Contributions to Symbolic Effective Qualitative Analysis of Dynamical Systems; Application to Biochemical Reaction Networks

Aslı Grimaud

Contributions à l'analyse qualitative symbolique effective des systèmes dynamiques; application aux réseaux de réactions biochimiques

présenté par

B. Aslı ÜRGÜPLÜ BLUMENFELD

asli.urguplu@lifl.fr
www.lifl.fr/~urguplu

pour obtenir le titre de

DOCTEUR en SCIENCES
(spécialité informatique)

à l'Université Lille 1
Laboratoire d'Informatique Fondamentale de Lille, UMR CNRS 8022
Bâtiment M3, 59655 Villeneuve d'Ascq, France
École Doctorale Sciences Pour l'Ingénieur, Université Lille Nord de France

Membres de la commission d'examen

Président :	Hélène TOUZET	Directeur de Recherche CNRS, Université Lille 1
Rapporteurs :	Hidde DE JONG	Directeur de Recherche INRIA, INRIA Grenoble - Rhône-Alpes
	Marc GIUSTI	Directeur de Recherche CNRS, Laboratoire d'Informatique de l'Ecole Polytechnique (LIX)
Examinateur :	Mohab SAFEY EL DIN	Maître de Conférence, Université Pierre et Marie Curie
Directeur :	François BOULIER	Maître de Conférence, HDR, Université Lille 1
Co-directeur :	Alexandre SEDOGLAVIC	Maître de Conférence, Université Lille 1

Lille, France - 13 Janvier 2010
Numéro d'ordre : 40187

Contributions to Symbolic Effective Qualitative Analysis of Dynamical Systems; Application to Biochemical Reaction Networks

by

B. Aslı ÜRGÜPLÜ BLUMENFELD

asli.urguplu@lifl.fr
www.lifl.fr/~urguplu

For the award of the degree of

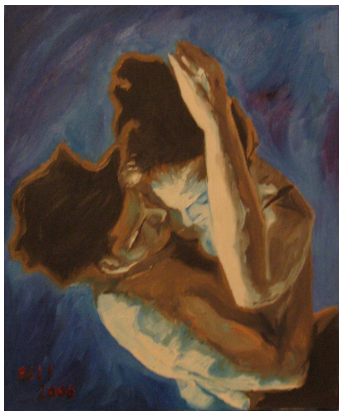
Doctor of Philosophy
(Computer Science)

at the
University of Lille 1
Laboratoire d'Informatique Fondamentale de Lille, UMR CNRS 8022
Building M3, 59655 Villeneuve d'Ascq, France
École Doctorale Sciences Pour l'Ingénieur, Université Lille Nord de France

Membership of Examining Committee

President:	Hélène TOUZET	CNRS Senior Research Scientist, University of Lille 1
External Examiners:		
(Rapporteur)	Hidde DE JONG	INRIA Senior Research Scientist, INRIA Grenoble - Rhône-Alpes
(Rapporteur)	Marc GIUSTI	CNRS Senior Research Scientist, Laboratory for Informatics at Polytechnique School (LIX)
(Examinateur)	Mohab SAFEY EL DIN	Assistant Professor, University of Pierre and Marie Curie
Advisor:	François BOULIER	Assistant Professor, HDR, University of Lille 1
Co-advisor:	Alexandre SEDOGLAVIC	Assistant Professor, University of Lille 1

Lille, France - January 13^{th}, 2010
Number: 40187



"One must work and dare if one really wants to live."

Letter to Theo van Gogh
Nuenen, April 11 th, 1885
Vincent van Gogh

Title

Contributions to Symbolic Effective Qualitative Analysis of Dynamical Systems; Application to Biochemical Reaction Networks

Abstract

The goal of my research is to make algorithmic, as much as possible, the study of models composed by parametric differential equations. I focus on the algorithms based on expanded Lie point symmetries for medium size (about twenty variables) models. I present two exact simplification methods: the reduction of the number of variables of a model and its reparametrization in order to distinguish the roles of its parameters. Simplified systems are equivalent to the original ones by implicit or explicit relationships (according to the chosen method). These algorithms, thanks to some computational strategies and restriction of studied objects, are of polynomial time complexity in the input size. They are implemented in the `MABSys` and the `ExpandedLiePointSymmetry` packages. Simplified models resulting from these methods allow to perform more easily various studies such as symbolic or numerical qualitative analysis. I illustrate my work on a family of genetic networks with a single self-regulated gene by a complete symbolic qualitative analysis. Even if my principal application example belongs to genetic regulatory networks field, the methods presented in my work are not limited to intracellular biology.

Titre

Contributions à l'analyse qualitative symbolique effective des systèmes dynamiques; l'application aux réseaux de réactions biochimiques

Résumé

Le but de mes travaux de recherche est de rendre, autant que possible, algorithmique l'étude des modèles composés par des équations différentielles paramétriques. Je me concentre aux algorithmes basés sur les symétries de Lie étendues pour les modèles de taille moyenne (environ vingt variables). Je présente deux méthodes de simplification exacte : la réduction du nombre des variables d'un modèle et sa reparamétrisation pour distinguer le rôle de ses paramètres. Les systèmes simplifiés sont équivalents aux systèmes originaux par des relations implicites ou explicites (suivant la méthode choisie). Ces algorithmes, grâce aux stratégies de calcul utilisées et aux restrictions sur les objets étudiés, ont une complexité temporelle polynomiale en la taille de l'entrée. Ils sont implémentés dans les paquetages `MABSys` et `ExpandedLiePointSymmetry`. Les modèles simplifiés issus de ces algorithmes facilitent diverses études comme l'analyse qualitative symbolique ou numérique. J'illustre mes travaux sur une famille de réseaux génétiques avec un seul gène autorégulé en faisant une analyse symbolique complète. Mon exemple principal appartient au domaine des réseaux de régulation génétique mais l'application des méthodes que je présente n'est pas limitée à la biologie intracellulaire.

Acknowledgements

I feel incredibly lucky to have received such an education all my life. The very document you are reading is a consequence of all these years. I owe, much more than I can possibly express, to the *Galatasaray* community in which I grew up. *İçinde büyüdüğüm Galatasaray camiasına ifade edebileceğimden daha fazlasını borçlu olduğumun farkındayım ve benim ben olmama katkıda bulunan herkese içtenlikle teşekkür ederim.* I am very happy where I am in my professional life and I hope to continue this job for many years. This dissertation would not have been possible without the help and the support of many people to whom I am greatly indebted.

I would like to thank my advisor, François Boulier, for having given me this opportunity to pursue and realize my scientific ambitions and for his encouragement. I thank also my co-advisor Alexandre Sedoglavic for his continuous availability for hours of discussions and for his patience to answer all of my questions.

I would like to express a deep appreciation to the members of Computer Algebra team. Many discussions that we had with François Lemaire are crucial in my work. I enjoyed my time spent working with him. I can't help but remember all the time I spent with Michel Petitot on many interesting scientific subjects. He was always available for a quick question or a lengthy discussion on seemingly inexhaustible open problems. Further thanks go to Léopold Weinberg, Éric Wegrzynowski and Nour-Eddine Oussous to be such pleasant to work with, always in a good mood, always with a smile.

I would like to express my deepest and warmest gratitude to my thesis committee. I thank Hidde de Jong and Marc Giusti to report my manuscript with their precious remarks. I also thank Hélène Touzet and Mohab Safey El Din to participate to my examining committee and for their attention to my work.

I cannot not to mention Sultan Turhan, Muhammed Uludağ, Moncef Meddeb and Guillaume Frazier who believed in me and supported me in my studies.

A special thanks goes to my friend Fadila Khadar for listening to my complaints and frustrations, for interesting discussions, for having fun together, for helping me with whatever needed help. She has always been the source of precious advice. And we *know* now that we enjoy the *champagne* :).

I also thank people I met during all these years of thesis and truly appreciate. I have the pleasure to cite alphabetically Benjamin Barbry, Gilles Grimaud, Frédéric Maquet, Anthony Martinet, Jérémy Ringard, Loïc Schmidt and also people with whom I acted in the play "Punaise de thèse", Tram Nguyen (best assistant that I ever met) and many others who offered their time. They have filled everyday of my thesis work with care, joy and excitement. I enjoyed every occasion where we spent time together. I would not have been able to carry out this research without all of you.

I would like to thank my family, especially my mother Gülgûn Ürgüplü for being there for me, for making life easier and for encouraging the scientist in me ever since I was young girl with interests in math.

Finally, I am everlastingly grateful to my husband Thomas Blumenfeld who helped me make it through these years. He gave me confidence when mine was lacking. With the comfort he and my little Küçük offered at home I could maintain hope and enjoy my daily life.

Contents

Résumé

Le but de mes travaux de recherche est de rendre, autant que possible, algorithmique l'étude des modèles composés par des équations différentielles paramétriques. Mon exemple principal appartient au domaine des réseaux de régulation génétique mais l'application des méthodes présentées dans ce document n'est pas limitée à la biologie intracellulaire. Les modèles mathématiques considérés sont des systèmes d'équations différentielles paramétriques et non linéaires (polynomiales). Ces systèmes peuvent être obtenus, par exemple, à partir de réseaux de réactions biochimiques généralisées en utilisant la loi d'action-masse (voir [55]). Je me concentre aux algorithmes de calcul formel développés pour les modèles de taille moyenne (environ vingt variables). Je considère principalement deux types d'algorithmes consacrés aux traitements de modèles : l'approximation quasi-stationnaire (basée sur la théorie de l'élimination différentielle) et les simplifications exactes (basées sur la théorie des symétries de Lie). Ces dernières correspondent à la réduction du nombre de variables d'un modèle et à sa reparamétrisation pour distinguer le rôle de ses paramètres. Les modèles simplifiés issus de ces algorithmes facilitent diverses études comme l'analyse qualitative symbolique ou numérique, les simulations numériques, etc. Par exemple, l'analyse qualitative symbolique est, dans le pire des cas, au moins de complexité exponentielle en le nombre de variables. La réduction du nombre des variables d'un modèle ou de ses expressions dérivées est utile pour faciliter son analyse.

Idées principales

La méthode de reparamétrisation que je propose est une méthode de simplification exacte basée sur un nouvel algorithme qui utilise la théorie classique des symétries de Lie. Elle améliore l'analyse préliminaire des modèles et facilite leur analyse qualitative. Pour établir la liaison entre le système original et le système simplifié, toute l'information relative à la simplification est aussi fournie. De plus, des stratégies de calcul comme la restriction de l'ensemble des symétries de Lie auquel on s'intéresse permettent à ce nouvel algorithme d'avoir une complexité polynomiale en la taille de l'entrée.

La méthode de réduction présentée dans ce document est une application classique de la théorie des symétries de Lie. La méthode de réduction classique (voir [24, 86]) et la méthode de réduction basée sur le repère mobile (voir [37, 38]) sont adaptées pour que la complexité temporelle soit polynomiale en la taille de l'entrée. Même dans ces cas, ils généralisent, entre autres, l'analyse dimensionnelle (voir ch. 1 de [6], [17]) pour laquelle il n'y a pas de résultat de complexité (voir [64]). De plus, nos algorithmes ne nécessitent pas de spécifier les dimensions des variables mais se basent sur des hypothèses de positivité. Ce processus est aussi appliqué partiellement aux systèmes dynamiques discrets. A notre connaissance, même si ces systèmes sont déjà traités à l'aide des

symétries de Lie (voir [70, 117]), aucune méthode algorithmique avec une complexité polynomiale en la taille de l'entrée n'est disponible.

La limitation à des types particuliers des symétries de Lie explique essentiellement le gain de complexité de nos algorithmes. L'algèbre linéaire sur un corps numérique et la factorisation des polynômes univariés sont suffisantes pour les opérations nécessaires. Les travaux existants sur ce sujet (voir par exemple [42, 59, 88]) considèrent essentiellement les symétries de Lie écrites sous forme générale. Le calcul de ces symétries générales nécessite des calculs de base de Gröbner et/ou la résolution des équations différentielles partielles. Leur complexité est donc en général au moins exponentielle.

Les algorithmes que je présente dans ce document sont disponibles dans les logiciels `MABSys` et `ExpandedLiePointSymmetry` (voir le paragraphe suivant pour plus de détails). Ces implémentations permettent de traiter les exemples de taille moyenne. L'étude préliminaire et l'analyse qualitative symbolique d'un tel exemple sont données dans le chapitre 1. Cet exemple illustre l'utilisation simultane d'une méthode de simplification inexacte (l'approximation quasi-stationnaire, voir [10]) et des méthodes de simplifications exactes (réduction et reparamétrisation). Ces algorithmes de réduction et de reparamétrisation adaptent la théorie des symétries de Lie au contexte de modélisation en biologie (voir [78] pour un point de vue sur les systèmes algébriques en biologie).

Contributions

Je propose des méthodes pour la gestion des modèles de taille moyenne et l'implémentation pilote de deux logiciels. Les détails de mes contributions ainsi que mes publications sont donnés ci-dessous.

Gestion des modèles de taille moyenne (environ vingt variables). Traiter ces modèles manuellement demande beaucoup de temps pour un résultat incomplet et peu fiable. Je propose des méthodes, aussi algorithmique que possible, de simplification et d'analyse. Je manipule à la fois des algorithmes nouveaux et des algorithmes connus. Tous ces algorithmes sont implémentés dans les logiciels `MABSys` et `ExpandedLiePointSymmetry`.

