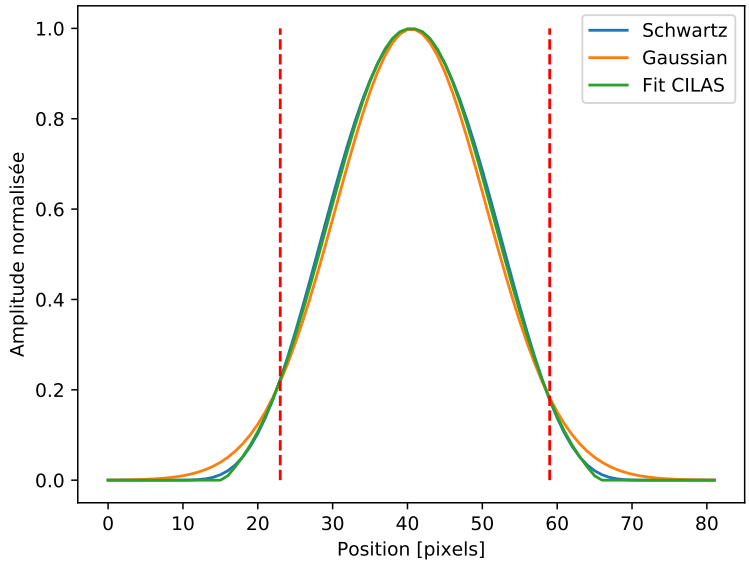

FIGURE 4.13 – Coupe le long de l'axe X de fonctions d'influence simulées par COMPASS avec un facteur de couplage de 0,2. L'actionneur est situé au centre, à la position du maximum, et les pointillés rouge indiquent la position des actionneurs voisins.

situé au centre et les pointillés rouge indiquent la position des actionneurs voisins le long de l'axe.

L'utilisateur peut également fournir un fichier contenant les fonctions d'influences qu'ils souhaitent utiliser, ainsi que les informations physiques sur le miroir déformable qu'il souhaite simuler (diamètre, positions des actionneurs, etc.). À partir de ces informations, le modèle de ce miroir déformable est inclus dans la simulation, après interpolation des fonctions d'influence fournies à la taille qui convient. À titre d'exemple, la Figure 4.14 représente un modèle du miroir M4 de l'ELT, obtenu à partir d'un fichier HDF5 généré à partir de données fournies par l'ESO. Le miroir génère ici un piston différent sur chacun de ses pétales.

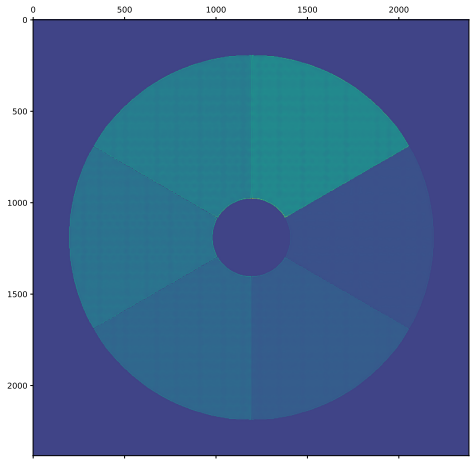

FIGURE 4.14 – COMPASS intègre un modèle de miroir déformable définit à partir d'un fichier fourni par l'utilisateur, comme ici le miroir M4 de l'ELT

Enfin, COMPASS intègre également le modèle d'un miroir tip-tilt et de modes de Karhunen-Loève en guise de miroir déformable.

4.4.11 Application de la commande sur le miroir

Avant d'appliquer la commande calculée par la loi de commande sur le miroir déformable, il convient de prendre en compte le délai de la boucle. En effet, comme nous sommes en simulation, le retard de la boucle d'OA est virtuellement nulle : ce dernier, défini comme paramètre de la simulation, est alors modélisé au moment de l'application de la commande sur le miroir. Les commandes obtenues lors des quelques itérations précédentes sont stockées, et le vecteur de "volts" v_k appliqués au miroir est calculé comme une interpolation linéaire de ces commandes pondérées par le retard τ de la boucle :

$$v_k = (a \times c_k + b \times c_{k-1} + e \times c_{k-2}) + p_k \tag{4.33}$$

avec :

$$\begin{cases} a = 1 - b \\ b = floor(\tau) \quad \text{si } 0 \leq \tau \leq 1 \\ e = 0 \end{cases}$$
$$\begin{cases} a = 0 \\ b = 1 - e \quad \text{si } 1 < \tau \leq 2 \\ e = floor(\tau) - 1 \end{cases} \tag{4.34}$$

et p_k des perturbations volontairement ajoutées et paramétrées par l'utilisateur. Pour une boucle standard, $p_k = 0$.

4.4.12 Formation de l'image

Pour terminer, le principal produit d'une simulation numérique avec COMPASS est la FEP du système simulé. Cette dernière est calculée comme la transformée de Fourier du champ complexe associé à la phase résiduelle obtenue après lancer de rayons au travers des couches d'atmosphères et des miroirs.

La taille du support de la FEP est choisie comme la puissance de 2 directement supérieure à deux fois la taille de la pupille, et ce afin d'assurer un échantilonnage correct de la FEP.

La FEP est ensuite normalisée par le carré du nombre de points dans la pupille afin que son maximum soit égal au rapport de Strehl. L'opération est réalisée à chaque itération de la boucle d'OA afin de produire des FEP courte pose, qui sont finalement moyennées progressivement pour obtenir la FEP longue pose.

4.5 Performances et *scalabilité*

4.5.1 La conférence High Performance Computing & Simulation

La conférence internationale High Performance Computing & Simulation est une conférence à comité de lecture. En effet, la participation à cette conférence implique la soumission d'un article de 8 pages maximum soumis à un comité de lecture constitué d'au moins 3 examinateurs, selon les standards de l'association IEEE [1]. Sur l'édition 2018 à laquelle cet article a été soumis, seuls 40 % des soumissions ont été retenues. L'article qui suit a été accepté après examen de 3 relecteurs. Cet article se concentre sur les performances de COMPASS en termes de vitesse d'exécution, en les comparant notamment aux performances de YAO. J'inclus ici la version finale corrigée qui paraîtra après la conférence qui s'est tenue en juillet 2018.

4.5.2 Article HPCS 2018

1. *Institute of Electrical and Electronics Engineers*

COMPASS: an efficient GPU-based simulation software for adaptive optics systems

Florian Ferreira, Damien Gratadour, Arnaud Sevin, Nicolas Doucet
Observatoire de Paris, LESIA, Univ. Paris Diderot
Meudon, France
Email: florian.ferreira@obspm.fr

***Abstract*— For ground-based astronomical telescopes, image resolution is directly dependent on aperture diameter. Practically however, atmospheric turbulence disturbs incoming wavefronts and sets dramatic limitations on imaging quality, which issue led to using Adaptive Optics (AO) systems to compensate wavefront disturbance and achieve optimal resolution. With the arrival of Extremely Large Telescopes, with aperture diameter above 20 meters, High Performance Computing (HPC) techniques are required to simulate and operate AO systems. With the advent of General Purpose Graphics Processing Units (GPGPU), many-core architectures of GPUs can be leveraged to accelerate AO simulation software. COMPASS (COMputing Platform for Adaptive opticS System) is an end-to-end AO simulation software based upon Nvidia's Compute Unified Device Architecture (CUDA) toolkit. Simulations performed on state-of-the-art Nvidia DGX-1 servers equipped with 8 Tesla V100-SXM2 and Dual 20-core Intel Xeon E5-2698 v4 show a speed up factor of up to 362 as compared to YAO, a feature-equivalent software with multi-threaded CPU architecture. The COMPASS implementation ensures compatibility with any Nvidia architecture. A speed up by 5 was measured between a DGX-1 and a comparable server with 6-year old GPUs, with no line of code changed.**

***Keywords*—GPGPU; Parallelization of Simulation; Applied Modeling and Simulation using HPC; Large Scale Scientific Computing**

I. Introduction

Astronomical sources are considered infinitely far from the ground-based telescope, and therefore the incoming light wavefront is expected to be flat, and image resolution must be limited only by the telescope diameter. Practically though, image resolution is limited by atmospheric turbulence, which induces disturbances on the wavefront, and thus loss of imaging resolution [1]. To overcome this phenomenon, Adaptive Optics (AO) systems have been developed to measure wavefront disturbances and compensate for them [2]. However, the complexity of AO systems scales as the square of telescope diameter, and with the upcoming Extremely Large Telescope (ELT) of diameter 40 m, numerical simulations of AO system performance becomes very demanding in terms of data flow and computation time [3].

General Purpose Graphics Processing Units (GPGPU) allow for applications – such as numerical simulations of complex systems – to be accelerated[4]. The GPU architecture, coupled to efficient libraries and toolkits such as Compute Unified Device Architecture (CUDA) or other programming frameworks (e.g. OpenCL), allows developers to achieve efficient parallel implementations of numerical models using the several thousands of cores on a GPU[5].

COMPASS (COMputing Platform of Adaptive opticS System) is an end-to-end AO system simulation tool able to address ELT-scaled simulations. COMPASS takes advantage of GPU acceleration through the Nvidia CUDA toolkit [6]. This paper overviews the capacities of this platform and compares it to other known AO simulation software. In Section II, we highlight the need of HPC techniques for adaptive optics systems in astronomy. Section III presents COMPASS software architecture and implementation. Section IV describes the benchmark procedure used to perform performance comparisons. Finally, Section V shows performances obtained in terms of framerate and scalability.

II. HPC for adaptive optics

A. Adaptive optics for astronomy

In ground-based astronomical observation, with a circular aperture, the Point Spread Function (PSF) of the telescope is an Airy function, leading to a theoretical diffraction-limited resolution R equal to its full width at half-maximum (FWHM):

$$R = \frac{\lambda}{D}, \tag{1}$$

where λ is the observation wavelength and D the telescope diameter.

However, temperature fluctuations in the atmosphere lead to refraction index variations, and so to wavefront disturbances. Consequently, the PSF of the telescope – and its resolution – are degraded by speckles, as shown in Fig. 1. In this case, the resolution is limited by the Fried parameter r_0, which may be interpreted as the diameter of a telescope with diffraction-limited resolution identical to the turbulence-limited telescope [7]. Typical values of r_0 in the visible are around 10 cm. An AO system aims at correcting the wavefront to retrieve the diffraction limited resolution of the telescope. Fig. 2 shows a schematic of a single conjugate AO (SCAO) system with all its components. As shown in this schematic, an AO system operates through following two main steps:

- measuring the wavefront using a wavefront sensor (WFS)
- compensate it by sending commands to a deformable mirror (DM), via a real-time controller (RTC)

A beam splitter is used to share the light between the AO system WFS and the science camera (not part of the AO

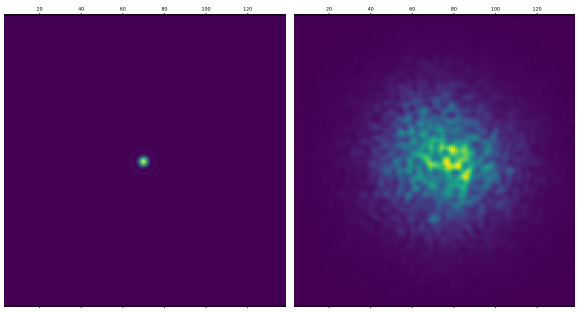

Fig. 1. Airy function (top) and PSF degraded by atmospheric turbulence (bottom)

system). Several WFS technologies are available, the most common being the Shack-Hartmann WFS [8] and the Pyramid WFS [9]. A Shack-Hartmann WFS is composed of a microlens array that samples the wavefront from a guide star into sub-apertures, each of which forming an image of the latter star on a CCD. A typical image from a SH WFS is displayed in the top-right part of Fig. 2. The position of each spot yields the local slope of the wavefront, and can be computed by the RTC using classical centroiding algorithms. The total number of measurements – or slopes – N_{slopes} is twice (X and Y axes) the number of illuminated subapertures N_{ssp} of the WFS.

A Pyramid WFS operates differently. The wavefront is focused on the tip of a pyramidal prism and – using a relay lens – each face of the prism produces a pupil image, resulting in a total of 4. Intensity differences between each pixel of those 4 images yields one measurement, leading to as many measurements as the number of pixels in a pupil image for a given axis. Then, N_{slopes} is twice the number of pixels in a pupil image.

The RTC has two main functions: computing the slopes from the WFS image and computing the commands to apply on the DM actuators from these measurements. Several

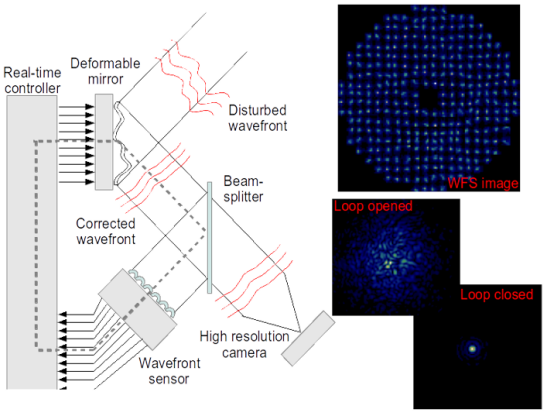

Fig. 2. Schematic of an AO system

reconstruction methods exist for command computation from measurements [10], [11], [12], with classical ones generally requiring a N_{act} by N_{slopes} command matrix, where N_{act} is the number of DM actuators. This command matrix is usually built, in the case of classical AO, as the pseudo-inverse of a so-called interaction matrix, which contains a calibration of the interaction between DM actuators and WFS slopes. A matrix-vector multiplication (MVM) between the command matrix and the slopes vector is then performed to obtain the command vector at each iteration of the AO loop. An important constraint is that those operations have to be performed in real-time, so as to follow the evolution of the atmospheric turbulence [13]. Typically, an AO loop needs to be run at several hundred iterations per second.

The correction brought by an AO system is not perfect: the corrected wavefront is not perfectly flat because of residual errors induced by e.g. noise or servolag-limited spatial sampling of the turbulence. The performance of an AO system is generally assessed in terms of Strehl ratio (SR), which is the ratio between the PSF maximum intensity actually obtained after the AO correction and the maximum intensity of the diffraction limited PSF:

$$SR = \frac{max(PSF)}{max(Airy)}. \tag{2}$$

The obtained PSF directly depends on the variance of the residual error σ^2_Φ, and the Strehl ratio can be approximated as [14]:

$$SR \approx \exp\left(-\sigma^2_\Phi\right). \tag{3}$$

Generally, the main contribution to this residual error is the temporal error due to AO loop latency [15]. Hence, the AO loop needs to be as fast as possible to minimize this contributor and increase performance. More complex AO systems exist, which include more than one WFS or DM, and sometimes Laser Guide Stars (LGS), such as Ground Layer AO (GLAO) [16], Multi Conjugate AO (MCAO) [17] or Multi Object AO (MOAO) [18]. These systems generally provide better sky coverage than a simple SCAO system, but complexity is dramatically increased.

B. ELT scale numerical challenge

The Extremely Large Telescope (ELT) is a giant telescope designed by the European Southern Observatory (ESO), with a 39 meter diameter, currently under construction in Chile for a first light foreseen around 2025 [19]. The number of degrees of freedom of an AO system – i.e. the number of sub-apertures for wavefront measurement and the number of DM actuators for wavefront correction – approximately scales as the square of the telescope diameter. There, the factor 5 diameter, and thus resolution, scaling of the next generation of extremely large telescopes as compared to the state-of-the-art Very Large Telescope [20] translates into a factor 25 in terms of degrees of freedom. As an example, the MCAO mode of the ELT first light instrument MAORY includes up to 9 WFS and 3 DMs, leading to a 90k×15k-sized command matrix to be multiplied by a 90k-elements vector to obtain the 15k DMs actuators

commands. The AO loop frequency requirement is fixed to 1 kHz. With such design, the computing power required by the AO RTC can reach up to 1.5 TMAC/s (Tera Multiply ACcumulate per seconds) [21]. Full length ELT simulations are therefore compute-intensive applications, and as such good candidates for using of hardware accelerators like manycore processors.

C. AO simulation tools

The cornerstone of design studies of a complex system is numerical simulation. It allows validation of conceptual design and testing the behavior of various system components under realistic conditions. It is thus essential for the ELT project contributors to have access to a high performance end-to-end AO simulation platform able to tackle the ELT scale. Due to the stochastic nature of the turbulence, Monte Carlo simulations provide the most realistic results, the extremely large diameter of the ELT making each iteration a large scale problem in itself. Several AO end-to-end simulation softwares are already available online such as:

- YAO [22] based on Yorick language with CPU multi-threading
- OOMAO [23] based on Matlab, with CPU multi-threading
- MAOS [24] based on C and CUDA C languages, with CPU multi-threading and/or GPU acceleration

In Section V, we will compare COMPASS performance with YAO. We chose to restrict our comparison to this software exclusively since our goal is to show the speed up obtained with a GPU implementation on equivalent physical models and not the absolute performance of our GPU implementation. Indeed, the MAOS implementation is targeting the Thirty Meters Telescope (TMT) instrumentation[25] and assumes a number of approximations related to these instruments, while COMPASS is a versatile platform targeting a wide range of telescope aperture sizes and instrument dimensioning. As such, MAOS complexity is not comparable to COMPASS: for instance, MAOS uses pre-computed turbulent phase screens while COMPASS computes those screens on the fly which allows for infinite turbulence generation length and an arbitrary number of turbulence screens without the intrinsic memory limitation due to preloading large turbulent screens in the simulation framework. Additionally, we have chosen to use fine grained spatial sampling in COMPASS to get high accuracy on the individual WFS images while no details have been provided by MAOS developers in previous publications on the sampling choices they made in the code. This makes comparison results hard to appreciate, since the execution profile is mostly dominated by WFS images generation. Concerning OOMAO, while the code is open source, it requires a Matlab licence. On top of that, OOMAO public repository has not been updated for several years now and does not include a number of features we need to perform realistic AO simulations in the context of the ELT. For this reasons, a performance comparison with COMPASS does not appear to be valuable.

Our team has also developed another framework relying on a pseudo-analytical approach leveraging only linear algebra to simulate AO systems behavior[26]. While it has proven to be very efficient to simulate extended observations on large scale systems, it is based on a number of assumptions that do not provide the same level of accuracy as the end-to-end framework described in this paper. However, it can be used to feed COMPASS with reconstructor matrices and we have built an transparent interface to make these two software stacks inter-operable.

III. COMPASS

A. Simulation architecture

The initial objective behind the development of COMPASS was to get an open source, licence free numerical simulation platform able to deliver short exposure PSFs (i.e. at each WFS frame), with a time-to-solution in the range of the 10th of seconds for an ELT scale AO system. To achieve such performance, using GPUs as hardware accelerators appeared as the best option.

A comprehensive software stack has been designed to provide simultaneously both high-performance computing – brought by core massively parallel algorithms running on GPU – and ease of use – brought by a user interface based on python and a graphical toolkit.

As shown in Fig. 3, the developed stack is composed of three main layers. The lowest level comprises the optimized libraries, in which the memory allocation routines can be found. The intermediate layer is a binding layer, where the arrays in memory are manipulated through addresses. The upper layer is the user interface in which python code interacts with the lower level classes and methods. A number of high-level routines can also be found in the upper layer, mostly for initialization and AO loop data post-processing. On top of this standard stack, an application programming interface (API) is available in python for running command-line or scripted simulations. A collection of widgets for an easy-to-use GUI is also available, based on the Qt toolkit and the pyQtGraph library. Fig. 4 depicts the COMPASS simulation process.

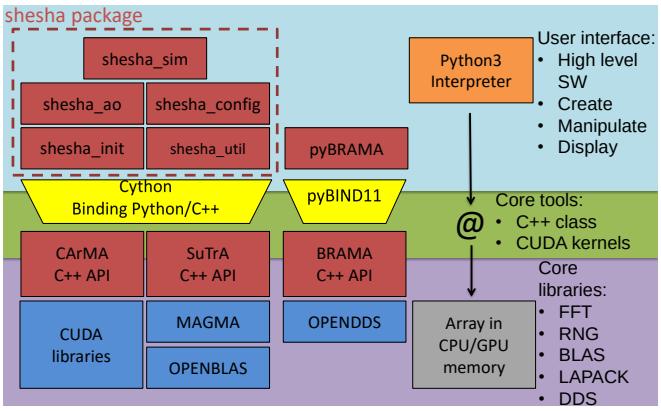

Fig. 3. COMPASS software architecture

First, simulation parameters are loaded from the user-defined parameter file, which is basically a python file instantiating parameter classes.

Then, these classes are used to initialize the simulation. This step mainly consist in the computation of sizes of simulation arrays and their allocation on the GPU. A critical parameter in the simulation model is the quantum pixel size, being the size (expressed in arcsec) of the smallest turbulence cell that can produce a significant optical path difference between two rays of the light beam. It defines how the turbulent phase is sampled in the simulation and sets constraints on the size of the largest arrays in the simulation.

In COMPASS, a special attention has been brought on this, as to ensure an appropriate sampling, and thus reliable results, all while keeping Fast Fourier Transform (FFT) optimal performance. This initialization step can be quite long (up to tens of minutes for the largest ELT MCAO scale), but then allows performing all loop computations directly on the GPU without any communication with the host CPU. As this kind of communication is a known bottleneck in HPC, avoiding them is an important optimization step. On the other hand, the platform uses the HDF5 file format to maintain a database – using the Pandas tool – of simulation parameters and results, which database allows to skip the initialization phase by loading previously computed simulation arrays, in the case a system has already been simulated with identical parameters.

Finally, the AO loop itself is performed on the GPU. Core GPU implementation relies on two libraries, which are detailed in Sect. III-B and III-C.

Diagnoses of simulation results can be optimized using the Bokeh tool. Finally, platform documentation is generated automatically using the Sphinx tool, and a website has been designed using GitHub Pages to provide installation instructions, an user guide, and feature details [27].

B. Core implementation: CArMA

CArMA is a low-level tool enabling easy manipulation of GPU memory allocated arrays. It shares several common features with the Thrust toolkit – available with CUDA – e.g. most noticeably the use of C++ templates from the Standard Template Library (STL). As shown on Fig. 5, CarMA provides a set of C++ classes for easy integration of GPU-accelerated numerical tools into complex applications.

CArMA is built on top of NVIDIA's CUDA toolkit, which provides a large collection of tools for performing scientific computing. Moreover, several libraries like MAGMA have been developed using CUDA and provide additional features most useful to scientific computing [28]. The purpose of CArMA is to provide easy access to rather low level GPU features, through the use of C++ classes. Three classes are used to control the CUDA context:

- the CArMA device providing information on a specific device
- the CArMA context, grouping information about overall system configuration
- the CArMA streams, being containers for a collection of CUDA streams wrappers

CArMA also provides two main classes for handling data in GPU and system memory:

- the CArMA object, providing a container for manipulating GPU data
- the CArMA host object, providing a container for system memory data, tagged and aligned so as to be accessed by the GPU DMA engine

Additionally, CArMA provides a set of wrappers to various libraries part of or based on the CUDA toolkit:

- cuBLAS for linear algebra operations
- cuRAND for random number generation
- cuFFT for FFT computations
- cuSPARSE for linear algebra on sparse matrices
- MAGMA for more complex linear algebra operations

CArMA can be used to easily build higher level applications, the SuTrA library being such an example. CArMA can also be bound to an interpreted language for simplified access to basic GPU features.

TABLE I
SUTRA LIBRARY AO FEATURES

Component	Features
Telescope	ELT pupils
	Phase aberration due to segments misalignment
Atmosphere	Independent turbulent layers (frozen flow)
	Kolmogorov or Von Karman spectrum
WFS	Shack-Hartmann
	Pyramid with multi-GPU support
	LGS including elongation and cone effect
	Average phase gradient
DM	Piezo-stack with multiple influence functions
	Tip-tilt mirror
	Pure Karhunen-Loëve modes
	Custom DM configuration from HDF5 file
Centroiding methods	Center of gravity (COG) ; Weighted COG
	Thresholded COG ; Brightest pixels [29]
	Correlation
Control strategy	Least squares ; Minimum variance
	CuReD [30] ; Phase projection

C. Core implementation: SuTrA

SuTrA – Simulation Tool for Adaptive optics – provides efficient tools for simulating a wide range of AO systems at various scales. SuTrA relies on CArMA and additional optimized CUDA kernels to model all the core components of an AO system. AO features available through the SuTrA library are listed in Table I, and details are provided in [27].

D. User interface: SHESHA

On top of the two previously detailed core components, two Python APIs codenamed NAGA and SHESHA have been developed through binding core features of CArMA and

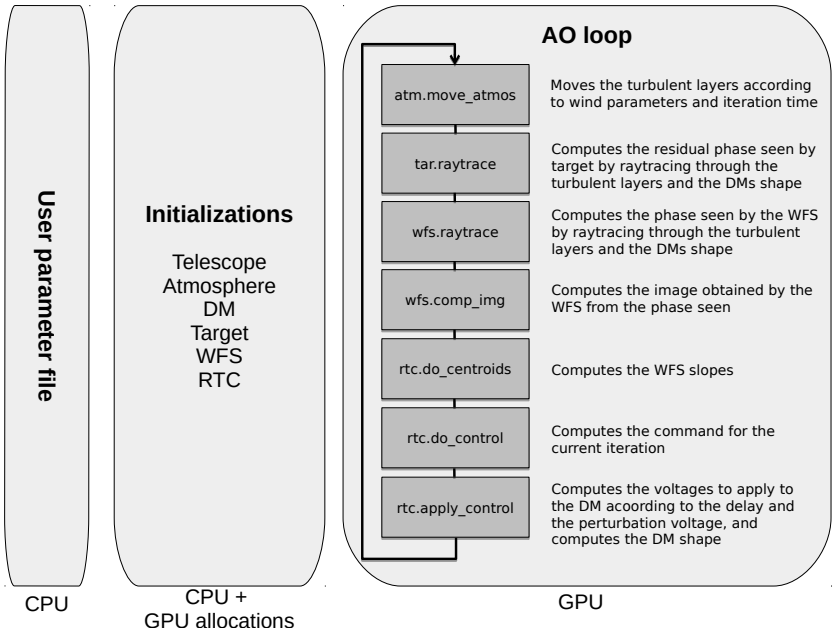

Fig. 4. COMPASS simulation process

SuTrA to the Python interpreter using Cython. While NAGA – which binds the low level features of CArMA – is not used in the COMPASS simulator, it is made available to users willing to develop extra features. SHESHA is the core of the COMPASS user interface; as shown in Fig. 3, it is made of 5 main modules:

- *shesha_config*, defining all parameter classes, including safe-typing and enumeration features to avoid unexpected errors due to unknown parameters.
- *shesha_init*, containing submodules that initialize the different components of the simulation. From user-defined parameters, it computes array sizes, and allocates and loads on the GPU all useful arrays and variables, as defined in the SuTrA library.

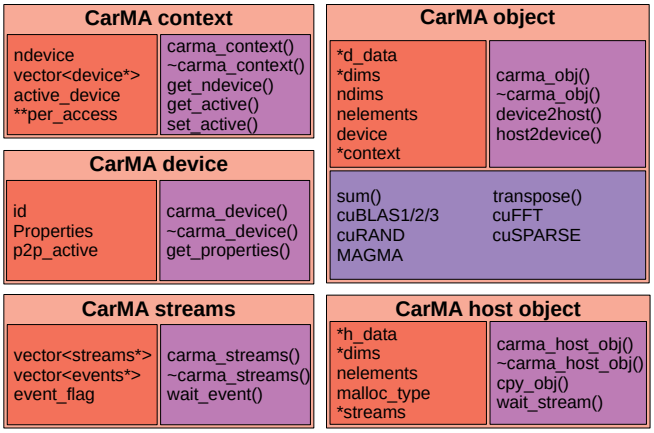

Fig. 5. CArMA classes

- *shesha_ao*, containing submodules that perform operations useful to AO loop optimization. It includes functions for computing interaction and command matrices, modal bases, modal optimization or tomographic matrices.
- *shesha_util*, including basic mathematical functions customized to code needs, such as rotation algorithms, indices generation, etc.
- *shesha_sim*, defining the simulator class, which acts as an abstraction layer for simulation scripts. Attributes of this class are the objects defined in the above-described initialization phase. These objects can then be manipulated through the simulator class. The AO loop process is embedded in a class method, making user scripts light and easy to write.

IV. Benchmark procedure

A. Timing process

To compare performances between COMPASS and YAO, we focus on AO loop speeds in terms of framerate, i.e. number of loop iterations per second. Indeed, the time needed for the initialization phase is not relevant as this step may only be performed once, then be re-used for other simulations.

YAO natively provides the average framerate at the end of simulations, using the embedded Yorick timing function. COMPASS follows identical behavior, using the time module provided by the Python Standard Library. Section V presents framerates obtained with either software for equivalent simulation cases.

The CArMA library also provides reliable GPU kernel timings using CUDA events. With this feature, we are able to precisely time each function called during the AO loop,

therefore producing a comprehensive execution profile of COMPASS as shown in Section V-C.

B. Test cases

Several simulations were performed, as described in Table II. Each simulation consists in a SCAO system with one WFS, one DM and one tip-tilt mirror. The SH WFS is on-axis and includes 8 pixels per sub-aperture. The atmosphere is composed of a single ground layer. The controller is a least-squares approach with a classical center of gravity for centroiding. Except for the atmosphere, these parameters are representative of AO systems currently operating and of future ELT AO systems. Pyramid modulation is computed on 32 discrete points.

The pyramid cases were performed on COMPASS only and are excluded from the comparison with YAO, as the pyramid model implemented in COMPASS differs from YAO's. This compass model has been designed for maximal realism with any simulation parameters, most noticeably pyramid modulation radii. Results obtained with the high resolution pyramid of COMPASS are shown for informative purposes, as the pyramid WFS is becoming more and more widespread in the AO community.

C. Environment

The simulations were performed on state-of-the-art Nvidia DGX-1 server, with Dual 20-core Intel Xeon E5-2698 v4 @ 2.2GHz and 8 Tesla V100-SXM2 GPUs. The Shack-Hartmann simulations were executed on a single GPU and the pyramid cases on 4 GPUs. The YAO simulations were multi-threaded over 8 cores, as it was the optimal configuration we found after testing.

V. Performance results

A. Time to solution

Time to solution is equivalent to framerate of the AO loop, as the solution is obtained when the loop is over. Fig. 6 shows the performance obtained for both COMPASS and YAO in terms of frame per second, averaged over 2,000 iterations. For the largest scale – SH 80x80 case – COMPASS computes 290 frames per second when YAO reaches 0.8 frames per second, yielding a speed up factor of 362 thanks to GPU acceleration. At lower scales, COMPASS becomes a "real-time" simulation with framerates of 2580, 1775 and 946 for cases SH 16x16, SH LGS 16x16 and SH 40x40 respectively. For those cases, the speed up compared to YAO is smaller compared to the ELT scale, with values 76, 42 and 135 respectively. Indeed GPUs are more efficient at large workloads, therefore the larger the simulation, the better the performance and speed up. We also note that simulating LGS reduces the speedup as compared to YAO. This can be explained as COMPASS leverages GPU acceleration to use high resolution models for LGS, leading to a better numerical precision, but slightly degrading overall computing performance. While the Laser spot is pre-computed in the initialization phase of YAO and reused unchanged in all AO loop iterations, COMPASS computes it on-the-fly at each loop iteration. This makes the simulation output more realistic, as the Laser spot shape is indeed expected to evolve along the duration of an observation, but also induces a larger workload.

These performance benchmarks show that COMPASS becomes a real-time simulator with framerate larger than the actual AO loop temporal sampling rate, when the performance of the multi-threaded code of YAO is around tens of frames per second (i.e. 10 times slower than the actual AO loop).

The high resolution pyramid model of COMPASS is obviously slower than SH models, with 24 frames per second at ELT scale computed on 4 GPUs. Note that using 2 GPUs instead of one leads to a speed up of 2, but this scaling factor decreases for more than 2 GPUs due to inter-GPU communication overheads. Best performance is obtained with 4 GPUs, with a speed up of 3 compared to a single GPU. Using more GPUs does not increase the performance further. These pyramid benchmark results depend on simulation parameters: as it is the WFS image computation that is distributed over the GPUs, a higher number of modulation points will lead to a larger workload per GPU, and hence to a better scaling factor. It also depends on the interconnect strategy between the GPUs. In our case, this is optimized by the use of the NVlink interconnect from NVIDIA, inside the DGX-1 server.

As mentioned in Section II-C, we focused on the compari-

TABLE II
Simulation parameters

Case name	Telescope Ø	WFS	N_{ssp}	N_{act}	GS
SH 16x16	8m	SH	368	222	NGS
SH LGS 16x16	8m	SH	368	222	LGS
SH 40x40	8m	SH	2400	1286	NGS
PYR 40x40	8m	Pyramid	2400	1286	NGS
SH 80x80	40m	SH	9792	5070	NGS
PYR 80x80	40m	Pyramid	9792	5070	NGS

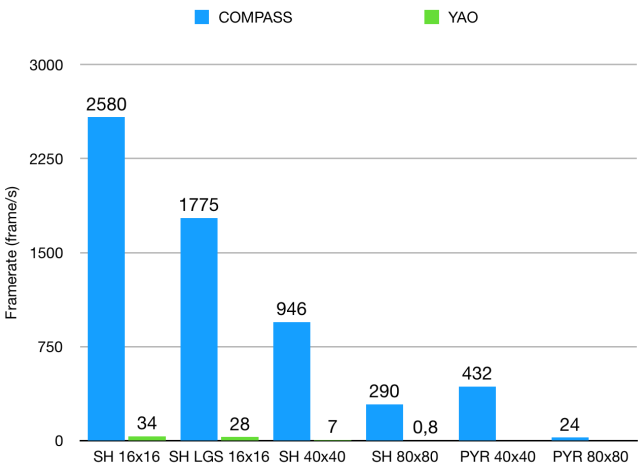

Fig. 6. Framerates obtained with COMPASS for simulation cases defined in Table II. Blue bars are COMPASS framerates, green bars are YAO framerates. Pyramid cases were not executed on YAO.

son with YAO because of the similarities between COMPASS and YAO models. MAOS is another AO simulation software using GPU acceleration and a quick comparison with COMPASS would be interesting, even if used models are different. We have performed MAOS simulations, one similar to the SH 80x80 case, and another based on MCAO system NFIRAOS, which includes 6 LGS WFS and 3 DM, and is designed for the TMT.

For the SH 80x80 case, we measured better performance for COMPASS as MAOS runs the simulation at 50 frames per second (almost 6 times slower). On the contrary, we obtained better performance with MAOS for the MCAO case: COMPASS runs at 60 frames per second when MAOS reaches 250. Rather surprisingly, MAOS is faster in MCAO than SCAO although the complexity is notably higher (several WFS using LGS and several DMs). We infer MAOS may use some approximations to be optimized for such kind of systems, and a comparison with COMPASS models might stand as unfair.

B. Scalability

COMPASS performance is dependent on the GPU used. The SH 80x80 case was benchmarked on the various GPU boards described in Table III. The performance obtained in terms of framerate with each of those cards is shown on Fig. 7. The plotted data shows that COMPASS performance has been

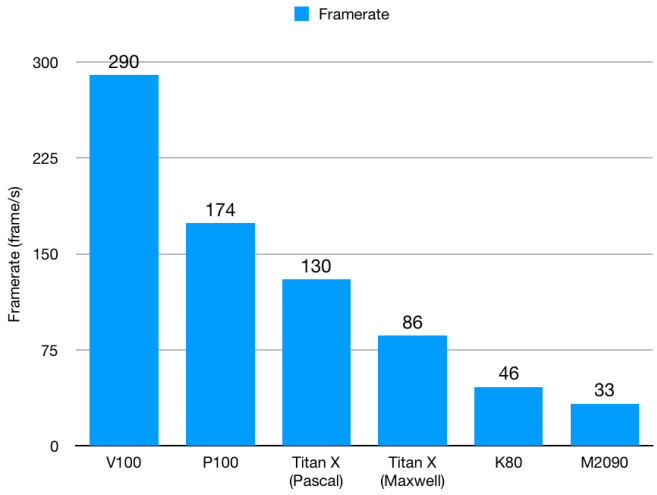

Fig. 7. Framerates obtained with COMPASS for the ELT SCAO scale case, using various Nvidia GPU architectures from Fermi to Volta

multiplied by a factor 5 within 6 years – M2090 had been released in 2011 – without changing the software, but just by using new generations of GPU which are more and more powerful. It also shows that COMPASS optimization is not architecture-dependent, but rather applies to any Nvidia GPU generation from Fermi to Volta, Nvidia's latest. COMPASS performance is therefore expected to keep improving while following performance trends in GPU boards.