*Logiciel : `MABSys` (**M**odeling and **A**nalysis of **B**iological **Sys**tems).* J'ai conçu et développé `MABSys` avec F. Lemaire. Ce logiciel inclut les méthodes de simplification exactes basées sur la théorie des symétries de Lie et l'approximation quasi-stationnaire qui a été rendue algorithmique par les auteurs de [10]. La théorie classique des symétries de Lie est adaptée d'une façon originale au contexte de la modélisation en biologie. De plus, ces fonctionnalités sont assemblées avec des outils classiques de l'analyse qualitative symbolique.

Logiciel : `ExpandedLiePointSymmetry`. J'ai contribué au développement du logiciel `ExpandedLiePointSymmetry`, initié par A. Sedoglavic. Les premières versions de ce logiciel ont été conçues pendant mon stage de master recherche [109, 110]. Il manipule le calcul des symétries de Lie et la réduction de modèle en complexité polynomiale en la taille de l'entrée.

Publications. J'ai contribué aux deux articles suivants.

1. L'article [12] appartient à la conférence internationale "Algebraic Biology". Ce papier illustre l'utilisation des méthodes de calcul formel pour déduire les conditions d'oscillation d'une famille de réseaux de régulation génétiques avec un seul gène auto-régulé. Cet exemple historique est traité manuellement.
2. L'article [15] appartient au numéro spécial "Symbolic Computation in Biology" du journal "Mathematics in Computer Science". Ce papier présente la première version d'un schéma algorithmique dédié au problème de réduction de modèle. Les modèles en question sont composés des équations différentielles ordinaires et polynomiales. Ces équations peuvent être obtenues à partir des systèmes de réactions chimiques généralisées. Cette publication montre le progrès accompli depuis la parution de [12] et illustre les travaux des trois dernières années de l'équipe Calcul Formel dans le domaine de la modélisation en biologie. Dans [15], on arrive aux mêmes résultats que ceux obtenus manuellement dans [12] en utilisant le plus possible les fonctionnalités de `MABSys`.

Les conclusions de ces articles ont poussées des chercheurs dans le domaine de la géométrie réelle à adapter leur méthodes (voir [106]) ou à interagir avec les auteurs de [12] (voir [97]). De plus, dans la prépublication [68], les auteurs présentent des stratégies de calcul en utilisant la représentation explicite (infinitésimale) des données dans le processus de la réduction de systèmes dynamiques. Le but est d'obtenir une complexité polynomiale en la taille de l'entrée. Ces stratégies mènent à des algorithmes efficaces utilisés dans les implémentations [102, 69].

Structure de document

Le premier chapitre, constituant avec l'introduction la première partie, est consacré à la présentation de mes résultats à travers un exemple historique, traité à l'aide du logiciel `MABSys`. La dernière section de ce chapitre discute de mes contributions en se basant sur cet exemple.

La deuxième partie constituée du deuxième chapitre est une introduction aux méthodes de réduction et de reparamétrisation. Les idées principales et les algorithmes utilisés dans le logiciel `MABSys` y sont décrits. La dernière section de ce chapitre positionne mes travaux par rapport à ceux existants.

La troisième partie de ce document généralise les algorithmes précédents. Les aspects techniques de mes travaux sont détaillées dans les chapitres 3, 4 et 5. Le troisième chapitre définit les notations et le cadre géométrique. Les propriétés des objets considérés et leur codage y sont également présentés. Le quatrième chapitre est entièrement consacré aux symétries de Lie étendues, à leur définition et à leur calcul dans le logiciel `ExpandedLiePointSymmetry`. Le dernier chapitre traite les processus de réduction et de reparamétrisation ainsi que leur implémentation.

L'annexe A présente quelques outils de l'analyse qualitative symbolique relatifs à la bifurcation de Hopf. L'annexe B détaille les réseaux biochimiques et leur codage dans `MABSys` ainsi que leur modélisation en système différentiel ordinare.

"...to look at the stars always makes me dream,..."

Letter to Theo van Gogh
Arles, July 9^{th}, 1888
Vincent van Gogh

I Introduction

The goal of my research is to make algorithmic, as much as possible, the study of models composed by parametric differential equations. The methods presented throughout this document are not limited to intracellular biology but my principal application example belongs to genetic regulatory networks field. Considered mathematical models are systems of nonlinear (polynomial) parametric differential equations. They may be obtained for instance from generalized biochemical reaction systems by application of the mass-action law (see [55]). I focus on the algorithms developed in the computer algebra discipline for medium size (about twenty variables) models. I work with mainly two different types of algorithms devoted to models treatment: quasi-steady state approximation (based on differential elimination theory) and exact simplification (based on Lie symmetry theory) methods. These methods involve the reduction of the number of variables of a model and its reparametrization in order to distinguish the roles of its parameters. Simplified models resulting from these algorithms allow to perform more easily various studies such as symbolic or numerical qualitative analysis, numerical simulations, etc. For instance, the symbolic qualitative analysis methods are of at least exponential complexity in the number of variables in worst cases. Thus reducing the number of variables of the models or some expressions derived from them is useful to ease their analysis.

Leading Ideas

The reparametrization method that I propose is an exact simplification method based on a new algorithm that uses the classical Lie symmetry theory. It improves the preliminary analysis of models and facilitates their qualitative analysis. All the information related to the simplification is kept in order to establish the link between the original and the simplified systems. Moreover, this new algorithm is of polynomial time complexity in the input size thanks to some computational strategies such as the restriction of the set of Lie symmetries of interest.

The reduction method presented in this document is a classical application of the Lie symmetry theory. The classical reduction method (see [24, 86]) and the moving frame based reduction method (see [37, 38]) are adapted so that the time complexity of our algorithms is polynomial in the input size. Still, they generalize, for example, the dimensional analysis (see ch. 1 of [6], [17]) for which one does not have any complexity result (see [64]). Furthermore, our algorithms do not require to specify the dimension of the variables but only some assumptions about their positivity. This processus is also partially applied to discrete dynamical systems. To our knowledge, even if such systems are already treated with Lie symmetries (see [70, 117]), no algorithmic method with polynomial time complexity in the input size is available.

The gain of the complexity of our algorithms arises mostly from the limitation to special kind Lie symmetries. The linear algebra over a number field and univariate polynomial factorization are thus sufficient for the required operations. The existing works about this subject (see for example [42, 59, 88]) consider Lie symmetries mostly with their general form. The computation of such general symmetries necessitates Gröbner bases computation and/or solving partial differential equations. Thus their complexity in general cases is at least exponential.

The algorithms that I present in this document are available in the software `MABSys` and `ExpandedLiePointSymmetry` (see the next paragraph for more details). Thanks to these implementations, medium size examples can be tackled. In the chapter 1, the preliminary study and the complete symbolic qualitative analysis of such a realistic example are given. This example illustrates also the usage of inexact (quasi-steady state approximation, see [10]) and exact (reduction and reparametrization) simplification methods together. Remark that the reduction and the reparametrization algorithms adapt the Lie symmetry theory to the context of modeling in biology (see [78] for an opinion about algebraic systems biology).

Contributions

I propose methodical ways of managing medium size models and the pilot implementations of two software. Details of these contributions along with my publications follow.

Management of medium size models (about twenty variables). Dealing with such models manually can be time-consuming, patchy and unreliable. I propose methodical and algorithmic (as much as possible) ways of simplification and analysis. For this issue, I manipulate some new and some known algorithms together. These algorithms are implemented in `MABSys` and `ExpandedLiePointSymmetry` packages.

*Software: `MABSys` (**M**odeling and **A**nalysis of **B**iological **Sys**tems).* I conceived and developed `MABSys` with F. Lemaire. This software involves the exact simplification methods based on the Lie symmetry theory and the quasi-steady state approximation which is made fully algorithmic by the authors of [10]. The classical Lie symmetry theory is adapted in an original way to the context of modeling in biology. Moreover, these tools are successfully coupled with each other and also with some classical symbolic qualitative analysis tools.

Software: `ExpandedLiePointSymmetry`. I contributed to the development of the software `ExpandedLiePointSymmetry` initiated by A. Sedoglavic. First sketches of this software were conceptualized during my master thesis [109, 110]. It handles Lie symmetries computations and model reductions in polynomial time complexity in the input size.

Publications. I contributed to the following two publications.

1. The article [12] is an article of the international conference "Algebraic Biology". This paper illustrates the usage of computer algebra methods to prove that gene regulatory networks can (or can not) oscillate. An historical example of a family of genetic networks with a single self-regulated gene is treated manually. It concludes that under some conditions, oscillations may be observed for the studied model.

2. The article [15] is in the journal "Mathematics in Computer Science" special issue "Symbolic Computation in Biology". This paper presents the first version of an algorithmic scheme dedicated to the model reduction problem, in the context of polynomial ordinary differential equation models derived from generalized chemical reaction systems. It shows the progress made w.r.t. the computations made in [12].

It required three years work in the domain of modeling in biology in the Computer Algebra team. In [15], the same results obtained in [12] manually are reached by using as much as possible the functionalities of `MABSys`.

The conclusions of these articles led some researchers in real geometry domain to adapt their methods (see [106]) or to interact with the authors of [12] (see [97]). Furthermore in the preprint [68], authors present the computational strategies using non explicit (infinitesimal) data representation in the reduction process of dynamical systems in order to obtain a polynomial time complexity in the input size. These strategies lead to efficient algorithms used in the implementations of [102, 69].

Document Structure

The first chapter, which is in the first part, is devoted to the presentation of my results through an historical example that is treated by the help of the software `MABSys`. The last section of this chapter discusses my contributions in the view of this example.

The second part, composed of the second chapter, is an introduction to the reduction and the reparametrization methods. We give the main ideas and the associated algorithms used in the software `MABSys`. Its last section points out the differences of my work w.r.t. the existing ones.

The third part of this document generalizes the previous algorithms. The technical aspects of my work are detailed in the chapters 3, 4 and 5. The third chapter defines the notations and the geometrical framework. Beside the properties of the associated objects, their encodings are also presented. The fourth chapter is completely devoted to the expanded Lie point symmetries and their computation in the `ExpandedLiePointSymmetry` package. The last chapter deals with the reduction and the reparametrization processes along with their implementations.

The appendix A presents some symbolic qualitative analysis tools related to the Hopf bifurcation. The appendix B details biochemical networks and their encoding in `MABSys`. It tackles also their modeling by means of ordinary differential systems.

1 Overview

The goal of this chapter is to present the results of my Ph.D. thesis through an historical example. First the main structure of the software `MABSys` (Modeling and Analysis of Biological Systems) is given. The family of genetic networks with a single self-regulated gene is then treated by the help of `MABSys`. The model conception of this example, the simplification of this model by exact simplification methods based on Lie symmetry theory and its symbolic qualitative analysis are detailed. The aim is to show how one can realize necessary computations in an algorithmic way as much as possible. This chapter is concluded by a discussion.

1.1 Main Structure of the Software `MABSys`

This section details the main structure of the software `MABSys` (Modeling and Analysis of Biological Systems). This is a pilot implementation (in `Maple`) that I conceived and developed with F. Lemaire (see [69]). It gathers, as much as possible, some functions to carry out the symbolic qualitative analysis of systems of ordinary differential equations (ODEs). It is oriented towards biochemical reaction networks. `MABSys` includes three main parts: modeling of biochemical reaction networks by means of ODEs, simplification of these dynamical systems and symbolic qualitative analysis tools to retrieve informations about the behavior of the model.

The main structure of `MABSys` is given in figure 1.1 page 8. Every box corresponds to a component composed of a set of functions. Solid arrows between two components indicate that the outputs of the first component can be used as inputs of the second one. Dotted arrows inside components show possible successions of computations.

In `MABSys`, biochemical reaction networks are denoted by the usual chemical notation which is used to generate models of ODEs. For instance, the reaction $A + B \rightarrow C$ between 3 chemical species is represented by its *reactants* (A,B), its *product* (C) and the *rate law* of the reaction (see § B.1 page 151 for more details). Two intimately related kinds of modeling by means of ODEs are available for these networks. On the one hand, one can use directly the rate laws of the reactions to construct the so called *basic model* (see § B.2.2 page 157). On the other hand, the new algorithm `ModelReduce`, which performs quasi-steady state approximation (QSSA, see [10] and § B.2.3 page 158), assures the modeling by reducing this basic model. The QSSA is preferable if the network possesses two timescales, fast and slow reactions, since it can lead to a simpler model than the basic one.

Any system of polynomial ODEs, coming from any scientific context, can be treated by the exact simplifications part i.e. by the reduction and the reparametrization (see ch. 5

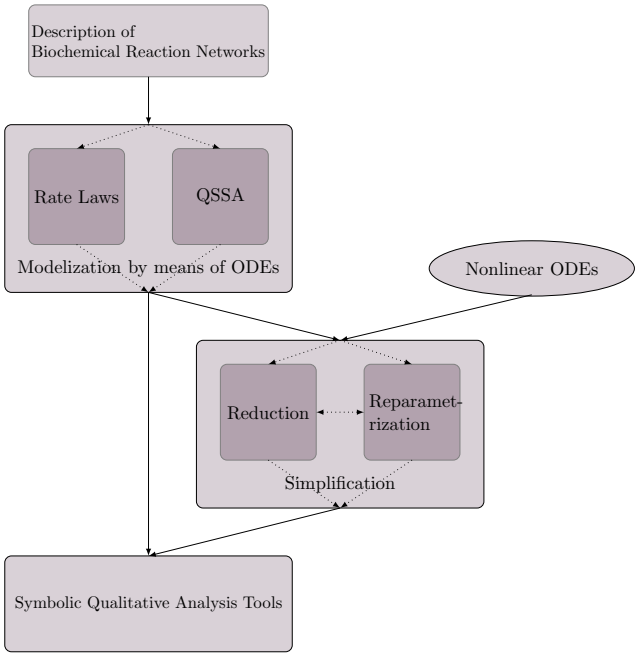


Figure 1.1: Main structure of `MABSys`.

page 103) methods. These simplifications are based on the classical Lie symmetry theory which is adapted in an original way to the modeling in biology (see ch. 2 page 33). The reduction of the parameter set of a model consists in *eliminating* some parameters thus in decreasing their number. The reparametrization of a model consists in distinguishing the roles of the parameters: the ones that decide the place of the steady points and the ones that decide their nature. The goal of these simplifications is to reorganize the model coordinates so that the resulting equivalent model can be more easily tractable. Even if the simplification algorithms are fully automatic, the choice of the parameters to eliminate, the assumptions to make, etc. still need human reflection.

The last part of `MABSys` involves some symbolic qualitative analysis tools for computing steady points, Hopf bifurcation conditions (see ch. A page 141), etc. They are implemented in coherence with the previous functionalities in order to complete, as much as possible, the study of models.

1.2 A Family of Genetic Networks with a Single Self-Regulated Gene

This section is devoted to the presentation of a family of genetic networks with a single self-regulated gene, borrowed from [12, 11, 15]. We give first the motivation and the problem statement of this chapter, then detail the networks of interest.

1.2.1 Motivation

The motivation of the choice of this family of genetic networks comes from the study of the green alga *ostreococcus tauri* (see figure 1.2) discovered in 1994, in the south of France.

This alga is one of the smallest known eukaryotes and its genome was published in 2006 (see [91]). Though very simple, this unicellular organism is endowed by a circadian clock i.e. a clock of period about 24 hours (see [41] and references therein). This clock permits the alga to raise itself at the top of water before sunrise. O. tauri is one of the main study domain of the laboratory "Observatoire Océanologique de Banyuls-Sur-Mer (OOB)" (see [2]).

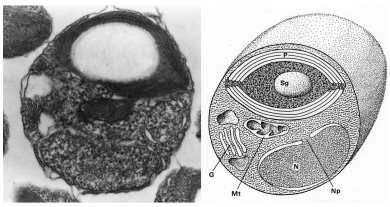


Figure 1.2: Ostreococcus tauri. Unicellular green alga characterized by minimal cellular organization. It possesses a nucleus (N) with a hole (Np), a chloroplast (P) with an amide ball (Sg), a mitochondrion (Mt) and a Golgi apparatus (G).

Some Computer Algebra team members, including myself, participated in a pluridisciplinary working group (see [79, 12, 11]) with biologists, computer scientists and physicists led by F.-Y. Bouget from OOB and by M. Lefranc from the Nonlinear Dynamics team of the Laboratory PhLAM (CNRS, UMR 8523). The issue was to model the gene regulatory network controlling the circadian clock of O. tauri.

1.2.2 Problem Statement and Biological Results

This subsection states the problem carried out in this chapter and quickly discusses its biological significance.

One of the main problems considered by the working group can be formulated as follows: *given a system of ODEs built using the mass-action law kinetics, does there exist ranges of positive values for the model parameters and state variables which are associated to oscillating trajectories (limit cycles)?* This issue is theoretically very difficult, in part because:

- systems of ODEs which oscillate may do so only for very restricted ranges of parameters values;
- the number of parameters arising in biochemical models can quickly become very large.

These difficulties led us to consider a related but easier problem: *searching for parameters and state variables positive values which give rise to Hopf bifurcations* (see § A.2

page 145). In general, a Hopf bifurcation may cause oscillating behaviors appear or disappear.

We consider a family of networks to tackle this problem. The networks of this family are indexed by the integer n that indicates the polymerization order of some protein (see § 1.2.3). The study of the associated model shows that Hopf bifurcations can occur if, and only if, n is greater than 9. It should be stressed that for a biological system, a cooperativity of order 9 is not unrealistic. In particular, gene regulation by an octamer has been reported (see [103]). Moreover, an effective cooperativity of order 9 may also be obtained as a consequence of reducing a higher-dimensional model.

1.2.3 Presentation of the Considered Family of Genetic Networks

In this subsection, the networks of figure 1.3 are presented. They feature a single gene regulated by an order n polymer of its own protein.

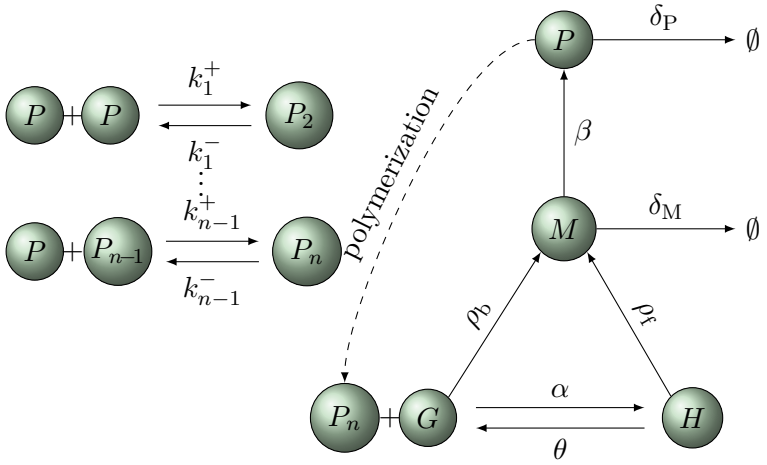


Figure 1.3: Family of networks of a single gene regulated by a polymer of its own protein.

These systems are indexed by n. One obtains different networks for different values of n. That is why the figure 1.3 is associated to a *family* of networks, not to a unique network. The variables G and H represent two states of the gene. This gene is *transcribed* into an mRNA, denoted by M, which is *translated* into a protein P. This regulatory protein forms polymers of order n. The system includes also the degradation of the mRNA and the protein. These networks involve a negative feedback loop, one of the core ingredients for generating oscillations (see ch. 9 of [36]). Greek letters and k_i^-, k_i^+ for all i in $\{1, \ldots, n-1\}$ represent parameters. Each parameter near an arrow stands for the associated mass-action law rate constant.

These abstract models are closely related to models studied in [47, 48, 49]. In particular, in [49] a model of a gene regulated by a polymer formed of n copies of its own protein is considered. Contrarily to this model, in this document, gene activation is not assumed to be faster than the other reactions. Only the polymerization reactions are supposed to be fast. Still, the results are consistent with those of [49].

1.3 Model Conception

This section is about constructing a model, as simple as possible, that represents the dynamic of the networks in figure 1.3 page 10. First, considered biochemical reactions are represented by `MABSys` in order to apply automatically in the sequel the modeling procedures. The networks are modeled by two methods. By using directly mass-action law one gets the *basic model* and by quasi-steady state approximation one gets the *raw reduced model*. The table 1.1 recapitulates the number of equations in these differential systems and the parameters of their parameters. As one can see, the raw reduced model involves less equations and parameters than the basic model. Finally we explain why

	Number of differential equations	Number of parameters of the differential system
Basic Model	$n+3$	$2\,n+5$
Raw Reduced Model	3	$n+8$

Table 1.1: Recapitulation of the basic and the raw reduced models.

the raw reduced model is a good approximation of the basic one.

1.3.1 Definition of the Biochemical Reactions in `MABSys`

Let us see now how one can use `MABSys` to represent the biochemical reactions of the networks given in figure 1.3 page 10. These $2\,n+5$ reactions are listed in figure 1.4.

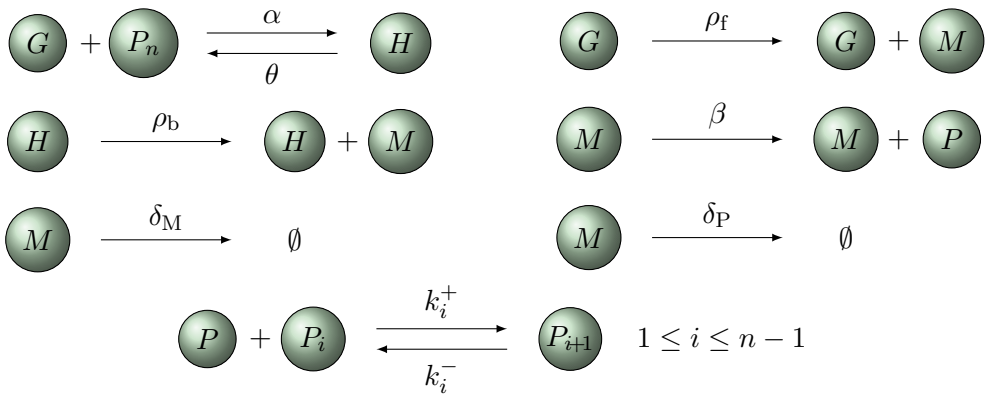


Figure 1.4: Reactions of the networks given in figure 1.3 page 10.

Unfortunately, neither the representations nor the computations can be performed by keeping a symbolic value for n. This is a classical restriction of symbolic computation methods. In the sequel the following arbitrary choice for n is made:

```
> n := 5;
       n := 5
```

Observe that, however, all the parameters and state variables are handled symbolically.

Remark 1.3.1. Because the packages `MABSys` and `ExpandedLiePointSymmetry` are developed in `Maple`, in the whole document, piece of codes are detailed by respecting the `Maple` syntax. Each line in code frames that begins with the symbol "`>`" stands for the commands sent to `Maple`. If a command ends with a semicolon then its output is displayed, but if it ends with a colon then its output is hidden. Each line without the symbol "`>`" stands for an output. The symbol "`#`" indicates a comment. Also, keywords that belong to `MABSys` or `ExpandedLiePointSymmetry` are written in bold so that one can distinguish them from standard `Maple` commands. ⌟

In `MABSys`, a biochemical reaction network is a list of biochemical reactions. A one-way biochemical reaction is defined by the constructor `NewReaction` (see § B.1.2 page 153). It takes as input the reactants, the products and the rate law of a reaction.

A fourth optional argument permits to set the speed of the reaction. The default value is slow (`fast=false`). This information is only used by the `ModelReduce` function (QSSA). In this example, the polymerization reactions are supposed to be faster than the other ones. Thus, the option `fast=true` is specified. The following commands define the biochemical reaction network for $n = 5$ that is composed of the reactions given in figure 1.4 page 11. Remark that the `cat` command of `Maple` concatenates its arguments in order to produce new identifiers and the `seq` command creates a sequence of expressions.

```
> RS := [
>     NewReaction(G+cat('P',n),H,MassActionLaw(alpha)),
>     NewReaction(H,G+cat('P',n),MassActionLaw(theta)),
>     NewReaction(G,G+M,MassActionLaw(rhof)),
>     NewReaction(H,H+M,MassActionLaw(rhob)),
>     NewReaction(M,M+P1,MassActionLaw(beta)),
>     NewReaction(M,0,MassActionLaw(deltaM)),
>     NewReaction(P1,0,MassActionLaw(deltaP)),
>     seq(NewReaction(cat('P',i)+P1,cat('P',i+1),
>                         MassActionLaw(cat('k_',i)),fast=true), i=1..n-1),
>     seq(NewReaction(cat('P',i+1),cat('P',i)+P1,
>                         MassActionLaw(cat('km_',i)),fast=true), i=1..n-1)
> ];
       RS := [Reaction([G, P5], [H], MassActionLaw(alpha), false),
              Reaction([H], [G, P5], MassActionLaw(theta), false),
              Reaction([G], [M, G], MassActionLaw(rhof), false),
              Reaction([H], [H, M], MassActionLaw(rhob), false),
              Reaction([M], [M, P1], MassActionLaw(beta), false),
              Reaction([M], [], MassActionLaw(deltaM), false),
              Reaction([P1], [], MassActionLaw(deltaP), false),
              Reaction([2 P1], [P2], MassActionLaw(k_1), true),
              Reaction([P2, P1], [P3], MassActionLaw(k_2), true),
              Reaction([P3, P1], [P4], MassActionLaw(k_3), true),
              Reaction([P4, P1], [P5], MassActionLaw(k_4), true),
              Reaction([P2], [2 P1], MassActionLaw(km_1), true),
              Reaction([P3], [P2, P1], MassActionLaw(km_2), true),
              Reaction([P4], [P3, P1], MassActionLaw(km_3), true),
              Reaction([P5], [P4, P1], MassActionLaw(km_4), true)]
```

1.3.2 Basic Modeling

The basic modeling is obtained by the direct application of the mass-action law (see § B.2.2 page 157). We apply it to the family of genetic networks given in figure 1.4 page 11.