We also note that both Pascal cards tested have the same number of CUDA cores, however the P100 has greater memory bandwidth than the Titan X. Results obtained with those cards show that COMPASS is memory bound. COMPASS algorithms require frequent read and write operations on the GPU memory, hence performance improvements with greater memory bandwidth.

C. Profiles

Execution profile is established by measuring the computation time of each function of the AO loop. The profiles for all test cases of Table II are plotted in Fig. 8. For all cases, the profile is dominated by the WFS image computation, more even for pyramid WFS cases where this computation represents more than 90% of the profile, even using 4 GPUs. The top bars shows the profile for SH 16x16 LGS case, showing the impact of using a LGS on the WFS image computation time for a Shack-Hartmann WFS.

VI. Conclusion

We have developed COMPASS, an end-to-end AO simulation that takes advantage of GPU acceleration to overcome the numerical challenges brought by the ELT scale. Our implementation relies on optimized libraries provided by CUDA and MAGMA, and on CUDA kernels specifically developed for this application.

Performance comparisons with a CPU multi-threaded equivalent software, YAO, show speed up factors up to 362, thanks to the state-of-the-art Nvidia Tesla V100 GPU.

Performance measured over the years has also shown that COMPASS is not dependent on the GPU architecture and can be used with any Nvidia GPU card, without changing a single line of code. This compatibility furthermore ensures that the performance will continue to increase with future Nvidia GPU board releases.

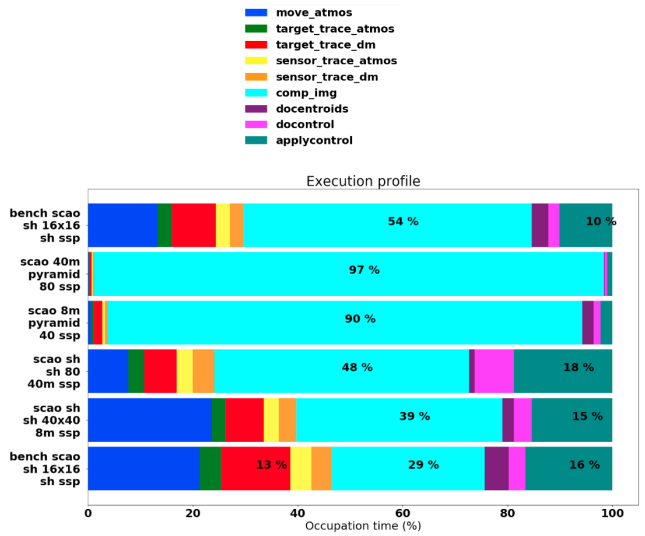

Fig. 8. Execution profiles of COMPASS AO loop for various simulation cases. Top to bottom: SH LGS 16x16, PYR 80x80, PYR 40x40, SH 80x80, SH 40x40, SH 16x16.

TABLE III
TESTED GPU CARDS PROPERTIES

	V100	P100	Titan X	Titan X	K80	M2090
Architecture	Volta	Pascal	Pascal	Maxwell	Kepler	Fermi
# CUDA cores	5,120	3,584	3,584	3,072	2,496	512
Memory bandwidth (GB/s)	760	515	351	255	96	75

VII. FUTURE WORKS

An incoming improvement of COMPASS will include distribution of the SH WFS computation over multiple GPUs in case the simulation requires multiple WFS, such as MCAO or MOAO setups. Currently multiple WFS are handled sequentially as to optimize GPU memory usage. As the WFS image computation is the main contributor in the execution profile, a significant improvement on the performance is expected.

We are also currently working on interfacing modules of COMPASS with prototyping optical benches. As the performances obtained are close to real-time requirements of the AO systems, our objective is to plug the RTC module to drive an AO loop on the bench. Finally, COMPASS is also being used for RTC prototyping by providing WFS images and a virtual DM, with an interface based on openDDS [31].

ACKNOWLEDGMENT

This work is sponsored through a grant from project #671662, a.k.a. Green Flash, funded by European Commission under program H2020-EU.1.2.2 coordinated in H2020-FETHPC-2014.

REFERENCES

[1] F. Roddier, "The effects of atmospheric turbulence in optical astronomy," *Progress in optics. Volume 19. Amsterdam, North-Holland Publishing Co., 1981, p. 281-376.*, vol. 19, pp. 281–376, 1981.

[2] G. Rousset, J. C. Fontanella, P. Kern, P. Gigan, and F. Rigaut, "First diffraction-limited astronomical images with adaptive optics," *A&A*, vol. 230, pp. L29–L32, Apr. 1990.

[3] D. Gratadour *et al.*, "Green FLASH: energy efficient real-time control for AO," in *Adaptive Optics Systems V*, ser. Proc. SPIE, vol. 9909, Jul. 2016, p. 99094I.

[4] S. Ayubian, S. Alawneh, M. Richard, and J. Thij, "Implementation and Performance of a GPU-Based Monte-Carlo Framework for Determining Design Ice Load," Jul. 2017.

[5] CUDA C Programming Guide. Nvidia corporation. [Online]. Available: http://docs.nvidia.com/cuda/cuda-c-programming-guide

[6] CUDA Toolkit documentation. Nvidia corporation. [Online]. Available: https://docs.nvidia.com/cuda/

[7] D. L. Fried, "Optical Resolution Through a Randomly Inhomogeneous Medium for Very Long and Very Short Exposures," *Journal of the Optical Society of America (1917-1983)*, vol. 56, p. 1372, Oct. 1966.

[8] R. G. Lane and M. Tallon, "Wave-front reconstruction using a Shack-Hartmann sensor," *Appl. Opt.*, vol. 31, pp. 6902–6908, Nov. 1992.

[9] R. Ragazzoni, "Pupil plane wavefront sensing with an oscillating prism," *Journal of Modern Optics*, vol. 43, pp. 289–293, Feb. 1996.

[10] C. Kulcsár, H.-F. Raynaud, C. Petit, J.-M. Conan, and P. Viaris de Lesegno, "Optimal control, observers and integrators in adaptive optics," *Optics Express*, vol. 14, p. 7464, Aug. 2006.

[11] E. Thiébaut and M. Tallon, "Fast minimum variance wavefront reconstruction for extremely large telescopes," *Journal of the Optical Society of America A*, vol. 27, p. 1046, Apr. 2010.

[12] F. Vidal, E. Gendron, M. Brangier, A. Sevin, G. Rousset, and Z. Hubert, "Tomography reconstruction using the Learn and Apply algorithm," in *Adaptative Optics for Extremely Large Telescopes*, 2010, p. 07001.

[13] D. L. Fried, "Time-delay-induced mean-square error in adaptive optics." *Journal of the Optical Society of America A*, vol. 7, pp. 1224–1227, Jul. 1990.

[14] J.-M. Conan, "Etude de la correction partielle en optique adaptative," Ph.D. dissertation, 1994, thèse de doctorat dirigée par Léna, Pierre Terre, océan, espace Paris 11 1994. [Online]. Available: http://www.theses.fr/1994PA112450

[15] F. Ferreira, E. Gendron, G. Rousset, and D. Gratadour, "Deriving comprehensive error breakdown for wide field adaptive optics systems using end-to-end simulations," in *Adaptive Optics Systems V*, ser. Proc. SPIE, vol. 9909, Jul. 2016, p. 990979.

[16] N. M. Milton *et al.*, "Commissioning the MMT ground-layer and laser tomography adaptive optics systems," in *Adaptive Optics Systems*, ser. Proc. SPIE, vol. 7015, Jul. 2008, p. 701522.

[17] E. Marchetti *et al.*, "MAD on sky results in star oriented mode," in *Adaptive Optics Systems*, ser. Proc. SPIE, vol. 7015, Jul. 2008, p. 70150F.

[18] E. Gendron *et al.*, "MOAO first on-sky demonstration with CANARY," *A&A*, vol. 529, p. L2, May 2011.

[19] R. Tamai, M. Cirasuolo, J. C. González, B. Koehler, and M. Tuti, "The E-ELT program status," in *Ground-based and Airborne Telescopes VI*, ser. Proc. SPIE, vol. 9906, Jul. 2016, p. 99060W.

[20] Very Large Telescope. ESO. [Online]. Available: http://www.eso.org/public/usa/teles-instr/paranal-observatory/vlt/

[21] L. Schreiber *et al.*, "Dimensioning the MAORY real time computer," in *Adaptive Optics Systems V*, ser. Proc. SPIE, vol. 9909, Jul. 2016, p. 99094L.

[22] F. Rigaut and M. Van Dam, "Simulating Astronomical Adaptive Optics Systems Using Yao," in *Proceedings of the Third AO4ELT Conference*, S. Esposito and L. Fini, Eds., Dec. 2013, p. 18.

[23] R. Conan and C. Correia, "Object-oriented Matlab adaptive optics toolbox," in *Adaptive Optics Systems IV*, ser. Proc. SPIE, vol. 9148, Aug. 2014, p. 91486C.

[24] L. Wang and B. Ellerbroek, "Fast End-to-End Multi-Conjugate AO Simulations Using Graphical Processing Units and the MAOS Simulation Code," in *Proceedings of the Second AO4ELT Conference*, 2011.

[25] J. Atwood, P. Byrnes, and G. Herriot, "NFIRAOS: the optical design of an adaptive optics system for the Thirty Meter Telescope," in *Astronomical and Space Optical Systems*, ser. Proc. SPIE, vol. 7439, Aug. 2009, p. 74390G.

[26] H. Ltaief, D. Gratadour, A. Charara, E. Gendron, and D. E. Keyes, "Adaptive Optics Simulation for the World's Biggest Eye on Multicore Architectures with Multiple GPUs." PASC16, June 2016.

[27] COMPASS. COMPASS team, LESIA. [Online]. Available: https://anr-compass.github.io/compass/

[28] S. Tomov, J. Dongarra, and M. Baboulin, "Towards dense linear algebra for hybrid GPU accelerated manycore systems," *Parallel Computing*, vol. 36, no. 5-6, pp. 232–240, Jun. 2010.

[29] A. G. Basden, R. M. Myers, and E. Gendron, "Wavefront sensing with a brightest pixel selection algorithm," *MNRAS*, vol. 419, pp. 1628–1636, Jan. 2012.

[30] M. Rosensteiner, "Wavefront reconstruction for extremely large telescopes via cure with domain decomposition," *J. Opt. Soc. Am. A*, vol. 29, no. 11, pp. 2328–2336, Nov 2012.

[31] openDDS. [Online]. Available: http://opendds.org/

4.6 Autres fonctionnalités

4.6.1 Base de données

L'initialisation d'une simulation COMPASS permet de calculer les tailles de tableaux nécessaires pour simuler la boucle d'OA sur GPU. Cette opération s'effectuant sur le CPU, elle peut être relativement longue : typiquement une dizaine de minutes pour une simulation ELT.

Afin d'accélérer le processus, notamment pour les campagnes de simulations explorant un espace de paramètres restreint, j'ai intégré un système de base de données. Au démarrage d'une simulation, la base de données est alimentée avec les paramètres utilisateurs de la simulation en cours, et les résultats de la phase d'initialisation (génération de turbulence, géométrie du miroir déformable, matrices d'interaction et de commandes, etc.) correspondante sont sauvegardés sous la forme de fichiers HDF5.

Au lancement d'une nouvelle simulation, la base de données est consultée : si un fichier existant a été généré avec des paramètres de simulation similaires, les données sont pré-chargées au lieu d'être recalculées, ce qui rend l'initialisation beaucoup plus rapide, de l'ordre de quelques secondes pour la même simulation ELT.

4.6.2 Modules autonomes

Les modules GPU incluant les modèles de miroir déformable et de calculateur temps-réel peuvent également fonctionner de manière autonome, en dehors d'ue simulation COMPASS.

Le modèle de miroir déformable peut ainsi permettre de contrôler un dispositif de type SLM [2] afin de modéliser un miroir déformable sur un banc optique par exemple (Deo et al., 2017). Dans cette configuration, le modèle reçoit des commandes venant du calculateur temps-réel de la boucle d'OA, et produit en sortie un écran de phase correspondant à la forme prise par le miroir déformable modélisé.

De la même façon, le module "RTC" de COMPASS, incluant quant à lui le calcul des mesures à partir de l'image d'un ASO et le calcul des commandes du miroir déformable, est également autonome. Il prend alors en entrée une image venant d'un ASO (Shack-Hartmann ou pyramide), et renvoie les mesures et/ou les commandes à envoyer au miroir déformable. Ce module a été testé au sein du projet GreenFlash : les résultats renvoyés par le modèle ont été comparés à d'autres modèles (en Python et en C++) réalisées dans le cadre de ce projet. La Figure 4.15 montre ainsi le résultat de ces comparaisons sous la forme de l'erreur relative entre les pentes et les commandes calculées par le module RTC de COMPASS et un module Python pour une image d'ASO donnée. Cette erreur relative est de l'ordre de la précision numérique pour des nombres flottants codés sur 32 bits.

2. *Spatial Light Modulator* en anglais

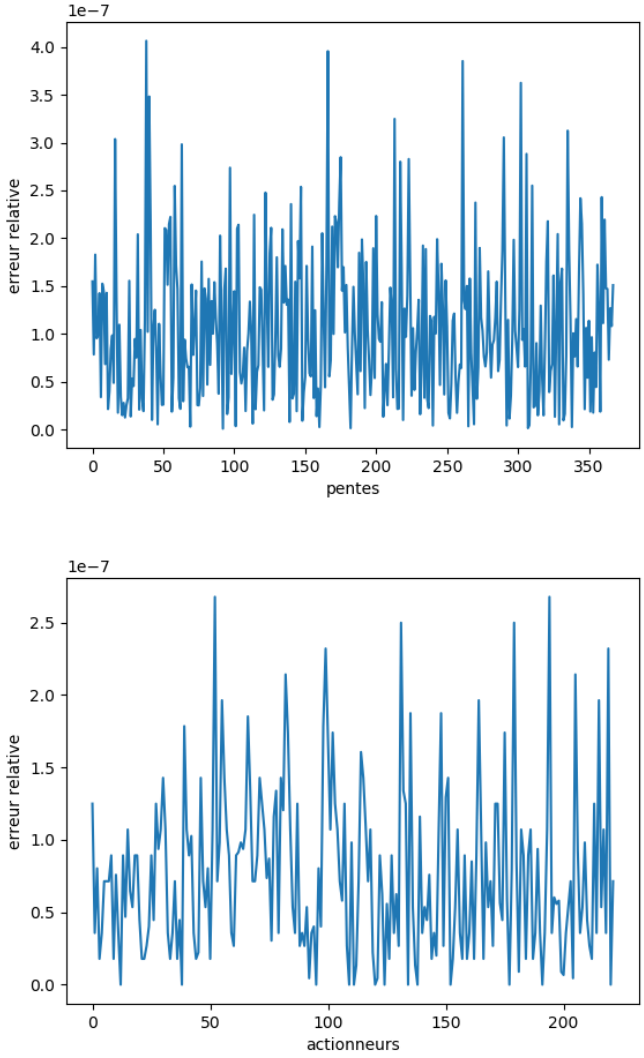

FIGURE 4.15 – Erreur relative entre les pentes (en haut) et les commandes (en bas) calculées par le module RTC de COMPASS et le module Python de test

4.6.3 CANAPASS et ADOPT

CANAPASS est une surcouche logicielle de COMPASS permettant à la simulation de recevoir et d'envoyer des informations depuis le réseau. Le module se base sur la librairie *openDDS*. Cette fonctionnalité permet de lancer une simulation en mode serveur et de communiquer avec elle depuis un processus différents.

ADOPT est une interface Python permettant alors d'interagir avec la simulation à travers le réseau. Dans cette configuration, tout se passe alors comme si l'utilisateur interagissait avec un instrument sur le ciel : envoi d'instructions (matrices de commandes ou d'interaction, contrôle de la boucle d'OA) et télémétrie (mesures, commandes, etc.)

Troisième partie

Estimation et modélisation du budget d'erreur en optique adaptative

Chapitre 5

ROKET : outil d'estimation numérique du budget d'erreur

Avec le développement d'un outil de simulation aussi rapide et polyvalent que COMPASS, l'idée est maintenant d'exploiter cet outil. Si sa première utilisation semble être de mener des campagnes de simulations numériques pour les études de design des futurs instruments des télescopes géants, un autre aspect de mon travail au cours de cette thèse a été de l'utiliser afin d'accéder à une information inaccessible sur le ciel, à savoir une description complète et précise du budget d'erreur du système simulé.

Au cours de ce chapitre, je m'attacherai tout d'abord à montrer l'intérêt d'établir un tel budget d'erreur pour un système d'optique adaptative, puis je décrirai l'outil d'estimation numérique que j'ai développé au sein de COMPASS et les résultats obtenus avec celui-ci.

Sommaire

5.1 Intérêt du budget d'erreur pour la reconstruction de FEP

5.1.1 La reconstruction de FEP

Comme nous l'avons vu dans la Section 2.6, un système d'OA n'est pas parfait, et une erreur résiduelle persiste sur le front d'onde après correction. Ces erreurs influent donc directement sur la FEP de l'instrument, et donc sur la qualité des images obtenues. Au delà de la correction du système d'OA, différentes techniques de traitement d'images peuvent être utilisées a posteriori dans le but d'améliorer la qualité des images obtenues. Néanmoins, ces dernières ne permettront pas d'augmenter la résolution de l'image qui est définie, je le rappelle, par la largeur à mi-hauteur de la FEP, et donc par la qualité de correction du système d'OA.

Une technique de traitement d'image couramment utilisée en optique adaptative est la déconvolution (Conan et al., 1998; Fusco et al., 1999; Mugnier et al., 2004). Comme déjà énoncé avec l'Éq. 1.1, l'image obtenue avec un télescope est le résultat de la convolution entre l'objet observé et la FEP. Ainsi, les méthodes de déconvolution visent à effectuer l'opération inverse afin de gagner en contraste dans l'image. Et cette opération nécessite naturellement de bien connaître la FEP de l'instrument utilisé.

Toutefois, obtenir cette FEP est loin d'être aisé. En effet, mesurer la FEP sur le ciel est contraignant dans la mesure où cela implique de pointer le télescope sur une étoile brillante et d'effectuer une pose suffisamment longue pour obtenir l'image de cet objet ponctuel, i.e. la FEP. Cette méthode est donc synonyme de perte de temps d'observation, puisque le temps passé à mesurer la FEP est donc du temps de perdu en terme d'observation à visée scientifique. D'autre part, la FEP n'est pas constante dans le champ : il faudrait alors effectuer cette mesure en plusieurs points en pointant des objets induisant des conditions de mesures équivalentes (flux, seeing, vent, etc.).

Afin de pallier ce problème, la reconstruction de FEP permet d'estimer cette dernière à partir des données de télémétrie de la boucle d'OA. Je vais résumer l'article de Véran et al. (1997) qui propose ainsi un algorithme de reconstruction basé sur l'utilisation de fonctions particulières, appelées U_{ij}, calculées sur une base modale du miroir déformable. En supposant que la phase est stationnaire sur la pupille, hypothèse vérifiée par Exposito et al. (2012), et que la phase peut être décomposée en un terme appartenant au sous-espace de la base modale et un autre terme orthogonal à ce sous-espace (cf. Éq. 2.22), la fonction de transfert optique monochromatique OTF du système d'OA peut s'écrire comme :

$$\langle OTF(\mathbf{f})\rangle \approx \exp\left(-0.5\bar{D}_{\Phi_{\varepsilon\parallel}}(\boldsymbol{\rho})\right) \times \exp\left(-0.5\bar{D}_{\Phi_{\varepsilon\perp}}(\boldsymbol{\rho})\right) \times OTF_{tel}(\mathbf{f}) \tag{5.1}$$

avec :

- $\mathbf{f}$ la fréquence spatiale angulaire
- $\bar{D}_{\Phi_{\varepsilon\parallel}}(\boldsymbol{\rho})$ la fonction de structure moyenne de la phase résiduelle parallèle, c'est à dire la partie de la phase corrigée qui appartient à l'espace du miroir déformable

- $\bar{D}_{\Phi_{\varepsilon\perp}}(\boldsymbol{\rho})$ la fonction de structure de la phase orthogonale à l'espace du miroir déformable
- $\boldsymbol{\rho} = \lambda \mathbf{f}$ le vecteur de séparation dans le plan pupille
- λ la longueur d'onde d'observation
- OTF_{tel} la fonction de transfert optique du télescope

Cette décomposition est basée sur d'autres hypothèses :

- l'amplitude du champ complexe associé à la phase incidente est constante sur la pupille
- les phases résiduelle parallèle et orthogonale suivent des statistiques gaussiennes
- la fonction de structure de phase est stationnaire sur la pupille
- la corrélation entre les modes du miroirs et la phase orthogonale est négligeable

Il montre ensuite que la fonction de structure moyenne de la phase résiduelle s'écrit sous la forme :

$$\bar{D}_{\Phi_{\varepsilon\parallel}}(\boldsymbol{\rho}) = \sum_{i=1}^{N}\sum_{j=1}^{N}\langle\varepsilon_i\varepsilon_j\rangle U_{ij}(\boldsymbol{\rho}) \tag{5.2}$$

où les fonctions U_{ij} sont définies comme :

$$U_{ij}(\boldsymbol{\rho}) = \frac{\int \left(\mathcal{B}_i(\mathbf{x}) - \mathcal{B}_i(\mathbf{x}+\boldsymbol{\rho})\right)\left(\mathcal{B}_j(\mathbf{x}) - \mathcal{B}_j(\mathbf{x}+\boldsymbol{\rho})\right) P(\mathbf{x})P(\mathbf{x}+\boldsymbol{\rho})\, d\mathbf{x}}{\int P(\mathbf{x})P(\mathbf{x}+\boldsymbol{\rho})\, d\mathbf{x}} \tag{5.3}$$

avec $P(\boldsymbol{\rho})$ la fonction pupille et $\mathcal{B}_i$ le mode i de la base modale.

Ne reste alors plus qu'à estimer la matrice de covariance $\langle\varepsilon_i\varepsilon_j\rangle$, avec ε_i le coefficient de décomposition de la phase résiduelle sur le mode i de la base modale :

$$\Phi_\varepsilon(\mathbf{x}) = \sum_{i=1}^{N}\varepsilon_i\mathcal{B}_i(\mathbf{x}) + \Phi_{\varepsilon\perp}(\mathbf{x}) \tag{5.4}$$

En définissant $\boldsymbol{\varepsilon}$ le vecteur composé de l'ensemble des coefficients ε_i, la matrice $\langle\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^t\rangle$ est alors la matrice de covariance de la phase résiduelle dans le sous-espace du miroir déformable, i.e. la matrice de covariance des erreurs résiduelles du système d'OA.

5.1.2 Estimation du budget d'erreur

Parvenir à estimer cette matrice de covariance de l'erreur résiduelle à partir de la télémétrie n'est pas facile, et les approches varient en fonction du type de système d'OA utilisé (Véran et al., 1997; Harder & Chelli, 2000; Jolissaint et al., 2004; Correia et al., 2011; Exposito et al., 2014; Gendron et al., 2014; Martin et al., 2016).

La difficulté provient (en partie) de la boucle d'OA elle même, tout du moins lorsqu'elle fonctionne en boucle fermée. Dans ce cas, les erreurs effectuées à chaque itération et appliquées sur le miroir déformable sont réinjectées dans la boucle et ont un impact sur les itérations suivantes. D'autre part, les approches citées précédemment font souvent l'hypothèse que les différents contributeurs du budget d'erreur sont statistiquement indépendants les uns des autres. Cependant, cette hypothèse reste difficilement vérifiable.

De ce constat vient l'idée d'utiliser les outils de simulation numérique bout-en-bout pour estimer le budget d'erreur de l'OA et en étudier le comportement. À l'instar de COMPASS, ces outils utilisent une série de modèles physiques afin de calculer la FEP finale du système simulé. En fonction des paramètres donnés à l'initialisation du modèle, le résultat obtenu peut inclure ou exclure certaines sources de bruit. Une méthode évidente pour obtenir un budget d'erreur est alors d'effectuer plusieurs simulations, chacune n'incluant qu'une seule source d'erreur. En "soustrayant" le résultat obtenu avec une simulation de référence sans source d'erreur, on obtient alors une estimation de l'impact du contributeur considéré.

Néanmoins, cette approche possède plusieurs défauts :

- l'obtention d'un unique budget d'erreur requiert les résultats de plusieurs simulations : si l'on considère un instrument ELT, cela peut représenter plusieurs jours de calcul. Et il faut encore multiplier par la dimension de l'espace de paramètres que l'on souhaite explorer
- par nature, cette approche suppose que les sources d'erreurs sont indépendantes statistiquement

L'objectif de l'estimateur de budget d'erreur que j'ai développé dans COMPASS, et que je vais décrire dans le reste du chapitre, est d'obtenir une estimation du budget d'erreur du système simulé en sortie d'une unique simulation, en plus de la FEP obtenue classiquement. L'idée originale est alors d'utiliser les calculs intermédiaires produits par la simulation pour en dériver des estimations de chaque contributeur au budget d'erreur sur les commandes appliquées au miroir déformable, et ce à chaque itération.

5.2 Article A&A

L'article qui suit a été accepté en avril 2018 par le journal Astronomy & Astrophysics. Il est composé de deux parties distinctes. La première à laquelle je vais m'intéresser ici présente ROKET, l'outil d'estimation numérique du budget d'erreur que j'ai développé et intégré dans COMPASS. L'article commence par rappeler les équations décrivant le contenu du budget d'erreur, puis j'y propose une étude sur les corrélations existantes entre les différents contributeurs.

La seconde partie quant à elle porte sur une modélisation des matrices de covariances des erreurs d'anisoplanétisme et temporelle. Je reviendrai sur cette modélisation dans le prochain chapitre, à la Section 6.2.1.

A&A 616, A102 (2018)
https://doi.org/10.1051/0004-6361/201832579

Astronomy & Astrophysics

Numerical estimation of wavefront error breakdown in adaptive optics

F. Ferreira, E. Gendron, G. Rousset, and D. Gratadour

LESIA, Observatoire de Paris, Université PSL, CNRS, Sorbonne Université, Univ. Paris Diderot, Sorbonne Paris Cité, 5 place Jules Janssen, 92195 Meudon, France
e-mail: florian.ferreira@obspm.fr

Received 3 January 2018 / Accepted 24 April 2018

ABSTRACT

Aims. Adaptive optics (AO) system performance is improved using post-processing techniques, such as point spread function (PSF) deconvolution. The PSF estimation involves characterization of the different wavefront (WF) error sources in the AO system. We propose a numerical error breakdown estimation tool that allows studying AO error source behavior such as their correlations. We also propose a new analytical model for anisoplanatism and bandwidth errors that were validated with the error breakdown estimation tool. This model is the first step for a complete AO residual error model that is expressed in deformable mirror space, leading to practical usage such as PSF reconstruction or turbulent parameters identification.
Methods. We have developed in the computing platform for adaptive optics systems (COMPASS) code, which is an end-to-end simulation code using graphics processing units (GPU) acceleration, an estimation tool that provides a comprehensive error breakdown by the outputs of a single simulation run. We derive the various contributors from the end-to-end simulator at each iteration step: this method provides temporal buffers of each contributor. Then, we use this tool to validate a new model of anisoplanatism and bandwidth errors including their correlation. This model is based on a statistical approach that computes the error covariance matrices using structure functions.
Results. On a SPHERE-like system, the comparison between a PSF computed from the error breakdown with a PSF obtained from classical end-to-end simulation shows that the statistics convergence limits converge very well, with a sub-percent difference in terms of Strehl ratio and ensquared energy at $5\frac{\lambda}{D}$ separation. A correlation analysis shows significant correlations between some contributors, especially WF measurement deviation error and bandwidth error due to centroid gain, and the well-known correlation between bandwidth and anisoplanatism errors is also retrieved. The model we propose for the two latter errors shows an SR and EE difference of about one percent compared to the end-to-end simulation, even if some approximations exist.

Key words. instrumentation: adaptive optics – methods: numerical

1. Introduction

Optical aberrations that are due to turbulence have a huge impact on the image resolution of ground-based large telescopes at visible and infrared wavelengths. Without any compensation, the resolution in the visible is equivalent to a diffraction-limited image of a telescope with a few tens of centimeters in diameter. Adaptive optics (AO) systems have been developed for several years in astronomy (Rousset et al. 1990) to compensate for these aberrations: a wavefront sensor (WFS) measures the wavefront deformation, and a deformable mirror (DM) is controlled in real time to flatten the wavefront for the scientific image.

However, the compensation by AO systems is not perfect: there is still a residual wavefront error that has variable impact on the image quality, depending on the system and the observing conditions. Post-processing techniques, such as point spread function (PSF) deconvolution algorithms, have been developed to improve the contrast on the final image (Mugnier et al. 2004). This approach requires estimating the PSF over the scientific field, which involves characterizing the error contributors of the AO system (Véran et al. 1997; Harder & Chelli 2000; Jolissaint et al. 2004; Correia et al. 2011; Gendron et al. 2014; Martin et al. 2016). Another approach is the use of end-to-end system simulation (Gilles et al. 2012). Estimating and distinguishing the various error contributors is a problem because of the propagation and filtering process of the errors in the AO loop (Vidal et al. 2014; Juvenal et al. 2015; Martin et al. 2017).

End-to-end AO simulation tools use a Monte Carlo approach to provide an AO-corrected PSF. It requires several thousand iterations to achieve sufficient convergence on the PSF computation. At the ELT (extremely large telescope) scale, it is particularly demanding in terms of computing power and data flow. Analytical approaches have been developed to provide analytical expression of the PSF (Neichel et al. 2008; Jolissaint 2010; Gendron et al. 2014). If these models do not have convergence problems, they usually require several simplificating assumptions such as stationarity and statistical independence.

In the first part of this paper, we present the development of a new estimation tool, called ROKET (error breakdown estimation tool), which provides a comprehensive error breakdown as an output of a single run of an end-to-end AO simulation (Ferreira et al. 2016). It is developed in the end-to-end graphics processing units (GPU)-based simulation tool COMPASS (computing platform for adaptive optics systems; Gratadour et al. 2014). The novelty of ROKET is that all the error terms are determined frame by frame at each single iteration step of the simulation, instead of only assessing their statistical behavior. This is done by using known internal quantities that are computed by the

simulation to estimate these errors on the fly without interrupting the main AO loop. Thus, this tool provides an error breakdown with no other assumption than those that are made by the end-to-end models, which are usually weaker than the assumptions needed by analytical models. The goal of this tool is to quantify a detailed wavefront error breakdown in an AO system design by numerical simulations. In particular, statistical correlations between contributors can be evaluated using ROKET. The code can be use to study the behavior of error breakdown contributors and to validate or disprove strong assumptions made by an analytical model.

In the second part of this paper, we propose an analytical model for estimating the anisoplanatism and bandwidth error terms, including their correlation. For instance, this model can be used in an a posteriori PSF reconstruction algorithm, complementary to a conventional analytical model as proposed by Véran et al. (1997) or Jolissaint (2010). This analytical model is dependent on a reduced number of parameters that might be identified on AO telemetry data and by turbulence profiling tools on site. The novelty of this model is that it does not rely on a Fourier analysis, but on the error covariance matrices directly expressed in the DM space. In addition, this model is a good candidate for parallel implementation, especially using GPU acceleration, leading to efficient computing that is suitable for ELT scale.

We have realized that some models that assume independence of classical AO error terms (bandwidth, anisoplanatism, etc.) sometimes fail to reproduce the PSF obtained with an end-to-end model. Therefore, it is quite understandable that mismatches of a reconstructed PSF on a real system exist. If progress has to be made in this field, it has to first address the mismatches between reconstructed PSF on simulation data. Our study intends to identify at the simplest and lowest simulation level how the errors are mixed up and what a PSF reconstruction model should take into account to reproduce end-to-end simulated data with a reliable level of accuracy. Our work in this paper intends to pave the way for a future work on PSF reconstruction that is based on telemetry data of real systems, but it remains preliminary to this future work.

After we describe the error breakdown estimation tool in Sect. 2, we discuss the error correlations that could be observed in Sect. 3. In Sect. 4 we propose a pseudo-analytical model for bandwidth and anisoplanatism errors. We also discuss the results obtained with this model and its limitations in this section. The computing performance is summarized in Sect. 5.

2. Error breakdown estimation tool

In this section, we present our error breakdown estimation tool. After recalling the equations that have been derived in Ferreira et al. (2016), we validate its outputs against a classical run of COMPASS. In order to make it as simple as possible in a first step, we consider an AO system with a single wavefront sensor and a single deformable mirror coupled to a tip-tilt mirror both conjugated to ground, also called single conjugated adaptive optics (SCAO) system.

2.1. AO loop error

A wavefront sensor requires a bright guide star (GS): the observed scientific object is sometimes too faint to allow the wavefront measurement. In that case, a guide star is chosen that lies as close as possible to the scientific object in order to limit the anisoplanatism error (Fried 1982). The residual phase $\Phi_\epsilon(\boldsymbol{x},\boldsymbol{\theta})$ seen by the WFS is the turbulent phase in the GS direction $\Phi(\boldsymbol{x},\boldsymbol{\theta})$, corrected by the DM, which adds a phase $\Phi_{\mathrm{DM}}(\boldsymbol{x})$:

$$\Phi_\epsilon(\boldsymbol{x},\boldsymbol{\theta}) = \Phi(\boldsymbol{x},\boldsymbol{\theta}) + \Phi_{\mathrm{DM}}(\boldsymbol{x}), \tag{1}$$

with $\boldsymbol{\theta}$ the GS direction. Hence, we note with $\Phi(\boldsymbol{x},0)$ the turbulent phase in the scientific direction. These quantities can be expressed on a basis $\mathcal{B}$ of the DM:

$$\Phi_\epsilon(\boldsymbol{x},\boldsymbol{\theta}) = \sum_{i=1}^{N} \epsilon_{i,\boldsymbol{\theta}}\, \mathcal{B}_i(\boldsymbol{x}) + \Phi_\perp(\boldsymbol{x},\boldsymbol{\theta}), \tag{2}$$

$$\Phi(\boldsymbol{x},\boldsymbol{\theta}) = \sum_{i=1}^{N} a_{i,\boldsymbol{\theta}}\, \mathcal{B}_i(\boldsymbol{x}) + \Phi_\perp(\boldsymbol{x},\boldsymbol{\theta}), \tag{3}$$

$$\Phi_{\mathrm{DM}}(\boldsymbol{x}) = \sum_{i=1}^{N} v_i\, \mathcal{B}_i(\boldsymbol{x}), \tag{4}$$

where N is the number of commanded modes and $\Phi_\perp(\boldsymbol{x},\boldsymbol{\theta})$ is the component of $\Phi(\boldsymbol{x},\boldsymbol{\theta})$ that is orthogonal to the DM space. We note with $\boldsymbol{\epsilon}$, $\boldsymbol{a}$, and $\boldsymbol{v}$ the vectors composed of the coefficients $\epsilon_{i,\theta}$, $a_{i,\theta,}$ and v_i, respectively. For clarity, we simplify the notation for the scientific direction:

$$\boldsymbol{\epsilon} = \boldsymbol{\epsilon}_{\theta=0}, \tag{5}$$

$$\boldsymbol{a} = \boldsymbol{a}_{\theta=0}. \tag{6}$$

Then, assuming here a linear wavefront sensor, a system with a one-frame delay, and using Eqs. (1)–(4), the measurement vector $\boldsymbol{w}_k$ of the sensor at the iteration k of the AO loop can be written as

$$\boldsymbol{w}_k = D\boldsymbol{a}_{\mathbf{k},\theta} + D\boldsymbol{v}_{\mathbf{k-1}} + \boldsymbol{r}_k + \boldsymbol{n}_k + \boldsymbol{u}_k, \tag{7}$$

where D is he interaction matrix, and $\boldsymbol{n}_k$ is the noise contribution on the measurement vector. $\boldsymbol{u}_k$ covers anything that is not noise and that deviates from the theoretical model of ideal wavefront slope measurements, such as centroid gain and truncation effect. We call this term the wavefront measurement deviation. $\boldsymbol{r}_k$ is the aliasing contribution in the GS direction (Véran et al. 1997; Rigaut et al. 1998):

$$\boldsymbol{r}_k = \mathcal{M}(\Phi^k_\perp(\boldsymbol{x},\boldsymbol{\theta})), \tag{8}$$

with $\mathcal{M}$ the linear operator that describes the ideal wavefront slope measurement from any input phase.

As the system works in closed loop, we assume that the temporal command law used in real time is an integrator with a gain g:

$$\boldsymbol{v}_k = \boldsymbol{v}_{k-1} - gR\boldsymbol{w}_k, \tag{9}$$

where R is the command matrix, which classically computed as the pseudo-inverse of the interaction matrix D.

Then, if we express the residual phase in the scientific direction on the DM basis using Eq. (1) considering a one-frame delay, we obtain a definition of the error without the fitting term:

$$\boldsymbol{\epsilon}_k = \boldsymbol{a}_k + \boldsymbol{v}_{k-1}. \tag{10}$$

Finally, we can derive the error at the iteration k from Eqs. (7), (9), and (10):

$$\begin{aligned}\boldsymbol{\epsilon}_k = {} & (1 - gRD)\boldsymbol{\epsilon}_{k-1} + (\boldsymbol{a}_k - \boldsymbol{a}_{k-1}) \\ & + gRD(\boldsymbol{a}_{k-1} - \boldsymbol{a}_{k-1,\boldsymbol{\theta}}) - gR\boldsymbol{r}_{k-1} - gR\boldsymbol{n}_{k-1} - gR\boldsymbol{u}_{k-1}.\end{aligned} \tag{11}$$

2.2. Error breakdown contributors

Writing Eq. (11) in this way allows us to reveal five contributors to $\boldsymbol{\epsilon}_k$:

$$\boldsymbol{\epsilon}_k = \boldsymbol{\beta}_k + \boldsymbol{\tau}_k + \boldsymbol{\rho}_k + \boldsymbol{\eta}_k + \boldsymbol{\mu}_k, \tag{12}$$

with

$$\boldsymbol{\beta}_k = (1 - gRD)\boldsymbol{\beta}_{k-1} + (\boldsymbol{a}_k - \boldsymbol{a}_{k-1}), \tag{13}$$
$$\boldsymbol{\tau}_k = (1 - gRD)\boldsymbol{\tau}_{k-1} + gRD(\boldsymbol{a}_{k-1} - \boldsymbol{a}_{k-1,\theta}), \tag{14}$$
$$\boldsymbol{\rho}_k = (1 - gRD)\boldsymbol{\rho}_{k-1} - gR\boldsymbol{r}_{k-1}, \tag{15}$$
$$\boldsymbol{\eta}_k = (1 - gRD)\boldsymbol{\eta}_{k-1} - gR\boldsymbol{n}_{k-1}, \tag{16}$$
$$\boldsymbol{\mu}_k = (1 - gRD)\boldsymbol{\mu}_{k-1} - gR\boldsymbol{u}_{k-1}. \tag{17}$$

2.2.1. Bandwidth error β_k

$\boldsymbol{\beta}_k$ is the temporal error that is due to the delay between the time when the command is computed and the time when this command is applied on the DM. During the computation, the turbulent phase is evolving, whereas the DM shape does not. This can be interpreted as the difference between the actual turbulent phase at time k and the command that is applied at the same time. This term only involves the phase in the direction of the scientific object.

2.2.2. Anisoplanatism error τ_k

$\boldsymbol{\tau}_k$ is an anisoplanatism error that is due to the wavefront difference between the scientific direction and the analysis direction. This anisoplanatism error is estimated in the DM space, and is filtered by the temporal response through the AO loop.

2.2.3. Aliasing error ρ_k

$\boldsymbol{\rho}_k$ is the aliasing term: the high frequencies of a turbulent phase are misinterpreted by the WFS and result in a non-null measurement, reconstructed and compensated for by the AO system, introducing an aliased phase component.

2.2.4. Noise measurement error η_k

The noise on the WFS measurements is the term $\boldsymbol{\eta}_k$, which results from both read-out noise and photon noise on the WFS image. Only the low-frequency part of this noise is injected in the command.

2.2.5. WF measurement deviation error μ_k

$\boldsymbol{\mu}_k$ describes anything that deviates from an ideal wavefront slope measurement. An example of it is the truncation effect, which occurs when the field of view of subapertures of a Shack–Hartmann is too small compared to the spot size. Another example is the undersampling error that is due to the extended size of the pixel of a Shack–Hartmann. It can be estimated numerically by the difference between the WFS measurements without noise and the measurements obtained by directly computing the phase gradient of the wavefront.

2.2.6. Fitting and filtered mode error $\Phi_\perp$

The residual phase $\Phi_\perp$ includes a term from the filtered modes and a fitting term. These two contributions are evaluated separately.

The filtered mode error is due to the filtering process used to invert the interaction matrix. It can be estimated easily if the subspace of the modes commanded through the system control matrix is orthogonal to the subspace of filtered ones.

The fitting term is derived as the residual phase that is left after the projection onto the DM basis. As this phase is orthogonal to the DM space, it is not expressed on any DM basis. In order to estimate its impact on the Strehl ratio (SR), we compute its spatial variance at each iteration.

2.3. Modal basis

In order to exploit the error breakdown estimation for a PSF reconstruction as an example, it is more convenient to express the different contributors as a phase variance spread over a modal basis. Such modal variances can be derived from the error terms obtained and the knowledge of the DM basis or influence function used to control it. To handle it, we built a modal basis $\mathcal{B}_{\mathrm{tt}}$ from the influence functions with the following properties:

- the modes are orthonormal;
- in particular, the subspace of the modes commanded through the system control matrix is orthogonal (same scalar product as above) to the subspace of filtered modes;
- the modes are orthogonal to piston and tip-tilt modes;
- two of these modes are pure tip-and-tilt modes.

See Appendix A for details on the computation of this modal basis.

2.4. Validation against end-to-end simulation

As ROKET provides the temporal buffers of each contributor estimated in the DM command space, we are able to compute any covariance matrix between two error terms and the full residual error covariance matrix as

$$\langle \boldsymbol{\epsilon}\boldsymbol{\epsilon}^t \rangle = \langle (\boldsymbol{\beta} + \boldsymbol{\tau} + \boldsymbol{\rho} + \boldsymbol{\eta} + \boldsymbol{\mu})(\boldsymbol{\beta} + \boldsymbol{\tau} + \boldsymbol{\rho} + \boldsymbol{\eta} + \boldsymbol{\mu})^t \rangle. \tag{18}$$

From this matrix, we are able to reconstruct the AO PSF using algorithms developed by Véran et al. (1997) or Gendron et al. (2006). These algorithms provide an estimate of the optical transfer function by computing a structure function from the AO residual error covariance matrix $\langle \boldsymbol{\epsilon}\boldsymbol{\epsilon}^t \rangle$. In this paper, we use the algorithm proposed by Gendron et al. (2006), which uses the so-called V_{ii} functions that can be computed on the fly. This makes it more convenient for a GPU implementation because only a limited amount of memory is available on the device.

Then, we now have two ways of producing a PSF after a simulation run. The built-in way is naturally produced by COMPASS itself as a simulation output. This is noted PSF_C and is regarded as the reference PSF. The other way is computed from ROKET data using a PSF reconstruction algorithm.

We propose to compare the two PSFs and discuss their difference. We have chosen to compare them in terms of maximum value (or SR) and ensquared energy (EE) at a half-width of $5\frac{\lambda}{D}$. The importance of the error contributors and their possible correlations are then studied in detail in Sect. 3.

2.4.1. Simulation parameters

We consider a telescope with a diameter of 8 m without central obstruction. The parameters used in all the simulations are listed in Table 1. It corresponds to a SPHERE-like system (Beuzit et al. 2008). The parameters regarding the atmospheric conditions

Table 1. Simulation parameters.

Telescope parameters		Target parameters	
Diameter	8 m	Wavelength λ_t	1.65 μm
Atmospheric parameters		**WFS parameters**	
Number of layers	12	Number of subapertures	40×40
r_0 (500 nm)	0.16 m	Wavelength λ_{wfs}	0.5 μm
L_0	100 m	Number of pixels per subap.	6
		Pixel size	0.5″
		Photons per subap.	760
		Readout noise	3 e^-
		Centroiding method	Classical CoG
		Guide star coordinates in FoV	(5″, 0″)
AO parameters		**DM parameters**	
Loop frequency	500 Hz	Number of DM actuators	41×41
Command law	Integrator	1 tip-tilt mirror	
Loop gain	0.3	Conjugation altitude	0 m (pupil)
Delay	1 frame		
Frames	20 000		

Table 2. 12-layer turbulent profile.

Alt. [m]	Wind speed (m s^{-1})	Wind dir. (deg)	Frac. of C_n^2
0	13	345	0.261
100	17	68	0.138
200	5	245	0.081
300	17	199	0.064
900	10	181	0.105
1800	10	94	0.096
4500	8	152	0.085
7100	6	185	0.053
11 000	14	265	0.048
12 800	9	116	0.037
14 500	8	6	0.021
16 500	17	272	0.011

are detailed in Table 2. The simulations was run over 20 000 iterations, which is equivalent to 40 s of observation. The sky coordinates and directions are defined in a reference frame where the center is the science target and the X-axis is oriented toward the WFS guide star.

2.4.2. Results

We can directly deduce from the ROKET outputs the conventionnal error breakdown as shown in Table 3. Cross-terms are not included in this table and are detailed in Sect. 3.

The error breakdwon obtained is dominated by the anisoplanatism and bandwidth terms, and it is compliant with common approximations used to estimate the contributors, such as the fitting error estimated from (Hudgin 1977) using $\sigma^2_{\text{fit}} \approx 0.23 \left(\frac{d}{r_0}\right)^{\frac{5}{3}}$ (with d the inter actuator distance), leading here to 46 nm rms. The variance of aliasing, set to 33% of the fitting (Rigaut et al. 1998), gives 31 nm rms.

To validate this error breakdown, we reconstructed the PSF$_R$ by directly computing the residual error covariance matrix from the sum of the error buffers returned by ROKET just as in Eq. (18), and we compared it to the PSF$_C$ simulated by COMPASS. Figure 1 shows these two PSFs and their absolute difference in log scale. Cuts along the X- and Y-axes are shown in Fig. 2.

Table 3. Error breakdown returned by ROKET for the test case.

Contributors	σ (nm rms)
Bandwidth	80
Anisoplanatism	109
Aliasing	31
Noise	11
WF measurement deviation	7.5
Fitting	54
Filtered modes	6

We note the perfect agreement between the two PSFs, with an estimated SR of 73.6% by ROKET compared to 73.8% computed by COMPASS, and an EE of 83.2% compared to 83.3%, respectively. This demonstrates that the estimation of ROKET is accurate, as we can reliably retrieve the PSF from it. Of course, the PSF$_C$ exhibits a speckled aspect because the simulation does not converge, while the PSF$_R$ is built from an averaged phase structure function (Véran et al. 1997) and thus appears smooth and symmetric.

3. Correlation analysis

This section takes advantage of ROKET capacities to highlight the correlations that exist between some of the error breakdown contributors. We also study the conditions that lead to these correlations.

3.1. Correlation between error terms

Developing Eq. (18) leads to 5 main terms that are the covariance matrices of each error breakdown contributor, and to 20 cross-terms that are often neglected in many studies (Jolissaint 2010; Vidal et al. 2014; Martin et al. 2016). A statistical independence between the error contributors is usually assumed, so that

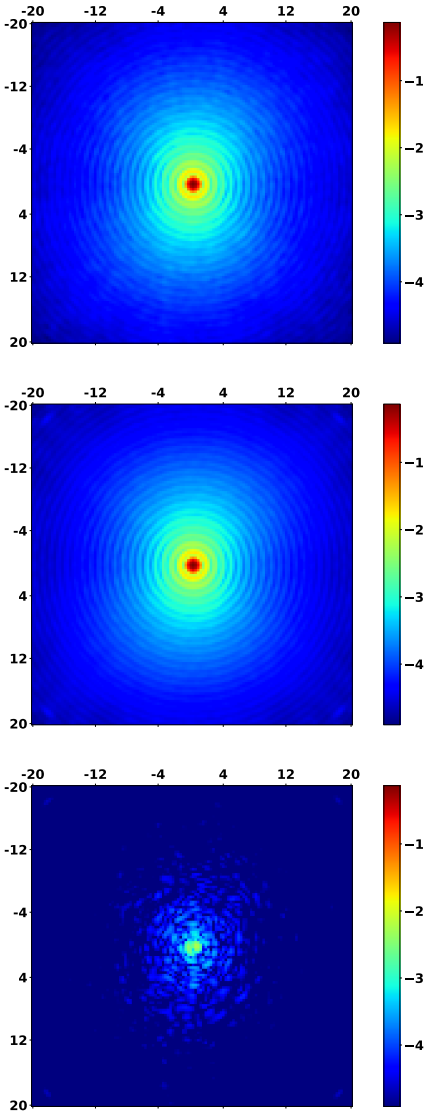

Fig. 1. PSFs in log scale. Axes are expressed in units of $\frac{\lambda}{D}$. *Top panel*: PSF_C simulated by COMPASS. *Middle panel*: PSF_R reconstructed from ROKET buffers. *Bottom panel*: $\|PS\,F_C - PS\,F_R\|$.

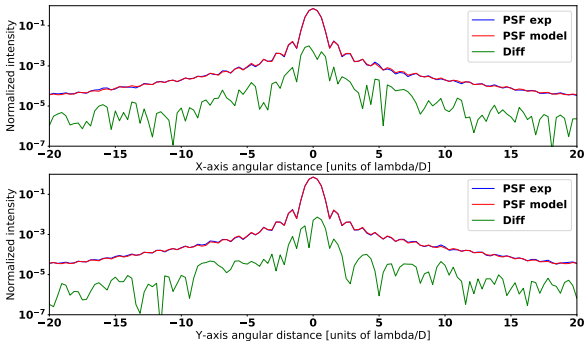

Fig. 2. Cuts of PSF_C in red and PSF_R in blue. The green curve is $\|PS\,F_C - PS\,F_R\|$. *Top panel*: cut along X-axis. *Bottom panel*: cuts along Y-axis.

Eq. (18) simplifies to

$$\langle \boldsymbol{\epsilon}\boldsymbol{\epsilon}^t \rangle = \langle \boldsymbol{\beta}\boldsymbol{\beta}^t \rangle + \langle \boldsymbol{\tau}\boldsymbol{\tau}^t \rangle + \langle \boldsymbol{\eta}\boldsymbol{\eta}^t \rangle + \langle \boldsymbol{\mu}\boldsymbol{\mu}^t \rangle + \langle \boldsymbol{\rho}\boldsymbol{\rho}^t \rangle, \tag{19}$$

which is, in many cases, a meaningful hypothesis.

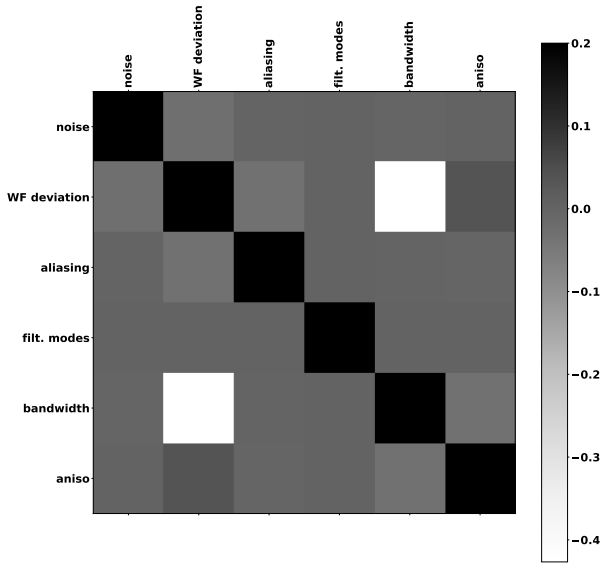

Fig. 3. Correlation matrix between the error contributors. The color scale has been changed to highlight small cross-correlations. The diagonal is still equal to 1.

However, we cannot infer with certainty that this assumption is always valid. Obviously, some contributors of the error breakdown are effectively independent (such as noise with respect to turbulent terms), but we have no guarantee that turbulence-related terms are independent among themselves. This assumption could have an impact in the PSF estimation as it directly affects the error covariance matrix. With ROKET, we will be able to assess the validity of assuming that the error term is statistically independent, and determine what to do if this is not true.

We are able to compute all covariance terms, including cross-terms, from the temporal buffers of each error contributor returned by ROKET. Figure 3 shows the correlation coefficients obtained between all error contributors for the simulated case described in Sect. 2.4.1. They are displayed in matrix form, with lines and rows associated with a given contributor. Thus, for two contributors x and y, the value of the cell that intersects line x and column y is the correlation coefficient between these terms. We note a strong anticorrelation of −0.43 between the WF measurement deviation and bandwidth. Other non-null correlations are a WF measurement deviation with aliasing (−0.03), anisoplanatism (0.03), noise (−0.02), and also bandwidth and anisoplanatism (0.03). The WF measurement deviation affects the measurements of the low-order modes, so that it is naturally correlated to the temporal and the aliasing error, which are both made of low orders. ROKET shows a correlation between WF measurement deviation and noise. The Shack–Hartmann image profile affects the centroiding noise $\boldsymbol{n}_k$ and also the centroid gain, which is part of the deviation term. Consequently, it is impossible to argue that these terms cannot be correlated, as they both come from the centroiding process of the image.

3.2. WF measurement deviation and bandwidth correlation

The correlation between deviation and bandwidth is first due to a centroid gain that can be modeled by a multiplicative scalar factor γ on the WFS measurement. By construction of ROKET, the effect of the centroid gain is counted as part of $\boldsymbol{u}_k$ – an odd behavior. On an other hand, the effect of a centroid gain is often regarded as the equivalent to an alteration of the loop gain

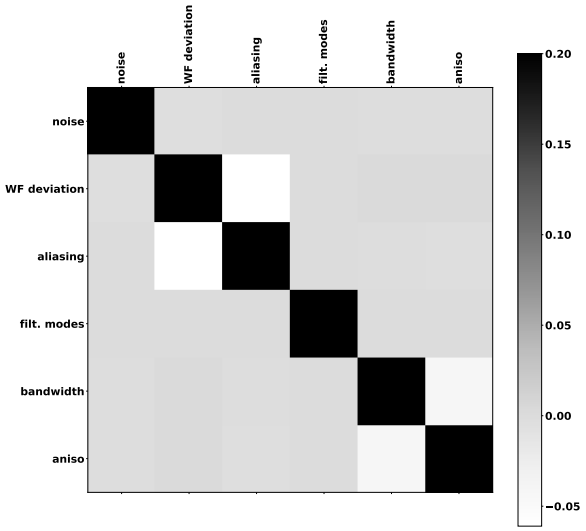

Fig. 4. Correlation matrix between the error contributors by taking into account the centroid gain. The color scale has been changed to highlight the vanishing of the deviation and bandwidth cross-correlation. The diagonal is still equal to 1.

and is expected to primarily affect the bandwidth term, and to a lesser extent the propagation of noise, aliasing, and other terms (Véran & Herriot 2000). It would make more sense to identify this centroid gain as such to establish a new error breakdown. We have modified the wavefront measurement models described in Ferreira et al. (2016) for this purpose. For the sake of clarity, the new equations are presented in Appendix B, Eq. (B.4). We highlight one point: the summation of the five equations obtained in Eq. (B.4) will always result in the same $\boldsymbol{\epsilon}_k$, by definition. If the decomposition is affected by γ, the total remains unchanged because the AO system is not retuned with respect to γ. As a result, different decompositions using different gamma will lead to exactly the same PSF. The γ parameter only changes the breakdown by transferring variances and covariances from one item to another. Now, using the particular value of γ that matches the system centroid gain has some particular property: it causes the covariances to vanish.

Of course, a new simulation run needs to be performed because the centroid gain γ also needs to be applied to the phase gradient WFS model. To compute the correct value of γ, ROKET performs a linear regression between the WFS measurement without noise and the phase gradient model measurement, in a first run. The value of γ is then estimated as the average of the regression coefficients over the iterations. Finally, we need to launch a new ROKET simulation run with this γ value in order to compute the error breakdown as given in Appendix B.

Figure 4 shows the correlation coefficients obtained with these equations. We note that the WF measurement deviation term is significantly reduced and the correlation between WF measurement deviation and bandwidth has been zeroed. We also note the disappearance of the correlation between WF measurement deviation and noise, which was then effectively due to the existence of this centroid gain. A correlation remains with aliasing. Introducing this centroid gain γ in the equations has moved variances and covariances around, as shown in Table 4. It is worth noticing that the error breakdown returned by ROKET remains valid in any case, since the sum of the time buffers remains unchanged. Similarly, the sum of all elements of the covariance tables in Figs. 3 and 4 leads to the same total variance, regardless of the value of γ. Taking γ into account is only a modification of the wavefront measurement models, leading to a new error breakdown.

Table 4. Variances and covariances obtained with $\gamma = 1$ (i.e., without taking the centroid gain into account) and with $\gamma = 0.95$ (value computed after the first run of ROKET).

γ	σ^2_η	σ^2_μ	σ^2_ρ	σ^2_β	σ^2_τ	$\sigma^2_{\mu\beta}$
1	0.114	0.057	0.948	6.43	11.8	−0.258
0.95	0.109	0.036	0.908	5.93	12.0	0

Notes. σ^2_η is the noise variance, σ^2_μ is the WF measurement deviation variance, σ^2_ρ is the aliasing variance, σ^2_β is the bandwidth variance, σ^2_τ is the anisoplanatism variance, and $\sigma^2_{\mu\beta}$ is the covariance between the WF measurement deviation and bandwidth. All are expressed in $10^{-3}\mu\text{m}^2$.

3.3. Anisoplanatism and bandwidth correlation

In most cases, bandwidth and anisoplanatism are the main contributors of the error breakdown of a SCAO system. Even if the correlation between anisoplanatism and bandwidth errors appears to be low, it is well known that it strongly depends on wind conditions. Under the frozen-flow assumption (Taylor 1938), temporal fluctuations of the turbulence can be expressed as spatial fluctuations (Greenwood 1977). Thus, we cannot neglect this correlation, especially in layer-by-layer error modeling as in Jolissaint et al. (2006).

To illustrate this purpose and prepare further model validation, we ran 60 ROKET simulations with the same conditions as described in Sect. 2.4.1, but with only a single layer at 5 km altitude and various wind conditions: a windspeed between 5 and 20 m s^{-1}, and a wind direction between 0° and 180°. We compare then the PSF$_R$ computed by ROKET and the PSF$_I$ computed by neglecting anisoplanatism and bandwidth correlation. The comparison is presented in terms of SR and EE at $\pm 5\frac{\lambda}{D}$. Table 5 summarizes the results obtained for each wind direction in terms of relative error on the SR and the EE. As expected, the anisoplanatism and bandwidth correlation depends on the wind direction. It is strongest when the wind is aligned with the off-axis GS direction. Neglecting it in this case leads to huge errors in the PSF estimate. Conversely, the correlation is negligible when the wind direction is orthogonal to that of the GS.

4. Bandwidth and anisoplanatism models

This work prepares some tools for PSF reconstruction based on the telemetry data of a real system. We propose an analytical model of the covariance matrix between bandwidth and anisoplanatism to be retrieved from these telemetry and turbulence profiling tools. Similar models already exist that aim at PSF computation (Jolissaint et al. 2006; Neichel et al. 2008). In this section, we propose a different approach that does not rely on a Fourier analysis, but on an expression of covariance matrices of the anisoplanatism and bandwidth errors, including their possible correlation, directly in the DM space. Applications and limitations of this approach are discussed.

4.1. Definitions

We define the bandwidth error as the temporal error due to the delay between the time when the wavefront is measured and the time when this command is applied on the DM. Then, we write the bandwidth error ϵ_b as a phase difference in the

Table 5. Mean and maximum relative error between PSF_I compared to PSF_C over the 60 simulation runs, in terms of SR and EE in $\pm 5\frac{\lambda}{D}$ for five values of the wind direction.

Wind direction	Aniso/bandwidth correlation	SR relative error		EE relative error	
		mean	max	mean	max
0	0.93	46%	84%	12%	14%
45	0.63	26%	47%	7%	8%
90	0.01	0.4%	2%	0.3%	0.4%
135	−0.66	24%	38%	8%	10%
180	−0.95	31%	48%	12%	14%

direction of the target between two moments that are separated by some delay τ':

$$\epsilon_b = \Phi_{\parallel}(\boldsymbol{x}, \boldsymbol{0}, t+\tau') - \Phi_{\parallel}(\boldsymbol{x}, \boldsymbol{0}, t). \tag{20}$$

We find the value of τ' assuming that we can approximate the rejection transfer function of the AO loop for one-frame delay $H(p) = \frac{1}{1+g\frac{F_s}{p}e^{-\tau p}}$ (Demerle et al. 1994) by the transfer function of substraction with pure delay $H'(p) = 1 - e^{-\tau' p}$. The variable p is the Laplace variable, F_s is the sampling frequency of the AO system, and $\tau = 1/F_s$ is the sampling period. Considering the Taylor series of both expressions around $p = 0$ at first order, they will match for the value

$$\tau' = \frac{\tau}{g}\,. \tag{21}$$

The validity of this assumption is discussed later. In a real AO system, we should include the filtering by the exposure time on the WFS and by the blocker of the DM command. In any case, however, our pure delay identification procedure can still be applied.

We define the anisoplanatism error as the phase difference between wavefront sensor and the target directions:

$$\epsilon_a = \Phi_{\parallel}(\boldsymbol{x}, \boldsymbol{\theta}, t) - \Phi_{\parallel}(\boldsymbol{x}, \boldsymbol{0}, t), \tag{22}$$

where $\boldsymbol{\theta}$ is the angular separation between the WFS guide star and the target.

Now, we assume the case of a phase in a single layer. The multi-layers case will come easily under the frozen flow assumption. This will allow us to transform angular coordinates of observation into spatial coordinates at the layer surface, knowing the turbulent layer altitude h. We also assume a frozen flow hypothesis, which allows us to turn time into space using the wind velocity vector $\boldsymbol{v}$. Equations (22) and (20) both can be rewritten in the target direction:

$$\epsilon_a = \Phi_{\parallel}(\boldsymbol{x} + h\boldsymbol{\theta}) - \Phi_{\parallel}(\boldsymbol{x}), \tag{23}$$
$$\epsilon_b = \Phi_{\parallel}(\boldsymbol{x} - \boldsymbol{v}\tau') - \Phi_{\parallel}(\boldsymbol{x}). \tag{24}$$

We emphasize that in both definitions, we consider only the phase $\Phi_{\parallel}$ that belongs to the DM modal space.

4.2. Covariance matrix model

From the previous definitions, we wish to derive an expression of the covariance matrix C_{bb} of the bandwidth error, of the covariance matrix C_{aa} of the anisoplanatism error, and of the covariance matrix C_{ba} between these errors. We first compute these covariance matrices on the DM actuators. Components (i, j) of the covariance matrix C_{ba} can be written as

$$C_{\mathrm{ba}}(i,j) = \langle (\Phi_{\parallel}(\boldsymbol{x_i} + h\boldsymbol{\theta}) - \Phi_{\parallel}(\boldsymbol{x_i}))(\Phi_{\parallel}(\boldsymbol{x_j} - \boldsymbol{v}\tau') - \Phi_{\parallel}(\boldsymbol{x_j}))\rangle, \tag{25}$$

where $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ are the position vectors in the pupil of the DM actuator numbers i and j, respectively. Using the identity

$$(A-a)(B-b) = \frac{1}{2}\left(-(A-B)^2 + (A-b)^2 \;\; +(a-B)^2 - (a-b)^2\right) \tag{26}$$

leads to

$$\begin{aligned} C_{\mathrm{ba}}(i,j) = \frac{1}{2}\Big(&-\langle(\Phi_{\parallel}(\boldsymbol{x_i} + h\boldsymbol{\theta}) - \Phi_{\parallel}(\boldsymbol{x_j} - \boldsymbol{v}\tau'))^2\rangle \\ &+\langle(\Phi_{\parallel}(\boldsymbol{x_i} + h\boldsymbol{\theta}) - \Phi_{\parallel}(\boldsymbol{x_j}))^2\rangle \\ &+\langle(\Phi_{\parallel}(\boldsymbol{x_i}) - \Phi_{\parallel}(\boldsymbol{x_j} - \boldsymbol{v}\tau'))^2\rangle - \langle(\Phi_{\parallel}(\boldsymbol{x_i}) - \Phi_{\parallel}(\boldsymbol{x_j}))^2\rangle\Big). \end{aligned} \tag{27}$$

We introduce $D_{\phi}^{\mathrm{low}}(\boldsymbol{r})$, the structure function of the turbulent phase restricted to the spatial frequencies of the DM space. We can write it as

$$D_{\phi}^{\mathrm{low}}(\boldsymbol{r}) = \langle(\Phi_{\parallel}(\boldsymbol{x} + \boldsymbol{r}) - \Phi_{\parallel}(\boldsymbol{x}))^2\rangle. \tag{28}$$

Then, considering the separation vector $\boldsymbol{x_{ij}} = \boldsymbol{x_j} - \boldsymbol{x_i}$ between actuators i and j, Eq. (27) can be written as

$$\begin{aligned} C_{\mathrm{ba}}(i,j) = \frac{1}{2}\Big(&-D_{\phi}^{\mathrm{low}}(\boldsymbol{x_{ij}} - h\boldsymbol{\theta} - \boldsymbol{v}\tau') + D_{\phi}^{\mathrm{low}}(\boldsymbol{x_{ij}} - h\boldsymbol{\theta}) \\ &+D_{\phi}^{\mathrm{low}}(\boldsymbol{x_{ij}} - \boldsymbol{v}\tau') - D_{\phi}^{\mathrm{low}}(\boldsymbol{x_{ij}})\Big). \end{aligned} \tag{29}$$

The same approach stands for covariance matrices C_{aa} and C_{bb} and leads to the very similar expressions:

$$C_{\mathrm{aa}}(i,j) = \frac{1}{2}\left(D_{\phi}^{\mathrm{low}}(\boldsymbol{x_{ij}} + h\boldsymbol{\theta}) + D_{\phi}^{\mathrm{low}}(\boldsymbol{x_{ij}} - h\boldsymbol{\theta}) - 2D_{\phi}^{\mathrm{low}}(\boldsymbol{x_{ij}})\right), \tag{30}$$

$$C_{\mathrm{bb}}(i,j) = \frac{1}{2}\left(D_{\phi}^{\mathrm{low}}(\boldsymbol{x_{ij}} + \boldsymbol{v}\tau') + D_{\phi}^{\mathrm{low}}(\boldsymbol{x_{ij}} - \boldsymbol{v}\tau') - 2D_{\phi}^{\mathrm{low}}(\boldsymbol{x_{ij}})\right). \tag{31}$$

These expressions are simple and fully analytical as we are able to compute the structure function $D_{\phi}^{\mathrm{low}}(\boldsymbol{r})$. We define f_c, and the Nyquist spatial frequency of the DM equals to $\frac{1}{2d}$, where d is the actuator pitch. Using a Fourier approach, this structure function limited to the spatial frequencies lower than f_c can be written as

$$\begin{aligned} D_{\phi}^{\mathrm{low}}(r) = D_{\phi}(r, L_0) - 2(2\pi)^{8/3}0.023&\left(\frac{r}{r_0}\right)^{5/3} \\ &\times\left(1.11833 - \int_0^{2\pi f_c r} u^{-8/3}(1 - J_0(u))\,\mathrm{d}u\right). \end{aligned} \tag{32}$$

The $D_{\phi}(r, L_0)$ expression can be found in Conan (2000). This structure function is an approximation that considers circular

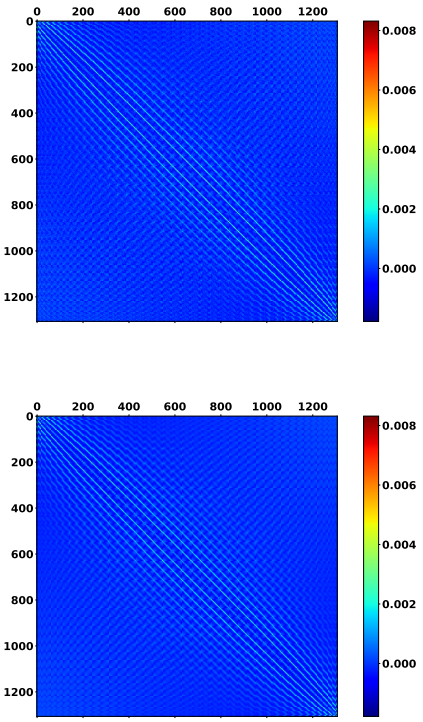

Fig. 5. Covariance matrices of the global contribution of the anisoplanatism and bandwidth errors. Both matrix diagonals have been nullified to emphasized side structures. *Top panel*: covariance matrix computed from ROKET. *Bottom panel*: covariance matrix model $C_{\text{ee.}}$

symmetry and not the actual actuator pattern. The detailed calculation and computation methods are reported in Appendix C. Then, we obtain the full contribution of the anisoplanatism and bandwidth errors by adding these covariance matrices:

$$C_{\text{ee}}^{\Phi} = C_{\text{bb}} + C_{\text{aa}} + C_{\text{ba}} + C_{\text{ba}}^{t}. \tag{33}$$

The covariance matrix of a complex, multi-layer profile can be obtained by summing all the single-layer matrices together.

Finally, we note with C_{ee} the covariance matrix expressed in the actuator space and filtered from piston and tip-tilt modes. The removed tip-tilt component is projected on the tip-tilt mirror actuators.

This model only needs a few parameters from the atmospheric turbulence (L_0, r_0, turbulent profile, and wind) and from the AO system (loop frequency, actuator positions, and guide star position).

4.3. Model result

To assess the validity of our model, we generated PSFs following two methods. The first method used the error buffers from ROKET of bandwidth and anisoplanatism, which are summed together and converted into a covariance matrix that will feed the PSF reconstruction algorithm. The second method used Eq. (33) to compute the C_{ee}, which also feeds the PSF reconstruction algorithm.

Figure 5 shows both covariance matrices expressed in the DM actuators space. We note that the model reproduces the features of the covariance matrix well. Figure 6 shows the PSFs obtained through the two methods and their difference. Cuts of these PSFs along the X- and Y-axes are shown in Fig. 7.

We note a good accuracy in the PSF reconstruction up to $5\frac{\lambda}{D}$, with a maximum intensity at 0.78 compared to 0.77, and an EE

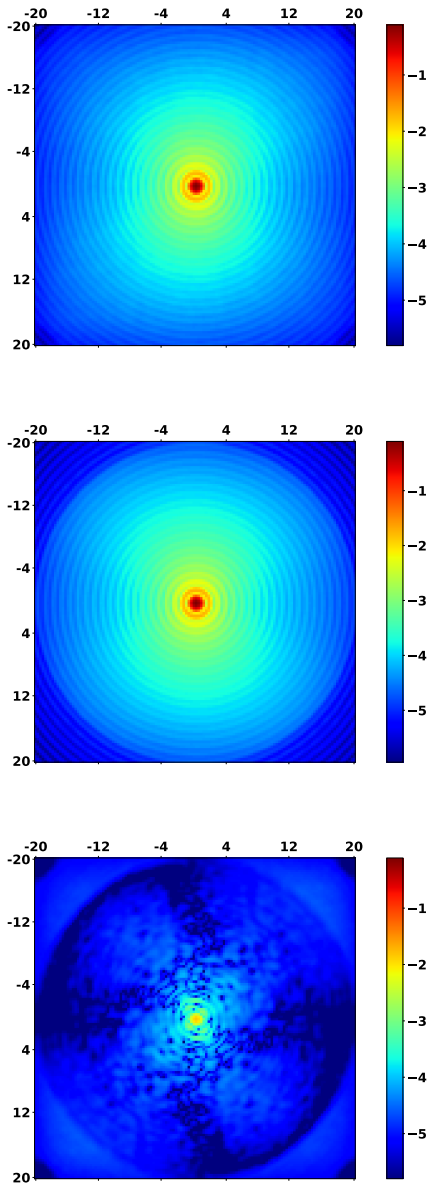

Fig. 6. *Top panel*: PSF obtained from ROKET buffers of anisoplanatism and bandwidth errors. *Middle panel*: PSF obtained from $C_{\epsilon\epsilon}$. *Bottom panel*: absolute difference between the two PSFs. The actuator pattern is a square array. All figures are in log scale, and axes are expressed in units of $\frac{\lambda}{D}$.

at $\pm 5\frac{\lambda}{D}$ that equals 87.5% compared to 86.8%. Then, the PSF estimation at distances farther than $10\frac{\lambda}{D}$ is less accurate.