The basic model of the biochemical reactions of figure 1.4 page 11 follows:

$$\left\{\begin{array}{rcl} \dot{G} &=& \theta\, H - \alpha\, G\, P_n, \\ \dot{H} &=& -\theta\, H + \alpha\, G\, P_n, \\ \dot{M} &=& \rho_{\mathrm{f}}\, G + \rho_{\mathrm{b}}\, H - \delta_{\mathrm{M}}\, M, \\ \dot{P} &=& \beta\, M - \delta_{\mathrm{P}}\, P + 2\, A_1 + A_2 + \cdots + A_{n-1}, \\ \dot{P}_i &=& -A_{i-1} + A_i \quad \text{with} \quad 2 \leq i \leq n-1, \\ \dot{P}_n &=& -A_{n-1} + \theta\, H - \alpha\, G\, P_n. \end{array}\right. \tag{1.1}$$

In this model $A_i := k_i^- P_{i+1} - k_i^+ P_i P$ for all i in $\{1, \ldots, n-1\}$ are introduced to lighten up notations. Capital letters $G, H, M, P = P_1, P_2, \ldots, P_n$ represent state variables and $\dot{X}$ represents the derivative of the variable X w.r.t. the independent variable t (time). They all correspond to species concentrations except G and H which should rather be viewed as rates of gene transcription. This system involves $n+3$ nonlinear (because of the multiplications between its coordinates) ODEs depending on $2\,n+5$ parameters. For interesting values of n, for instance for $n = 9$ (see § 1.5 page 22), this is a medium size system.

The basic model (1.1) can be obtained automatically for a given n by calling the function `ReactionSystem2ODEs`. In its current version, `MABSys` cannot compute directly the general form of the basic model. However it can be applied for many different values of n and the general formulas can be inferred. The arguments of the `ReactionSystem2ODEs` function are the reaction system and an order on the set of the involved species, given by a list. The commands for $n = 5$ follow.

```
> # The order of the chemical species
> X := [G,H,M,seq(cat('P',n-i+1),i=1..n)];
       X := [G, H, M, P5, P4, P3, P2, P1]

> # The basic model
> BasicModel := ReactionSystem2ODEs(RS, X);
BasicModel :=
 d
[-- G(t) = -alpha G(t) P5(t) + theta H(t),
 dt
 d
 -- H(t) = alpha G(t) P5(t) - theta H(t),
 dt
 d
 -- M(t) = rhof G(t) + rhob H(t) - deltaM M(t),
 dt
 d
 -- P5(t) = -alpha G(t) P5(t) + theta H(t) + k_4 P4(t) P1(t) - km_4 P5(t),
 dt
 d
 -- P4(t) = k_3 P3(t) P1(t) - k_4 P4(t) P1(t) - km_3 P4(t) + km_4 P5(t),
 dt
 d
 -- P3(t) = k_2 P2(t) P1(t) - k_3 P3(t) P1(t) - km_2 P3(t) + km_3 P4(t),
 dt
 d                    2
 -- P2(t) = k_1 P1(t)  - k_2 P2(t) P1(t) - km_1 P2(t) + km_2 P3(t),
 dt
 d                                                  2
 -- P1(t) = beta M(t) - deltaP P1(t)- 2 k_1 P1(t)  - k_2 P2(t) P1(t) + km_2 P3(t)
 dt
    - k_3 P3(t) P1(t) - k_4 P4(t) P1(t) + 2 km_1 P2(t) + km_3 P4(t) + km_4 P5(t)]
```

1.3.3 Quasi-Steady State Approximation

The function `ModelReduce` attempts to construct a raw reduced model that represents the biochemical reaction networks behavior with less chemical species thus simpler than the basic modeling. It applies the quasi-steady state approximation (QSSA) and it is based on differential elimination theory (see [10, 8, 7, 118]). In fact, this process takes into account the speed of the reactions. In the sequel, it is applied to our family of genetic networks.

The QSSA requires for the network to have two different timescales. In our case, these timescales are provided by the speed of the reactions (fast and slow). For the networks of figure 1.3 page 10, the $n-1$ polymerization reactions are assumed to be fast compared to the other ones. That means that the rate constants k_i^-, k_i^+ for all i in $\{1, \ldots, n-1\}$ are supposed to be *greater* than the remaining parameters.

According to the technique sketched in [10] (see § B.2.3 page 158), one gets an approximation of the system (1.1) page 13 by replacing each A_i by a new dependent variable F_i for all i in $\{1, \ldots, n-1\}$ and by augmenting this system by the $n-1$ following algebraic equations:

$$k_i^+ \, P \, P_i - k_i^- \, P_{i+1} = 0 \quad \text{with} \quad 1 \leq i \leq n-1. \tag{1.2}$$

It is sufficient to eliminate the F_i from the so obtained differential-algebraic system to get the raw reduced model.

In order to clarify the model presentation, a few extra manual changes are done. G and H are two variables that are bound by the relation $G + H = \gamma_0$ where the new positive parameter γ_0 is equal to the total quantity of the gene. That is why H is replaced by $\gamma_0 - G$ manually. Finally n new parameters K_i are introduced for legibility with the convention $K_0 = 1$:

$$K_i = \frac{k_1^+ \cdots k_i^+}{k_1^- \cdots k_i^-}. \tag{1.3}$$

The obtained raw reduced model writes:

$$\left\{ \begin{array}{rcl} \dot{G} & = & (\gamma_0 - G)\,\theta - \alpha\, K_{n-1}\, G\, P^n, \\ \dot{M} & = & (\gamma_0 - G)\,\rho_{\mathrm{b}} + \rho_{\mathrm{f}}\, G - \delta_{\mathrm{M}}\, M, \\ \dot{P} & = & \dfrac{(\gamma_0 - G)\, n\, \theta - n\, \alpha\, K_{n-1}\, G\, P^n - \delta_{\mathrm{P}}\, P + \beta\, M}{\displaystyle\sum_{i=0}^{n-1} (i+1)^2\, K_i\, P^i}. \end{array} \right. \tag{1.4}$$

The system resulting from the quasi-steady state approximation contains also $n-1$ ODEs on the derivatives of P_i with $2 \leq i \leq n-1$ depending only on P. Removing these equations does not influence the dynamics of G, M and P.

The `subs` command of `Maple` substitutes subexpressions into an expression and the `map` command applies a procedure to each operand of an expression. The function `subs` performs the extra changes done for user convenience. Following commands show how one can obtain the system (1.4) for $n = 5$. The generic formula valid for any n can be deduced from several applications of `ModelReduce` to different values of n.

```
> RawReducedModel := ModelReduce(RS,X) [1,1]:
> RawReducedModel :=
>         subs(H(t)=gamma0-G(t),
>         P1(t)=P(t),
>         seq(cat('k_',i)=cat('km_',i)*cat('K_',i)/cat('K_',i-1),i=1..n-1),
>         K_0=1,
>         [RawReducedModel[1],RawReducedModel[3],RawReducedModel[-1]]):
> RawReducedModel := map(simplify, RawReducedModel);
                     d                                                5
RawReducedModel := [-- G(t) = theta gamma0 - theta G(t) - P(t)  K_4 alpha G(t),
                     dt

           d
           -- M(t) = rhof G(t) + rhob gamma0 - rhob G(t) - deltaM M(t),
           dt

                                                                      5
d         5 theta gamma0-5 theta G(t)+beta M(t)-deltaP P(t)-5 P(t) K_4 alpha G(t)
-- P(t)= ------------------------------------------------------------------------]
dt                   4               3                  2
              25 P(t)  K_4 + 16 P(t)  K_3 + 1 + 9 P(t)  K_2 + 4 K_1 P(t)
```

1.3.4 Basic Model vs Raw Reduced Model

In the sequel we prefer to keep the approximative raw reduced model to study the behavior of the networks of figure 1.3 page 10. The figure 1.5 shows the comparison between the basic model (1.1) page 13 and the raw reduced model (1.4) page 14, for different values of the parameters k_i^-, k_i^+ for all i in $\{1, \ldots, n-1\}$. We take the same value for all these parameters i.e. $k = k_1^- = \cdots = k_{n-1}^- = k_1^+ = \cdots = k_{n-1}^+$.

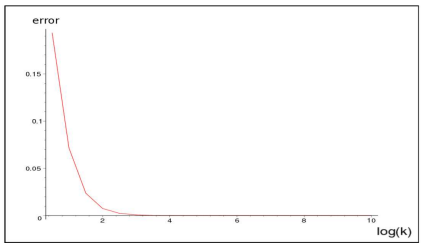


Figure 1.5: Comparison between the basic model (1.1) page 13 and the raw reduced model (1.4) page 14 for different values of the parameters k_i^-, k_i^+ for all i in $\{1, \ldots, n-1\}$. We vary the value of $k = k_1^- = \cdots = k_{n-1}^- = k_1^+ = \cdots = k_{n-1}^+$ and we take: $\theta = 10$, $\alpha = 1$, $\rho_{\mathrm{f}} = 10$, $\rho_{\mathrm{b}} = 5$, $\delta_{\mathrm{M}} = 5$, $\delta_{\mathrm{P}} = 10$, $\beta = 50$, $\gamma_0 = 1$, $K_1 = 1$, $K_2 = 1$, $K_3 = 1$, $K_4 = 1$, $P(0) = 1$, $M(0) = 1$, $G(0) = 0.5$, $H(0) = 0.5$, $P_2(0) = 1$, $P_3(0) = 1$, $P_4(0) = 1$ and $P_5(0) = 1$. On the graphics, the letters P, M and G represent the state variables of the basic model and the letters P', M' and G' represent these of the raw reduced model. We trace $\sqrt{((|P'-P|)/P)^2 + ((|M'-M|)/M)^2 + ((|G'-G|)/G)^2}$, in function of $\log(k)$. The comparison shows that more the value k of the parameters k_i^-, k_i^+ has greater value, more the approximative raw reduced model approaches the basic model. ⌟

In the figure 1.5 page 15, the horizontal axes represent the logarithm of the values of k. The vertical axes represent the absolute value of the difference between the values of the state variables of the basic model (1.1) page 13 and these of the raw reduced model (1.4) page 14. Observe that, as expected, when the value of k increases the raw reduced model tends to the basic one. See [11] for more details and another standpoint of this comparison.

The next section is devoted to the simplification of the parameter set of the raw reduced model (1.4) page 14. Observe that, in principle, one should not have introduced the parameters K_i since this operation should be performed in the next section. However, the `MABSys` package does not find this particular change of coordinates but a slightly more complicated one. In this document, these new parameters K_i are kept for the sake of legibility and in order to conserve the coherence with the article [15].

1.4 Exact Simplification of the Raw Reduced Model

This section is devoted to the simplification of the parameter set of the raw reduced model (1.4) page 14. Two intimately related but different exact methods, reduction and reparametrization (see ch. 2 page 33 and ch. 5 page 103) are presented. For each algorithm the result of the simplification, the change of coordinates that led to this result and the `MABSys` function that performs these computations are given.

The reduction and the reparametrization algorithms are based on the classical Lie symmetry theory. In `MABSys`, the symmetries are restricted to scalings for several reasons. Indeed parameters and state variables of a model representing a biochemical reaction network are supposed to be positive and scalings preserve this property. Furthermore the special form of scalings permits to find these new coordinates without solving any differential system.

The following computations are not fully automatic: the choice of parameters to get rid of, the parameters and variables to adjust, etc. remain under the control of the user. One has many possibilities for such choices. Here, we choose to make a simplification slightly different than that of [15]. This slightly simplifies the qualitative analysis (see remark 1.5.2 page 24).

1.4.1 Reduction of the Parameter Set

The objective of the reduction method is to *eliminate* some parameters by constructing new coordinates and thus to decrease their number. In this subsection first we give the reduced system of ODEs written in a new coordinate set and the explicit relations between the raw reduced model (1.4) page 14 and this one. Second we detail the function that performs this reduction automatically in `MABSys`.

There are many ways to choose the parameters to eliminate but their maximum number is limited by the number and the structure of the Lie symmetries of the system (see th. 5.1.1 page 103). In this chapter I decided to state the reduction of the parameters α and θ because the resulting reduced system is more interesting for the symbolic qualitative analysis (see remark 1.5.2 page 24) than the one treated in [15]. The reduction of

the parameters α and θ from the model (1.4) page 14 (in [15], β and δ_{P} are eliminated) leads to the following system of ODEs:

$$\left\{\begin{array}{rcl} \dot{\overline{G}} &=& \overline{\gamma}_0 - \overline{G} - \overline{K}_{n-1}\,\overline{G}\,\overline{P}^n, \\ \dot{\overline{M}} &=& \left(\overline{\gamma}_0 - \overline{G}\right)\overline{\rho}_{\mathrm{b}} + \overline{\rho}_{\mathrm{f}}\,\overline{G} - \overline{\delta}_{\mathrm{M}}\,\overline{M}, \\ \dot{\overline{P}} &=& \dfrac{\left(\overline{\gamma}_0 - \overline{G}\right) n - n\,\overline{K}_{n-1}\,\overline{G}\,\overline{P}^n - \overline{\delta}_{\mathrm{P}}\,\overline{P} + \overline{\beta}\,\overline{M}}{\displaystyle\sum_{i=0}^{n-1} (i+1)^2\,\overline{K}_i\,\overline{P}^i}. \end{array}\right. \tag{1.5}$$

This system is written in a new coordinate set where each new coordinate is overlined. Remark that this model involves, as expected, two less parameters than the raw reduced model. The relation between the coordinates of (1.4) page 14 and these of (1.5) can be expressed by the following change of coordinates:

$$\begin{array}{llll} \overline{t} = t\,\theta, & \overline{G} = \dfrac{G\,\alpha}{\theta}, & \overline{M} = M, & \overline{P} = \dfrac{P\,\alpha}{\theta}, \\ \overline{\beta} = \dfrac{\beta\,\alpha}{\theta^2}, & \overline{\gamma}_0 = \dfrac{\gamma_0\,\alpha}{\theta}, & \overline{\rho}_{\mathrm{b}} = \dfrac{\rho_{\mathrm{b}}}{\alpha}, & \overline{\rho}_{\mathrm{f}} = \dfrac{\rho_{\mathrm{f}}}{\alpha}, \\ \overline{\delta}_{\mathrm{M}} = \dfrac{\delta_{\mathrm{M}}}{\theta}, & \overline{\delta}_{\mathrm{P}} = \dfrac{\delta_{\mathrm{P}}}{\theta}, & \overline{K}_i = \dfrac{K_i\,\theta^i}{\alpha^i} & \end{array} \tag{1.6}$$

for all i in $\{1, \ldots, n-1\}$.