4.4. Model limitations

In this section, we apply the model we developed to the 60 simulations used in Sect. 3.3. The results obtained highlight some limitations of the model that we explain below.

We now apply the same methods as we used in Sect. 4.3 to compute PSFs for the same 60 simulation cases. Figure 8 shows the SR obtained from a PSF reconstructed from $C_{\epsilon\epsilon}$ versus the one obtained from PSF reconstructed from ROKET buffers of anisoplanatism and bandwidth errors. Globally, the faster the wind and the lower the loop gain, the less efficient the model. This is explained by the approximation we made in Sect. 4.1 concerning the rejection transfer function. This assumption is made

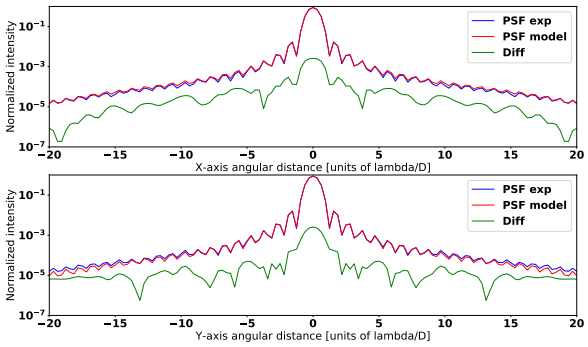

Fig. 7. Cuts of the PSFs in log scale. *Top panel*: along the X-axis. *Bottom panel*: along the Y-axis. Blue curves: PSF reconstructed from ROKET buffers. Red curves: PSF reconstructed from the model. Green curves: Absolute difference between the two PSFs.

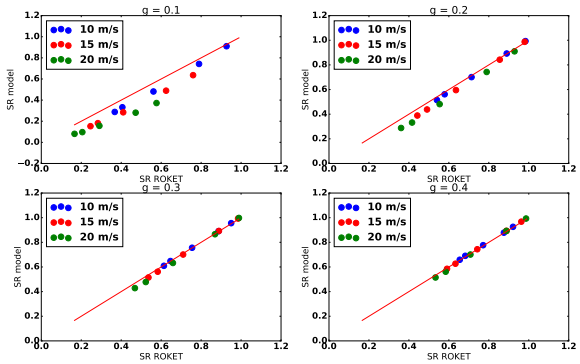

Fig. 8. SR of PSF reconstructed from the model vs. SR of PSF reconstructed from ROKET buffers. *Top left panel*: loop gain of 0.1. *Top right panel*: loop gain of 0.2. *Bottom left panel*: loop gain of 0.3. *Bottom right panel*: loop gain of 0.4. Blue points: wind speed of 10 m s^{-1}. Green: 15 m s^{-1}. Red: 20 m s^{-1}. The red line is $y = x$.

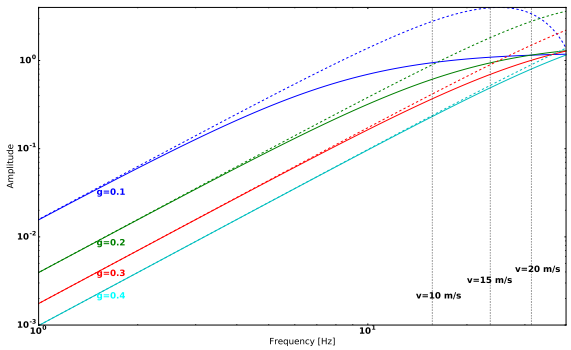

Fig. 9. Square modulus of the rejection transfer functionand the delay transfer function. Solid line: rejection transfer function, dashed line: delay transfer function, black dot lines: turbulence cut-off frequency depending on the wind speed.

to obtain a simple model that could be computed fast and accurate in most cases, as shown in Sect. 4. However, the validity of this assumption depends on the loop gain, the loop frequency, and the wind speed. Figure 9 compares the modulus of the rejection transfer function and the modulus of the delay transfer function for various gains. The higher the gain, the closer the transfer functions, and so the more valid the assumption. Moreover, the approximation appears to be better for low frequencies than for high frequencies. As the cut-off frequency of the turbulence is given by $f_c \approx 0.3\frac{v}{d}$ (Conan et al. 1995), where d is

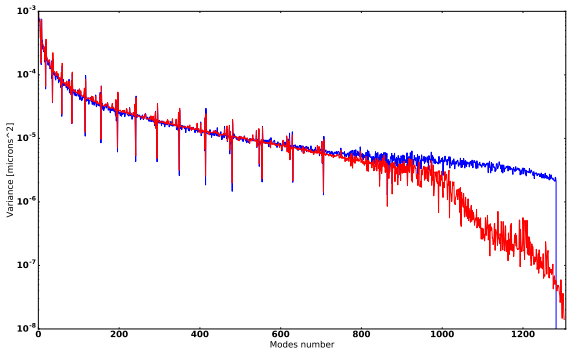

Fig. 10. Diagonal of the error covariance matrix. Blue curve: covariance computed from ROKET. Red curve: covariance computed from the model.

the subaperture diameter, the approximation, and so the model that we have developed, is better for low wind speed. The results obtained with Fig. 8 confirm this behavior.

It also leads to a poor estimation of the high-frequency components of the error. Figure 10 shows the diagonal of the covariance matrix $C_{\epsilon\epsilon}$ in the modal basis space, compared to the covariance matrix computed from ROKET. The model is accurate for modes from 0 to 800, then the covariance for higher modes is underestimated.

However, the results obtained with the model encourage us to keep the assumptions we made. For most cases, the accuracy is relevant and the computing performance obtained with this model is promising.

4.5. Model applications

As described above, models for anisoplanatism and bandwidth errors are available that use a power spectral density analysis of the AO residual phase (Jolissaint et al. 2006). The model that we propose is based on a slightly different approach that will be extended in future works to model the other error breakdown contributors. It will provide another method for estimating the PSF of an AO system. It will allow quick first-order AO performance evaluation even at the ELT scale.

Moreover, this approach can easily take into account the DM actuator geometry such as the hexagonal pattern and the shape of the influence functions within the limit of a modal basis that is restricted to a circular frequency domain. It could also be used in a PSF reconstruction algorithm. It might also be useful in practical cases since it is based on an estimation of covariance matrices in the DM actuator space, which is a space available from AO loop telemetry. For example, a possible application could be the identification of turbulent parameters from AO loop data using a method similar to the one developed by Vidal et al. (2010) based on a fitting algorithm that minimizes the difference between the modeled covariance matrix with the matrix computed from telemetry data. Thus, retrieving the turbulence parameters from an SCAO system will be possible.

5. Computing performance

The simulations presented in this paper have been run on a Nvidia DGX-1 server, with Dual 20-core Intel Xeon E5-2698 v4 @ 2.2 Ghz and 8 Tesla V100-SXM2 GPUs. The GPU acceleration enables COMPASS to run the 12-layer simulation, without the error estimation feature, at approximately 330 frames per second. Performing the error breakdown estimation, the

computation speed decreases by a factor 5 at 60 frames per second.

At the ELT scale, COMPASS is able to run a 35-layer SCAO simulation at 73 frames per second, and the error estimation is performed at 15 frames per second.

We have also developed a multi-GPU module to perform PSF reconstruction using the algorithm proposed by Gendron et al. (2006). This implementation allows reconstructing a 4096 by 4096 ELT PSF on 5000 modes in 35 s on two GPUs.

The covariance matrix model described in Sect. 4 is highly parallelizable as each element of the matrix can be computed independently of the other elements. Hence, we have developed a GPU module dedicated to the computation of those matrices that lead to a computation of an ELT scale covariance matrix 5000 by 5000 over 35 layers in half a second.

6. Conclusion and perspective

We have developed a numerical tool, called ROKET, which is included in the COMPASS AO simulation code. It produces a comprehensive breakdown of the wavefront error for an AO loop. The estimated contributors are the temporal error, the anisoplanatism, the aliasing, the WF measurement deviation including centroid gain, the fitting, and the filtered mode error. The tool allows us to evaluate not only the variances, but the possible correlations between the different contributors to the error breakdown.

Our results show that the error breakdown estimation is accurate and fast. The PSF, reconstructed from a ROKET error breakdown, is very similar to the empirical long-exposure PSF computed during the simulation run by COMPASS. The computed error breakdowns show that correlations exist between WF measurement deviation, noise, aliasing, temporal error, and anisoplanatism. We focused on the bandwidth and anisoplanatism errors, which are two main contributors in off-axis WFS. Their correlation often cannot be neglected.

We have developed an analytical model of these two main contributors and their correlation. It allows us to compute the resulting error covariance matrix that can be directly used by classical PSF reconstruction algorithms. Our results show that this model is accurate, even if the accuracy reached for the highest order controlled modes is lower than the one for low-order modes.

This model only requires a few parameters of the AO system and of the atmospheric conditions to compute the covariance matrix. The novelty compared to other analytical models is that it is expressed directly on the DM actuator space, allowing then convenient implementation for post-facto PSF reconstruction, for instance. In addition, the computing efficiency can be greatly improved by a GPU implementation, which allows easily handling an ELT scale.

Future works will use ROKET as a reference tool to develop the same type of models for the other error breakdown contributors. With such models, fast and accurate PSF estimation of an AO system will be possible, even at the ELT scale. We plan to use ROKET for the validation by simulation of a turbulent pro- file identification tool from SCAO telemetry data. Another step will be to upscale ROKET and the covariance model for MCAO or MOAO systems.

Acknowledgements. This work is sponsored through a grant from project #671662, a.k.a. Green Flash, funded by European Commission under program H2020-EU.1.2.2 coordinated in H2020-FETHPC-2014. We also wish to thank the anonymous referee for their meaningful comments on this article.

References

Beuzit, J.-L., Feldt, M., Dohlen, K., et al. 2008, Ground-based and Airborne Instrumentation for Astronomy II, Proc. SPIE, 7014, 701418
Conan, R. 2000, PhD Thesis, Sciences de l'univers Nice
Conan, J.-M., Rousset, G., & Madec, P.-Y. 1995, J. Opt. Soc. Am. A, 12, 1559
Correia, C., Véran, J.-P., Ellerbroek, B., Gilles, L., & Wang, L. 2011, Second International Conference on Adaptive Optics for Extremely Large Telescopes, 70
Demerle, M., Madec, P. Y., & Rousset, G. 1994, in NATO Advanced Science Institutes (ASI) Series C, eds. D. M., Alloin, & J. M., Mariotti, 423, 73
Ferreira, F., Gendron, E., Rousset, G., & Gratadour, D. 2016, in Adaptive Optics Systems V, Proc. SPIE, 9909, 990979
Fried, D. L. 1982, J. Opt. Soc. Am., 72, 52
Gaffard, J. P., & Boyer, C. 1987, Appl. Opt., 26, 3772
Gendron, E. 1995, Theses, Université Denis Diderot (Paris 7)
Gendron, E., Clénet, Y., Fusco, T., & Rousset, G. 2006, A&A, 457, 359
Gendron, É., Charara, A., Abdelfattah, A., et al. 2014, in Adaptive Optics Systems IV, Proc. SPIE, 9148, 91486L
Gilles, L., Correia, C., Véran, J.-P., Wang, L., & Ellerbroek, B. 2012, Appl. Opt., 51, 7443
Gratadour, D., Puech, M., Vérinaud, C., et al. 2014, in Adaptive Optics Systems IV, Proc. SPIE, 9148, 91486O
Greenwood, D. P. 1977, J. Opt. Soc. Am., 67, 390
Harder, S., & Chelli, A. 2000, A&AS, 142, 119
Hudgin, R. 1977, J. Opt. Soc. Am., 67, 393
Jolissaint, L. 2010, J. Eur. Opt. Soc., 5, [arXiv:1009.1581]
Jolissaint, L., Veran, J.-P., & Marino, J. 2004, in Advancements in Adaptive Optics, eds. D., Bonaccini Calia, B. L., Ellerbroek, & R., Ragazzoni, Proc. SPIE, 5490, 151
Jolissaint, L., Véran, J.-P., & Conan, R. 2006, J. Opt. Soc. Am. A, 23, 382
Juvenal, R., Kulcsar, C., Raynaud, H.-F., Conan, J.-M., & Sivo, G. 2015, in Adaptive Optics for Extremely Large Telescopes IV (AO4ELT4), E63
Martin, O. A., Correia, C. M., Gendron, E., et al. 2016, J. Astron. Telesc. Instrum. Syst., 2, 048001
Martin, O. A., Gendron, É., Rousset, G., et al. 2017, A&A, 598, A37
Mugnier, L. M., Fusco, T., & Conan, J.-M. 2004, J. Opt. Soc. Am. A, 21, 1841
Neichel, B., Fusco, T., & Conan, J.-M. 2008, J. Opt. Soc. Am. A, 26, 219
Rigaut, F. J., Veran, J.-P., & Lai, O. 1998, in Adaptive Optical System Technologies, eds. D., Bonaccini & R. K., Tyson, Proc. SPIE, 3353, 1038
Rousset, G., Fontanella, J. C., Kern, P., Gigan, P., & Rigaut, F. 1990, A&A, 230, L29
Taylor, G. I. 1938, Math. Phys. Sci. Proc. Roy. Soc. London Ser. A, 164, 476
Véran, J.-P., & Herriot, G. 2000, J. Opt. Soc. Am. A, 17, 1430
Véran, J.-P., Rigaut, F., Maitre, H., & Rouan, D. 1997, J. Opt. Soc. Am. A, 14, 3057
Vidal, F., Gendron, E., & Rousset, G. 2010, J. Opt. Soc. Am. A, 27, A253
Vidal, F., Gendron, É., Rousset, G., et al. 2014, A&A, 569, A16

Appendix A: Modal basis $\mathcal{B}_{\rm tt}$ computation

As specified in Sect. 2.3, the error breakdown needs to be estimated on a modal basis with specific properties:

- the modes span the full DM space;
- the modes are normalized ($\frac{1}{S}\int B_i^2 \mathrm{d}S = 1\mu\mathrm{m}^2, \forall i$ with the integral defined over the pupil area);
- they are orthogonal, in the sense that the scalar product over the pupil area is null ($\frac{1}{S}\int B_i B_j \mathrm{d}S = 0$ for $i \neq j$);
- the subspace of the modes commanded through the system control matrix is orthogonal (same scalar product as above) to the subspace of filtered ones;
- one of the filtered modes is constructed as the best DM least-squares fit to piston, so that all modes are orthogonal to the piston ($\forall i, \int B_i \mathrm{d}S = 0$). This property is important for deriving phase variances that do not include any component along the piston term;
- two of these modes are pure tip-tilt and are associated with the tip-tilt mirror, while the tip-tilt that can be produced by the DM was suppressed.

To ensure that our modal basis $\mathcal{B}_{\rm tt}$ spans the full DM space, it is computed from the influence functions of the DM. Let IF the basis of influence functions of the DM, that is, any phase produced by the DM can be decomposed as

$$\Phi_{\rm DM} = \sum_{i=1}^{N_{\rm actus}} a_i\, IF_i. \tag{A.1}$$

Then, the dimensions of this basis are $N_\Phi \times N_{\rm actus}$, where N_Φ is the number of points in the pupil area and $N_{\rm actus}$ is the number of DM actuators. The geometric covariance matrix Δ (Gaffard & Boyer 1987) is then defined as

$$\Delta = IF^t \cdot IF. \tag{A.2}$$

As IF is a basis, Δ is invertible and $\Delta^{-1} \cdot IF^t$ projects a phase onto the IF basis.

Let T_p be the matrix containing the phase corresponding to a piston and pure tip-tilt. The dimensions of this matrix are therefore $N_\Phi \times 3$. Using the projection matrix defined above, we can retrieve the coefficients a_i on the basis IF that fits the piston, tip-tilt modes, and store them in a matrix τ:

$$\tau = \Delta^{-1} \cdot IF^t \cdot T_p. \tag{A.3}$$

Our $\mathcal{B}_{\rm tt}$ is to be orthogonal to the piston mode, and this includes pure tip-tilt modes. Then, we have to generate a set of generators G from IF that cannot produce those modes (Gendron 1995):

$$G = I - \tau \cdot \left(\tau^t \cdot \Delta \cdot \tau\right)^{-1} \cdot \tau^t \cdot \Delta, \tag{A.4}$$

where I is the identity matrix. Now, the diagonalization of G provides a basis B':

$$G^t \cdot \Delta \cdot G = B' \cdot \lambda \cdot B'^t, \tag{A.5}$$

where λ are the eigenvalues of $G^t \cdot \Delta \cdot G$. After truncation of the three last columns of B' corresponding to piston, tip-tilt modes, and normalization,

$$B = G \cdot B' \cdot \sqrt{\lambda}^{-1}, \tag{A.6}$$

we obtain an orthonormal basis B such that $B^t \cdot \Delta \cdot B = I$.

Finally, we have to add the pure tip-tilt modes in the basis B to obtain the modal basis $\mathcal{B}_{\rm tt}$:

$$\mathcal{B}_{\rm tt} = \begin{pmatrix} & & & 0 & 0 \\ & B & & \vdots & \vdots \\ & & & 0 & 0 \\ 0 & \cdots & 0 & \frac{1}{\mu} & 0 \\ 0 & \cdots & 0 & 0 & \frac{1}{\mu} \end{pmatrix}, \tag{A.7}$$

where μ are the eigenvalues of the geometric covariance matrix of the tip-tilt modes, corresponding to its diagonal as these two modes composed a basis.

Appendix B: AO loop error with centroid gain

A centroid gain γ will have an effect on the equations described in Ferreira et al. (2016) and Sect. 2. To take it into account in ROKET, we have to rewrite the equations with a centroid gain. It can be modeled as a gain γ on the WFS measurement, so that Eq. (7) becomes

$$\boldsymbol{w}_k = \gamma D \boldsymbol{a}_{\mathbf{k},\theta} + \gamma D \boldsymbol{v}_{\mathbf{k-1}} + \gamma \boldsymbol{r}_k + \boldsymbol{n}_k + \boldsymbol{u}_k. \tag{B.1}$$

In this expression, $\boldsymbol{u}_k$ is no longer the same vector as in Eq. (7) as it does not include the centroid gain error.

Then, the command vector $\boldsymbol{v}_k$ can be written as

$$\boldsymbol{v}_k = (1 - \gamma g R D)\boldsymbol{v}_{k-1} - \gamma g R D \boldsymbol{a}_{k,\theta} - \gamma g R \boldsymbol{r}_k - g R \boldsymbol{n}_k - g R \boldsymbol{u}_k. \tag{B.2}$$

Finally, the residual error $\boldsymbol{\epsilon}_k$ becomes

$$\begin{aligned}\boldsymbol{\epsilon}_k = {} & (1 - \gamma g R D)\boldsymbol{\epsilon}_{k-1} + (\boldsymbol{a}_k - \boldsymbol{a}_{k-1}) + \gamma g R D(\boldsymbol{a}_{k-1} - \boldsymbol{a}_{k-1,\theta}) \\ & - \gamma g R \boldsymbol{r}_{k-1} - g R \boldsymbol{n}_{k-1} - g R \boldsymbol{u}_{k-1.}\end{aligned} \tag{B.3}$$

Hence, the equations for the estimation of the error breakdown contributors described in Sect. 2.2 become

$$\begin{aligned}\boldsymbol{\beta}_k &= (1 - \gamma g R D)\boldsymbol{\beta}_{k-1} + (\boldsymbol{a}_k - \boldsymbol{a}_{k-1}), \\ \boldsymbol{\tau}_k &= (1 - \gamma g R D)\boldsymbol{\tau}_{k-1} + \gamma g R D(\boldsymbol{a}_{k-1} - \boldsymbol{a}_{k-1,\theta}), \\ \boldsymbol{\rho}_k &= (1 - \gamma g R D)\boldsymbol{\rho}_{k-1} - \gamma g R \boldsymbol{r}_{k-1}, \\ \boldsymbol{\eta}_k &= (1 - \gamma g R D)\boldsymbol{\eta}_{k-1} - g R \boldsymbol{n}_{k-1}, \\ \boldsymbol{\mu}_k &= (1 - \gamma g R D)\boldsymbol{\mu}_{k-1} - g R \boldsymbol{u}_{k-1}.\end{aligned} \tag{B.4}$$

Appendix C: Calculation of the low-frequency structure function $D_{\phi}^{\rm low}$

To compute a model for the covariance between anisoplanatism and bandwidth error as a sum of structure functions, we have to compute these functions. However, as we are searching for errors made on the DM command, we need to calculate the structure function that excludes spatial frequencies higher than the Nyquist frequency $f_c = \frac{1}{2d}$ of the DM.

Starting from the Kolmogorov spectrum,

$$W(k) = 0.023\, r_0^{-5/3} (k)^{-11/3}, \tag{C.1}$$

the structure function that we search for can be written as

$$\begin{aligned}D_{\phi}^{\rm low}(r) &= 2\int_0^{2\pi}\int_0^{f_c} W(k)(1 - \cos(2\pi k r) k \mathrm{d}k \mathrm{d}\theta \\ &= 2 \times 0.023\, r_0^{-5/3}\end{aligned} \tag{C.2}$$

$$\times \int_0^{f_c} \int_0^{2\pi} k^{-8/3}(1-\cos(2\pi kr\cos(\theta))\mathrm{d}k\mathrm{d}\theta \quad \text{(C.3)}$$

$$= 2\times 0.023\, r_0^{-5/3} \times 2\pi \int_0^{f_c} k^{-8/3}(1-J_0(2\pi kr))\mathrm{d}k. \quad \text{(C.4)}$$

With the variable change $u = 2\pi kr$, we obtain

$$D_\phi^{\mathrm{low}}(r) = 2\times 0.023\, r_0^{-5/3} \times 2\pi \int_0^{2\pi f_c r} u^{-8/3}(2\pi r)^{8/3}(1-J_0(u))\frac{\mathrm{d}u}{2\pi r} \quad \text{(C.5)}$$

$$= 2\times 0.023\,(2\pi)^{8/3}\left(\frac{r}{r_0}\right)^{5/3} \times \int_0^{2\pi f_c r} u^{-8/3}(1-J_0(u))\mathrm{d}u. \quad \text{(C.6)}$$

However, this expression does not take into account the outer scale of the turbulence L_0. To include this, we note that

$$\int_0^{\infty} u^{-8/3}(1-J_0(u))\mathrm{d}u = 1.11833. \quad \text{(C.7)}$$

From this result and Eq. (C.6), we derive an expression of the structure function limited to the frequencies higher than f_c:

$$D_\phi^{\mathrm{high}} = 2\times 0.023\,(2\pi)^{8/3}\left(\frac{r}{r_0}\right)^{5/3} \times\left(1.11833 - \int_0^{2\pi f_c r} u^{-8/3}(1-J_0(u))\mathrm{d}u\right). \quad \text{(C.8)}$$

Moreover, we know that the effect of the outer scale on the spectrum is a saturation effect on low frequencies. Hence, the expression of D_ϕ^{high} is not affected since $f_c > \frac{1}{L_0}$. Then, we can obtain the final $D_\phi^{\mathrm{low}}(r, L0)$ as the difference between the complete structure function $D_\phi(r, L_0)$ and $D_\phi^{\mathrm{high}}(r)$:

$$D_\phi^{\mathrm{low}}(r, L_0) = D_\phi(r, L_0) - D_\phi^{\mathrm{high}}(r). \quad \text{(C.9)}$$

5.3 Quelques compléments

Je profite de cette section pour revenir sur certains points de cet article. Par soucis de concision, certains points n'y ont été que rapidement évoqués, et je souhaite y apporter quelques précisions et/ou développements.

5.3.1 Intégration dans COMPASS

Je reviens ici sur la méthode utilisée par ROKET pour estimer les contributions des sources d'erreur à chaque itération. La Figure 5.1 schématise les différentes opérations réalisées par ROKET pour obtenir le budget d'erreur.

Le lecteur reconnaîtra au centre de schéma l'exécution classique d'une simulation COMPASS telle que déjà représentée par la Figure 4.3 :

- À partir de l'écran turbulent *atmos*, la phase Φ *WFS* vue par l'ASO est déterminée par lancés de rayons
- Les *spots WFS* sont ensuite calculés pour obtenir l'image de l'ASO
- Le bruit est ajouté
- Les mesures $\mathbf{m_k}$ sont calculées avec un centre de gravité *COG*
- La matrice de commande est appliquée sur le vecteur de mesure pour déterminer la commande courante $\mathbf{com}_k = R\,\mathbf{m}_k$
- La loi de commande et le délai sont appliqués afin de déterminer le vecteur de "volts" à envoyer au miroir déformable $\mathbf{v_k} = \mathbf{v}_{k-1} - g\,R\,\mathbf{m}_k$
- Enfin, la phase résiduelle est calculée et la FEP déterminée

À l'issue de chaque itération de cette boucle, ROKET en sauvegarde l'état, i.e. l'écran de phase résiduelle, la commande courante, l'intégrateur, etc., avant d'effectuer les opérations nécessaires à l'estimation de chaque terme du budget d'erreur. Une fois tous les contributeurs obtenus, la boucle est relancée à partir de cet état, ce qui permet de ne pas modifier son comportement en comparaison avec une simulation COMPASS classique.

Détermination du bruit η_k

Comme nous l'avons vu, l'image de l'ASO est calculée par COMPASS comme une l'image non bruitée sur laquelle le bruit est ensuite appliqué. Pour déterminer la contribution du bruit sur la commande, ROKET récupère donc cette image non bruitée et calcule la commande qui aurait été obtenue avec cette dernière. La soustraction de la commande obtenue par COMPASS avec cette dernière donne alors la contribution du bruit à cette itération, à savoir $R\,\mathbf{n}_k$. L'application du filtre de la boucle me permet alors d'aboutir à l'Éq. (16) de l'article.

Détermination des déviations μ_k

Comme décrit dans l'article, on appelle ici déviations tout ce qui fait différer la mesure obtenue par l'ASO de la mesure théorique de ce dernier telle que définie dans

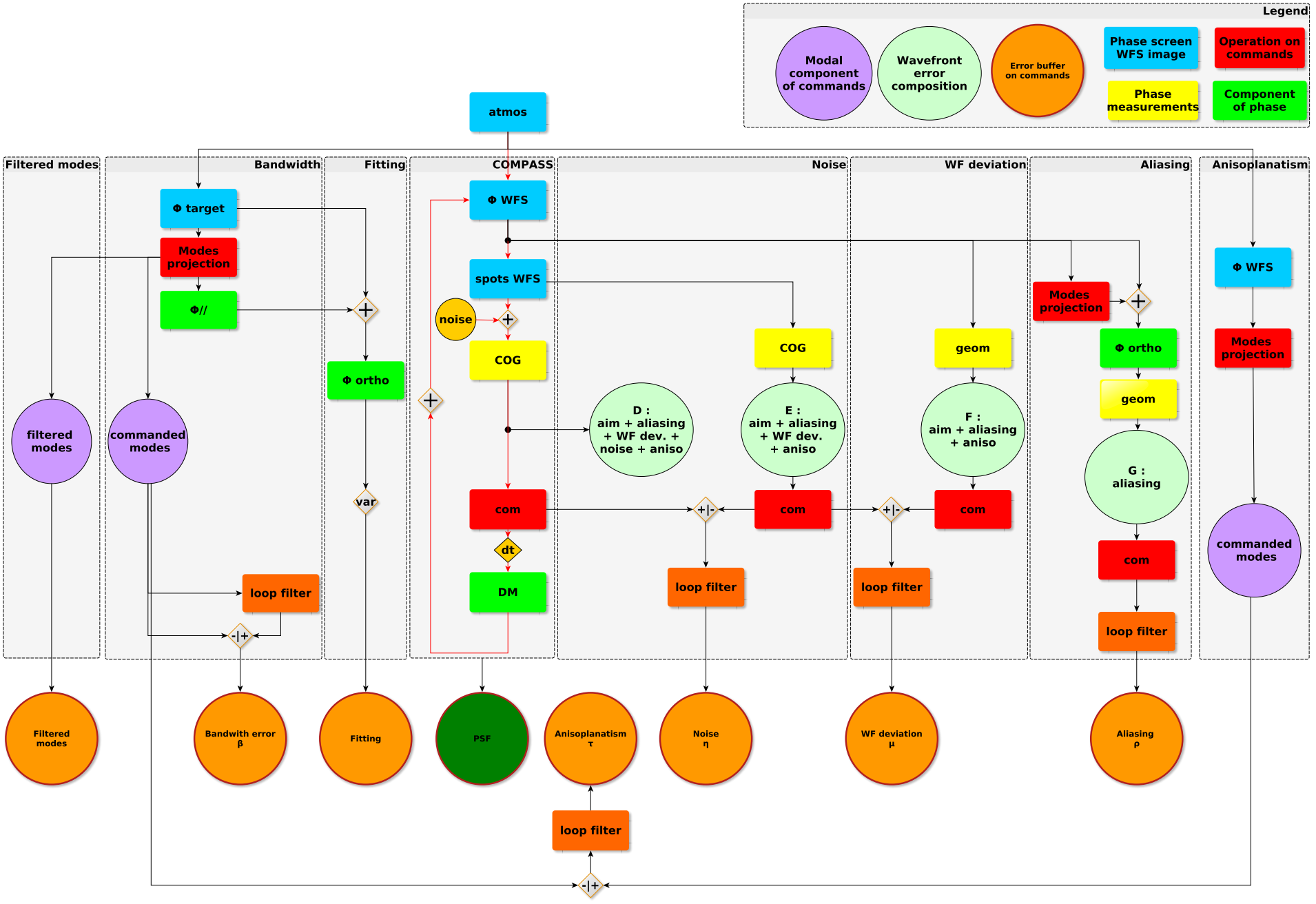

FIGURE 5.1 – Schéma de l'implémentation de ROKET dans COMPASS

l'Éq. 2.3 et qui n'est pas du bruit. Cela inclut donc par exemple le sous-échantillonnage des sous-pupilles, la troncation des spots, le gain optique, etc.

Ainsi, ROKET obtient cette contribution en calculant cette mesure théorique grâce au modèle géométrique décrit en fin de Section 4.4.8. À nouveau, ROKET calcule les commandes qui auraient été obtenues avec ces mesures, et la différence avec les commandes obtenues à partir de l'image non bruitée donne la contribution des déviations, i.e. $R\,u_k$. Ne reste alors plus qu'à appliquer le filtre de la boucle pour aboutir à l'Éq. (17) de l'article.

Détermination de l'erreur de repliement ρ_k

L'erreur de repliement est due aux hautes fréquences spatiales qui sont repliées par l'échantillonnage de l'ASO et interprétées comme des basses fréquences dans les mesures. Le modèle intégré dans ROKET permet de l'estimer en soustrayant de la phase sa projection sur les modes du miroir déformable de façon à ne garder que la composante haute fréquence orthogonale à l'espace du miroir, puis d'utiliser le modèle géométrique de l'ASO définit dans la Section 4.4.8 pour calculer les mesures théoriques des hautes fréquences ainsi obtenues. La commande qui en découle est la contribution $R\,\mathbf{r}_k$ du repliement. À nouveau, le filtre de la boucle nous permet d'obtenir l'Éq. (15) de l'article.

Détermination de l'erreur due aux modes filtrés

Grâce à la base modale décrite dans l'article et sur laquelle je reviendrai dans la Section 5.3.2, cette contribution est obtenue simplement en projetant la phase de la direction d'observation scientifique sur cette base : les modes qui ont été filtrés lors de l'inversion de la matrice d'interaction sont ensuite projetés dans le sous-espace des fonctions d'influences pour obtenir cette contribution dans l'espace des commandes du miroir déformable.

Détermination de l'erreur temporelle β_k

À partir de la projection précédente sur la base modale, le résultat obtenu est soustrait de celui obtenu à l'itération précédente (pour un délai d'une trame) pour obtenir la contribution de l'erreur temporelle $R\,(\mathbf{a}_k - \mathbf{a}_{k-1})$. Le filtre de la boucle est alors appliqué pour aboutir à l'Éq. (13).

Détermination de l'anisoplanétisme τ_k

L'anisoplanétisme est déterminé par différence entre la projection sur la base modale de la phase dans direction scientifique $\mathbf{a}_k$ à celle obtenue dans la direction d'analyse $\mathbf{a}_{k,\theta}$. L'Éq. (14) est ensuite obtenue avec le filtre de la boucle.

Détermination de l'erreur de sous-modélisation

L'erreur de sous-modélisation ne peut être estimée sur les commandes du miroir déformable. L'estimation de cette erreur par ROKET se fait alors sous la forme d'une FEP produite uniquement à partir de la composante de la phase orthogonale à l'espace du miroir déformable. Cette FEP est moyennée sur l'ensemble des itérations, et ROKET estime également la variance spatiale due à cette erreur, exprimée en secondes d'arc carrées.

Sauvegarde des données

Une fois toutes les itérations effectuées, ROKET génère un fichier HDF5 où il sauvegarde l'ensemble des buffers temporels des contributeurs au budget d'erreur. Le fichier contient également l'ensemble des paramètres qui ont été utilisés pour générer la simulation, ainsi que la télémétrie et la FEP produites par COMPASS au cours de la simulation.

5.3.2 La base modale

Pour simplifier l'exploitation du budget d'erreur estimé par ROKET, ce dernier est exprimé sur une base modale dérivée à partir des fonctions d'influence du miroir déformable utilisé. Cette base modale est construite dans le but d'obtenir certaines propriétés importantes :

- Les modes couvrent l'ensemble de l'espace du miroir déformable
- Les modes sont orthogonaux entre eux au sens du produit scalaire nul sur la pupille ($\int_{pup} B_i B_j = 0$ si $i \neq j$)
- Les modes sont normalisés sur la pupille ($\int_{pup} B_i^2 = 1\mu m^2$)
- Le sous-espace des modes commandés par la boucle d'OA doit être orthogonal au sous-espace des modes filtrés
- Les modes sont orthogonaux aux modes piston et tip-tilt. Les modes tip-tilt sont ajoutés à la base modale par la suite comme des modes purs associés à un miroir tip-tilt.

La suppression du mode piston est nécessaire afin d'éliminer toute composante de piston dans les modes de la base, et qui serait sinon comptabilisée plus tard dans les calculs de variances des coefficients.

La suppression du mode tilt est effectuée dans un but pratique, afin de simplifier la mise en œuvre modale de la base en évitant de distribuer chacun des modes sur deux miroirs (le tip-tilt et le miroir déformable).

Cette base, appelée $\mathcal{B}_{tt}$, est donc construite à partir de la base IF des fonctions d'influence du miroir déformable. De cette façon, on s'assure que la première propriété est vérifiée. Je calcule ensuite la matrice de covariance géométrique Δ du miroir déformable Gaffard & Boyer (1987) :

$$\Delta = IF^t.IF \tag{5.5}$$

Puisque que IF est une base, alors la matrice Δ est inversible et la matrice $\Delta^{-1}.IF^t$ permet de projeter toute phase sur l'espace du miroir déformable.

Je note T_p la matrice contenant les modes piston (pur, au sens de Zernike) et tip-tilt (ceux du miroir tip-tilt). Je peux alors obtenir l'expression de leur projection τ dans l'espace du miroir déformable :

$$\tau = \Delta^{-1}.IF^t.T_p \tag{5.6}$$

Cette dernière permet alors d'obtenir une famille génératrice G orthogonale aux modes piston et tip-tilt, et qui appartient à l'espace du miroir déformable Gendron (1995) :

$$G = Id - \tau.\left(\tau^t.\Delta.\tau\right)^{-1}.\tau^t.\Delta \tag{5.7}$$

où Id est la matrice identité.

Dès lors, la diagonalisation de la famille G donne une base B' :

$$G^t.\Delta.G = B'.\lambda.B'^t \tag{5.8}$$

où λ sont les valeurs propres de $G^t.\Delta.G$. Après troncation des trois dernières colonnes correspondant aux modes piston et tip-tilt, je normalise :

$$B = G.B'.\sqrt{\lambda}^{-1} \tag{5.9}$$

et j'obtiens ainsi une base B orthonormée. Il suffit alors de rajouter les modes du miroir tip-tilt à cette base pour obtenir la base $\mathcal{B}_{tt}$ voulue.

5.3.3 GAMORA : reconstruction de FEP sur GPU

À titre de validation du budget d'erreur évalué par ROKET, je propose dans l'article d'utiliser un algorithme de reconstruction de FEP. Je souhaiterais revenir sur l'implémentation de cet algorithme que j'ai réalisé sur GPU.

GAMORA (Gpu Accelerated Module fOr psf Reconstruction Algorithms) est le module Python que j'ai développé à cette fin et qui sert d'interface utilisateur au cœur de calcul, codé en C++/CUDA. Ce module inclus l'algorithme de reconstruction de FEP proposé par Gendron et al. (2006), dérivé de celui proposé par Véran et al. (1997) et que j'ai rapidement décrit dans la Section 5.1.1. Un inconvénient de ce dernier était que le nombre de fonctions U_{ij} à calculer étaient proportionnel au carré du nombre de modes du miroirs déformables, ce qui le rend difficilement utilisable dans le cadre des ELT. Et ceci est d'autant plus vrai dans le cadre d'une implémentation sur GPU où l'espace mémoire est limité.

Gendron et al. (2006) propose de diagonaliser la matrice de covariance des erreurs $\langle \boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^t \rangle$:

$$\Lambda = V^t \langle \varepsilon\varepsilon^t \rangle V \tag{5.10}$$

où Λ est la matrice diagonale contenant les $\lambda_{ii=1...N}$ valeurs propres de $\langle \boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^t \rangle$ et V est la matrice des vecteurs propres. Dans cette nouvelle base dans laquelle la matrice de covariance est diagonale, l'Éq. 5.2 s'écrit :

$$\bar{D}_{\Phi_{\varepsilon\parallel}}(\boldsymbol{\rho}) = \sum_{i=1}^{N} \lambda_i V_{ii}(\boldsymbol{\rho}) \tag{5.11}$$

où les $V_{ii}(\boldsymbol{\rho})$ sont l'équivalent dans cette nouvelle base des fonctions $U_{ij}(\boldsymbol{\rho})$. Elles présentent alors l'avantage d'être moins nombreuses (une par mode du miroir), mais elles doivent être calculées à chaque reconstruction puisqu'elles dépendent de la matrice de covariance des erreurs.

Je résume ici l'algorithme complet tel que proposé par Gendron et al. (2006) :

1. Calcul de la fonction de transfert optique du télescope OTF_{tel} comme la fonction d'autocorrélation de la pupille
2. Diagonalisation de la matrice $\langle \boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^t \rangle$
3. Détermination des modes propres sur la pupille
4. Boucle sur chacun de ces modes, calcul de la V_{ii} puis de la fonction de structure correspondante (cf. Éq. 5.11)
5. Calcul de la fonction de transfert optique (Éq. 5.1)

L'implémentation GPU que j'en ai faite suit globalement le même algorithme en utilisant les librairies CUDA pour effectuer les transformées de Fourier. Elle propose également de répartir la boucle sur de multiples GPUs. La Figure 5.2 donne les temps d'exécution obtenus sur CPU ainsi que sur un et deux GPUs. On notera ainsi un facteur d'accélération de 110 entre le code CPU et la version sur 2 GPUs pour une FEP de 2048 par 2048 reconstruite sur 1304 modes, et un facteur 390 pour une FEP à l'échelle de l'ELT de 4096 par 4096 reconstruite sur 4880 modes. Ces résultats ont été obtenus sur un serveur Nvidia DGX-1, équipé d'un CPU Dual 20-core Intel Xeon E5-2698v4 cadencé à 2,2 Ghz, et de 8 cartes graphiques Tesla V100-SXM2. Notons tout de même que le code CPU de référence n'a pas été optimisé, et n'utilisent donc que les optimisations natives de la librairie NumPy de Python.

À noter que l'utilisation d'un plus grand nombre de GPU n'apporte pas plus de performances, le temps gagné en terme de calcul étant perdu en communications entre les GPUs, en tout cas pour ces dimensionnements. Enfin, il faut également noter que ces résultats peu flatteurs pour le code CPU doivent être mis en perspective par le fait que la version CPU testée ici n'était pas spécialement optimisée et ne servait qu'à vérifier l'exactitude du résultat retourné par le code GPU. Il est donc très certainement possible d'obtenir de meilleurs temps d'exécution sur CPU également.

5.4 Résultats à l'échelle de l'ELT

Le lecteur attentif aura remarqué que les résultats de ROKET présentés dans l'article concernent un système d'OA type SPHERE, i.e. sur un télescope de 8m et

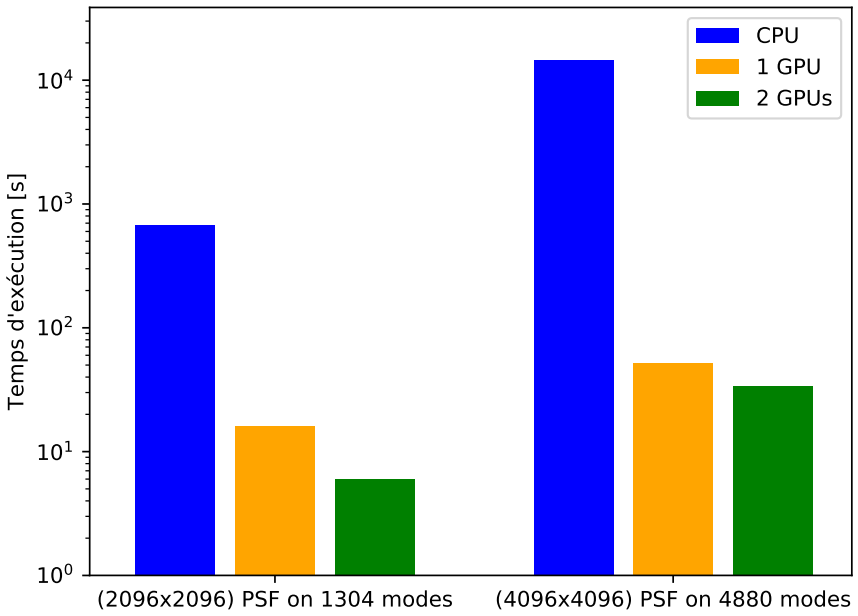

FIGURE 5.2 – Temps d'exécution de GAMORA sur CPU et GPU pour un dimensionnement typique d'une FEP sur un télescope de 8m et sur un télescope de 39m

utilisant 40 par 40 sous-pupilles. D'autre part, ce même lecteur s'est lancé dans la lecture de ce mémoire en voyant écrit sur la première page *"dans la perspective des ELT"*. Donc, l'ELT dans tout ça ? Je propose dans cette section de *mettre à jour* les résultats de l'article sur un système ELT cette fois-ci.

5.4.1 Paramètres de simulation ELT

Les paramètres de simulations utilisés sont listés dans le tableau 5.1. Le profil

Télescope		**Cible**	
Diamètre	39 m	Longueur d'onde λ_t	1.65 μm
Atmosphère		**ASO**	
Nombre de couches	35	Nombre de sous-pupilles	78×78
r_0 (500 nm)	0.15 m	Longueur d'onde λ_{wfs}	0.5 μm
L_0	100 m	Pixels par sous-pupille	6
		Taille des pixels	0.5"
		Photons par sous-pupille	4730
		Bruit de lecture	3 e^-
		Calcul du centre de gravité	Classical CoG
		Coordonnées de l'étoile guide	(5",0")
Boucle d'OA		**Miroir déformable**	
Fréquence de boucle	500 Hz	Nombre d'actionneurs	79×79
Loi de commande	Intégrateur	1 miroir tip-tilt	
Gain de boucle	0.3	Altitude de conjugaison	0 m (pupille)
Retard	1 trame		
Trames	20 000		

Tableau 5.1 – Paramètres de simulation ELT

turbulent est composé de 35 couches et la Figure 5.3 représente la fraction de $r_0^{5/3}$ et les vitesses de vents de chaque couche, les barres horizontales étant positionnées selon leur altitude. La direction du vent sur chacune des couches a été choisi aléatoirement.

5.4.2 Budget d'erreur

Le Tableau 5.2 résume le budget d'erreur estimé par ROKET pour le cas ELT décrit précédemment. À nouveau, ce budget d'erreur est cohérent avec les approximations couramment utilisées pour estimer ces erreurs et présentées en Section 2.6. Ce dernier est d'ailleurs confirmé par la comparaison des FEP issues d'une part de la simulation bout-en-bout, et d'autre part des erreurs estimées par ROKET et utilisées pour opérer une reconstruction de FEP. Cette comparaison est illustrée par les Figures 5.4 et 5.5. Ces figures donnent des résultats semblables à ceux commentés dans l'article, avec un rapport de Strehl de 56,5 % pour la FEP reconstruite à partir

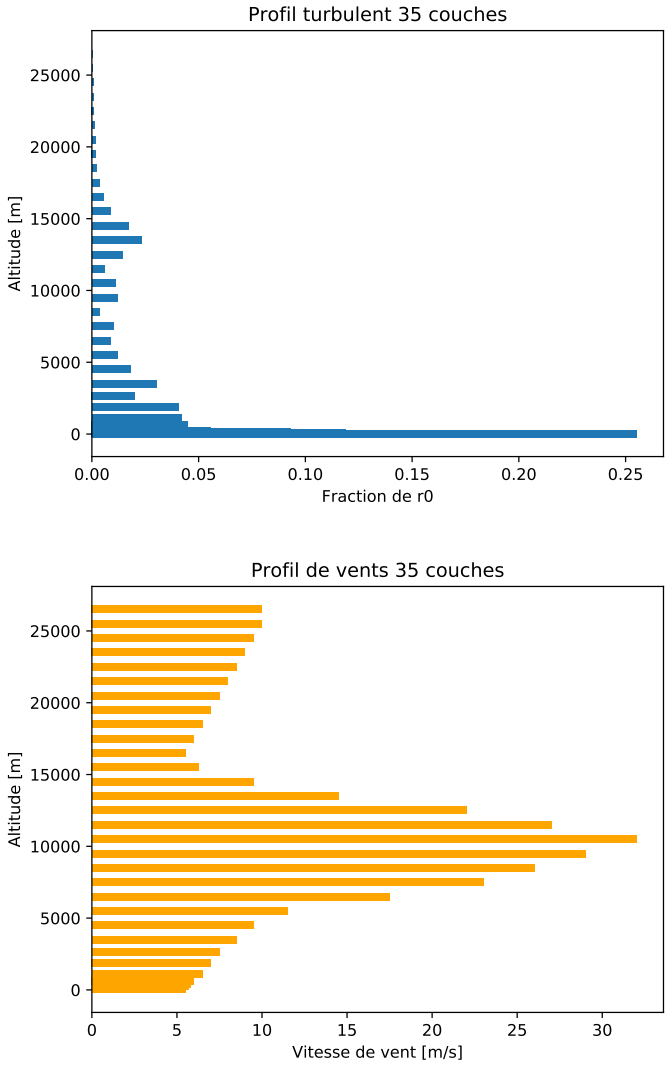

FIGURE 5.3 – Profil turbulent à 35 couches utilisé pour les simulations. En haut : fraction de $r_0^{5/3}$ de chaque couche, en bas : vitesse de vent

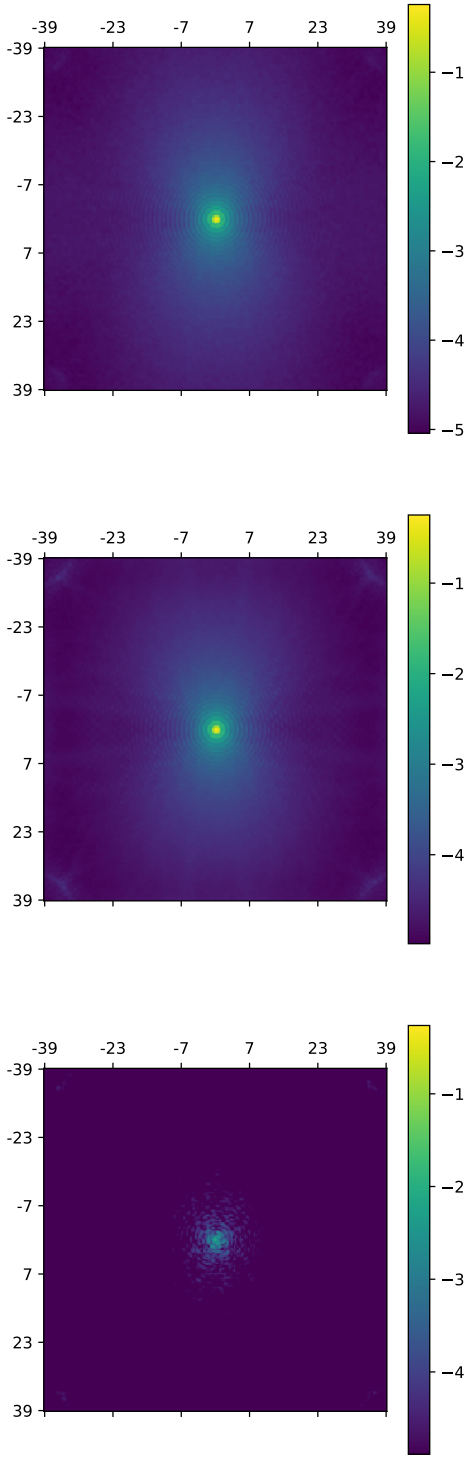

FIGURE 5.4 – En haut : FEP calculée par COMPASS ; au milieu : FEP reconstruite à partir des estimations d'erreur de ROKET ; en bas : différence des 2 FEP. Les échelles sont toutes logarithmiques.

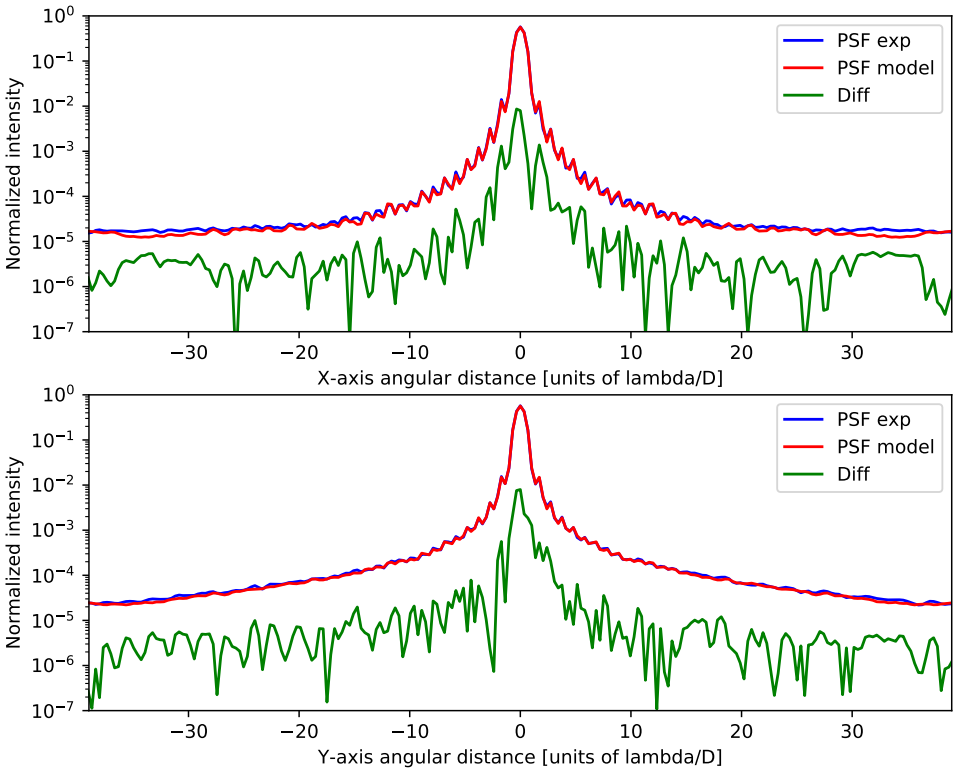

FIGURE 5.5 – Coupes en X et en Y des FEP. En bleu : FEP calculée par COMPASS ; en rouge : FEP reconstruite à partir des estimations de ROKET ; en vert : différence des 2 FEP précédentes. Échelle logarithmique.

Contributeurs	σ **[nm rms]**
Erreur temporelle	57
Anisoplanétisme	114
Repliement	76
Bruit	26
Déviations de la mesure	7.5
Sous modélisation	120
Modes filtrés	15
Termes croisés	37

Tableau 5.2 – Budget d'erreur retourné par ROKET pour le cas ELT

des estimations d'erreur de ROKET contre 57,3 % de Strehl obtenue via la simulation bout-en-bout. Idem en termes d'énergie encadrée dans un carré de $10\frac{\lambda}{D}$ centré, avec 65,3 % contre 65,8 %.

5.4.3 Performances

Côté performances, ROKET profite naturellement de l'accélération GPU de COMPASS. Le temps d'exécution d'une simulation réalisée avec estimation du budget d'erreur est environ 5 fois plus long que celui d'une simulation COMPASS classique. Ce facteur est évidemment dû aux opérations supplémentaires effectuées par ROKET dans le but d'estimer chacun des contributeurs au budget d'erreur.

Ainsi, toujours avec le serveur DGX-1 décrit dans la Section 5.3.3, ROKET s'exécute à 60 trames par seconde, contre 330 pour une simulation COMPASS classique.

5.4.4 Perspectives

En l'état, l'implémentation de ROKET ne permet une estimation du budget d'erreur que pour les systèmes d'optique adaptative de type SCAO. Le travail à mener pour aboutir à la possibilité de prendre en charge des systèmes plus complexes type MOAO ou MCAO nécessite de redéfinir certaines erreurs. Je pense notamment à l'erreur d'anisoplanétisme qui devra plutôt être considérée comme une erreur de tomographie.

Le temps nécessaire pour faire aboutir ces développements est conséquent, dans la mesure où l'intégration dans COMPASS n'est pas nécessairement aisée. J'ai alors fait le choix au cours de ma thèse de me contenter de cette première implémentation de ROKET, même si elle me limite à des cas simples. L'un des objectifs visés étant la reconstruction de FEP, l'outil en l'état permet déjà d'étudier le comportement des erreurs d'un système d'OA et fournit déjà une quantité d'informations importante qu'il a été nécessaire d'assimiler et de comprendre, notamment dans l'optique de développer des modèles analytiques de ces erreurs. La transition menant au prochain chapitre de cette thèse portant justement sur ces modélisations est donc toute trouvée.

Chapitre 6

GROOT : modèle de matrice de covariance des erreurs

Maintenant que je dispose d'un outil numérique d'estimation des termes du budget d'erreur d'un système d'OA, ce dernier me permet de produire des données de référence pour le développement de modèles analytiques (ou pseudo-analytiques) de ces termes d'erreur. Je reviendrai sur l'intérêt de tels modèles dans la première section de ce chapitre, qui est donc consacré à la dernière partie des travaux que j'ai menés pendant cette thèse. J'y présente GROOT [1], une nouvelle approche pour la modélisation des erreurs en OA. Basée sur la modélisation de matrices de covariance, l'implémentation GPU de GROOT permet d'obtenir des estimations rapides et précises de ces matrices, pouvant par la suite être utilisées par des algorithmes de reconstruction de FEP.

Je présente donc ici les différents modèles que j'ai pu développer, les résultats obtenus à l'échelle de l'ELT ainsi que leurs limitations. Je proposerai également une méthode d'identification des paramètres nécessaires à ces modèles à partir des données de télémétrie de la boucle d'OA, utile pour une possible utilisation de ces modèles sur des observations ciel.

Sommaire

1. *Gpu-base Residual errOr cOvariance maTrix*

6.1 L'intérêt d'une approche analytique du budget d'erreur

À la lumière des chapitres précédents, revenons quelque peu sur ce que j'ai d'ores et déjà évoqué dans la Section 3.1.1 au sujet des différentes approches possible pour la simulation des systèmes d'OA.

Comme nous l'avons vu dans le chapitre 4, l'implémentation d'un outil de simulation bout-en-bout comme COMPASS peut être une tâche très significative. Leur structure repose sur différents modèles de chaque composants du système, et la fidélité du résultat à la réalité dépend de ces derniers. Ainsi, la complexité numérique de ces outils résulte d'un compromis entre temps de calcul raisonnable et précision du résultat. De plus, les simulations de type Monte-Carlo nécessitent l'exécution d'un grand nombre d'itérations pour faire converger le résultat final.

Les approches analytiques permettent, au prix d'hypothèses simplificatrices supplémentaires, d'obtenir une estimation plus rapide du résultat. Les deux approches ne doivent cependant pas être mises en opposition, elles sont plutôt complémentaires. Les modèles analytiques peuvent par exemple être utilisés lors des phases préliminaires du design d'un instrument afin d'explorer rapidement l'espace des paramètres. Les outils bout-en-bout permettront ensuite de confirmer certains compromis plus fins dans les phases plus avancées du design.

L'un des objectifs du développement de ROKET, présenté au chapitre précédent, était justement de pouvoir disposer d'un outil permettant d'obtenir une estimation précise des différents termes du budget d'erreur en OA afin d'être capable de les étudier et de les modéliser le plus fidèlement possible. Pendant la dernière partie de ma thèse, je me suis donc attelé au développement de différents modèles analytiques de ces erreurs.

Comme je l'ai évoqué précédemment, de tels modèles existent déjà. La plupart cherchent à modéliser le spectre de la phase résiduelle obtenue après correction par l'OA, en décomposant cette phase comme une somme de contributions au budget d'erreur. La modélisation de ces différents termes permet alors de remonter à la FEP de l'instrument (cf. Éq. (3.1)). Ces approches cherchent généralement à modéliser, dans le domaine de Fourier, le spectre des différents contributeurs au budget d'erreur et supposent l'indépendance statistique des contributeurs à l'erreur totale.

L'idée derrière l'approche que je propose est de plutôt chercher à modéliser la matrice de covariance de l'erreur résiduelle, telle que nécessaire dans les algorithmes de reconstruction de FEP (Éq. (5.2)). L'intérêt de cette approche est détaillée dans la

deuxième partie de l'article A&A, section 4.5, qui suit. Pour résumer, elle permet non seulement d'obtenir une estimation rapide de la FEP par simulation, mais pourrait également être utilisée sur des observations ciel pour la reconstruction de FEP.

Les modèles développés incluent les erreurs d'anisoplanétisme, temporelle, de sous-modélisation et de repliement. Les prochaines sections sont dédiées à leur description.

6.2 Modélisation analytique du budget d'erreur en OA

6.2.1 Erreur d'anisoplanétisme et erreur temporelle : seconde partie de l'article A&A

J'ai présenté en Section 5.2 la première partie de l'article A&A portant sur l'estimation du budget d'erreur en OA. La seconde partie, i.e. à partir de la section 4, traite de la modélisation des erreurs d'anisoplanétisme et temporelle. Ce modèle inclut la corrélation des deux erreurs, dépendante de la direction du vent et de la position de l'étoile guide, comme expliqué dans la première partie de l'article. Pour un système de type SCAO, ces deux termes d'erreur sont en général les principaux contributeurs au budget d'erreur sur les sources brillantes. Je profite de cette section pour apporter quelques précisions sur l'article.

Les Éq. (29) à (31) qui apparaissent dans la section 4.2 de l'article définissent le modèle analytique des matrices de covariance de ces erreurs. L'Éq. (33) donne ensuite la matrice de covariance totale, incluant les deux erreurs et leur corrélation. Cette matrice C_{ee}^{Φ} décrit la covariance de l'erreur sur la phase produite par le miroir déformable, exprimée au sommet des actionneurs. Or, ce que l'on souhaite calculer est plutôt le poids à appliquer à chaque fonction d'influence du miroir déformable pour obtenir cette phase résiduelle. La différence entre ces deux approches est liée au couplage qui existe entre les actionneurs du miroir déformable. Ne pas le prendre en compte conduira nécessairement à surestimer la covariance des erreurs. En supposant que les fonctions d'influence ont une extension limitée dans la pupille, i.e. qui n'impacte que les actionneurs directement voisins, on construit une matrice M de couplage des actionneurs comme une matrice avec une diagonale égale à 1 et où les éléments qui correspondent à deux actionneurs voisins sont égaux au coefficient de couplage. Ainsi, l'inverse de cette matrice permet alors de transformer la valeur du front d'onde au sommet des actionneurs en un poids à appliquer aux fonctions d'influence.

D'autre part, il faut noter que les fonctions de structure utilisées pour produire la matrice C_{ee}^{Φ} incluent les modes de piston et de tip-tilt. Comme expliqué dans la Section 5.3.2, j'utilise une base modale où les modes tip-tilt sont produits exclusivement par le miroir tip-tilt du système. Ainsi, ces 2 modes et le mode piston doivent être filtrés de la matrice C_{ee}^{Φ}, et les composantes tip-tilt ainsi filtrées doivent être projetées sur les commandes du miroir tip-tilt.

Au final, en notant T la matrice de filtrage de ces modes, la matrice de covariance des erreurs d'anisoplanétisme et temporelle s'exprime, dans l'espace des actionneurs,

comme :

$$C_{ee} = T\,M^{-1}\,C_{ee}^{\Phi}\,(T\,M^{-1})^t \tag{6.1}$$

6.2.2 L'erreur de repliement

Je me place ici dans l'espace des mesures d'un ASO de type Shack-Hartmann. Le repliement $\mathbf{r}$ sur les mesures de l'ASO peut s'écrire (Véran et al., 1997; Rigaut et al., 1998) :

$$\mathbf{r} = \mathcal{M}(\phi_\perp(\mathbf{x}, \boldsymbol{\theta})) \tag{6.2}$$

où $\mathcal{M}$ est l'opérateur linéaire incluant l'échantillonnage spatial de l'ASO et décrivant la mesure théorique des pentes du front d'onde (cf. Éq. (2.3)), et $\phi_\perp(\mathbf{x}, \boldsymbol{\theta})$ la composante de la phase orthogonale à l'espace du miroir déformable. À partir de l'Éq. (2.3), le repliement r_i sur la sous-pupille i selon les deux axes X et Y s'écrit donc :

$$\begin{aligned} r_x^i &= \frac{\lambda}{2\pi}\frac{1}{S}\int_{subap.} \frac{\partial \phi_\perp^i}{\partial x} dxdy \\ r_y^i &= \frac{\lambda}{2\pi}\frac{1}{S}\int_{subap.} \frac{\partial \phi_\perp^i}{\partial y} dxdy \end{aligned} \tag{6.3}$$

avec S la surface d'une sous-pupille du Shack-Hartmann. Par la suite, je supposerai qu'elles sont toutes identiques et carrées de côté d.

Pour alléger quelque peu ce qui suit, concentrons-nous uniquement sur un seul des 2 axes, l'axe X par exemple. Les calculs sont strictement identiques sur les deux axes : le résultat pour l'autre axe pourra être immédiatement déduit de celui-ci.

Sur l'axe X donc, l'Éq. (6.3) peut se réécrire (Martin, 2014) :

$$r_x^i = A\int_0^d (\phi_\perp(x_i + d, y_i + l) - \phi_\perp(x_i, y_i + l))\,dl \tag{6.4}$$

avec $A = \frac{\lambda}{2\pi}\frac{1}{d^2}$ et où les coordonnées (x_i, y_i) définissent la position du bord inférieur gauche de la sous-pupille i.

Afin de pouvoir calculer numériquement cette expression, il nous faut la discrétiser. Je choisis de suivre la méthode de Martin (2014), basée sur une intégration par une méthode de Simpson. L'équation précédente peut donc être discrétisée sur N points, avec N impair et supérieur à 2, telle que :

$$r_x^i \approx A\frac{h}{3}\left(\sum_{k=0}^{N-1} w_k\,(\phi_\perp(x + d, y + kh) - \phi_\perp(x, y + kh))\right) \tag{6.5}$$

avec $h = \frac{d}{N-1}$ l'intervalle entre 2 points d'échantillonnages et w_k un poids défini comme :

$$w_k = \begin{cases} 1 & \text{si } k = 0 \text{ ou } k = N-1 \\ 4 & \text{si } k \text{ est impair} \\ 2 & \text{si } k \text{ est pair} \end{cases} \tag{6.6}$$

À partir de l'Éq.(6.5), il est possible d'obtenir une expression de la matrice de covariance $C_{alias}^{xx} = \langle r_x r_x^t \rangle$ de l'erreur de repliement sur l'axe X :

$$C_{alias}^{xx}(i,j) \approx A^2 \frac{h^2}{9} \sum_{k=0}^{N-1} \sum_{p=0}^{N-1} w_k w_p \langle (\phi_\perp(x_i + d, y_i + kh) - \phi_\perp(x_i, y_i + kh)) \\ (\phi_\perp(x_j + d, y_j + ph) - \phi_\perp(x_j, y_j + ph)) \rangle \quad (6.7)$$

Comme pour les erreurs d'anisoplanétisme et temporelle, il est possible d'écrire l'équation précédente comme une somme de fonction de structure en supposant que la phase est stationnaire et en utilisant l'identité :

$$2(A-a)(B-b) = (A-b)^2 + (a-B)^2 - (A-B)^2 - (a-b)^2 \quad (6.8)$$

Dès lors, l'Éq. (6.7) s'écrit :

$$C_{alias}^{xx}(i,j) \approx \frac{A^2}{2} \frac{h^2}{9} \sum_{k=0}^{N-1} \sum_{p=0}^{N-1} w_k w_p \Big(D_\phi^{high}(x_{ij} + d, y_{ij} + (k-p)h) \\ + D_\phi^{high}(x_{ij} - d, y_{ij} + (k-p)h) \\ - 2D_\phi^{high}(x_{ij}, y_{ij} + (k-p)h) \Big) \quad (6.9)$$

avec $x_{ij} = x_i - x_j$ la séparation entre les sous-pupilles i et j selon l'axe X, $y_{ij} = y_i - y_j$ celle selon l'axe Y et D_ϕ^{high} la fonction de structure de phase limitée aux fréquences spatiales supérieures à la fréquence de coupure du miroir déformable, telle que définie dans l'article A&A précédent (appendice C). Il faut noter ici que le calcul de cette fonction de structure fait l'hypothèse que la phase génératrice du repliement est spatialement stationnaire, et que la fonction de structure est indépendante de la direction, i.e. centro-symétrique.

En suivant le même raisonnement, on obtient également les expressions de la matrice de covariance C_{alias}^{yy} de l'erreur de repliement selon l'axe Y et de la matrice de covariance croisée C_{alias}^{xy} :

$$C_{alias}^{yy}(i,j) \approx \frac{A^2}{2} \frac{h^2}{9} \sum_{k=0}^{N-1} \sum_{p=0}^{N-1} w_k w_p \Big(D_\phi^{high}(x_{ij} + (k-p)h, y_{ij} + d) \\ + D_\phi^{high}(x_{ij} + (k-p)h, y_{ij} - d) \\ - 2D_\phi^{high}(x_{ij} + (k-p)h, y_{ij}) \Big) \quad (6.10)$$

$$C_{alias}^{xy}(i,j) \approx \frac{A^2}{2} \frac{h^2}{9} \sum_{k=0}^{N-1} \sum_{p=0}^{N-1} w_k w_p \Big(D_\phi^{high}(x_{ij} + \frac{d}{2} - ph, y_{ij} + \frac{d}{2} + kh) \\ + D_\phi^{high}(x_{ij} - \frac{d}{2} - ph, y_{ij} - \frac{d}{2} + kh) \\ - D_\phi^{high}(x_{ij} + \frac{d}{2} - ph, y_{ij} - \frac{d}{2} + kh) \\ - D_\phi^{high}(x_{ij} - \frac{d}{2} - ph, y_{ij} + \frac{d}{2} + kh) \Big) \quad (6.11)$$

Au final, la matrice de covariance totale de l'erreur de repliement sur les mesures de l'ASO C_{alias} est la méta-matrice composée par les matrices C_{alias}^{xx}, C_{alias}^{yy} et C_{alias}^{xy} :

$$C_{alias}^{m} = \begin{pmatrix} C_{alias}^{xx} & C_{alias}^{xy} \\ C_{alias}^{xy} & C_{alias}^{yy} \end{pmatrix} \tag{6.12}$$

Une nouvelle fois, je précise que la matrice C_{alias}^{m} est exprimée dans l'espace des mesures. Pour passer dans l'espace des commandes du miroir déformable, il faudra appliquer à chaque côté de cette matrice le reconstructeur R utilisé par la boucle d'OA :

$$C_{alias} = R\, C_{alias}^{m}\, R^{t} \tag{6.13}$$

Je néglige ici l'impact du filtrage de la boucle d'OA. Je reviendrai sur ce point dans la Section 6.3.2.

6.2.3 Erreur de sous-modélisation

La fonction de structure $D_{\phi}^{high}(r)$, développée pour les modèles présentés précédemment, permet également d'obtenir directement une estimation de l'erreur de sous-modélisation. Contrairement aux modèles présentés jusqu'ici, cette erreur ne peut pas, par définition, s'exprimer sous la forme d'une matrice de covariance dans l'espace de commande du miroir. On cherchera plutôt exprimer directement la composante de la fonction de transfert optique due à cette erreur (cf. Éq. (5.1)), qui peut s'écrire à partir de la fonction de structure $D_{\phi}^{high}(r)$:

$$OTF_{fit}(\frac{\boldsymbol{\rho}}{\lambda}) = \exp\left(-0.5\, D_{\phi}^{high}(\boldsymbol{\rho})\right) \tag{6.14}$$

Cette approche est alors similaire à ce qui est proposé par Gendron et al. (2014), ou encore par Jolissaint et al. (2006).

6.2.4 Les modes filtrés

Les modes filtrés appartiennent à l'espace du miroir déformable, mais ne sont pas contrôlés par la boucle puisque mal vus par l'ASO. Grâce aux propriétés de la base modale présentée dans la Section 5.3.2, il est possible d'estimer leur contribution au budget d'erreur en calculant la matrice de covariance de la phase sur les actionneurs du miroir déformable. Une fois obtenue, il suffira alors de projeter cette matrice dans l'espace modal et d'isoler la contribution des N modes filtrés.

Ainsi, je pose $\phi_{\parallel}(\mathbf{x}_i)$ la phase moyennée au voisinage de l'actionneur i, dans l'espace du miroir déformable, $\mathbf{x}_i$ est le vecteur position de cet actionneur dans la pupille. La matrice de covariance Δ_{ϕ} de la phase sur les actionneurs du miroir peut alors s'écrire comme :

$$\Delta_{\phi}(i,j) = \langle (\phi_{\parallel}(\mathbf{x}_i) - \phi_{\parallel}(\mathbf{0}))(\phi_{\parallel}(\mathbf{x}_j) - \phi_{\parallel}(\mathbf{0})) \rangle \tag{6.15}$$

où $\phi_{\parallel}(\mathbf{0})$ est la phase au centre de la pupille.

À partir de cette équation, et en utilisant à nouveau l'identité de l'Éq. 6.8, la matrice de covariance Δ_ϕ s'écrit finalement :

$$\Delta_\phi(i,j) = \frac{1}{2}\left(D_\phi^{low}(\mathbf{x}_i) + D_\phi^{low}(\mathbf{x}_j) - D_\phi^{low}(\mathbf{x}_{ij})\right) \tag{6.16}$$

Comme dans le cas des erreurs d'anisoplanétisme et temporelle, il conviendra de filtrer le piston et les modes de tip-tilt de cette matrice, ainsi que de prendre en compte le couplage des actionneurs.

Enfin, comme on peut le voir dans le budget d'erreur présenté par le Tableau 5.2, cette erreur pourra être, la plupart du temps, négligée devant les autres contributeurs. Le bon dimensionnement du système viendra de plus assurer que cette erreur est petite.

6.2.5 Le bruit de mesure

Le bruit de mesure a d'ores et déjà fait l'objet de modélisation de sa covariance. En effet, Rousset (1999) donnait déjà des expressions pour la variance du bruit de photons et du bruit de lecture électronique (cf. Section 2.6.4). Le bruit étant indépendant entre les sous-pupilles de l'ASO, sa matrice de covariance dans l'espace des mesures est diagonale. Il suffira alors de construire cette matrice à partir des expressions connues de la variance du bruit. Enfin, il conviendra de prendre en compte le filtrage temporel de la boucle d'OA.

D'autre part, pour ce qui est de la reconstruction de FEP, il a déjà été montré qu'il est possible de mesurer le bruit sur les mesures de l'ASO à partir de la télémétrie de l'OA (Gendron & Léna, 1994).

6.2.6 Les déviations de la mesure

Reste donc les déviations de la mesure l'ASO. Malheureusement, la nature non linéaire de cette erreur, due notamment à la troncation et au sous-échantillonnage des images, rend difficile l'établissement d'un modèle similaire aux précédents.