This reduced model and the associated change of coordinates can be obtained easily by using `MABSys`. The arguments of the function `InvariantizeByScalings` (see § 2.2.1 page 41) are the model to reduce and some indications to guide this reduction:

- the parameters to remove from the model by priority order;
- the coordinates to keep unchanged;
- a boolean that indicates if the time variable can be scaled or not during the reduction.

The algorithm takes each parameter in the first list and tries to eliminate it before considering the next one. It may happen that some of these parameters cannot be eliminated. Associated `MABSys` commands for $n = 5$ follow.

```
> # Reduction
> ParametersToRemove := [alpha,theta]:
> RemainingCoords := [G,M,P,seq(cat('K_',i), i=1..n-1),
>                     beta, deltaM, deltaP, gamma0, rhob, rhof]:
> out := InvariantizeByScalings(RawReducedModel, ParametersToRemove,
>                               RemainingCoords, scaletime=true):
> IntermediateModel1 := out[1];

IntermediateModel1 :=
        d                                  5
       [-- G(t) = gamma0 - G(t) - P(t)  K_4 G(t),
        dt
        d
        -- M(t) = rhof G(t) + rhob gamma0 - rhob G(t) - deltaM M(t),
        dt
```

```
                                                                        5
    d          5 gamma0 - 5 G(t) + beta M(t) - deltaP P(t) - 5 P(t)  K_4 G(t)
    -- P(t) = ------------------------------------------------------------]
    dt                 4                 3                 2
                25 P(t)  K_4 + 16 P(t)  K_3 + 1 + 9 P(t)  K_2 + 4 K_1 P(t)

> ChangeOfCoord1 := out[2];
                                      G alpha       P alpha          beta alpha
ChangeOfCoord1 := [t = t theta, G = -------,  P = -------, beta = -----------,
                                       theta         theta                  2
                                                                      theta

          gamma0 alpha           deltaM           deltaP        rhob         rhof
gamma0 = ------------, deltaM = ------, deltaP = ------, rhob= -----, rhof= -----,
             theta                theta            theta        alpha        alpha

                                    2                 3                 4
       K_1 theta         K_2 theta          K_3 theta         K_4 theta
 K_1 = ---------, K_2 = ----------, K_3 = ----------, K_4 = ----------]
         alpha                  2                 3                 4
                         alpha             alpha             alpha
```

The output contains two elements: the reduced model and the associated change of coordinates. Remark that in the implementation, the new coordinates are also denoted as the old ones for the sake of legibility and computational simplicity. The output must be interpreted as in (1.5) page 17 and (1.6) page 17.

1.4.2 Reparametrization

The objective of the reparametrization method is to make some parameters to appear as factors in the right-hand side of ODEs. This property simplifies the expressions of the associated steady points and permits to distinguish the roles of the parameters. For example, some parameters matter to decide just the nature (attractor or repellor) of steady points. Other ones are involved for the place and the nature of steady points. In this subsection first we give the reparametrized system of ODEs written again in a new coordinate set and the exact relations between the model (1.5) page 17 and this one. Second, we detail the function that performs this reparametrization automatically in `MABSys`.

The reparametrization of the system (1.5) page 17 leads to the following system of ODEs:

$$\left\{\begin{array}{rcl} \dot{\widehat{G}} & = & \widehat{\gamma}_0 - \widehat{G} - \widehat{G}\,\widehat{P}^n, \\ \dot{\widehat{M}} & = & \left((\widehat{\rho}_{\mathrm{f}} - 1)\,\widehat{G} + \widehat{\gamma}_0 - \widehat{M}\right)\widehat{\delta}_{\mathrm{M}}, \\ \dot{\widehat{P}} & = & \dfrac{\left(\left(\widehat{M} - \widehat{P}\right)\widehat{\rho}_{\mathrm{b}} + \left(\widehat{\gamma}_0 - \widehat{G} - \widehat{G}\,\widehat{P}^n\right) n\right)\widehat{\delta}_{\mathrm{P}}}{\left(n^2\,\widehat{K}_{n-1}\,\widehat{P}^{n-1} + \displaystyle\sum_{i=0}^{n-2} (i+1)^2\,\widehat{K}_i\,\widehat{K}_{n-1}^{-i}\,\widehat{P}^i\right)\widehat{\rho}_{\mathrm{b}}}. \end{array}\right. \tag{1.7}$$

This system is written in a new coordinate set where each new coordinate is endowed with a hat. The relation between the coordinates of (1.5) page 17 and these of (1.7) can

be expressed by the following change of coordinates:

$$\begin{array}{llll} \widehat{t} = \overline{t}, & \widehat{G} = \dfrac{\overline{G}\,\overline{K}_{n-1}^{(1/n)}\,\overline{\rho}_{\mathrm{b}}\,\overline{\beta}}{\overline{\delta}_{\mathrm{M}}\,\overline{\delta}_{\mathrm{P}}}, & \widehat{M} = \dfrac{\overline{M}\,\overline{K}_{n-1}^{(1/n)}\,\overline{\beta}}{\overline{\delta}_{\mathrm{P}}}, & \widehat{P} = \overline{P}\,\overline{K}_{n-1}^{(1/n)}, \\ \widehat{\beta} = \overline{\beta}, & \widehat{\gamma}_0 = \dfrac{\overline{\gamma}_0\,\overline{K}_{n-1}^{(1/n)}\,\overline{\rho}_{\mathrm{b}}\,\overline{\beta}}{\overline{\delta}_{\mathrm{M}}\,\overline{\delta}_{\mathrm{P}}}, & \widehat{\rho}_{\mathrm{b}} = \dfrac{\overline{\rho}_{\mathrm{b}}\,\overline{\beta}}{\overline{\delta}_{\mathrm{M}}}, & \widehat{\rho}_{\mathrm{f}} = \dfrac{\overline{\rho}_{\mathrm{f}}}{\overline{\rho}_{\mathrm{b}}}, \\ \widehat{\delta}_{\mathrm{M}} = \overline{\delta}_{\mathrm{M}}, & \widehat{\delta}_{\mathrm{P}} = \overline{\delta}_{\mathrm{P}}, & \widehat{K}_{n-1} = \overline{K}_{n-1}^{(1/n)}, & \widehat{K}_i = \overline{K_i} \end{array} \tag{1.8}$$

for all i in $\{1, \ldots, n-2\}$. Remark that, thanks to the positivity assumption of the parameters, the rational fraction powers of coordinates used in this change of coordinates remain real.

The reparametrization simplifies the steady points equations. The steady points of the reduced system (1.5) page 17 depend on 6 parameters. After the reparametrization, the steady points of (1.7) page 18 depend on only 1 parameter, namely $\widehat{\rho}_f$ (see (1.12) page 22 for the equations of the steady points but remark that the coordinates are not endowed with a hat). The complexity of computations on the steady points decreases (see § 1.5.1 page 22).

This reparametrized model and the associated change of coordinates can be obtained relatively easily by using `MABSys`. The function `CylindrifySteadyPoints` (see § 2.2.2 page 45) needs the model to reduce and some indications to guide this reparametrization. One must specify:

- the coordinates considered as strictly positive. Their order has an influence on the form of the change of coordinates;
- remaining coordinates that are supposed to be positive or null;
- a boolean that indicates if the algorithm must or must not use the regular chains (see [67]).

The resulting change of coordinates is a sequence of expressions of the form $\bar{V} = F\,V$ where V belongs to the second list and F is an expression involving elements of the first list only. In practice state variables are given in the second list. This implies that state variables are not multiplied together and that parameters are always transformed into parameters. The use of regular chains increases the complexity of the computation but sometimes permits to find more symmetries. Associated `MABSys` commands for $n = 5$ follow.

```
> # Reparametrization
> Positive := [seq(cat('K_',n-i), i=1..n-1),
>         beta, deltaM, deltaP, rhob, rhof, gamma0]:
> RemainingCoords := [G,M,P]:
> out := CylindrifySteadyPoints(IntermediateModel1, Positive, RemainingCoords,
>                               triangularization=true):
> IntermediateModel2 := out[1,1];

IntermediateModel2 :=
        d                                5
       [-- G(t) = gamma0 - G(t) - P(t)  G(t),
        dt
```

```
        d
        -- M(t) = deltaM (rhof G(t) + gamma0 - G(t) - M(t)),
        dt
               3                                                              5
 d           K_4  deltaP ((M(t) - P(t)) rhob + 5 gamma0 - 5 G(t) - 5 P(t)  G(t))
 -- P(t) = ------------------------------------------------------------------------]
 dt                        4   4    3         3           2                        2
           rhob (25 P(t) K_4 +K_4 +16 P(t) K_3 +9 P(t) K_2 K_4 +4 K_1 P(t) K_4 )

> ChangeOfCoord2 := out[1,3];
                              1/5                      1/5
                            K_4    G rhob beta       K_4    M beta          (1/5)
ChangeOfCoord2 := [ G = ------------------, M = -------------, P = K_4      P,
                          deltaM deltaP              deltaP
             1/5
          K_4    gamma0 rhob beta          rhob beta          rhof
 gamma0 = ------------------------, rhob = ---------, rhof = ----,
              deltaM deltaP                deltaM             rhob

          (1/5)
 K_4 = K_4     ]
```

In the output one can find mainly the reparametrized model and the associated change of coordinates. As for the reduction, in the implementation the notation of the coordinates does not change even if the output must be interpreted as in (1.7) page 18 and (1.8) page 19.

1.4.3 Final Model and associated Change of Coordinates

The task of the simplification procedures is achieved. In the sequel, we finalize the model that is going to be used for the symbolic qualitative analysis, compute the global change of coordinates and show the associated `MABSys` commands.

We choose to make two extra change of coordinates on the model (1.7) page 18 for the sake of readability of the result and in order to keep the coherence with [15]. They do not correspond to any scaling. A final new coordinate set is constructed where new coordinates are endowed with a tilde. For n greater than or equal to 2, one has:

$$\widetilde{\delta}_{\mathrm{P}} = \frac{1}{\widehat{\delta}_{\mathrm{P}}}, \quad \widetilde{\rho}_{\mathrm{b}} = \frac{1}{\widehat{\rho}_{\mathrm{b}}} \tag{1.9}$$

and all other coordinates remain the same (the case $n = 1$ is slightly different). The global change of coordinates that is applied to the system of ODEs (1.4) page 14 is the composition of the three changes of coordinates (1.6) page 17, (1.8) page 19 and (1.9):

$$\begin{array}{llll}
\widetilde{t} = t\,\theta, & \widetilde{G} = \dfrac{G\,K_{n-1}^{(1/n)}\,\alpha^{(1/n)}\,\rho_{\mathrm{b}}\,\beta}{\theta^{(1/n)}\,\delta_{\mathrm{P}}\,\delta_{\mathrm{M}}}, & \widetilde{M} = \dfrac{M\,K_{n-1}^{(1/n)}\,\alpha^{(1/n)}\,\beta}{\theta^{(1/n)}\,\delta_{\mathrm{P}}}, & \widetilde{P} = \dfrac{P\,K_{n-1}^{(1/n)}\,\alpha^{(1/n)}}{\theta^{(1/n)}}, \\
\widetilde{\beta} = \dfrac{\beta\,\alpha}{\theta^2}, & \widetilde{\gamma}_0 = \dfrac{\gamma_0\,K_{n-1}^{(1/n)}\,\alpha^{(1/n)}\,\rho_{\mathrm{b}}\,\beta}{\theta^{(1/n)}\,\delta_{\mathrm{P}}\,\delta_{\mathrm{M}}}, & \widetilde{\rho}_{\mathrm{b}} = \dfrac{\delta_{\mathrm{M}}\,\theta}{\rho_{\mathrm{b}}\,\beta}, & \widetilde{\rho}_{\mathrm{f}} = \dfrac{\rho_{\mathrm{f}}}{\rho_{\mathrm{b}}}, \\
\widetilde{\delta}_{\mathrm{M}} = \dfrac{\delta_{\mathrm{M}}}{\theta}, & \widetilde{\delta}_{\mathrm{P}} = \dfrac{\theta}{\delta_{\mathrm{P}}}, & \widetilde{K}_{n-1} = \dfrac{K_{n-1}^{(1/n)}\,\theta^{((n-1)/n)}}{\alpha^{((n-1)/n)}}, & \widetilde{K}_i = \dfrac{K_i\,\theta^i}{\alpha^i}
\end{array} \tag{1.10}$$

for all i in $\{1, \ldots, n-2\}$.