L'un des atouts des modèles présentés jusqu'ici est qu'ils sont "simples" dans la mesure où ils ne nécessitent que peu de paramètres d'entrée et identifiables sur le ciel. Le développement d'un modèle qui ajusterait une erreur majoritairement et réellement non linéaire demanderait un temps conséquent que je n'ai pas eu pendant cette thèse, et dont je doute qu'il eut été utilisé à bon escient. En effet, le terme des déviations, censé initialement contenir les termes non-linéaires de la mesure, s'est avéré être dominé par l'impact du gain optique qui corrèle les termes du budget d'erreur. Je l'ai montré et développé dans la section 3 de l'article A&A (Section 5.2), l'estimation du gain optique et sa prise en compte en tant que phénomène à part entière permet de retirer sa contribution du terme des déviations, et par là de la rendre négligeable dans le budget d'erreur total, à condition bien sûr que le dimensionnement de l'ASO soit correct vis à vis du seeing.

6.3 Résultats

Après la présentation théorique des différents modèles inclut dans GROOT, il est temps de passer à la pratique. J'utilise ici à nouveau le dimensionnement ELT présenté dans la Section 5.4.1. La procédure de test est la même que celle que j'ai utilisée dans l'article : ROKET me permet d'avoir accès à la matrice de covariance de chaque erreur séparément les unes des autres. De l'autre côté, les modèles développés dans la section précédente me donnent également une estimation de ces mêmes matrices. Je compare alors les FEP reconstruites à partir de ces matrices de covariance en termes de rapport de Strehl et d'énergie encadrée.

6.3.1 Erreur d'anisoplanétisme et erreur temporelle

La Figure 6.1 montre ainsi les FEP obtenues en ne considérant que les erreurs d'anisoplanétisme et temporelle, et la Figure 6.2 compare les coupes en X et en Y des FEP obtenues à partir de ROKET et des modèles.

Comme pour les résultats obtenus dans l'article sur un télescope de 8m, le modèle montre une bonne précision sur la reconstruction de la FEP, avec un rapport de Strehl de 79% comparé aux 77% obtenus à partir des données de ROKET. Même constat pour l'énergie encadrée entre $\pm 5\frac{\lambda}{D}$, avec 82 % contre 81%.

Les limites du modèle, exposées dans la section 4.4 de l'article Section 6.2.1, restent les mêmes, à savoir une précision moindre au-delà de 20 $\frac{\lambda}{D}$. J'invite donc le lecteur à se référer à l'article pour obtenir plus de précisions à ce sujet.

6.3.2 Erreur de repliement

De la même façon que précédemment, la Figure 6.3 montre les FEP obtenues en ne considérant que l'erreur de repliement, avec une discrétisation sur $N = 5$ points, et la Figure 6.4 les coupes associées.

En termes de rapport de Strehl, le modèle nous donne une FEP reconstruite à 93 % de Strehl, contre 92 % pour ROKET. Pour ce qui est de l'énergie encadrée, on obtient 90 % avec le modèle pour 93 % avec ROKET.

Même si les résultats en terme de reconstruction de FEP restent corrects, on observe néanmoins une différence entre les FEP plus marquée que pour les erreurs d'anisoplanétisme et temporelle. Sur la Figure 6.4 notamment, on constate que les "ailes" de la FEP sont surestimées par le modèle au-delà de 30 $\frac{\lambda}{D}$, et on remarque également que les anneaux sont plus "creusés" autour du cœur cohérent. Cela semble donc traduire une sous-estimation des basses fréquences spatiales ainsi qu'une surestimation des hautes fréquences.

Ces observations sont confirmées par la Figure 6.5 qui représente la variance de l'erreur de sous-repliement estimée sur chaque mode de la base modale. Si le niveau moyen du spectre est respecté, on observe cependant une sous-estimation des basses fréquences par le modèle, sur les 200 premiers modes environ, puis une surestimation

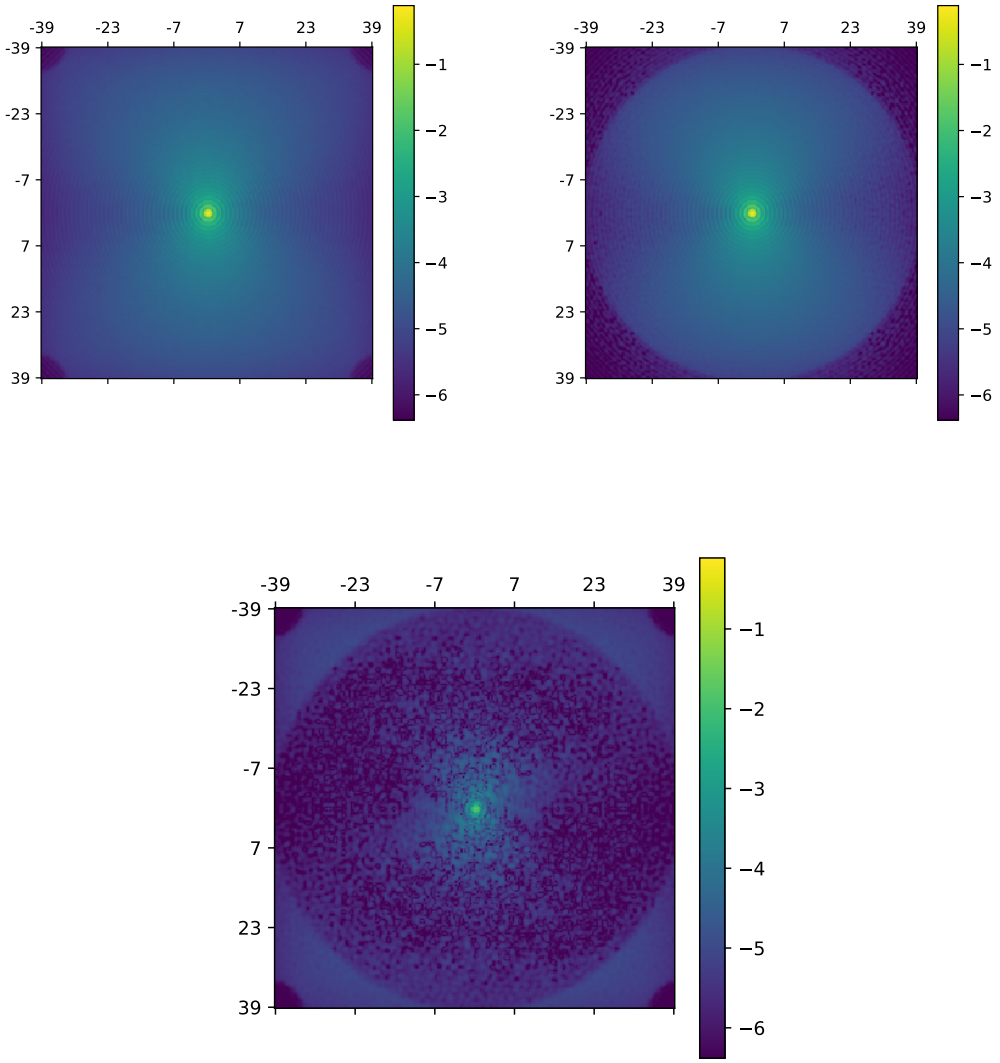

FIGURE 6.1 – FEP reconstruites exclusivement à partir de la matrice de covariance de l'erreur d'anisoplanétisme et de l'erreur temporelle. Tous les autres termes du budget d'erreur sont ommis. En haut à gauche : la matrice a été calculée à partir des données de ROKET. En haut à droite : la matrice a été modélisée par GROOT. En bas : différence entre les 2 FEP précédentes

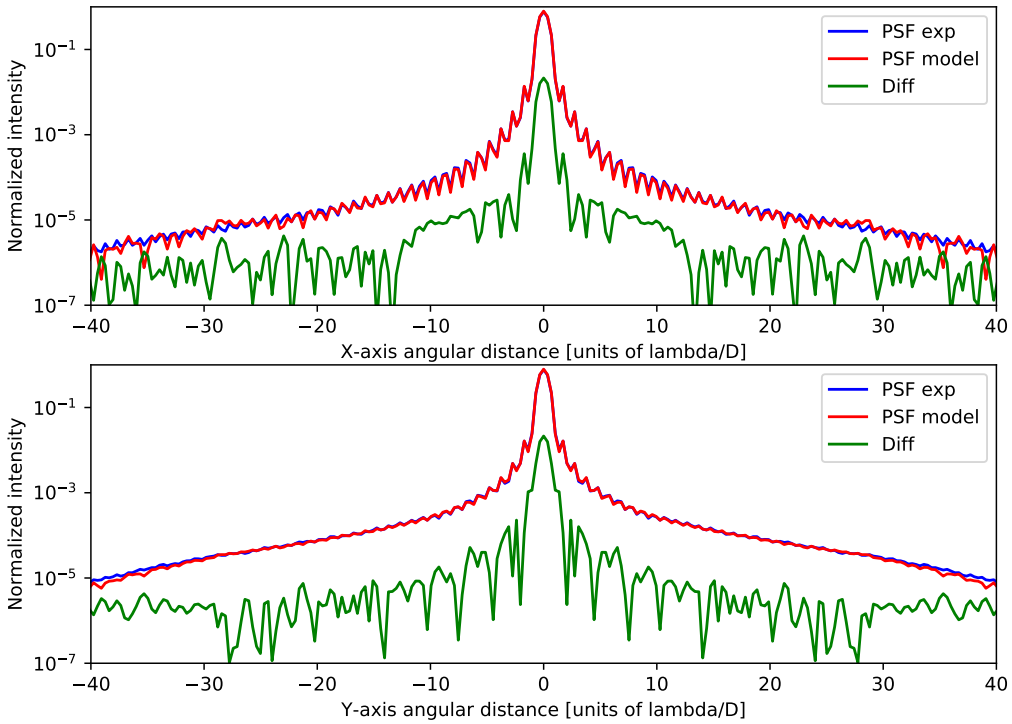

FIGURE 6.2 – Coupes de la FEP reconstruite à partir de la matrice de covariance des erreurs d'anisoplanétisme et temporelle. En haut : coupe selon l'axe X, en bas : coupe selon l'axe Y. En bleu : FEP obtenue à partir de ROKET, en rouge : FEP obtenue à partir du modèle, en vert : différence entre les 2 FEP. Échelle logarithmique

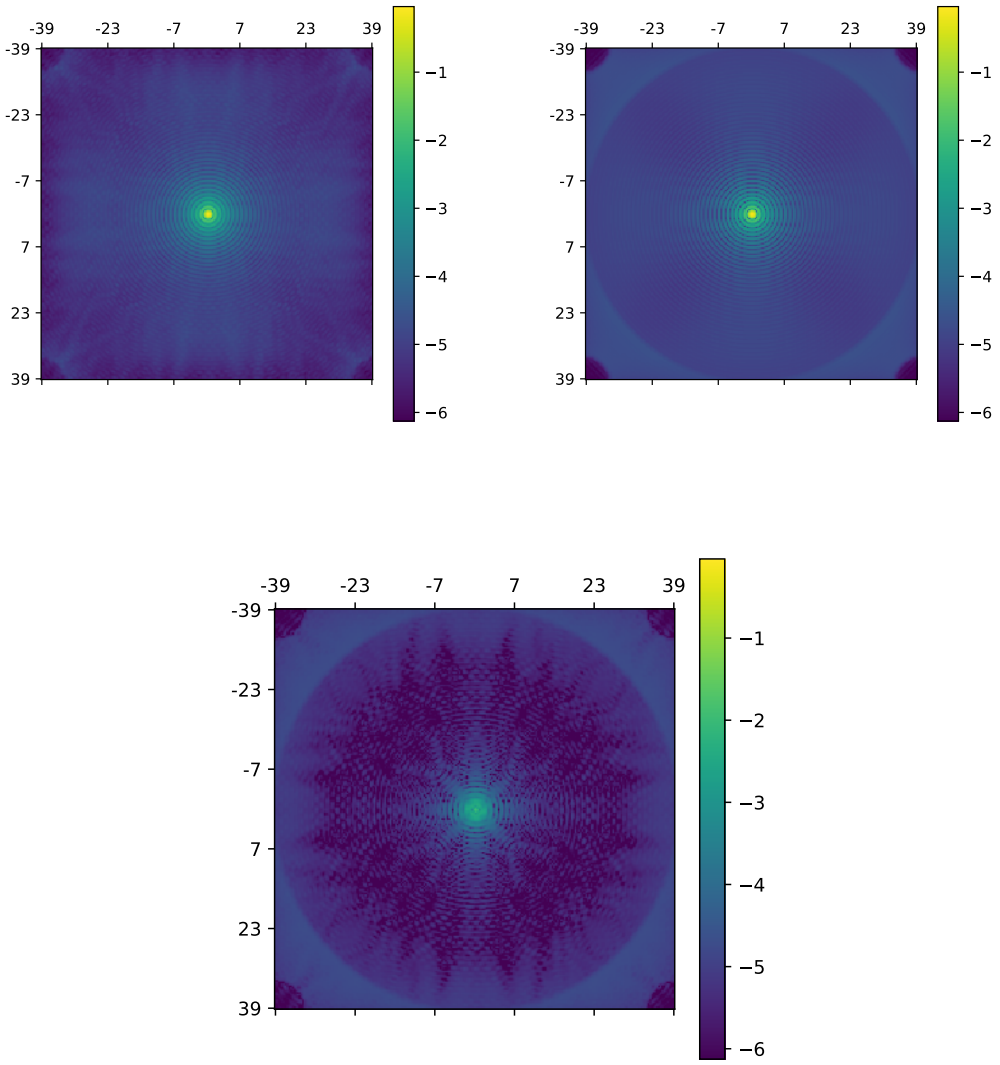

FIGURE 6.3 – FEP reconstruites à partir de la matrice de covariance de l'erreur de repliement. En haut à gauche : la matrice a été calculée à partir des données de ROKET. En haut à droite : la matrice a été modélisée par GROOT. En bas : différence entre les 2 FEP précédentes

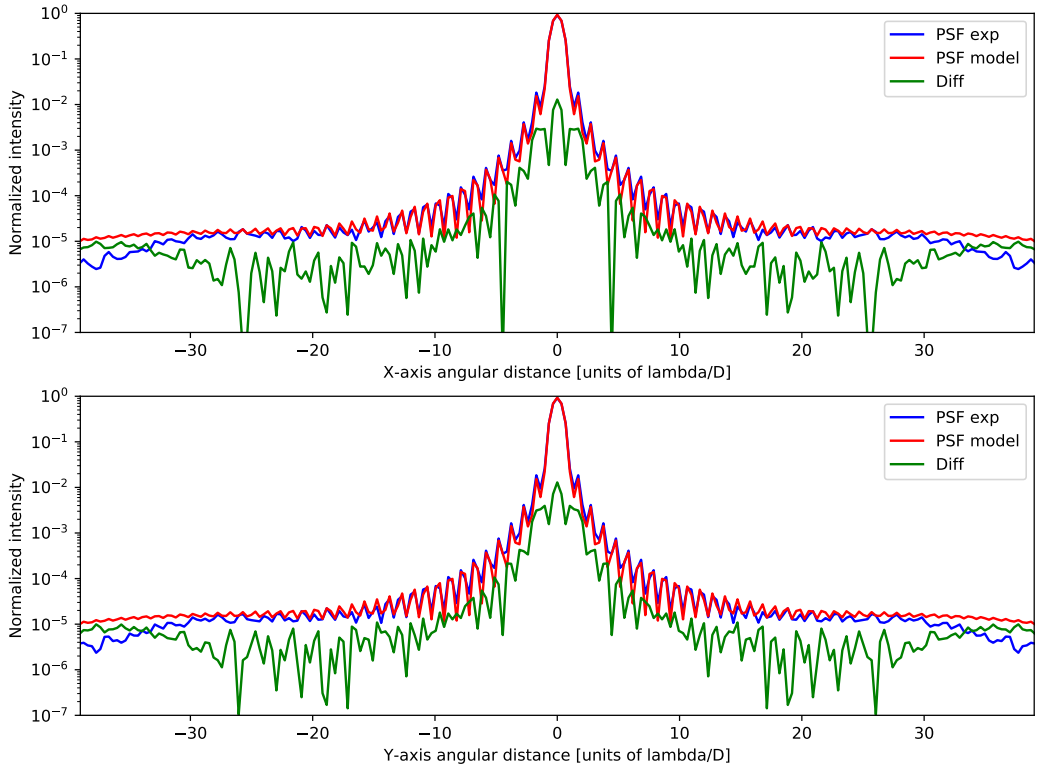

FIGURE 6.4 – Coupes de la FEP reconstruite à partir de la matrice de covariance de l'erreur de repliement. En haut : coupe selon l'axe X, en bas : coupe selon l'axe Y. En bleu : FEP obtenue à partir de ROKET, en rouge : FEP obtenue à partir du modèle, en vert : différence entre les 2 FEP. Échelle logarithmique

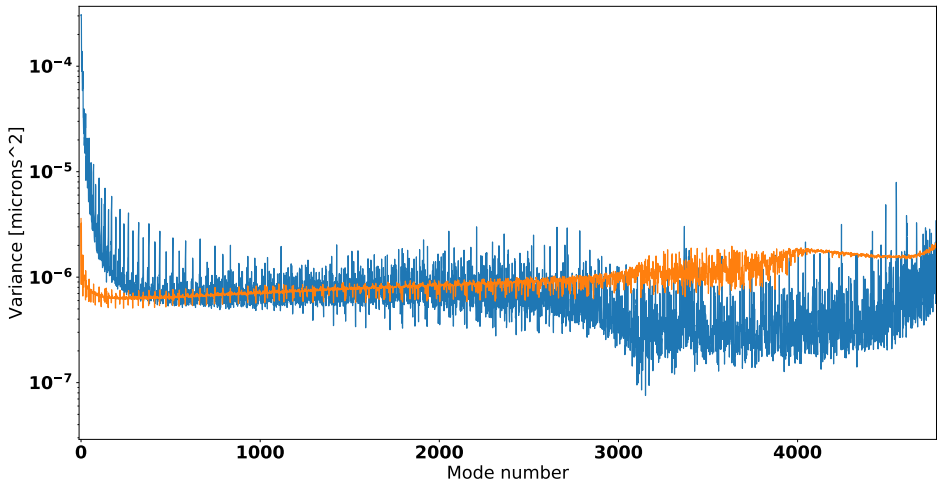

FIGURE 6.5 – Variance de l'erreur de repliement estimée sur chaque mode de la base modale. En bleu : estimation à partir des données de ROKET, en orange : estimation à partir du modèle

des hautes fréquences au-delà du 3000^e mode. L'estimation des fréquences intermédiaires semble, quant à elle, correcte.

D'où peut provenir une telle différence d'allure entre ces deux spectres ? J'ai exploré plusieurs pistes.

Le nombre de points de discrétisation

Je me suis tout d'abord intéressé à la discrétisation que j'avais choisi, à savoir sur 5 points (cf. Éq. (6.5)). J'ai fait varier N de 3 à 9 points. La Figure 6.6 montre alors une coupe des FEP obtenues pour chaque valeur de N. On constate qu'au-delà de 3 points de discrétisation, l'impact sur la FEP reconstruite est négligeable. En effet, les FEP obtenues pour $N = 5$ et $N = 9$ peuvent à peine être distinguées sur la Figure 6.6, alors que l'on peut observer une estimation légèrement différente des hautes fréquences pour $N = 3$.

Il faut néanmoins noter que même pour $N = 3$, la différence reste acceptable et il peut alors être intéressant de n'utiliser que 3 points de discrétisation afin de minimiser la complexité numérique de l'algorithme.

Le filtrage temporel de la boucle

Le modèle de l'erreur de repliement que j'ai développé dans la Section 6.2.2 n'inclut pas le filtrage temporel de la boucle d'OA. Pour estimer l'impact de ce filtrage sur l'erreur de repliement, j'ai modifié ROKET pour qu'il enregistre à chaque itération l'erreur de repliement sur la mesure telle que définie par l'Éq. (6.2). La composante

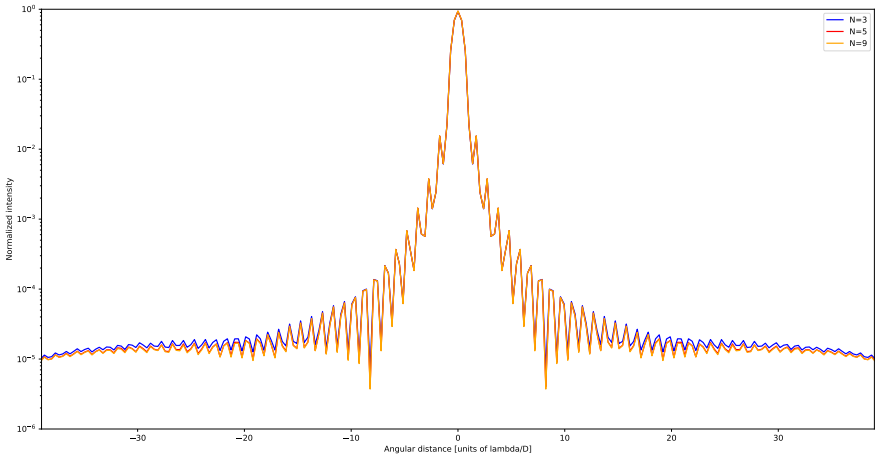

FIGURE 6.6 – Coupe de FEP obtenues pour à partir du modèle de repliement pour différentes discrétisations. Courbe bleue : $N = 3$, courbe rouge : $N = 5$, courbe orange : $N = 9$

orthogonale $\phi_\perp$ de la phase n'étant pas impactée par la boucle d'OA, la matrice de covariance de ces mesures est alors équivalente à la matrice calculée par le modèle. En appliquant alors la même série d'opérations aux deux matrices pour aboutir aux variances sur les modes, je suis capable de calculer le spectre de l'erreur de repliement avec et sans filtrage par la boucle. Ces spectres sont représentés sur la Figure 6.7 et, comme on peut le constater, il s'avère que le filtrage par la boucle n'a que très peu d'effet sur ce dernier.

Les sous-pupilles en bord de pupille et la stationnarité de la phase

Je reviens ici sur l'une des hypothèses émises pour aboutir au modèle de l'erreur de repliement, à savoir que l'on considère que toutes les sous-pupilles de l'ASO sont strictement identiques. Dans la pratique, certaines sous-pupilles situées au bord de la pupille du télescope ou autour de l'obstruction centrale ne sont pas uniformément illuminées.

Pourquoi m'intéresser à cette hypothèse particulièrement ? Il suffit de regarder l'allure de la carte de covariance de l'erreur de repliement obtenue avec ROKET (Figure 6.8). La carte de covariance permet une représentation spatiale de la matrice de covariance. On observe ainsi que la covariance entre les sous-pupilles diamétralement opposées est plus élevée qu'ailleurs.

Une nouvelle fois, j'ai voulu estimer l'impact de ces covariances plus élevées. À partir de la carte de covariance précédente, j'annule les covariances situées sur le bord de la carte, et je reconstruis la matrice de covariance correspondante à cette carte modifiée. Notons également que, ce faisant, je stationnarise la phase puisque la carte

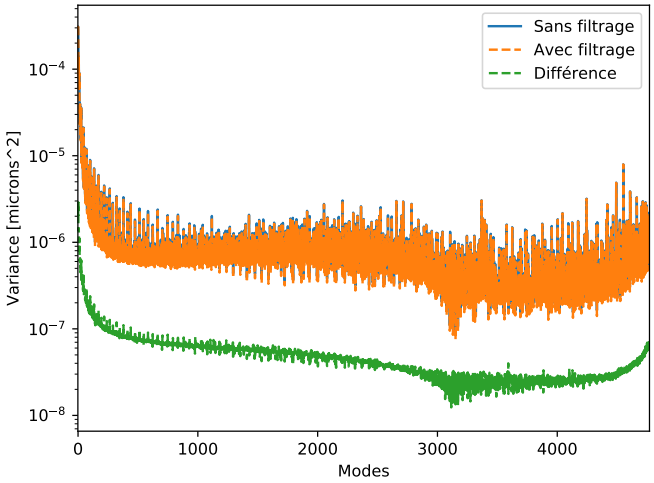

FIGURE 6.7 – Variance de l'erreur de repliement estimée sur chaque mode de la base modale. En bleu : sans filtrage par la boucle d'OA. En orange : avec filtrage par la boucle. En vert : valeur absolue de la différence entre les courbes bleue et verte.

de covariance moyenne les sous-pupilles ayant une séparation identique.

Au final, ces modifications n'influent que très peu sur le spectre, qui conserve toujours l'allure observable sur la Figure 6.5.

Échantillonnage de la simulation

Une autre possibilité serait que le repliement estimé par ROKET ne soit pas tout à fait juste. Pour rappel, ROKET estime le repliement sur les mesures de l'ASO en calculant directement le gradient moyen de la phase $\phi_\perp$ sur chaque sous-pupille. Cette phase n'étant composée que de hautes fréquences, son échantillonnage spatial de cette dernière a un impact direct sur la mesure du gradient.

Pour écarter la possibilité que l'échantillonnage ne soit pas assez fin, j'ai effectué une nouvelle simulation en sur-échantillonnant très largement par rapport à l'échantillonnage classique de COMPASS, à hauteur de 4 fois ce dernier. Une nouvelle fois, le spectre du repliement n'a pas changé par rapport à celui de la Figure 6.5.

Conclusion

Au final, il m'est difficile d'identifier l'origine d'une telle différence entre le spectre du modèle et celui obtenu à partir des estimations de ROKET. Intuitivement, on s'attendrait plutôt à avoir un spectre ressemblant à celui du modèle dans la mesure où le repliement des hautes fréquences a un impact plus conséquent autour de la fréquence de coupure. Le spectre du modèle est ainsi quasiment plat jusqu'à ce qu'il s'accroisse sur les hautes fréquences. L'origine des basses fréquences spatiales sur le

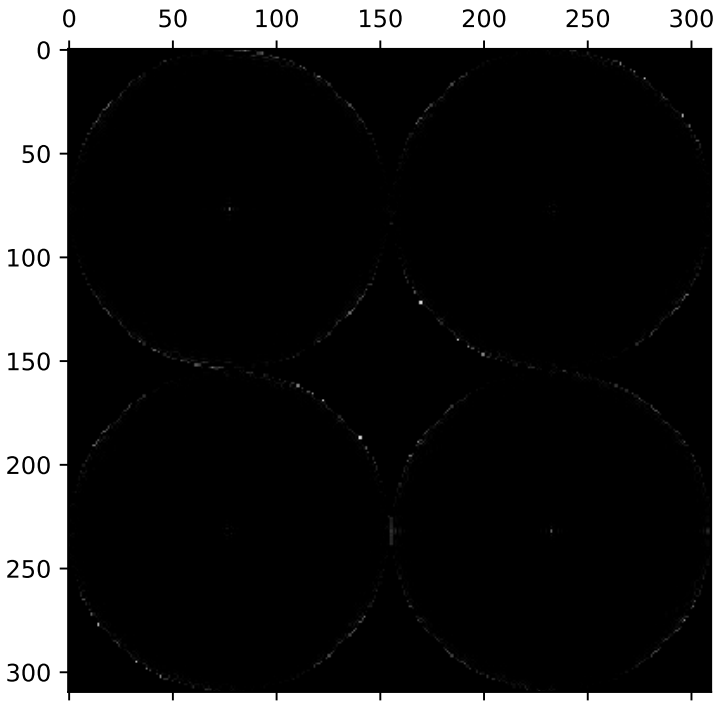

FIGURE 6.8 – Carte de covariance de l'erreur de repliement obtenue à partir de la matrice de covariance de ROKET. L'échelle de couleur a été adaptée afin de faire plus nettement les covariances en bord de pupille.

spectre obtenu à partir de ROKET est difficilement explicable de ce point de vue, ce qui ne me permet pas d'exclure l'existence d'un problème résiduel qui conduirait à une mauvaise estimation des basses fréquences.

D'un autre côté, j'ai montré dans l'article A&A Section 5.2 ainsi que dans la Section 5.4 que les estimations faites par ROKET permettaient bien de reconstruire la FEP obtenue par COMPASS. Si le repliement est mal estimé, l'impact de l'erreur commise est faible, ce qui est d'ailleurs confirmé par la Figure 6.4 où l'on a constaté que les différences entre les FEP restaient largement acceptables.

À mon sens, le modèle développé ici pour l'erreur de repliement reste donc utilisable dans la mesure où les résultats obtenus sur un dimensionnement typique d'ASO Shack-Hartmann sur l'ELT permettent malgré tout d'obtenir une estimation de l'impact de cette erreur sur la FEP finale.

6.3.3 Erreur de sous-modélisation

La Figure 6.9 montre les FEP obtenues en ne considérant que l'erreur de sous-modélisation, et la Figure 6.10 compare les coupes en X et en Y de ces FEP. La FEP modélisée présente un rapport de Strehl de 84 % contre 82 % pour celle estimée par ROKET, avec une énergie encadrée de 81 % contre 80 %.

La modélisation radiale de la fonction de structure se traduit de manière évidente sur la FEP du modèle alors que la FEP ROKET présente une structure carrée du fait du maillage carré du miroir déformable simulé. On notera sur la Figure 6.9 que la différence entre la FEP modélisée et la FEP estimée par ROKET fait apparaître des motifs carrés.

Cette différence de structure explique les différences visibles sur les FEP. L'introduction dans le modèle d'une fonction de structure prenant en compte un maillage carré ou hexagonal pourrait être intéressante pour aboutir à de meilleures performances, mais cela se fera sûrement au prix d'une complexité numérique accrue.

6.3.4 Estimation de la FEP totale à partir des modèles

À partir des modèles développées dans cette section, et en considérant que les déviations des mesures de l'ASO (hors gain optique) sont négligeables, ce qui n'est pas une hypothèse forte dans le cas présent compte tenu du budget d'erreur obtenu (cf. Tableau 5.2), nous pouvons estimer la FEP du système et la comparer à la FEP obtenue par la simulation COMPASS, toujours à partir des paramètres donnés par le Tableau 5.1. L'estimation faites à partir des modèles inclut donc les erreurs d'anisoplanétisme, temporelle, de repliement et de sous-modélisation. L'erreur due au bruit sur les mesures est directement ajoutée à partir de l'estimation qui en est faites par ROKET.

La Figure 6.11 représente ainsi ces 2 FEP ainsi que leur différence, et la Figure 6.12 les coupes en X et en Y correspondantes.

La FEP modélisée se caractérise par un rapport de Strehl de 61 % contre 57 % pour la FEP simulée par COMPASS, et par une énergie encadrée de 65 % contre 66 %

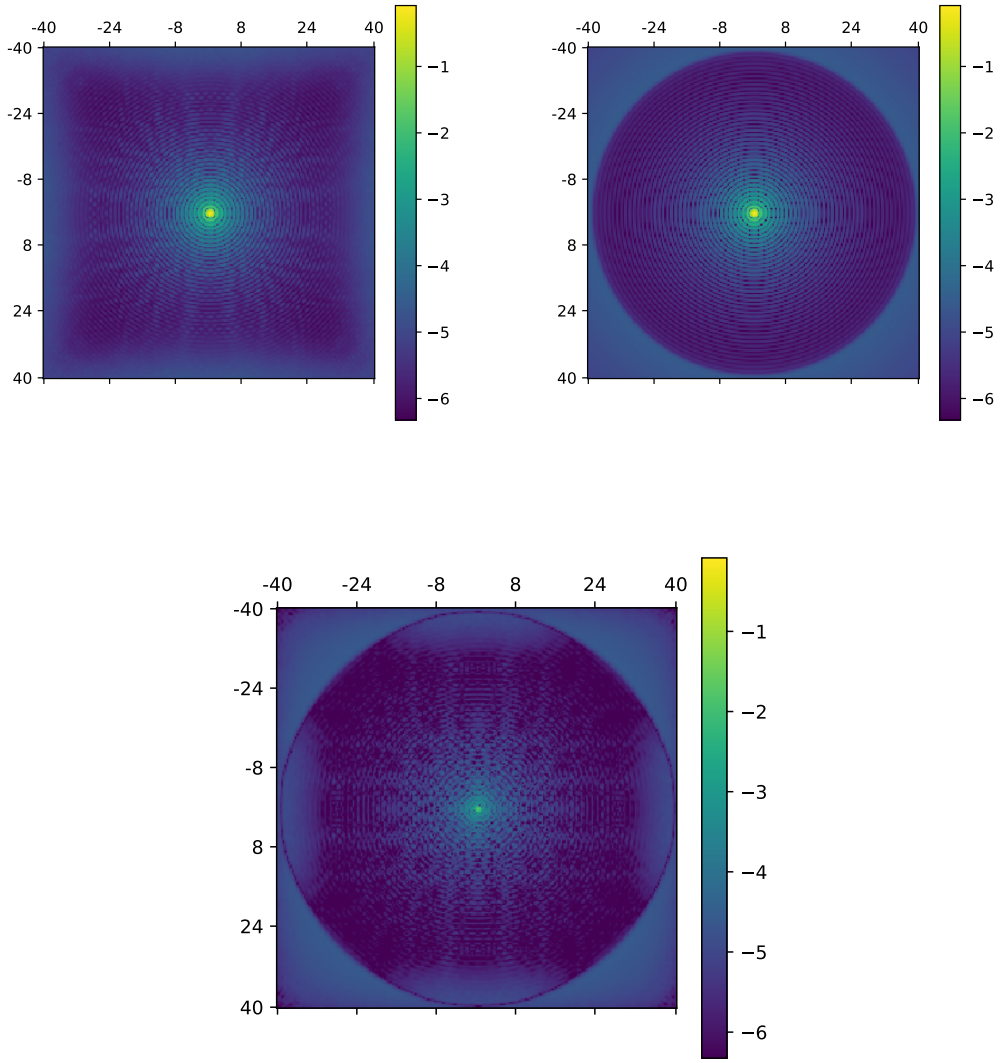

FIGURE 6.9 – FEP obtenues en ne considérant que l'erreur de sous-modélisation. En haut à gauche : FEP estimée par ROKET. En haut à droite : FEP modélisée. En bas : différence entre les 2 FEP précédentes

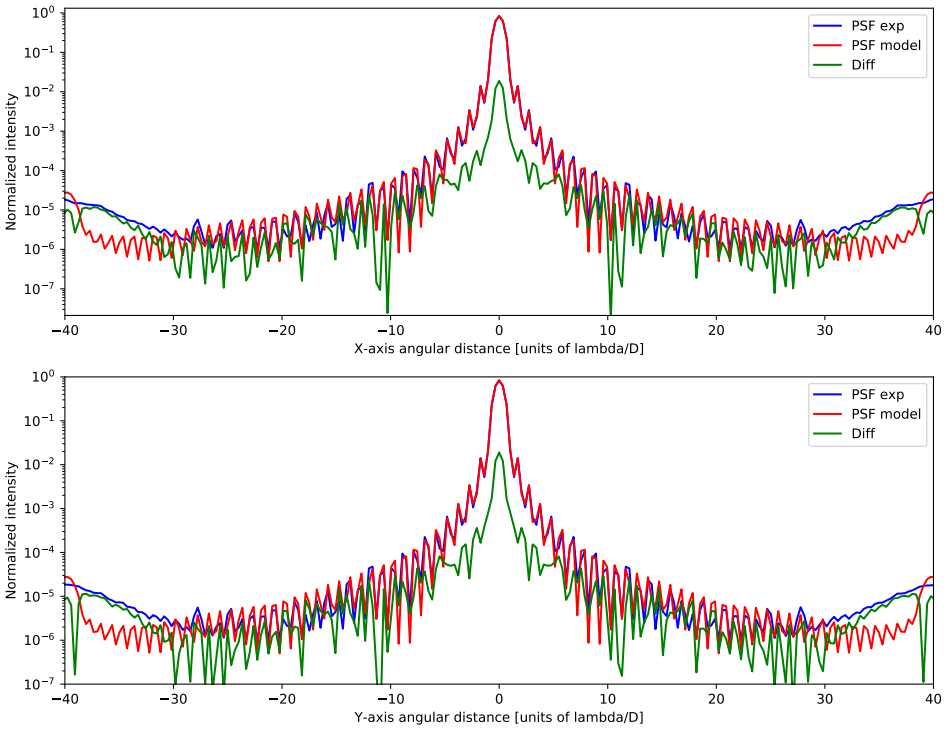

FIGURE 6.10 – Coupes de la FEP obtenue en ne considérant que l'erreur de sous-modélisation. En haut : coupe selon l'axe X, en bas : coupe selon l'axe Y. En bleu : FEP estimée par ROKET, en rouge : FEP obtenue à partir du modèle, en vert : différence entre les 2 FEP. Échelle logarithmique