Removing afterward the tildes to help the legibility, one gets the simplified system of ODEs:

$$\left\{\begin{array}{rcl} \dot{G} &=& \gamma_0 - G - G\,P^n, \\ \dot{M} &=& ((\rho_{\mathrm{f}} - 1)\,G + \gamma_0 - M)\,\delta_{\mathrm{M}}, \\ \dot{P} &=& \dfrac{M - P + (\gamma_0 - G - G\,P^n)\,n\,\rho_{\mathrm{b}}}{\left(n^2\,K_{n-1}\,P^{n-1} + \displaystyle\sum_{i=0}^{n-2}(i+1)^2\,K_i\,K_{n-1}^{-i}\,P^i\right)\delta_{\mathrm{P}}}. \end{array}\right. \tag{1.11}$$

`MABSys` commands that compute above system for $n = 5$ and the final change of coordinates follow.

```
> # Interactive change of coordinates
> if n=1 then
>     ChangeOfCoord3 := [deltaP=1/deltaP,rhob=deltaM/(rhob*beta)]:
> else
>     ChangeOfCoord3 := [deltaP=1/deltaP,rhob=1/rhob]:
> end if:
> # The final model
> FinalModel := ApplyChangeOfCoord(IntermediateModel2,ChangeOfCoord3);
                d                                  5
FinalModel := [-- G(t) = gamma0 - G(t) - P(t)  G(t),
                dt
                d
                -- M(t) = (rhof G(t) + gamma0 - G(t) - M(t)) deltaM,
                dt
              3                                                          5
d          K_4  (M(t) - P(t) + 5 rhob gamma0 - 5 rhob G(t) - 5 rhob P(t)  G(t))
-- P(t) = ------------------------------------------------ ---------------------]
dt                       4   4    3        3           2                       2
          deltaP(25 P(t) K_4 +K_4 +16 P(t) K_3 +9 P(t) K_2 K_4 +4 K_1 P(t) K_4 )
> # The final change of coordinates is the composition of the three
> # computed above.
> FinalChangeOfCoord := ComposeChangeOfCoord(ChangeOfCoord1,
>                                            ChangeOfCoord2,
>                                            ChangeOfCoord3);

FinalChangeOfCoord :=
                     1/5      1/5                      1/5      1/5
                  K_4    alpha    G rhob beta      K_4    alpha    M beta
[t = t theta, G = --------------------------, M = ---------------------,
                         1/5                                 1/5
                    theta    deltaM deltaP            theta    deltaP
       1/5      1/5                                  1/5      1/5
    K_4    alpha    P          rhof            K_4    alpha    gamma0 rhob beta
 P = ----------------, rhof = ------, gamma0 = ------------------------------,
             1/5               rhob                    1/5
          theta                                      theta    deltaM deltaP
        deltaM theta          beta alpha            deltaM           theta
 rhob = ------------, beta = ----------, deltaM = ------, deltaP = ------,
         rhob beta                  2               theta            deltaP
                                theta
                                   2                  3           1/5      4/5
        K_1 theta         K_2 theta          K_3 theta          K_4    theta
  K_1 = ---------, K_2 = ----------, K_3 = ----------, K_4 = ---------------]
          alpha                 2                 3                  4/5
                            alpha             alpha              alpha
```

1.5 Symbolic Qualitative Analysis of the Considered Model

This section discusses Hopf bifurcation conditions at steady points found via the Routh-Hurwitz criterion (see th. 13.4 of [51] and § A.2.3 page 147). We are in fact interested in the oscillations of the model (1.11) page 21 associated to a Hopf bifurcation. During this analysis, the `MABSys` package functions are used, as much as possible, to ease computations. This section aims to prove the following proposition (see [15]).

Proposition 1.5.1. *For positive values of state variables and parameters, the system of ODEs (1.11) page 21 can exhibit a Hopf bifurcation if, and only if, n is greater than or equal to* 9. ⌟

1.5.1 Steady Points

For a system of ODEs, a steady point is a point where the derivatives vanish (see § A.1.1 page 142). Here we explicit the steady points of the model (1.11) page 21, their properties and how we can compute them with the help of `MABSys`.

Thanks to Gröbner basis (see [20]) computations, the steady points of (1.11) page 21 are defined explicitly by:

$$\gamma_0 = \frac{P\,(1+P^n)}{P^n+\rho_{\mathrm{f}}}, \quad M = P, \quad G = \frac{P}{P^n+\rho_{\mathrm{f}}}. \tag{1.12}$$

As shown above, they depend on only 1 parameter ρ_{f}. The following paragraph details the choices made to find the expressions (1.12) but it is not essential to understand the main ideas of this section.

One computes a Gröbner basis of the ideal generated by the algebraic system defining steady points of (1.11) page 21 w.r.t. the lexicographical ordering $G > M > \gamma_0$. Other variables and parameters are considered as algebraically independent elements of the base field of the equations. Observe that one does not need to distinguish the roles of variables from the ones of the parameters at this step. In general, one cannot compute a Gröbner basis if a symbolic n is left as an exponent, but in our case, a generic Gröbner basis exists. The ordering was chosen carefully in order to achieve two important properties: the leading monomials are plain variables and the right-hand sides of the Gröbner basis equations are positive. The first property implies that the quotient ring defined by the Gröbner basis is a polynomial ring. The second property implies that there are no constraints on the values that can be assigned to the variables and parameters occurring in the right-hand sides of the Gröbner basis equations.

Computation of steady points expressions for $n = 5$ follows. Gröbner basis operations are performed by using `Groebner` package of `Maple`. The `normal` command of `Maple` normalizes a rational expression.

```
> SteadyPoint := SteadyPointSystem(FinalModel);
                                  5
SteadyPoint := [gamma0 - G - P  G, (rhof G + gamma0 - G - M) deltaM,
```

```
                3                                                    5
              K_4  (-M + P - 5 rhob gamma0 + 5 rhob G + 5 rhob P  G)
    - ------------------------------------------------------------------]
                  4    4      3       3          2                    2
      deltaP (25 P  K_4  + K_4  + 16 P  K_3 + 9 P  K_2 K_4 + 4 K_1 P K_4 )

> SteadyPointOrder := plex(G, M, gamma0):
> SteadyPoint := Groebner:-Basis(SteadyPoint, SteadyPointOrder);
                             6           5                          5
       SteadyPoint := [-P - P  + (rhof + P ) gamma0, M - P, -P + (rhof + P ) G]

> SteadyPoint :=
>     seq(Groebner:-LeadingMonomial(SteadyPoint[i], SteadyPointOrder) =
>         normal(Groebner:-NormalForm
>                (Groebner:-LeadingMonomial(SteadyPoint[i], SteadyPointOrder),
>                SteadyPoint,
>                SteadyPointOrder)),
>         i = 1..nops(SteadyPoint));
                                  5
                            P (1 + P )                  P
       SteadyPoint := gamma0 = ----------, M = P, G = ---------
                                  5                       5
                            rhof + P                rhof + P
```

1.5.2 Linearization

In order to study the Hopf bifurcations of the system (1.11) page 21 using linearization (see § A.1.2 page 143), one needs to consider the Jacobian matrix of that system evaluated over its steady points. Thanks to the striking properties of the steady points equations, in the next subsection, evaluating the Jacobian matrix at the steady points just amounts to replacing each element of the generic Jacobian matrix by its normal form. The following paragraphs detail this matrix and how one can reach it by the help of `MABSys`.

The Jacobian matrix writes:

$$J = \begin{pmatrix} -1 - P^n & 0 & -\dfrac{n\,P^n}{P^n + \rho_\mathrm{f}} \\ (\rho_\mathrm{f} - 1)\,\delta_\mathrm{M} & -\delta_\mathrm{M} & 0 \\ -\dfrac{(1 + P^n)\,n\,\rho_\mathrm{b}}{B} & \dfrac{1}{B} & -\dfrac{n^2\,\rho_\mathrm{b}\,P^n + P^n + \rho_\mathrm{f}}{(P^n + \rho_\mathrm{f})\,B} \end{pmatrix} \tag{1.13}$$

where $B = \left(n^2\,K_{n-1}\,P^{n-1} + \sum_{i=0}^{n-2} (i+1)^2\,K_i\,K_{n-1}^{-i}\,P^i \right) \delta_\mathrm{P}$.

The parameters δ_P and K_i only occur in B. It is thus possible to assign arbitrary positive values to B without disturbing the values of other expressions involved in the matrix elements. One can consider B as a new parameter.

Computations that lead to this final form of the Jacobian matrix for $n = 5$ follow.

```
> J := JacobianMatrix(FinalModel, statevars=[G,M,P]):
> J0 := map(normal, subs(SteadyPoint, J)):
```

```
> denominator := subs(K_0 = 1,
>                     n^2*P(t)^(n-1)*cat(K_,n-1)+
>                     add((i+1)^2*cat(K_,i)*cat(K_,n-1)^(-i)*P(t)^i,
>                         i=0..n-2))*deltaP:
> J0 := map(normal,subs(K_1 = solve(B - subs(P(t)=P,denominator), K_1), J0));
             [                                                  5          ]
             [           5                                   5 P           ]
             [     -1 - P              0              - ---------         ]
             [                                                   5         ]
             [                                           rhof + P          ]
             [                                                             ]
       J0 := [(rhof - 1) deltaM    -deltaM                 0               ]
             [                                                             ]
             [          5                                  5    5         ]
             [  5 (1 + P ) rhob                    25 rhob P  + P  + rhof]
             [- ---------------       1/B       - ----------------------]
             [          B                                    5           ]
             [                                        B (rhof + P )      ]
```

Remark 1.5.2. This Jacobian J involves one less parameter than that computed in [15]. This improvement is due to the difference of the simplification part. In [15], the parameters β and δ_P are eliminated from the raw reduced model whereas in this document the parameters α and θ are eliminated (see § 1.4.1 page 16). ⌟

1.5.3 Hopf Bifurcation Detection

The last part of our symbolic qualitative analysis requires the computation of Hopf bifurcation (see § A.2.2 page 146) conditions and their verification. First, we compute these conditions thanks to the Routh-Hurwitz criterion (see § A.2.3 page 147). Then we verify the proposition 1.5.1 page 22 by using some basic heuristics and supposing that all variables are positive. Finally we specify a set of positive numerical values for models variables that give birth to oscillations in order to illustrate our assertions.

If a Hopf bifurcation occurs for a given system of ODEs then one can expect the existence of parameters and state variables values for which the system oscillates. The proposition 1.5.1 page 22 implies that no Hopf bifurcation arises for positive values of state variables and parameters whenever n is less than or equal to 8 for the model (1.11) page 21. In order to prove that, it is sufficient to show that three Hurwitz determinants $c_{0,0}, c_{1,0}$ and $c_{2,0}$ are positive, thanks to theorem A.2.3 page 146 and theorem A.2.5 page 147. These polynomials are deduced from the characteristic polynomial of the above Jacobian matrix. Showing that some polynomial is always positive is an example of the real geometry problems which has an exponential complexity in the number of variables of the polynomial in the worst case (see [106, 97]). That is one of the reasons also why the simplification of the parameter set of a system of ODEs, thus associated Jacobian matrix, is very important for the symbolic qualitative analysis. Remark that the slightly different simplification of (1.4) page 14 comparing to [15] (see remark 1.5.2) permits to have one less parameter in the Hurwitz determinants. The following commands compute these Hurwitz determinants for $n = 5$.

```
> CP := normal(LinearAlgebra:-CharacteristicPolynomial(J0,x)):
> HD := HurwitzDeterminants(CP,x):
> c[0,0] := HD[1];
       c[0, 0] := 1
```

```
> c[1,0] := collect(HD[2],[P,rhof]);
c[1, 0] :=
   10                                                5
B P   + (1 + deltaM B + 25 rhob + B + B rhof) P  + B rhof + rhof + deltaM B rhof
------------------------------------------------------------------------------
                                          5
                                 B (rhof + P )

> c[2,0] := HD[3]:
```

One can compute the Hurwitz determinants as above for different values of n and then get the generic form of it or work directly with a symbolic exponent n as done in [14]. From now on, in computations a symbolic exponent n is used. Moreover the output of function calls are not detailed because of their hugeness (see following paragraphs for their size).

Positivity of Hurwitz Determinants

Recall that in this paragraph, all parameters and state variables are supposed to be positive because they represent biological quantities. The first determinant is equal to 1 thus positive. The second determinant

$$c_{1,0} = \frac{B\,P^{2\,n} + \left(1 + n^2\,\rho_\mathrm{b} + \delta_\mathrm{M}\,B + B + \rho_\mathrm{f}\,B\right)P^n + (1 + \delta_\mathrm{M}\,B + B)\,\rho_\mathrm{f}}{(P^n + \rho_\mathrm{f})\,B} \tag{1.14}$$

is also positive since the numerator and the denominator of its generic form are linear combinations of power products of positive state variables and parameters, with positive coefficients. Such a reasoning is possible thanks to simplifications that keep the positivity assumption of the coordinates. Now, let us simplify as much as possible the third Hurwitz determinant $c_{2,0}$. The denominator $B\,(P^n + \rho_\mathrm{f})^2$ of this rational fraction is positive so it can be removed. The numerator is a sum of more than 50 monomials, only two of which have negative coefficients. This is a polynomial in P^n. One performs a change of coordinates that consists of renaming P^n as P. At this step, one can try again the simplification techniques on this numerator but there is no more possible simplification. In the sequel, we show the positivity of this final polynomial denoted by $\tilde{c}_{2,0}$, as in [15]. The following proposition is the main heuristic used while studying the positivity of $\tilde{c}_{2,0}$.

Lemma 1.5.3. *A linear combination of power products of positive quantities with positive coefficients is always positive.* ⌟

The polynomial $\tilde{c}_{2,0}$ has the form:

$$d_0\,\rho_\mathrm{b}^2 + d_1\,\rho_\mathrm{b} + d_2 \tag{1.15}$$

where d_0 and d_1 are positive thanks to the lemma 1.5.3. Thus, $c_{2,0}$ is positive for each $\rho_\mathrm{b} > 0$ if and only if d_2 is nonnegative. One thus studies the non-negativity of d_2, factoring out the positive term $P + \rho_\mathrm{f}$. This polynomial has the form:

$$\frac{d_2}{P + \rho_\mathrm{f}} = e_0\,\rho_\mathrm{f} + e_1 \tag{1.16}$$

where e_1 is positive thanks to the lemma 1.5.3 page 25. Thus, d_2 is nonnegative for $\rho_\mathrm{f} > 0$ if and only if e_0 is nonnegative. One thus studies the non-negativity of e_0, which has the form:

$$e_0 = f_0\,P^2 + f_1\,P + f_2 \tag{1.17}$$

where f_0 and f_2 are positive again thanks to the lemma 1.5.3 page 25. Thus e_0 is nonnegative if and only if e_0 (viewed as a univariate polynomial in P) has no positive root. Since f_2/f_0 is the product of roots, two roots have the same sign and are nonzero. In these conditions, both roots are positive if and only if f_1 is negative (since it is the opposite of the sum of the roots) and the discriminant of e_0 is positive w.r.t. P (in order to have real roots). These polynomials write:

$$\begin{array}{l} q_1 = \delta^2 + 2\,B\,\delta + 2\,\delta - n\,\delta + 1 + 2\,B, \\ q_2 = \delta^4 - 2\,n\,\delta^3 - 4\,B\,n\,\delta^2 - 4\,n\,\delta^2 + n^2\,\delta^2 - 2\,\delta^2 - 4\,B\,n\,\delta - 2\,n\,\delta + 1 \end{array} \tag{1.18}$$

with $\delta = \delta_\mathrm{M}\,B$. One has e_0 nonnegative if and only if one does not have $q_1 < 0$ and $q_2 > 0$ at the same time. Let B_1 and B_2 vanish q_1 and q_2 respectively. Remark also that the coefficient of B in q_1 is always positive and that in q_2 is negative. Finally, one has e_0 nonnegative if and only if conditions $0 < B < B_1$ and $0 < B < B_2$ are not satisfied simultaneously i.e. if and only if B_1 and B_2 are not positive simultaneously. For this argument, one gets rid of denominators of B_1 and B_2 because they are positive. Also, denoting $\lambda = \delta + 1/\delta$, one must *not* have simultaneously:

$$\left\{ \begin{array}{l} -\lambda + n - 2 > 0, \\ \lambda^2 - 2\,n\,\lambda + n^2 - 4\,n - 4 > 0. \end{array} \right. \tag{1.19}$$

For the first inequality to hold, it is necessary that $n > \lambda + 2$. For the second inequality to hold, it is necessary that $n < \lambda + 2 - 2\sqrt{\lambda + 2}$ or $n > \lambda + 2 + 2\sqrt{\lambda + 2}$. Thus, for both conditions in (1.19) to hold, it is necessary that $n > \lambda + 2 + 2\sqrt{\lambda + 2}$. Since $\lambda = \delta + 1/\delta$ with $\delta > 0$, one has $\lambda \geq 2$ whence $n \geq 9$. Thus $c_{2,0}$ is positive if and only if $n \leq 8$. This concludes the proof of the left to right implication of the proposition 1.5.1 page 22.

Oscillations

Let us prove now right to the left implication of the proposition 1.5.1 page 22 by finding numerical values for (1.11) page 21 with n greater than or equal to 9 that give birth to a Hopf bifurcation. Let n be an integer strictly greater than 8. According to the theorem A.2.3 page 146, one gets a Hopf bifurcation if the following conditions, deduced from the characteristic polynomial, are satisfied: $c_{0,0} > 0, c_{1,0} > 0, c_{2,0} = 0$ and $c_{2,1} < 0$. Remember that all state variables and parameters are assumed to be positive.

Take $\delta = 1$. The conditions $0 < B < B_1$ and $0 < B < B_2$ permit to take $B = 1/10$. Since one has $\delta = \delta_\mathrm{M}\,B$, take $\delta_\mathrm{M} = 10$. Then the polynomial e_0 has two positive roots (in the P variable). One can take the value $P = -11 + 5\,n/2$, which is enclosed between the two roots of e_0, in order to ensure that $e_0 < 0$. Now, the curve $d_2 = 0$ is a decreasing function of n, bounded by (say) 600. Taking $\rho_\mathrm{f} = 600$ thus ensures that $d_2 < 0$ (for $9 \leq n \leq 230$, for greater values of n, one must take greater values of ρ_f). The positive root of $d_0\,\rho_\mathrm{b}^2 + d_1\,\rho_\mathrm{b} + d_2 = 0$ provides a value of ρ_b which cancels $c_{2,0}$.

In summary, for the model (1.11) page 21 has a Hopf bifurcation, one can take for any $n \geq 9$, the parameters values $\delta_{\mathrm{M}} = 10,\ B = 1/10,\ P = (-11 + 5\,n/2)^{1/n},\ \rho_{\mathrm{f}} = 600$ (for $n \leq 230$) and

$$\rho_{\mathrm{b}} = \frac{-75\,n^2 - 17670\,n + \sqrt{625\,n^4 + 2690500\,n^3 + 588647300\,n^2 - 2483789440\,n}}{40\,(-22 + 5\,n)\,n^2}.$$

These values ensure that $c_{0,0} > 0, c_{1,0} > 0, c_{2,0} = 0$ and $c_{2,1} < 0$ i.e. that a Hopf bifurcation occurs. Figure 1.6 corresponds to an oscillation in the neighborhood of a Hopf bifurcation for $n = 15$. This concludes the proof of the proposition 1.5.1 page 22.

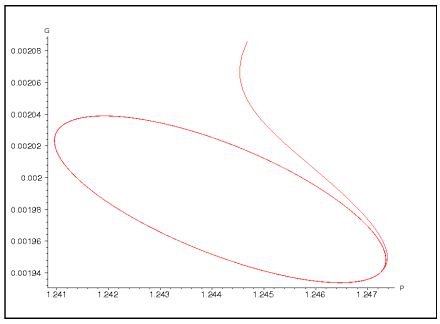


Figure 1.6: An oscillation between state variables $P(t)$ (horizontally) and $G(t)$ (vertically) obtained by simulating the system of ODEs (1.11). Parameter values for $n = 15$ are $\delta_{\mathrm{M}} = 10,\ \rho_{\mathrm{f}} = 600,\ \delta_{\mathrm{P}} \simeq 0.666 \times 10^{-5},\ \rho_{\mathrm{b}} \simeq 0.086,\ \gamma_0 \simeq 0.055,\ K_i = 1$ for all i in $\{1, \ldots, n-1\}$ and initial conditions are $P(0) \simeq 1.244,\ M(0) \simeq 1.244,\ G(0) \simeq 0.002$. ⌟

Way Back to the Raw Reduced and Basic Model

Because the simplified model (1.11) page 21 can have a Hopf bifurcation for n greater than or equal to 9, one can conclude also that the raw reduced model (1.4) page 14 can have a Hopf bifurcation for equivalent numerical values. This is possible thanks to the structure of the reduction and the reparametrization methods. Indeed, it is sufficient to apply the inverse of the change of coordinates (1.10) page 20 on the above state variables and parameters values to get these that cause a Hopf bifurcation to the raw reduced model (1.4) page 14. For example, when $n = 15$, parameter values $\delta_{\mathrm{M}} = 10,\ \theta = 1,\ \rho_{\mathrm{f}} \simeq 69751.35,\ \rho_{\mathrm{b}} \simeq 116.25,\ \gamma_0 \simeq 705.4,\ \alpha = 1,\ K_i = 1$ for all i in $\{1..n-1\},\ \beta = 1,\ \delta_{\mathrm{P}} = 150155.7$ and initial conditions $P(0) \simeq 1.25,\ G(0) \simeq 25.55,\ M(0) \simeq 186820.68$ lead to a limit cycle (oscillation).

Because the raw reduced model is an approximation of the basic model and the behavior of the system do not change (under fast polymerization assumption). The above numerical values and $k_i^+ = k_i^- = 10^{12}$ for all i in $\{1, \ldots, n-1\}$ cause a Hopf bifurcation for the basic model and oscillations are observed.

1.6 Contributions Summary

The aim of this section is to discuss, in the view of the motivating example, this contribution of this document to the study of medium size models composed of parametric differential equations and some related topics.

The table 1.2 summarizes the simplification of the medium size example treated by the help of `MABSys` in this chapter (see § 1.2.3 page 10). The basic model of the considered

	Number of differential equations	Number of parameters of the differential system	Number of parameters of steady points
Basic Model	$n+3$	$2\,n+5$	$2\,n+5$
Raw Reduced Model	3	$n+8$	8
After reduction	3	$n+6$	6
After reparametrization	3	$n+6$	1

Table 1.2: Recapitulation of the simplification done to the model of the considered family of networks (see figure 1.3 page 10).

genetic networks involves $n+3$ nonlinear ODEs depending on $2\,n+5$ parameters. The quasi-steady state approximation leads to a raw reduced model composed of 3 ODEs with $n+8$ parameters and associated steady points depend on 8 parameters. After application of exact simplification methods, one obtains a model that involves 3 ODEs with $n+6$ parameters and associated steady points depend on only 1 parameter. This concrete example shows that the approximative and exact methods detailed in this document simplify noticeably the system of ODEs of interest and thus their symbolic qualitative analysis.

In `MABSys` several computer algebra techniques with new and known algorithms are combined successfully. Quasi-steady state approximation, reduction and reparametrization are the key methods that permit the model simplification. The example of this chapter shows that many steps of the model simplification process are automated.

In the treatment of medium size models, there is an obvious progress since the article [12]. In [15] manual computations are replaced, as much as possible by the functions of `MABSys`. These functions demand to the user no knowledge about the Lie symmetry theory. Thanks to `MABSys`, most of the computations are systematic, human free and more reliable. As a result, the user can try many different simplifications relatively easily.

`MABSys` and `ExpandedLiePointSymmetry` are two packages written in `Maple` version 12. A computer with AMD Athlon(tm) Dual Core Processor 5400B, 3.4 GiB memory and Ubuntu 8.04 Hardy Heron as operating system is used for the computations. The execution of all the commands of the sections 1.3 page 11, 1.4 page 16 and 1.5 page 22 with the numerical simulations takes less than one minute. That means that the presented algorithms and the associated functions are practical to use on medium size models.

There exist software packages such as AUTO (vers. 0.6 of Jan. 13, 2009, see [32]) or XPPAUT (see [35]) which locate Hopf bifurcations by means of numerical calculations. They give evidence about the existence of Hopf bifurcations but do not prove their absence. Theoretically, the existence or the absence of Hopf bifurcations can be decided algebraically (see [50, 63, 43, 115, 84]). There are computer algebra libraries that tackle this kind of problems and much more. For instance, GNA (Genetic Network Analyzer) is the implementation of a method for the qualitative modeling and simulation of genetic regulatory networks (see [61, 93, 3]). The input of GNA consists of a model of the regulatory network in the form of a system of piecewise-linear differential equations, supplemented by inequality constraints on the parameters and initial conditions. From this information GNA generates a state transition graph summarizing the qualitative dynamics of the system. RAGlib (see [97]) is a library for real solving polynomial systems of equations and inequations based on critical points algorithms. For systems of reasonable size (up to 7,8 coordinates) it can work quite well. Unfortunately the complexity of these algorithms is exponential on the number of coordinates (in the worst case) which makes difficult their application in more general cases. That is also why exact simplification methods are very useful. Using RAGlib, one can show some of the above results for different values of n. Regular quantifier elimination of QEPCAD (see [19]) or REDLOG (see [33]) with their default settings (versions of August 08) was unable to decide the positivity of the polynomial $\tilde{c}_{2,0}$ for different values of n (see § 1.5.3 page 24). However, the new concept of positive quantifier elimination, designed thanks to the conclusions of the article [12], gives useful results. In [106] authors show, for a given n, if a Hopf bifurcation may arise or not for the single gene networks of figure 1.3 page 10 modeled as in the article [12]. Remark that these methods do not let have a generic conclusion as in the previous section. One must perform similar computations for every given n. With these libraries, even if one can decide the existence of a Hopf bifurcation for a particular value of n, one can not proof it for any n as done in this document.

I also contributed to the development of the software `ExpandedLiePointSymmetry` which has a more mathematical standpoint. It treats the Lie symmetry theory for algebraic systems, systems of ODEs and systems of ordinary recurrence equations (OREs). Some additional kinds of symmetries, more complicated than scalings and also the moving frame based reduction process which permits to avoid high complexity computations are available. The usage of this software does not appear in this chapter. However, `MABSys` uses some functionalities of the package `ExpandedLiePointSymmetry` for the computation of scalings. The conception of these softwares requires the acquisition of an important mathematical background detailed in the part III page 57.