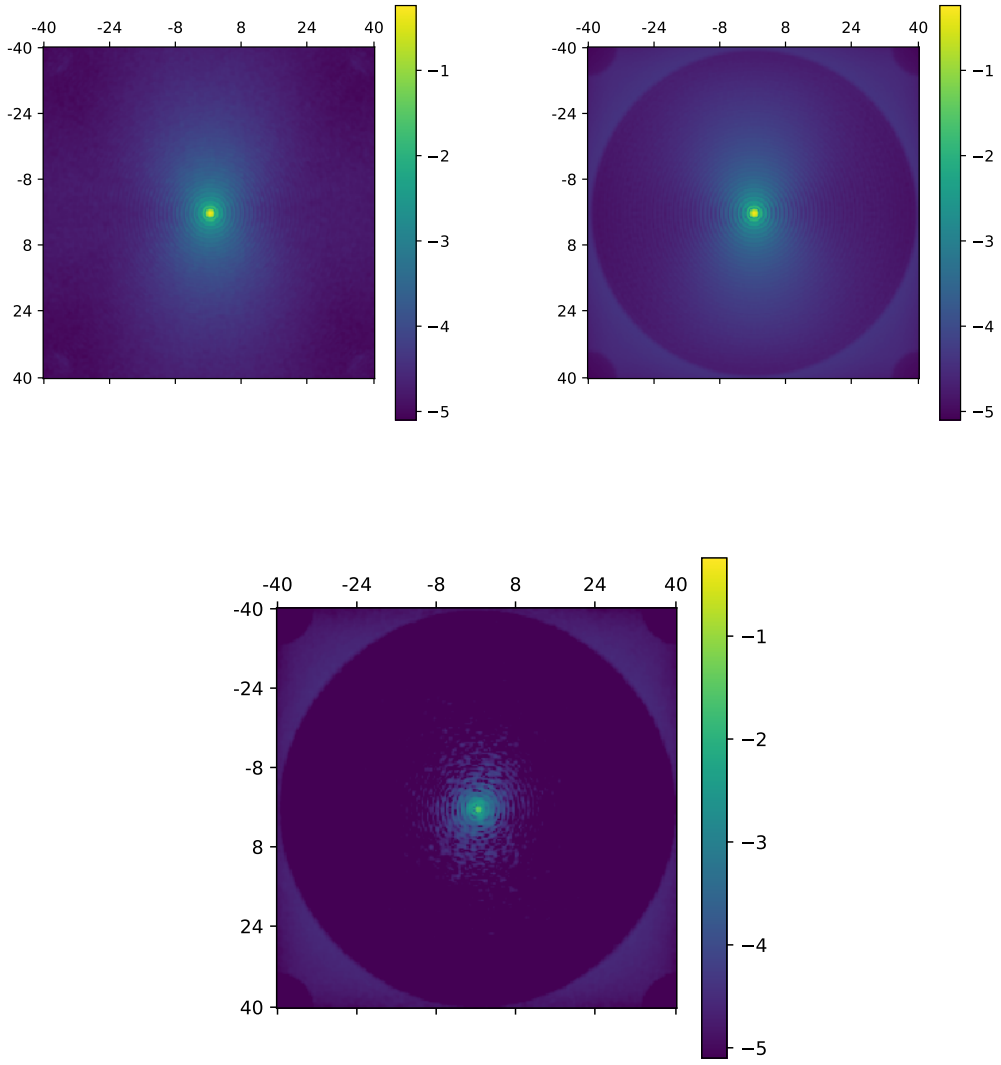

FIGURE 6.11 – FEP du système d'OA modélisée avec GROOT. En haut à gauche : FEP simulée par COMPASS. En haut à droite : FEP modélisée par GROOT. En bas : différence entre les 2 FEP précédentes

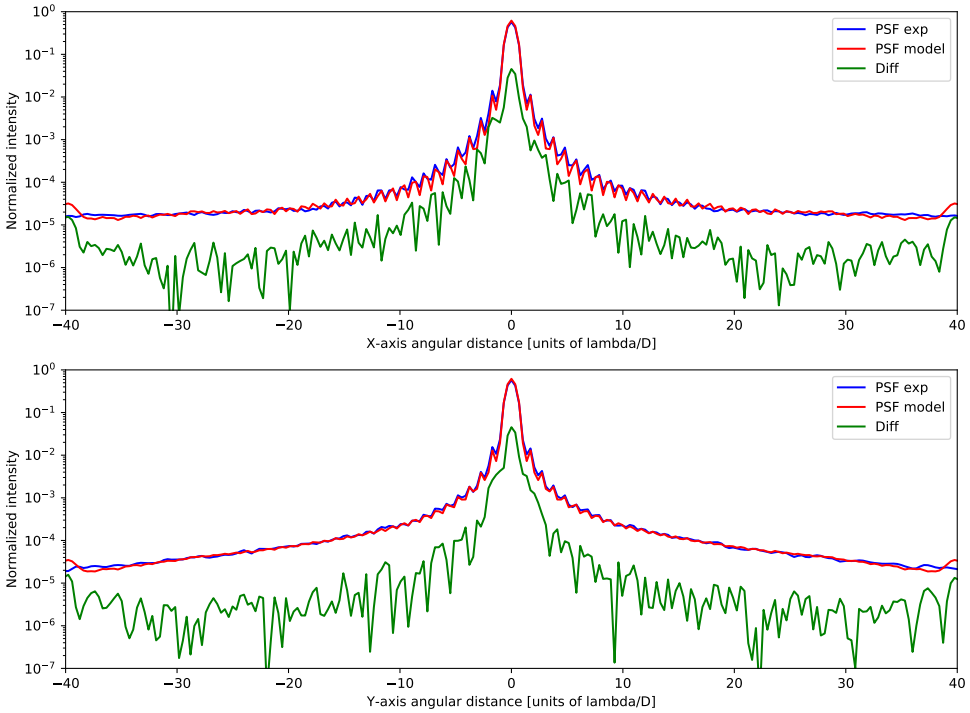

FIGURE 6.12 – Coupes de la FEP modélisée par GROOT ainsi que de la FEP COMPASS. En haut : coupe selon l'axe X, en bas : coupe selon l'axe Y. En bleu : FEP estimée par ROKET, en rouge : FEP obtenue à partir du modèle, en vert : différence entre les 2 FEP. Échelle logarithmique

pour COMPASS.

Au final, les modèles développés au sein de GROOT permettent d'estimer la FEP avec une différence de 4 % sur le rapport de Strehl, et dans les limites définies par les différents modèles. Cette différence équivaut, à la longueur d'onde d'observation (1,65 μm), à une erreur sur le front d'onde de 68 nm rms.

Ce résultat est évidemment à mettre en perspectives avec le temps de calcul nécessaire à obtenir la FEP. Ainsi, toujours sur le serveur DGX-1 décrit dans la Section 5.3.3, il faut à COMPASS environ 140 secondes pour effectuer les 40000 itérations qui ont conduit à la FEP précédente, auxquelles il faut ajouter environ 5 minutes de temps d'initialisation de la simulation. Pour GROOT, le calcul des matrices de covariance se prêtant parfaitement à la parallélisation sur GPU étant donné que chaque élément de cette matrice peut être calculé indépendamment des autres, il faut compter environ 2 secondes pour obtenir la matrice de covariance de l'erreur résiduelle incluant les erreurs d'anisoplanétisme, temporelle et de repliement. Le module GAMORA (Section 5.3.3) permet alors de reconstruire la FEP à partir de cette matrice en 30 secondes sur 2 GPUs.

6.4 Identification des paramètres nécessaires aux modèles

6.4.1 Paramètres nécessaires aux modèles

Je propose de faire l'examen des paramètres nécessaires aux modèles développés dans la section précédente. Ces paramètres peuvent être divisés en deux catégories distinctes :

- **Les paramètres géométriques** qui dépendent uniquement du système d'OA utilisé.
- **Les paramètres turbulents** qui dépendent de la turbulence atmosphérique

Les premiers sont identifiables pendant la phase de calibration de l'instrument. Pour alimenter les modèles composant GROOT, on aura ainsi besoin de déterminer :

- la position des actionneurs dans la pupille
- les fonctions d'influences du miroir déformable
- la position des sous-pupilles dans la pupille
- la position de l'étoile guide dans le champ
- la fréquence de fonctionnement de l'OA
- le gain de la boucle d'OA
- le retard de la boucle

Pour ce qui est des paramètres turbulents, les modèles nécessitent la connaissance :

- du paramètre de Fried r_0
- du profil turbulent (nombre couches, altitude, $C_n^2(h)$,...)
- du profil de vent (vitesse et direction)
- de l'échelle externe L_0 : si cette dernière dépend de l'altitude, on pourra néanmoins se contenter du L_0 moyen étant donné qu'il est difficile de déterminer

un tel profil

L'identification de ces paramètres n'est pas triviale. De nombreux travaux traitent de méthodes permettant d'estimer ces paramètres, soit à partir des données de télémétrie, soit avec un instrument dédié. Ainsi, Véran et al. (1997) donne une méthode itérative permettant d'obtenir une estimation du paramètre de Fried à partir des données de télémétrie de l'OA, et Vernin & Roddier (1973) utilise les motifs de scintillation propres à chaque couche turbulente pour identifier le profil turbulent par triangulation. La méthode connue sous le nom de SCIDAR [2] apparaît avec Rocca et al. (1974) par observation d'une étoile double et corrélation spatio-temporelle et spatio-angulaire des motifs de scintillation.

La méthode du SLODAR [3] se rapproche du SCIDAR, à la différence qu'elle se base sur les mesures faites par un ASO de type Shack-Hartmann sur une étoile binaire (Wilson, 2002). En calculant la corrélation spatiale des mesures entre les deux étoiles, les pics de corrélation permettent d'identifier le nombre de couches turbulentes, leurs altitudes et C_n^2 respectifs. Wang et al. (2008) propose également de calculer la corrélation temporelle : les mouvements des pics de corrélation permettent alors en plus d'identifier la vitesse et la direction du vent sur chaque couche.

Plusieurs auteurs ont déjà proposé de calculer la corrélation spatio-temporelle des mesures, et montré l'existence de couches turbulentes (Gendron & Léna, 1996; Schoeck & Spillar, 1998). L'arrivée sur le ciel des optiques adaptatives multi-analyseurs (GeMs, puis CANARY en 2010) re-déclenche la ruée vers la caractérisation du profil turbulent à l'aide des mesures des analyseurs. Dans cet ensemble de publications, les auteurs cherchent dans tous les cas à établir les cartes de covariances spatiales des mesures avec décalage temporel. En 2009 pourtant, un autre type d'optique adaptative entre en jeu : les optiques adaptatives pour le haut contraste, telles que GPI ou SPHERE. Dans ce type d'instrument où la meilleure performance est souhaitée, le contrôle prédictif a une place privilégiée et l'hypothèse de la turbulence gelée et de sa caractérisation devient importante (Poyneer et al., 2009).

Si la méthode peut être utilisée avec les mesures de système d'OA de type tomographique, i.e. multi-sources et multi-analyseurs (Vidal et al., 2010; Cortés et al., 2012; Vidal et al., 2014b; Sivo et al., 2018), son application directe s'avère plus difficile pour des systèmes de type SCAO qui ne dispose que d'un unique ASO. Je souhaite proposer ici une méthode inspirée du SLODAR, mais applicable à partir de la télémétrie d'un système d'OA SCAO, et permettant d'obtenir une estimation du nombre de couches turbulentes et des conditions de vent pour chacune d'entre elles.

6.4.2 Principe de la méthode

Sous l'hypothèse de turbulence gelée, une couche turbulente se déplace uniformément dans la pupille selon un vent $\mathbf{v}$. Dès lors, la même "mesure" d'une portion de cette couche doit se retrouver sur différentes sous-pupilles de l'ASO à des instants

2. *SCIntillation Detection And Ranging* en anglais
3. *SLOpes Detection And Ranging* en anglais

différents. La covariance spatiale des mesures est une fonction assez large : on sait que les pentes locales du front d'onde mesurées par les sous- pupilles sont corrélées sur de grandes distances. Retrouver les contributions individuelles des couches peut s'avérer délicat car, comme dans la méthode du SLODAR, il faut mettre en œuvre un processus d'inversion pour les retrouver, lequel s'avère d'autant plus risqué que la réponse impulsionnelle des couches est large. D'autre part, le spectre temporel des mesures est fortement décroissant. Il contient une grande quantité d'énergie à basse fréquence temporelle. Ce sont ces basses fréquences qui sont responsables de la corrélation spatiale élargie et importantes sur l'ensemble de la pupille. En effet, qui dit corrélation spatiale large et étendue dit forcément mode d'ordre bas, associé à un phénomène lent. Or, il semble physiquement improbable que des modes de bas ordre, à variation lente, puisse nous renseigner sur le déplacement des couches turbulentes, pour lequel on s'attendrait plutôt à trouver l'information dans les hautes fréquence spatiales, aux courtes échelles de temps. Ce sont également ces basses fréquences temporelles qui sont responsables des lenteurs et donc des difficultés de convergence des matrices de covariance (Martin et al., 2012).

Pour ces raisons, j'ai décidé de m'interesser à la dérivée temporelle des pentes, donc à la vitesse de déplacement des spots. Si des couches sont en mouvement, la vitesse des spots devrait également être corrélée spatialement mais avec une pondération moindre des mouvements lents. Cependant, la dérivée temporelle des pentes est extrêmement bruitée. Je propose donc de calculer la covariance entre les pentes et leur dérivée temporelle, en utilisant la formule mathématique symétrique suivante :

$$C(\delta_{i,j}, \delta t) = \frac{\langle \sum_{i,j} (s_i(t) + s_i(t+\delta t))(s_{i+\delta_{i,j}}(t+\delta t) - s_{i+\delta_{i,j}}(t)) \rangle}{O(\delta_{i,j})} \tag{6.17}$$

ayant la même séparation $\delta_{i,j}$ entre les sous-pupilles i et j. $\sum_{i,j}$ désigne la somme sur tous les couples de sous-pupilles (i, j) ayant une séparation $\delta_{i,j}$, et $O(\delta_{i,j})$) le nombre de ces couples.

Pour ce faire, on commencera par exemple par calculer la matrice de covariance des mesures avec leurs dérivées temporelles C_{ssdt} :

$$C_{ssdt} = \langle (s_i(t) + s_i(t+\delta t))(s_j(t+\delta t) - s_j(t)) \rangle \tag{6.18}$$

Cette formulation symétrique présente l'avantage de minimiser le bruit sur les mesures du calcul. En effet, en supposant que le bruit n'est pas corrélé entre différentes sous-pupilles, ce dernier ne peut donc pas avoir d'impact sur la covariance d'une sous-pupille avec elle-même. Or, pour $i = j$, l'équation précédente se réécrit aisément sous la forme :

$$C^{ii}_{ssdt} = \langle s_i(t)^2 \rangle - \langle s_i(t+\delta t)^2 \rangle \tag{6.19}$$

ce qui est évidemment nul.

Une fois la matrice de covariance calculée, la carte de corrélation spatiale peut être aisément obtenue en moyennant les covariances correspondantes à tous les couples de sous-pupilles avec une séparation identique. Le mouvement des pics de corrélation

obtenus pour différent décalage δt nous permet alors d'estimer la vitesse et la direction du vent.

L'intérêt de cette méthode est qu'elle permettrait de remonter au profil du vent sur les couches turbulentes directement à partir de la télémétrie de l'OA, même sur un système SCAO. J'intuite également que la détermination du profil $C_n^2(h)$ doit être possible par cette méthode, même si nous verrons que cela est plus difficile. L'absence de triangulation la rend cependant moins complète que les techniques classiques de SLODAR, dans la mesure où elle ne permettra pas de remonter aux altitudes des couches turbulentes.

6.4.3 Simulation numérique

Je considère ici un système SCAO identique à celui décrit dans l'article A&A (Section 5.2, section 2.4.1 de l'article). L'atmosphère est constituée de 3 couches distinctes situées au sol, à 5000 et 9000 m respectivement. Sur une simulation de 20000 itérations, les mesures en boucle fermée de l'ASO ont été enregistrées à chaque itération.

La Figure 6.13 représente la carte de corrélation des mesures selon l'axe X avec leurs dérivées temporelle, pour différents décalage δt. Les cartes font clairement apparaître 3 pics de corrélations et d'anti-corrélation dont les positions varient selon le décalage temporel utilisé, et dont on notera la finesse. Connaissant la taille physique d'une sous-pupille d, la position d'un pic de covariance par rapport au centre nous donne la vitesse du vent correspondante, et la direction du déplacement du pic donne la direction du vent. Le Tableau 6.1 donne alors les estimations réalisées à partir de ces cartes de corrélation ainsi que les valeurs effectivement utilisées par la simulation. Les résultats montrent ainsi qu'une estimation correcte des conditions de vent

Couche	**Vitesse du vent estimée [m/s]**	**Vitesse du vent simulée [m/s]**
1	5,2	5,6
2	11,6	12,1
3	7,0	6,4
	Direction du vent estimée [deg]	**Direction du vent simulée [deg]**
1	339	344
2	216	217
3	21	24

Tableau 6.1 – Estimations des vitesses et directions de vent sur chaque couche faites à partir de la carte de corrélation et comparaison avec les valeurs effectives de la simulation

sur les couches turbulentes est bien possible grâce à cette méthode. J'ai d'ailleurs pu l'appliquer sur des données ciel provenant de l'instrument CANARY.

Il pourrait également être intéressant de considérer une reconstruction des mesures en boucle ouverte à partir des mesures en boucles fermées, des commandes envoyées au miroir déformable et de la matrice d'interaction du système. L'exploitation de la

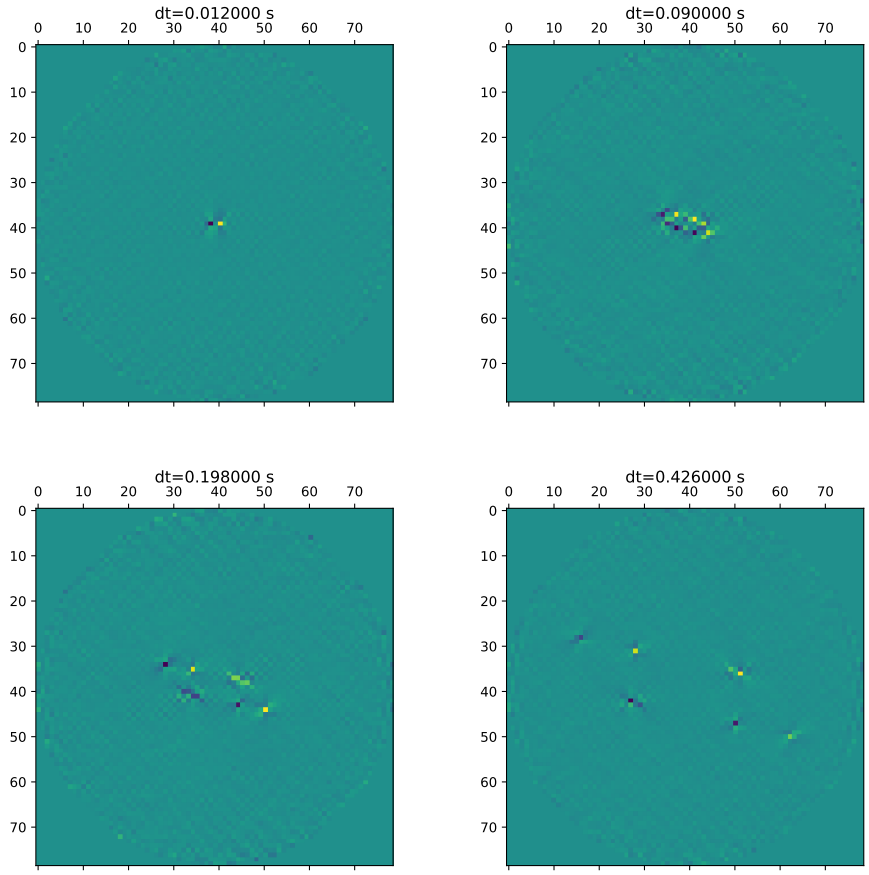

FIGURE 6.13 – Carte de corrélation spatiale des mesures de l'ASO selon l'axe X avec leurs dérivées temporelles pour différents décalages temporels δt

méthode dans ce contexte pourrait permettre de remonter plus facilement au profil $C_n^2(h)$. Malheureusement, le temps m'a manqué pour pouvoir tester cette hypothèse.

6.4.4 Résultats sur données CANARY

CANARY est le démonstrateur de MOAO installé au William Herchel Telescope, aux Iles Canaries (Gendron et al., 2011). Pour tester la méthode d'identification, j'ai utilisé des mesures prises lorsque l'instrument fonctionnait en mode SCAO, lors d'une nuit observation d'octobre 2015. L'ASO était alors un Shack-Hartmann de 14x14 sous-pupilles (Gendron et al., 2016). Avec ces données, je dispose également de données SCIDAR mesurées au même moment et qui permettent donc de connaître le profil turbulent auquel était soumis le système d'OA.

La carte de corrélation de ces mesures avec leurs dérivées temporelles, présentée par la Figure 6.14, fait ainsi clairement apparaître 2 couches turbulentes, avec des vitesses de vent similaires mais des directions légèrement différentes. Ainsi, à partir de

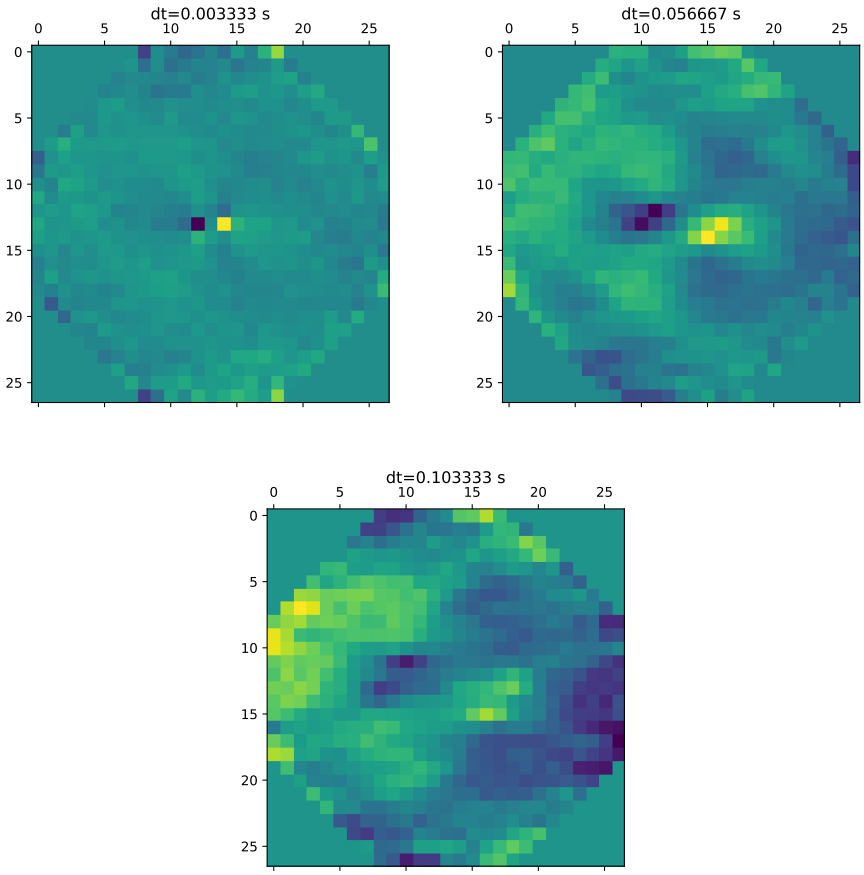

FIGURE 6.14 – Carte de corrélation spatiale des mesures de l'ASO sur CANARY selon l'axe X avec leurs dérivées temporelles pour différents décalages temporels δt

cette carte, on mesure une vitesse de vent de 12 m/s et 14 m/s pour les deux couches, ainsi que des directions de 90 et 124 degrés.

À titre de comparaison, la Figure 6.15 montre les images du SCIDAR obtenues en parallèle des mesures de l'ASO qui ont été utilisées précédemment (Osborn et al., 2015). On y voit ainsi clairement apparaître plusieurs pics de corrélation se déplaçant

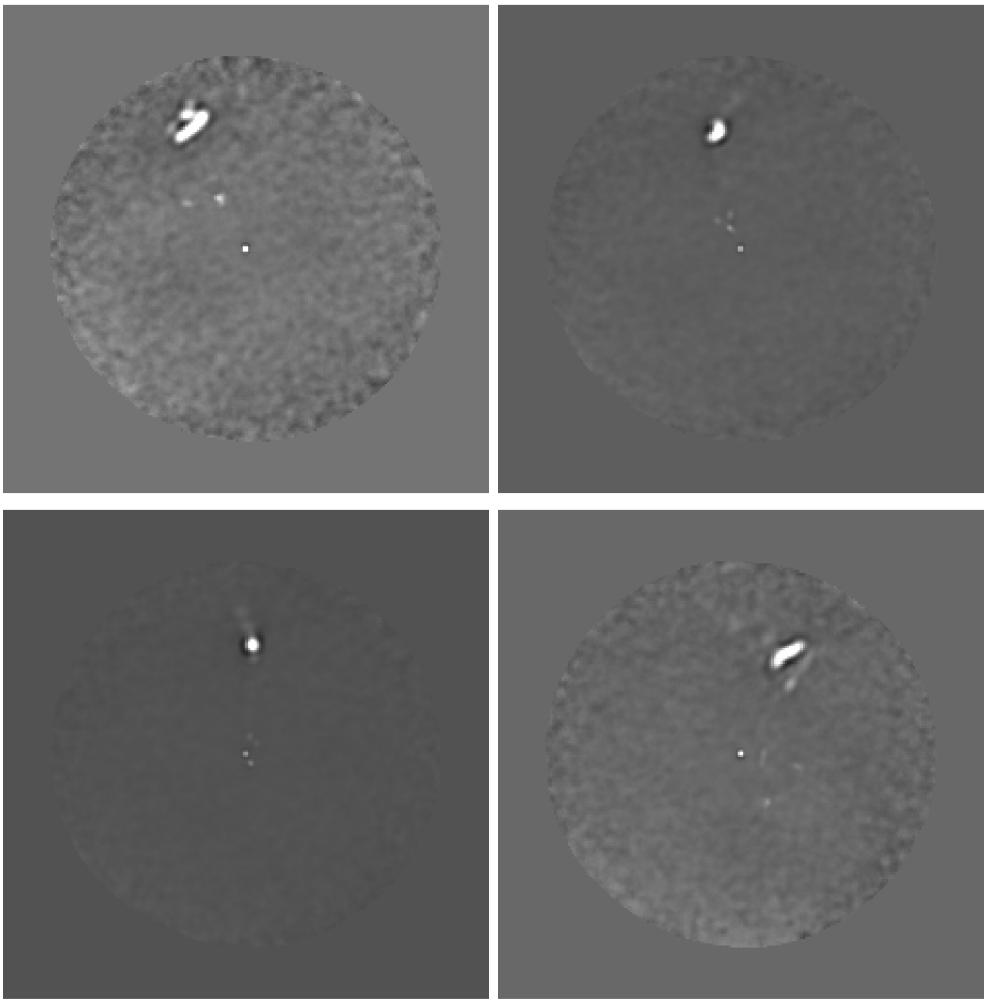

FIGURE 6.15 – Images du SCIDAR lors de l'observation CANARY. La chronologie démarre en haut à gauche, puis continue en haut à droite, en bas à gauche, et enfin en bas à droite. Les données ont été enregistrées au même moment que les mesures de l'ASO

sensiblement dans la même direction et avec des vitesses semblables également. On notera que la direction du déplacement correspond à celle du déplacement observé dans la Figure 6.14. Au final, le SCIDAR détecte une épaisse couche en altitude (ou plusieurs couches proches les unes des autres) qui domine (le profil mesuré est donné dans la Figure 6.16), avec des vitesses de vent comprises entre 10,5 et 14 m/s, et des

directions entre 92 et 115 degrés.

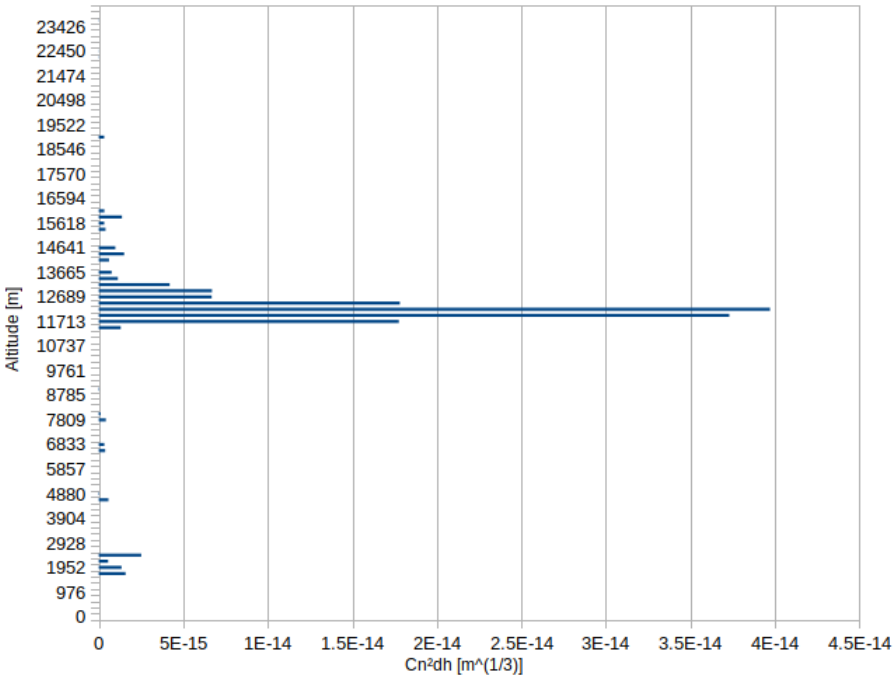

FIGURE 6.16 – Profil de $C_n^2(h)$ détecté par le SCIDAR

Au final, la résolution de la méthode présentée ici étant limitée par le dimensionnement du Shack-Hartmann, les mesures de la vitesse et de la direction du vent qu'elle fournit sont en accord avec les données SCIDAR.

6.5 Conclusion

Les modèles développés au sein de GROOT permettent une estimation rapide de la FEP. En effet, nous avons pu constater que l'erreur faite dans l'estimation de la FEP, comparée à une FEP simulée par COMPASS, représente une erreur sur la reconstruction du front d'onde de l'ordre de 70 nm rms. Cette erreur s'explique par certaines limitations induites par ces modèles, comme la symétrie circulaire des fonctions de structure utilisées par exemple.

De plus, le caractère hautement parallélisable du calcul des matrices de covariances nécessaires aux modèles convient parfaitement à une implémentation efficace sur GPU : il en résulte que la FEP peut être estimée en quelques secondes pour un système ELT.

D'autre part, les modèles ne nécessitent que peu de paramètres d'entrée dont la plupart peuvent être identifiés à partir des données de télémétrie enregistrées par le système d'OA en opération. La solution proposée précédemment permet en effet d'estimer la vitesse et la direction du vent simplement à partir des mesures de l'ASO,

et ce même sur un système de type SCAO. La comparaison entre les estimations faites par cette méthode sur des données d'observation ciel et des données SCIDAR montre des résultats prometteurs.

Je n'ai pas eu le temps de poursuivre le développement de cette méthode avant la fin de ma thèse. Faisant intervenir une dérivée temporelle, il semble logique que notre détermination soit d'autant plus sensible que les couches sont rapides.

Je m'attendais à trouver un coefficient de proportionnalité entre la hauteur du pic et le produit de la force et la vitesse de la couche : c'est faux, et visiblement les choses ne sont pas aussi simples. Une étude approfondie sera nécessaire si on souhaite utiliser l'intensité des pics pour le relier au Cn2(h). Cependant, la méthode offre déjà une identification des couches, et elle peut être couplée aux méthodes classiques dont j'ai déjà parlé pour retrouver l'intensité du profil, et retrouver également les couches très lentes et la couche au sol. Je pense que cette méthode vient en complément des méthodes traditionnelles d'analyse de la turbulence.

Conclusion & Perspectives

Synthèse

Dans la première partie de ce mémoire de thèse, j'ai introduit les principes fondateurs de l'optique adaptative ainsi que ses évolutions. Je suis ainsi revenu sur la formation d'images avec un télescope, et sur l'impact de la turbulence atmosphérique sur la qualité de ces dernières. J'ai alors décrit les différents composants d'un système d'OA, ainsi que les principes de contrôle temps-réel qu'un tel système implique. Nous avons vu que l'OA n'est pas parfaite, et j'ai décrit les différentes sources d'erreur qui avaient un impact sur ses performances. Ensuite, j'ai introduit quelques évolutions des systèmes d'OA avec notamment les étoiles laser et les systèmes d'OA grand champ utilisant plusieurs ASO et miroirs déformables. Enfin, j'ai rapidement décrit les caractéristiques du futur ELT et les défis qu'il représente en termes de design pour les systèmes d'optique adaptative prévus en première lumière.

Dans la seconde partie, j'ai introduit le calcul haute performance sur carte graphique. Nous avons vu que le défi numérique que représente l'ELT, que ce soit en termes de besoin de simulations numériques ou pour le contrôle temps-réel de l'OA, pouvait bénéficier de l'apport du calcul haute performance. Plus précisément, je me suis intéressé au calcul sur carte graphique : l'architecture matérielle de ce type de carte permet d'effectuer efficacement du calcul parallèle. Comme nous l'avons vu, cette dernière est particulièrement bien adaptée aux problèmes numériques à résoudre en OA.

J'ai alors décrit le développement de la plateforme de simulation pour les systèmes d'optique adaptative, COMPASS, à laquelle j'ai contribué significativement. Basée sur un cœur de calcul exécuté sur cartes graphiques, j'ai passé en revue les fonctionnalités de cet outil ainsi que son implémentation. Enfin, les performances de COMPASS en termes de temps d'exécution ont été comparées à celles d'un outil de simulation équivalent et bien connu de la communauté, à savoir YAO. J'ai ainsi montré un facteur d'accélération pouvant aller jusqu'à plus de 300 sur les cas les plus lourds à simuler.

Dans la dernière partie, j'ai décrit le développement, au sein de la plateforme COMPASS, d'un outil d'estimation du budget d'erreur de l'OA, appelé ROKET. ROKET permet ainsi d'obtenir une estimation des erreurs d'anisoplanétisme, temporelle, de repliement, de déviations de la mesure, des modes filtrés et du bruit. Cette estimation est faite directement sur les commandes envoyées au miroir déformable, et ce à chaque itération. Le budget d'erreur ainsi obtenu a pu être validé en l'utilisant pour alimenter un algorithme de reconstruction de FEP, et comparant la FEP ainsi obtenue à celle produite par COMPASS.

Enfin, j'ai décrit GROOT, un ensemble de modèles des contributions composant le budget d'erreur de l'OA. Ces modèles, basés sur le calcul de fonctions de structure de phase permettant d'obtenir une matrice de covariance, peuvent aussi bien être

utile pour la simulation numérique que pour la reconstruction de FEP. En effet, nous avons vu que ces modèles, couplés à GAMORA, un module GPU de reconstruction de FEP, pouvaient estimer la FEP d'un système d'OA en quelques secondes. De plus, ces modèles ne nécessitent que peu de paramètres, et nous avons vu que la plupart d'entre eux pouvaient être estimés à partir de l'étalonnage et de la télémétrie du système d'OA. Ces caractéristiques rendent ainsi ces modèles utiles pour la reconstruction de FEP sur le ciel.

Perspectives

Comme nous l'avons vu, l'ELT représente un défi considérable, notamment pour les systèmes d'optique adaptative. Dans le contexte de cette thèse, je me suis principalement intéressé au défi numérique que représente la simulation d'un système d'OA à cette échelle. En ne considérant que le système d'OA le plus simple, la SCAO, nous avons déjà un aperçu des difficultés liées au grand diamètre de ce télescope, notamment en termes d'efficacité du calcul et de gestion de l'empreinte mémoire.

L'exploitation du champ de vue de l'ELT nécessitera le développement de systèmes d'OA multi-analyseurs novateurs. Les difficultés numériques rencontrées avec la SCAO seront alors amplifiées, et ces systèmes d'OA complexes apporteront leur lot de difficultés spécifiques à appréhender. Ces dernières ne seront évidemment pas uniquement numériques : l'échelle de l'ELT et de ces instruments d'OA amènent également des questions plus fondamentales en terme d'instrumentation. À titre d'exemple, je pourrai citer l'élongation extrême des étoiles lasers en bord de pupille, qui pose le problème de l'efficacité des algorithmes de calcul du centre de gravité dans ces conditions.

Dans ce contexte, les travaux que j'ai pu mener durant cette thèse posent les bases de l'estimation numérique du budget d'erreur de l'OA et de sa modélisation dans le cadre de l'ELT. Sur ces bases pourront reposés les développements supplémentaires nécessaire aux systèmes d'OA les plus complexes. Je présente ici quelques uns de ces développements.

COMPASS

COMPASS est un outil en perpétuelle évolution : la perspective de l'ELT rend les performances de COMPASS attractives, si bien que son développement ne s'arrête pas, avec des mises à jour régulières. La distribution publique, via GitHub, ouvre également la voie à des développements extérieurs à l'équipe du LESIA.

Pour ce qui est des futurs développements internes au LESIA, plusieurs améliorations et réaménagements du code sont prévus à court et moyen termes. Ainsi, l'architecture logicielle va être allégée par l'utilisation de pyBind11 en lieu et place de Cython, pour la gestion de l'interface entre la partie utilisateur en Python et le cœur de calcul en C++/CUDA. Cette transition permettra d'homogénéiser l'architecture de COMPASS en supprimant un langage, et le temps de compilation sera également amoindri.

Côté performances, ces dernières pourront être encore améliorées par l'implémentation multi-GPU du modèle de Shack-Hartmann, plus précisément dans les cas multi-analyseurs. Ainsi, les analyseurs seront distribués sur les différents GPU disponibles et les modèles pourront alors être exécutés en parallèle. Le profil d'exécution actuel du code étant dominé par le modèle de l'ASO, cette future implémentation devrait fournir une accélération intéressante.

Côté fonctionnalité, l'outil sera continuellement enrichi. Je peux citer une généralisation du modèle de pyramide permettant de simuler une pyramide à N faces, ou encore de nouvelles méthodes de reconstruction du front d'onde. Ainsi, le code MOAO (Doucet et al., 2018), qui permet de calculer efficacement un reconstructeur tomographique pour différentes configurations comme la MOAO ou la MCAO, pourra être interfacé avec la simulation.

ROKET

Si ROKET fournit des résultats intéressants, il est pour le moment limité à l'étude de la SCAO avec un ASO de type de Shack-Hartmann. Les développements de ROKET viseront ainsi la prise en charge de la pyramide, ainsi que le passage aux systèmes multi-analyseurs et multi-miroirs déformables.

Pour la prise en charge de la pyramide, l'implémentation devra être modifiée pour prendre en compte les spécificités de cet ASO, comme sa courbe de réponse non linéaire par exemple. Pour les systèmes d'OA plus complexes, comme la MCAO ou la MOAO, certaines erreurs devront être redéfinies. La partie analyse des erreurs liées à l'ASO sera dupliquée pour chaque ASO du système et ces termes seront à propager sur la commande avec le reconstructeur. Un cas particulier important est l'erreur d'anisoplanétisme qui devra être transformée en une erreur de tomographie : on pourra alors s'inspirer de travaux existants pour la déterminer numériquement (Gendron et al., 2014; Martin et al., 2017). Il faudra également prendre en compte les erreurs spécifiques dues à l'utilisation d'étoile laser, telle que l'effet de cône.

Enfin, toujours pour les systèmes d'OA de type tomographique, il faudra développer une estimation du budget d'erreur multi-directionnelle, permettant ainsi d'estimer la FEP du système dans le champ pour chaque cible d'intérêt.

GROOT

J'ai pu montrer dans la section dédiée que GROOT produisait des résultats prometteurs. Néanmoins, nous avons aussi pu constater que les modèles utilisés présentaient quelques limitations. La plus importante est celle que j'ai discuté en Section 6.2.2 concernant le modèle de l'erreur de repliement, plus spécifiquement la différence que l'on peut observer entre les spectres obtenus. Les investigations que j'ai pu mener pour tenter d'expliquer cette différence méritent d'être prolongées, afin d'améliorer le modèle, ou de trouver une possible erreur dans l'estimation produite par ROKET.

Une autre limitation provient de la symétrie centro-circulaire des fonctions de structure utilisées. Le développement de méthodes numériques permettant de calculer

ces fonctions de structures sur des motifs carrés ou hexagonaux pourrait permettre d'améliorer la précision des différents modèles, et donc la qualité de la FEP estimée grâce à GROOT.

L'implémentation sur GPU de GROOT permet un calcul très efficace des matrices de covariance des erreurs. Ainsi, une perspective intéressante de GROOT serait de l'utiliser "en ligne" à partir de la télémétrie de l'OA afin d'associer une matrice de covariance, et donc une FEP, à chaque image enregistrée pendant les observations.

Enfin, j'ai exposé une méthode pouvant permettre d'identifier certains paramètres nécessaires aux modèles à partir de la télémétrie produite par l'OA, afin notamment de les rendre utiles pour la reconstruction de FEP. Les résultats préliminaires montrent qu'il est possible d'estimer la vitesse et la direction du vent sur les couches turbulentes. Il serait intéressant de prolonger cette étude afin de déterminer s'il est possible d'identifier le C_n^2 par couche à partir de cette méthode, peut être en passant par une reconstruction des mesures en boucle ouverte. Certaines informations, comme l'altitude des couches et leur échelle externe, resteront cependant à déterminer. Si la méconnaissance de ces paramètres a un impact moindre pour les systèmes SCAO, ils seront au contraire essentiels dans le cadre des systèmes d'OA multi-analyseurs. L'altitude des couches pourra alors être estimées à partir des mesures provenant des différentes directions d'analyse. Restera à identifier le profil de l'échelle externe dont l'impact sur un télescope dont le diamètre est supérieur ou du même ordre de grandeur sera plus important sur les mesures que sur les systèmes actuels avec des télescopes de 4 ou 8 m.

Une idée à développer, que je n'ai malheureusement pas eu le temps de mettre en œuvre, serait de réussir à produire un modèle de la matrice de covariance des mesures avec leurs dérivées temporelles. Ce modèle pourrait être alimenté par l'identification précédente, et fournirait une matrice par couche turbulente. L'idée serait alors d'ajuster les paramètres des modèles afin de faire converger la différente entre la somme de ces matrices analytiques sur toutes les couches et la matrice calculée depuis la télémétrie (Vidal et al., 2010).

GuARDIAN

Un dernier point sur les développements logiciels que j'ai mené lors de cette thèse. ROKET, GROOT et GAMORA forment une pile logicielle qui a pour but d'être utilisée dans le développement des projets instrumentaux de l'ELT. Ainsi, ces 3 modules sont regroupés au sein du paquet GuARDIAN[4] afin d'être distribués (sous peu) et maintenus sur le long terme. Ce paquet forme alors un outil majeur pour l'analyse de performances des systèmes d'OA, que ce soit lors de la phase de design de l'instrument ou lors de son exploitation. Son architecture logicielle, proche de celle de COMPASS avec un cœur de calcul sur carte graphique et une interface utilisateur modulaire, permet une maintenance facile sur le long terme et définit un environnement simple pour le développement de nouvelles fonctionnalités.

4. *packaGe for Ao eRror breakDown estImation And exploitatioN* en anglais

Bibliographie

Amdahl, G. M. 1967, in Proceedings of the April 18-20, 1967, Spring Joint Computer Conference, AFIPS '67 (Spring) (New York, NY, USA : ACM), 483–485 (cité à la page 61)

Assémat, F., Wilson, R., & Gendron, E. 2006, Optics Express, 14, 988 (cité aux pages 83 et 84)

Babcock, H. W. 1953, PASP, 65, 229 (cité à la page 2)

Bardou, L., Gendron, E., Rousset, G., et al. 2016, in Proc. SPIE, Vol. 9909, Adaptive Optics Systems V, 990944 (cité à la page 45)

Basden, A., Bharmal, N., Jenkins, D., et al. 2018, SoftwareX, 7, 63 (cité à la page 59)

Basden, A. G., Myers, R. M., & Gendron, E. 2012, MNRAS, 419, 1628 (cité aux pages 24, 77, 80 et 92)

Beckers, J. M. 1989, in Proc. SPIE, Vol. 1114, Active telescope systems, ed. F. J. Roddier, 215–217 (cité à la page 48)

Beckers, J. M., Eisenhardt, P., Goad, L., & Roddier, F. 1985, in BAAS, Vol. 17, Bulletin of the American Astronomical Society, 571 (cité à la page 2)

Bernard, J., Gratadour, D., Lainé, M., Perret, D., & Sevin, A. 2017, in AO4ELT5 (cité aux pages 61 et 62)

Bertram, T., Absil, O., Bizenberger, P., et al. 2018, in Proc. SPIE, Vol. 10703, Adaptive Optics Systems VI, 10703 – 10703 – 11 (cité à la page 76)

Borgnino, J., Martin, F., & Ziad, A. 1992, Optics Communications, 91, 267 (cité à la page 11)

Boyer, C., Michau, V., & Rousset, G. 1990, in Proc. SPIE, Vol. 1271, Adaptive optics and optical structures, ed. R. K. Tyson & J. Schulte In den Baeumen, 63–81 (cité aux pages 29 et 30)

Brandl, B. R., Feldt, M., Glasse, A., et al. 2014, in Proc. SPIE, Vol. 9147, Ground-based and Airborne Instrumentation for Astronomy V, 914721 (cité à la page 52)

Carbillet, M., La Camera, A., Folcher, J.-P., Perruchon-Monge, U., & Sy, A. 2016, in Proc. SPIE, Vol. 9909, Adaptive Optics Systems V, 99097J (cité à la page 59)

Clénet, Y., Buey, T., Rousset, G., et al. 2016, in Proc. SPIE, Vol. 9909, Adaptive Optics Systems V, 99090A (cité aux pages 50 et 76)

Conan, J.-M. 1994, PhD thesis, thèse de doctorat dirigée par Léna, Pierre Terre, océan, espace Paris 11 1994 (cité à la page 37)

Conan, J.-M., Mugnier, L. M., Fusco, T., Michau, V., & Rousset, G. 1998, Appl. Opt., 37, 4614 (cité à la page 116)

Conan, R. & Correia, C. 2014, in Proc. SPIE, Vol. 9148, Adaptive Optics Systems IV, 91486C (cité à la page 59)

Correia, C., Véran, J.-P., Ellerbroek, B., Gilles, L., & Wang, L. 2011, in Second International Conference on Adaptive Optics for Extremely Large Telescopes, 70 (cité à la page 117)

Cortés, A., Neichel, B., Guesalaga, A., et al. 2012, MNRAS, 427, 2089 (cité à la page 165)

Davies, R., Schubert, J., Hartl, M., et al. 2016, in Proc. SPIE, Vol. 9908, Ground-based and Airborne Instrumentation for Astronomy VI, 99081Z (cité à la page 50)

Demerle, M., Madec, P. Y., & Rousset, G. 1994, in NATO Advanced Science Institutes (ASI) Series C, Vol. 423, NATO Advanced Science Institutes (ASI) Series C, ed. D. M. Alloin & J. M. Mariotti, 73 (cité à la page 32)

Deo, V., Vidal, F., Gendron, E., et al. 2017, in 5th AO4ELT conference-Adaptive Optics for Extremely Large Telescopes (cité aux pages 77 et 110)

Diolaiti, E., Ciliegi, P., Abicca, R., et al. 2016, in Proc. SPIE, Vol. 9909, Adaptive Optics Systems V, 99092D (cité à la page 50)

Doucet, N., Gratadour, D., Ltaief, H., et al. 2017, in AO4ELT5 (cité à la page 62)

Doucet, N., Kriemann, R., Gendron, E., et al. 2018, in Adaptive Optics Systems VI, Proc. SPIE (cité à la page 175)

Du, P., Weber, R., Luszczek, P., et al. 2012, Parallel Comput., 38, 391 (cité à la page 65)

Ellerbroek, B. L. 1994, Journal of the Optical Society of America A, 11, 783 (cité aux pages 46, 47 et 48)

Ellerbroek, B. L. 2005, Journal of the Optical Society of America A, 22, 310 (cité à la page 58)

Exposito, J., Gratadour, D., Clénet, Y., Rousset, G., & Mugnier, L. 2012, in Proc. SPIE, Vol. 8447, Adaptive Optics Systems III, 84475X (cité à la page 116)

Exposito, J., Gratadour, D., Rousset, G., et al. 2014, in Proc. SPIE, Vol. 9148, Adaptive Optics Systems IV, 91484P (cité aux pages 2 et 117)

Fedrigo, E., Bourtembourg, R., Donaldson, R., et al. 2010, in Proc. SPIE, Vol. 7736, Adaptive Optics Systems II, 77362I (cité à la page 61)

Foy, R. 2000, in NATO Advanced Science Institutes (ASI) Series C, Vol. 551, NATO Advanced Science Institutes (ASI) Series C, ed. N. Ageorges & C. Dainty, 107 (cité à la page 44)

Foy, R. & Labeyrie, A. 1985, A&A, 152, L29 (cité à la page 42)

Fried, D. L. 1966, Journal of the Optical Society of America (1917-1983), 56, 1372 (cité à la page 12)

Fried, D. L. 1982, Journal of the Optical Society of America (1917-1983), 72, 52 (cité à la page 39)

Fried, D. L. 1990, Journal of the Optical Society of America A, 7, 1224 (cité à la page 13)

Fried, D. L. & Belsher, J. F. 1994, Journal of the Optical Society of America A, 11, 277 (cité à la page 44)

Fried, D. L. & Clark, T. 2008, Journal of the Optical Society of America A, 25, 463 (cité aux pages 83 et 84)

Fusco, T., Nicolle, M., Rousset, G., et al. 2004, in Proc. SPIE, Vol. 5490, Advancements in Adaptive Optics, ed. D. Bonaccini Calia, B. L. Ellerbroek, & R. Ragazzoni, 1155–1166 (cité à la page 92)

Fusco, T., Véran, J.-P., Conan, J.-M., & Mugnier, L. M. 1999, A&AS, 134, 193 (cité à la page 116)

Gaffard, J. P. 1988, in European Southern Observatory Conference and Workshop Proceedings, Vol. 30, European Southern Observatory Conference and Workshop Proceedings, ed. M.-H. Ulrich, 729 (cité à la page 98)

Gaffard, J. P. & Boyer, C. 1987, Appl. Opt., 26, 3772 (cité à la page 134)

Gardner, C. S., Welsh, B. M., & Thompson, L. A. 1989, in Proc. SPIE, Vol. 1114, Active telescope systems, ed. F. J. Roddier, 191–202 (cité à la page 42)

Gendron, E. 1995, Theses, Université Denis Diderot (Paris 7) (cité aux pages 35, 94 et 135)

Gendron, É., Charara, A., Abdelfattah, A., et al. 2014, in Proc. SPIE, Vol. 9148, Adaptive Optics Systems IV, 91486L (cité aux pages 47, 58, 95, 117, 148 et 175)

Gendron, E., Clénet, Y., Fusco, T., & Rousset, G. 2006, A&A, 457, 359 (cité aux pages 135 et 136)

Gendron, E. & Léna, P. 1994, A&A, 291, 337 (cité aux pages 93 et 149)

Gendron, E. & Léna, P. 1996, Ap&SS, 239, 221 (cité aux pages 13 et 165)

Gendron, E., Morris, T., Basden, A., et al. 2016, in Proc. SPIE, Vol. 9909, Adaptive Optics Systems V, 99090C (cité à la page 169)

Gendron, E., Vidal, F., Brangier, M., et al. 2011, A&A, 529, L2 (cité aux pages 20, 49 et 169)

Gilles, L. & Ellerbroek, B. 2006, Appl. Opt., 45, 6568 (cité à la page 45)

Gonsalves, R. A. 1982, Optical Engineering, 21, 829 (cité à la page 42)

Goodman, J. W. 1995, Introduction to Fourier optics (cité à la page 8)

Gratadour, D., Dipper, N., Biasi, R., et al. 2016, in Proc. SPIE, Vol. 9909, Adaptive Optics Systems V, 99094I (cité aux pages 61 et 62)

Gratadour, D., Gendron, E., & Rousset, G. 2010, Journal of the Optical Society of America A, 27, A171 (cité aux pages 45 et 92)

Gratadour, D., Puech, M., Vérinaud, C., et al. 2014, in Proc. SPIE, Vol. 9148, Adaptive Optics Systems IV, 91486O (cité à la page 76)

Gratadour, D., Sevin, A., Brulé, J., Gendron, É., & Rousset, G. 2012, in Proc. SPIE, Vol. 8447, Adaptive Optics Systems III, 84475R (cité à la page 82)

Greenwood, D. P. 1977, Journal of the Optical Society of America (1917-1983), 67, 390 (cité aux pages 13 et 38)

Hammer, F., Sayède, F., Gendron, E., et al. 2002, in Scientific Drivers for ESO Future VLT/VLTI Instrumentation, ed. J. Bergeron & G. Monnet, 139 (cité à la page 49)

Harder, S. & Chelli, A. 2000, A&AS, 142, 119 (cité à la page 117)

Hardy, J. W., Lefebvre, J. E., & Koliopoulos, C. L. 1977, Journal of the Optical Society of America (1917-1983), 67, 360 (cité à la page 2)

Hudgin, R. 1977, Journal of the Optical Society of America (1917-1983), 67, 393 (cité à la page 38)

Jolissaint, L. 2010, Journal of the European Optical Society, 5 (cité à la page 58)

Jolissaint, L., Véran, J.-P., & Conan, R. 2006, Journal of the Optical Society of America A, 23, 382 (cité aux pages 3 et 148)

Jolissaint, L., Veran, J.-P., & Marino, J. 2004, in Proc. SPIE, Vol. 5490, Advancements in Adaptive Optics, ed. D. Bonaccini Calia, B. L. Ellerbroek, & R. Ragazzoni, 151–163 (cité à la page 117)

Kasper, M., Fedrigo, E., Looze, D. P., et al. 2004, Journal of the Optical Society of America A, 21, 1004 (cité à la page 30)

Kellerer, A., Vidal, F., Gendron, E., et al. 2012, in Proc. SPIE, Vol. 8447, Adaptive Optics Systems III, 844765 (cité à la page 30)

Kolb, J., Madec, P.-Y., Arsenault, R., et al. 2017, in AO4ELT5 (cité à la page 42)

Kolmogorov, A. 1941a, Akademiia Nauk SSSR Doklady, 30, 301 (cité à la page 11)

Kolmogorov, A. N. 1941b, Akademiia Nauk SSSR Doklady, 32, 16 (cité à la page 11)

Kulcsár, C., Raynaud, H.-F., Petit, C., Conan, J.-M., & Viaris de Lesegno, P. 2006, Optics Express, 14, 7464 (cité à la page 34)

Le Roux, B., Conan, J.-M., Kulcsár, C., et al. 2004, 21, 1261 (cité à la page 34)

Li, S. & Zhang, S. 2014, Research in Astronomy and Astrophysics, 14, 1504 (cité à la page 98)

M. Obukhov, A. 1968, 13, 14 (cité à la page 11)

Madec, P.-Y. 1999, in Adaptive Optics in Astronomy, ed. F. Roddier (cité aux pages 32 et 35)

Madec, P. Y. 2012, in Adaptive Optics Systems III, Vol. 8447, 844705 (cité à la page 22)

Marchetti, E., Brast, R., Delabre, B., et al. 2008, in Proc. SPIE, Vol. 7015, Adaptive Optics Systems, 70150F (cité à la page 48)

Martin, O. 2014, Theses, Université Paris-Diderot - Paris VII (cité à la page 146)

Martin, O., Gendron, É., Rousset, G., & Vidal, F. 2012, in Proc. SPIE, Vol. 8447, Adaptive Optics Systems III, 84472A (cité à la page 166)

Martin, O. A., Correia, C. M., Gendron, E., et al. 2016, Journal of Astronomical Telescopes, Instruments, and Systems, 2, 048001 (cité à la page 117)

Martin, O. A., Gendron, É., Rousset, G., et al. 2017, A&A, 598, A37 (cité à la page 175)

McCarthy, P. J., Fanson, J., Bernstein, R., et al. 2016, in Proc. SPIE, Vol. 9906, Ground-based and Airborne Telescopes VI, 990612 (cité à la page 49)

Meimon, S., Petit, C., & Fusco, T. 2015, Optics Express, 23, 27134 (cité à la page 30)

Metropolis, N. & Ulam, S. M. 1949, Journal of the American Statistical Association, 44, 335 (cité à la page 59)

Michau, V., Rousset, G., & Fontanella, J. 1993, in Real Time and Post Facto Solar Image Correction, ed. R. R. Radick, 124 (cité à la page 92)

Milton, N. M., Lloyd-Hart, M., Baranec, C., et al. 2008, in Proc. SPIE, Vol. 7015, Adaptive Optics Systems, 701522 (cité à la page 48)

Molodij, G. & Rousset, G. 1997, Journal of the Optical Society of America A, 14, 1949 (cité à la page 44)

Mugnier, L. M., Fusco, T., & Conan, J.-M. 2004, Journal of the Optical Society of America A, 21, 1841 (cité aux pages 2 et 116)

Muller, N., Michau, V., Fusco, T., et al. 2010, in Proc. SPIE, Vol. 7736, Adaptive Optics Systems II, 773628 (cité à la page 45)

Neichel, B., Fusco, T., & Conan, J.-M. 2008, Journal of the Optical Society of America A, 26, 219 (cité à la page 58)

Neichel, B., Fusco, T., Sauvage, J.-F., et al. 2016, in Proc. SPIE, Vol. 9909, Adaptive Optics Systems V, 990909 (cité à la page 52)

Nicolle, M., Fusco, T., Rousset, G., & Michau, V. 2004, Optics Letters, 29, 2743 (cité à la page 24)

Noll, R. J. 1976, Journal of the Optical Society of America (1917-1983), 66, 207 (cité aux pages 15 et 16)

Osborn, J., Butterley, T., Föhring, D., & Wilson, R. 2015, in Journal of Physics Conference Series, Vol. 595, Journal of Physics Conference Series, 012022 (cité à la page 170)

Perret, D., Lainé, M., Bernard, J., Gratadour, D., & Sevin, A. 2016, in Proc. SPIE, Vol. 9909, Adaptive Optics Systems V, 99094M (cité à la page 62)

Petit, C., Conan, J.-M., Kulcsár, C., & Raynaud, H.-F. 2009, Journal of the Optical Society of America A, 26, 1307 (cité à la page 34)

Pfrommer, T. & Hickson, P. 2010, Journal of the Optical Society of America A, 27, A97 (cité à la page 45)

Poyneer, L., van Dam, M., & Véran, J.-P. 2009, Journal of the Optical Society of America A, 26, 833 (cité à la page 165)

Rabien, S., Ageorges, N., Barl, L., et al. 2010, in Proc. SPIE, Vol. 7736, Adaptive Optics Systems II, 77360E–77360E–12 (cité à la page 44)

Ragazzoni, R. 1996, Journal of Modern Optics, 43, 289 (cité aux pages 24, 27 et 92)

Ragazzoni, R. & Farinato, J. 1999, A&A, 350, L23 (cité à la page 27)

Riccardi, A., Bindi, N., Ragazzoni, R., Esposito, S., & Stefanini, P. 1998, in Proc. SPIE, Vol. 3353, Adaptive Optical System Technologies, ed. D. Bonaccini & R. K. Tyson, 941–951 (cité à la page 27)

Rigaut, F. 2002, in European Southern Observatory Conference and Workshop Proceedings, Vol. 58, European Southern Observatory Conference and Workshop Proceedings, ed. E. Vernet, R. Ragazzoni, S. Esposito, & N. Hubin, 11 (cité à la page 48)

Rigaut, F. & Gendron, E. 1992, A&A, 261, 677 (cité à la page 45)

Rigaut, F., Neichel, B., Boccas, M., et al. 2014, MNRAS, 437, 2361 (cité à la page 42)

Rigaut, F. & Van Dam, M. 2013, in Proceedings of the Third AO4ELT Conference, ed. S. Esposito & L. Fini, 18 (cité aux pages 3, 59 et 98)

Rigaut, F. J., Ellerbroek, B. L., & Flicker, R. 2000, in Proc. SPIE, Vol. 4007, Adaptive Optical Systems Technology, ed. P. L. Wizinowich, 1022–1031 (cité aux pages 47 et 48)

Rigaut, F. J., Veran, J.-P., & Lai, O. 1998, in Proc. SPIE, Vol. 3353, Adaptive Optical System Technologies, ed. D. Bonaccini & R. K. Tyson, 1038–1048 (cité aux pages 34, 38 et 146)

Rocca, A., Roddier, F., & Vernin, J. 1974, Journal of the Optical Society of America (1917-1983), 64, 1000 (cité à la page 165)

Roddier, F. 1981, Progress in optics. Volume 19. Amsterdam, North-Holland Publishing Co., 1981, p. 281-376., 19, 281 (cité aux pages 11, 12 et 41)

Roddier, F. 1999, Adaptive optics in astronomy (cité aux pages 2 et 20)

Roddier, N. 1990, Optical Engineering, 29, 1174 (cité à la page 16)

Rosensteiner, M. 2012, J. Opt. Soc. Am. A, 29, 2328 (cité aux pages 34 et 80)

Rousset, G. 1999, in Adaptive Optics in Astronomy, ed. F. Roddier (Camb. Univ. Press) (cité aux pages 24 et 149)

Rousset, G., Fontanella, J. C., Kern, P., Gigan, P., & Rigaut, F. 1990, A&A, 230, L29 (cité aux pages 2 et 34)

Rousset, G., Primot, J., & Fontanella, J. C. 1987, LEST Foundation, Technical Report, 28, 17 (cité à la page 39)

Sanders, J. & Kandrot, E. 2010, CUDA by Example : An Introduction to General-Purpose GPU Programming, 1st edn. (Addison-Wesley Professional) (cité à la page 57)

Schoeck, M. & Spillar, E. J. 1998, in Proc. SPIE, Vol. 3353, Adaptive Optical System Technologies, ed. D. Bonaccini & R. K. Tyson, 1092–1099 (cité à la page 165)

Schreiber, L., Diolaiti, E., Arcidiacono, C., et al. 2016, in Proc. SPIE, Vol. 9909, Adaptive Optics Systems V, 99094L (cité à la page 52)

Schwartz, L. 1966, Théorie des distributions, ed. Hermann (cité à la page 98)

Séchaud, M. 1999, in Adaptive optics in astronomy / edited by Francois Roddier. Cambridge ; New, 57 (cité à la page 22)

Shack, R. V. & Platt, B. C. 1971, Journal of the Optical Society of America (1917-1983) (cité à la page 24)

Simard, L., Ellerbroek, B., Bhatia, R., Radovan, M., & Chisholm, E. 2016, in Proc. SPIE, Vol. 9908, Ground-based and Airborne Instrumentation for Astronomy VI, 99081V (cité à la page 49)

Sivo, G., Turchi, A., Masciadri, E., Guesalaga, A., & Neichel, B. 2018, MNRAS, 476, 999 (cité à la page 165)

Summers, D., H. Bouchez, A., Chin, J., et al. 2004, 5490 (cité à la page 45)

Tallon, M. & Foy, R. 1990, A&A, 235, 549 (cité à la page 46)

Tamai, R., Cirasuolo, M., González, J. C., Koehler, B., & Tuti, M. 2016, in Proc. SPIE, Vol. 9906, Ground-based and Airborne Telescopes VI, 99060W (cité à la page 49)

Tatarskii, V. I. 1971, The effects of the turbulent atmosphere on wave propagation (cité à la page 11)

Taylor, G. I. 1938, Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences, 164, 476 (cité à la page 13)

Thiébaut, E. & Tallon, M. 2010, Journal of the Optical Society of America A, 27, 1046 (cité à la page 94)

Thomas, S., Fusco, T., Tokovinin, A., et al. 2006, MNRAS, 371, 323 (cité à la page 92)

Thompson, L. A. & Gardner, C. S. 1989, in Proc. SPIE, Vol. 1114, Active telescope systems, ed. F. J. Roddier, 184–190 (cité à la page 42)

Tyler, G. A. 1994, Journal of the Optical Society of America A, 11, 325 (cité à la page 88)

Van Trees, H. 1968, Detection, Estimation, and Modulation Theory Part I (cité à la page 94)

Véran, J.-P. & Herriot, G. 2000, Journal of the Optical Society of America A, 17, 1430 (cité à la page 41)

Véran, J.-P., Rigaut, F., Maitre, H., & Rouan, D. 1997, J. Opt. Soc. Am. A, Vol. 14, No. 11, p. 3057 - 3069, 14, 3057 (cité aux pages 2, 116, 117, 135, 146 et 165)

Vérinaud, C. 2004, Optics Communications, 233, 27 (cité à la page 27)

Vernet, E., Cayrel, M., Hubin, N., et al. 2014, in Proc. SPIE, Vol. 9148, Adaptive Optics Systems IV, 914824 (cité à la page 50)

Vernin, J. & Roddier, F. 1973, Journal of the Optical Society of America (1917-1983), 63, 270 (cité à la page 165)

Vidal, F., Ferreira, F., Deo, V., et al. 2017, in Proc. AO4ELT5 (cité aux pages 76 et 77)

Vidal, F., Gendron, E., Clénet, Y., et al. 2014a, in Proc. SPIE, Vol. 9148, Adaptive Optics Systems IV, 91486I (cité à la page 76)

Vidal, F., Gendron, E., & Rousset, G. 2010, Journal of the Optical Society of America A, 27, A253 (cité aux pages 20, 165 et 176)

Vidal, F., Gendron, É., Rousset, G., et al. 2014b, A&A, 569, A16 (cité à la page 165)

Wallner, E. P. 1983, Journal of the Optical Society of America (1917-1983), 73, 1771 (cité aux pages 34 et 94)

Wang, L. & Ellerbroek, B. 2011, in Proceedings of the Second AO4ELT Conference (cité aux pages 3 et 59)

Wang, L., Schöck, M., & Chanan, G. 2008, Appl. Opt., 47, 1880 (cité à la page 165)

Welsh, B. M. & Gardner, C. S. 1991, Journal of the Optical Society of America A, 8, 69 (cité à la page 46)

Wilson, R. W. 2002, MNRAS, 337, 103 (cité à la page 165)

Wizinowich, P. L., Le Mignant, D., Bouchez, A., et al. 2004, in Proc. SPIE, Vol. 5490, Advancements in Adaptive Optics, ed. D. Bonaccini Calia, B. L. Ellerbroek, & R. Ragazzoni, 1–11 (cité à la page 42)

Liste de publications

Publications en tant que premier auteur

- **F. Ferreira**, E. Gendron, G. Rousset et D. Gratadour, *"Numerical estimation of wavefront error breakdown in adaptive optics"*, Astronomy & Astrophysics, accepté en avril 2018
- **F. Ferreira**, D. Gratadour, A. Sevin et N. Doucet, *"COMPASS : an efficient GPU-based simulation software for adaptive optics systems"*, Proc. of the International Conference on High Performance Computing & Simulation, 2018
- **F. Ferreira**, D. Gratadour, A. Sevin, N. Doucet, F. Vidal, V. Deo et E. Gendron, *"Real-time end-to-end AO simulations at ELT scale on multiple GPUs with the COMPASS platform"*, Proc. of SPIE conference, 2018
- **F. Ferreira**, E. Gendron, G. Rousset et D. Gratadour, *"Modeling of PSF corrected by adaptive optics systems"*,Proc. of SPIE conference, 2018
- **F. Ferreira**, E. Gendron, G. Rousset et D. Gratadour,*"AO error breakdown : anisoplanatism and bandwidth error correlation with ROKET"*, Proc. of AO4ELT 5, 2017
- **F. Ferreira**, E. Gendron, G. Rousset et D. Gratadour, *"Deriving comprehensive error breakdown for wide field adaptive optics systems using end-to-end simulations"*,Proc. of SPIE conference, 2016

Publications en co-auteur

- Y. Clénet, J.-T. M. Buey, E. Gendron, Z. Hubert, F. Vidal, M. Cohen, F. Chapron, A. Sevin, P. Fedou, G. Barbary, P. Baudoz, V. Deo, O. Dupuis, S. Durand, **F. Ferreira**, J. Gaudemard, D. Gratadour, E. Huby, J.-M. Huet, B. Le Ruyet, N. Nguyen-Tuong, C. Perrot, S. Thijs, Y. Younès, G. Rousset, P. Feautrier, G. Zins, E. Diolaiti, P. Ciliegi, S. Esposito, L. Busoni, J. Schubert, M. Hartl, V. Hörmann, R. Davies, *"The MICADO first-light imager for the ELT : towards the preliminary design review of the MICADO-MAORY SCAO"*, Proc. of SPIE conference, 2018
- D. Gratadour, T. J. Morris, R. Biasi, H. Deneux, J. Bernard, J.-T. M. Buey, N. Doucet, **F. Ferreira**, M. Lainé, D. Perret, A. Sevin, A. G. Basden, D. Geng, J. Osborn, L. Staykov, M. J. Townson, E. J. Younger, M. Andrighettoni, C. Patauner, D. Pescoller, J. Lemaitre, P. Palazzari, D. Pretet, C. Rouaud, *"Prototyping AO RTC using emerging high performance computing technologies with the Green Flash project"*, Proc. of SPIE conference, 2018
- N. Doucet, D. Gratadour, H. Ltaeif, E. Gendron, A. Sevin, **F. Ferreira**, F. Vidal, R. Kriemann, D. Keyes, *"Efficient Supervision Strategy for Tomographic AO Systems on E-ELT"*, Proc. of AO4ELT 5, 2017

- F. Vidal, **F. Ferreira**, V. Deo, A. Sevin, E. Gendron, Y. Clénet, S. Durand, D. Gratadour, N. Doucet, G. Rousset, R. Davies, *"End-to-End simulations for the MICADO-MAORY SCAO mode"*, Proc. of AO4ELT 5, 2017
- D. Gratadour, **F. Ferreira**, A. Sevin, N. Doucet, Y. Clénet, E. Gendron, M. Lainé, F. Vidal, J. Brulé, M. Puech, C. Vérinaud, A. Carlotti, *"COMPASS : status update and long term development plan"*, Proc. of SPIE conference, 2016
- D. Gratadour, N. Dipper, R. Biasi, H. Deneux, J. Bernard, J. Brule, R. Dembet, N. Doucet, **F. Ferreira**, E. Gendron, M. Laine, D. Perret, G. Rousset, A. Sevin, U. Bitenc, D. Geng, E. Younger, M. Andrighettoni, G. Angerer, C. Patauner, D. Pescoller, F. Porta, G. Dufourcq, A. Flaischer, J.-B. Leclere, A. Nai, P. Palazzari, D. Pretet, C. Rouaud, *"Green FLASH : energy efficient real-time control for AO"*, Proc. of SPIE conference, 2016
- Y. Clénet, T. Buey, G. Rousset, E. Gendron, S. Esposito, Z. Hubert, L. Busoni, M. Cohen, A. Riccardi, F. Chapron, M. Bonaglia, A. Sevin, P. Baudoz, P. Feautrier, G. Zins, D. Gratadour, F. Vidal, F. Chemla, **F. Ferreira**, N. Doucet, S. Durand, A. Carlotti, C. Perrot, L. Schreiber, M. Lombini, P. Ciliegi, E. Diolaiti, J. Schubert, R. Davies, *"Joint MICADO-MAORY SCAO mode : specifications, prototyping, simulations and preliminary design"*, Proc. of SPIE conference, 2016
- D. Gratadour, M. Puech, C. Vérinaud, P. Kestener, M. Gray, C. Petit, J. Brulé, Y. Clénet, **F. Ferreira**, E. Gendron, M. Lainé, A. Sevin, G. Rousset, F. Hammer, I. Jégouzo, M. Paillous, S. Taburet, Y. Yang, J.-L. Beuzit, A. Carlotti, M. Westphal, B. Epinat, M. Ferrari, T. Gautrais, J. C. Lambert, B. Neichel, S. Rodionov, *"COMPASS : an efficient, scalable and versatile numerical platform for the development of ELT AO systems"*, Proc. of SPIE conference, 2014