However, despite all these improvements, there is still need to many progress in model simplification domain. On the one hand, some non-algorithmic computations, for instance manual addition of some changes of coordinates (see equation (1.3) page 14 for parameters K_i) can be avoided. This problem requires to find more scalings that take into account such cases. The extension of the set of symmetries to other symmetries than scalings can also help the automation of these computations. On the other hand, many progresses require long-term research. For example, argument choices for these simplifications are for now in the control of the user. Deciding which parameters to

eliminate, how to reorganize the coordinates, etc. is a difficult task because of the huge number of coordinates. In the future maybe dynamical programming with a cost function that indicates the level of the simplification can be used. This could help to extend these methods to larger examples.

"Nature is a thing about which one can learn a great deal."

Letter to Theo van Gogh
Nuenen, early July 1884
Vincent van Gogh

II Exact Model Simplification in `MABSys`

2

Introduction to Reduction and Reparametrization

The reduction and the reparametrization methods implemented in `MABSys` aim to simplify the parameter set of a system of ODEs or the associated algebraic system that defines its steady points. In `MABSys`, both exact simplification methods are based on the same three main instructions that can be summarized as below:

1. *find the scalings associated to the system of ODEs or to the algebraic system that defines its steady points;*
2. *deduce a new coordinate set from these scalings;*
3. *rewrite the system of ODEs in these new coordinates in order to obtain the simplified system of ODEs.*

In this chapter, we assume that the scalings are given. The user splits the set of coordinates in two parts: the ones which are strictly positive and the ones which may be also zero or negative. In the sequel, we give an answer to the following question: *how do the reduction and the reparametrization algorithms of* `MABSys` *use these data to perform their tasks?*

The first section gives some mathematical preliminaries. We survey scaling type Lie symmetries and now new coordinates are deduced from them. These notions are essential in the forthcoming explanations. In the second section we summarize the reduction and the reparametrization algorithms as they were conceived in `MABSys`. The third section is devoted to the algorithm which construct new coordinates from some given scalings. Finally we compare these methods with existing ones..

The following example illustrates the inputs and the outputs of these exact model simplification methods.

Example 2.0.1. Let us consider the following system of ODEs defined on the coordinate set $Z = (t, x, y, k_1, k_2, a, b)$:

$$\begin{cases} \frac{\mathrm{d}x}{\mathrm{d}t} = a - k_1\, x + k_2\, x^2\, y, \\ \frac{\mathrm{d}y}{\mathrm{d}t} = b - k_2\, x^2\, y. \end{cases} \tag{2.1}$$

This is a two-species oscillator (see § 7.4 of [83] and [99]). The evolution of the state variables x and y w.r.t. time t depend on 4 parameters, namely a, b, k_1 and k_2. These

parameters are all supposed to be strictly positive because they correspond to some biological quantities. The steady points of (2.1) page 33 are defined by the algebraic system

$$\begin{cases} a - k_1\, x + k_2\, x^2\, y = 0, \\ b - k_2\, x^2\, y = 0 \end{cases} \tag{2.2}$$

thus they depend also on these 4 parameters. The reduction and the reparametrization algorithms of this chapter transform the system of ODEs (2.1) page 33 into the following equivalent system of ODEs with less parameters:

$$\begin{cases} \frac{\mathrm{d}\widetilde{x}}{\mathrm{d}\widetilde{t}} = 1 - \widetilde{x} + \widetilde{x}^2\, \widetilde{y}, \\ \frac{\mathrm{d}\widetilde{y}}{\mathrm{d}\widetilde{t}} = \left(\widetilde{b} - \widetilde{x}^2\, \widetilde{y}\right) \widetilde{k}_2 \end{cases} \tag{2.3}$$

written in a new coordinate set $\widetilde{Z} = \left(\widetilde{t}, \widetilde{x}, \widetilde{y}, \widetilde{k}_2, \widetilde{b}\right)$. Remark that the parameters a and k_1 are eliminated from the system (2.1) page 33. The steady points of this new system are defined by the algebraic system

$$\begin{cases} 1 - \widetilde{x} + \widetilde{x}^2\, \widetilde{y} = 0, \\ \widetilde{b} - \widetilde{x}^2\, \widetilde{y} = 0 \end{cases} \tag{2.4}$$

that involves just 1 parameter $\widetilde{b}$ ($\widetilde{k}_2$ is strictly positive). These algorithms provide also the exact relations between the original and the simplified systems of ODEs. The following relations define the new coordinates of $\widetilde{Z}$ in function of the old ones in Z:

$$\widetilde{t} = t\, k_1, \quad \widetilde{x} = \frac{x\, k_1}{a}, \quad \widetilde{y} = \frac{y\, a\, k_2}{k_1^2}, \quad \widetilde{k}_2 = \frac{k_2\, a^2}{k_1^3}, \quad \widetilde{b} = \frac{b}{a}\cdot \tag{2.5}$$

⌟

2.1 Mathematical Preliminary

In this section, mathematical notions that are important to understand the reduction and the reparametrization algorithms are presented. We are interested in *expanded Lie point symmetries* (see [104] and ch. 4 page 87) and more specifically in *scalings*. First, we explain such symmetries and show their representation. Second, we give the idea of how to deduce a change of coordinates (thus new coordinates) from some given scalings in order to perform the exact simplification methods.

2.1.1 Lie Symmetries; Special Case of Scaling

The aim of this section is to survey the notion of Lie symmetries on which the reduction and the reparametrization algorithms rely. We focus on the special case of scalings.

Definition of Lie Symmetries

For a system of ODEs or an algebraic system, a symmetry expresses an invariance w.r.t. a transformation of the variables (see § 1.2 of [86]). The formal definition of Lie symmetries requires a detailed mathematical background that is available in the part III page 57.

In this document, Lie symmetries are studied by an infinitesimal approach (see [86, 104, 101]). They can be represented by differential operators that act on the coordinates $Z = (z_1, \ldots, z_n)$ of the system as follows:

$$\delta_{\mathcal{S}} = \sum_{i=1}^{n} \xi_{z_i}(Z)\, \frac{\partial}{\partial z_i} \tag{2.6}$$

where the coefficients ξ_{z_i} are expressions in Z with real coefficients i.e. are in $\mathbb{R}(Z)$. Observe that the coordinate set involves the time variable, the state variables and the parameters. Such differential operators are called *infinitesimal generators* (see § 3.2.1 page 64).

Forthcoming examples illustrate Lie symmetries, associated transformation groups and their influence on the system of interest.

Example 2.1.1. Let us consider the system of ODEs (2.1) page 33 defined on the coordinate set $Z = (t, x, y, k_1, k_2, a, b)$. These equations are invariant under the one-parameter ν_1 invertible transformation group (the group law being the composition of two transformations):

$$\begin{cases} t \to t, & x \to x\,\nu_1, & y \to y\,\nu_1, \\ k_1 \to k_1, & k_2 \to \dfrac{k_2}{\nu_1^2}, & a \to a\,\nu_1, \\ b \to b\,\nu_1. \end{cases} \tag{2.7}$$

If one applies (2.7) over any equation of (2.1) page 33, one gets the same equation, possibly up to some nonzero multiplicative factor. The transformation (2.7) is said to be a *Lie symmetry* (see ch. 4 page 87) of (2.1) page 33. It can be represented by the differential operator

$$\delta_{\mathcal{S}_1} = x\frac{\partial}{\partial x} + y\frac{\partial}{\partial y} - 2\,k_2\frac{\partial}{\partial k_2} + b\frac{\partial}{\partial b} + a\frac{\partial}{\partial a}\cdot \tag{2.8}$$

The transformation (2.7) transforms the solutions of (2.1) page 33 into other solutions of (2.1) page 33. This means that the application of the transformation does not change the expressions verified by the state variables x and y. ⌟

Example 2.1.2. The system of ODEs (2.1) page 33 possesses also a second symmetry with one-parameter ν_2 transformation group defined by:

$$\begin{cases} t \to \dfrac{t}{\nu_2^2}, & x \to x\,\nu_2, & y \to y\,\nu_2, \\ k_1 \to k_1\,\nu_2^2, & k_2 \to k_2, & a \to a\,\nu_2^3, \\ b \to b\,\nu_2^3. \end{cases} \tag{2.9}$$

This symmetry can be represented by the differential operator

$$\delta_{\mathcal{S}_2} = -2\,t\frac{\partial}{\partial t} + x\frac{\partial}{\partial x} + y\frac{\partial}{\partial y} + 3\,a\frac{\partial}{\partial a} + 3\,b\frac{\partial}{\partial b} + 2\,k_1\frac{\partial}{\partial k_1}\cdot \tag{2.10}$$

⌟

Example 2.1.3. The algebraic system defined on $Z = (x, y, k_2, b)$

$$\begin{cases} 1 - x + k_2\,x^2\,y = 0, \\ b - k_2\,x^2\,y = 0 \end{cases} \tag{2.11}$$

is invariant under the following one-parameter ν_3 transformation group that is given with the associated differential operator:

$$\begin{cases} x \to x, & y \to y\,\nu_3, \\ k_2 \to \dfrac{k_2}{\nu_3}, & b \to b \end{cases} \quad \text{and} \quad \delta_{\mathcal{S}_3} = y\frac{\partial}{\partial y} - k_2\frac{\partial}{\partial k_2}\cdot \tag{2.12}$$

That means that $\delta_{\mathcal{S}_3}$ is a differential operator that represents a symmetry of (2.11). ⌟

Every linear combination of symmetries, with real coefficients, is also a symmetry. The set of symmetries thus forms a vector space over $\mathbb{R}$ (and even a *Lie algebra*, see § 1.4 of [86] and § 3.3 page 68). One can associate a *matrix representation* (see § 12.4 of [112]) to a set of symmetries as follows.

Definition 2.1.4 (Matrix Representation)**.** Let $\mathcal{B} = \{\delta_1, \ldots, \delta_r\}$ be a set of infinitesimal generator expressed on the canonical basis $\{\partial/\partial z_1, \ldots, \partial/\partial z_n\}$. For all i in $\{1, \ldots, r\}$ one has $\delta_i = \sum_{j=1}^{n} \xi^i_{z_j}\, z_j \partial/\partial z_j$ with $\xi^i_{z_j}$ in $\mathbb{R}(Z)$. The *matrix representation* associated to $\mathcal{B}$ is the following matrix with n columns and r rows:

$$M := \left(\xi^i_{z_j}\right)_{1\le i\le r,\, 1\le j\le n}\cdot \tag{2.13}$$

⌟

Scalings

The `MABSys` software is restricted to particular type of Lie symmetries, namely *scalings*.

Definition 2.1.5 (Scaling)**.** Let Σ be a system of ODEs or an algebraic system defined on the coordinate set $Z = (z_1, \ldots, z_n)$. A *scaling* of Σ is a Lie symmetry that can be represented by differential operators acting on the coordinates of the system as follows:

$$\delta_{\mathcal{S}} := \sum_{i=1}^{n} \alpha_i\, z_i \frac{\partial}{\partial z_i} \tag{2.14}$$

where α_i are in $\mathbb{R}$. ⌟

Example 2.1.6. The differential operator $\delta_{\mathcal{S}_1}$ given in the equation (2.8) page 35 of the example 2.1.1 page 35 is a scaling of the system (2.1) page 33 defined on the coordinate set $Z = (t, x, y, k_1, k_2, a, b)$. ⌟

Semi-rectified symmetries (see § 2.2 of [104] for *normal forms* of infinitesimal generators) are particular cases of scalings such that all coefficients α_i are zero except one.

Definition 2.1.7 (Semi-Rectified Symmetry)**.** A *semi-rectified symmetry* is a Lie symmetry that can be represented by a differential operator that acts on just one coordinate z as follows:

$$\delta_{\mathcal{S}} := \alpha\, z \frac{\partial}{\partial z} \tag{2.15}$$

with α in $\mathbb{R}$. ⌟

In the forthcoming reduction and reparametrization algorithms, the scalings of the system of interest are handled all together thanks to the associated *matrix of scaling*. This matrix is a special case of the matrix representation (see definition 2.1.4 page 36).

Definition 2.1.8 (Matrix of Scaling)**.** Let $\mathcal{B} = \{\delta_1, \ldots, \delta_r\}$ be a set of scalings expressed on the canonical basis $\{z_1\partial/\partial z_1, \ldots, z_n\partial/\partial z_n\}$. For all i in $\{1, \ldots, r\}$, one has $\delta_i = \sum_{j=1}^{n} \alpha_j^i\, z_j\partial/\partial z_j$ with α_j^i in $\mathbb{R}$. The *matrix of scaling* associated to $\mathcal{B}$ is defined thanks to the coefficients α_j^i:

$$M := \left(\alpha_j^i\right)_{1\leq i\leq r,\, 1\leq j\leq n}. \tag{2.16}$$

⌟

Example 2.1.9. The differential operators $\delta_{\mathcal{S}_1}$ given in the equation (2.8) page 35 and $\delta_{\mathcal{S}_2}$ given in the equation (2.10) page 36 are two scalings defined on the coordinate set $Z = (t, x, y, k_1, k_2, a, b)$. The matrix of scaling associated to the set $\mathcal{B} = \{\delta_{\mathcal{S}_1}, \delta_{\mathcal{S}_2}\}$ follows:

$$M = \begin{pmatrix} 0 & 1 & 1 & 0 & -2 & 1 & 1 \\ -2 & 1 & 1 & 2 & 0 & 3 & 3 \end{pmatrix}. \tag{2.17}$$

More precisely, this matrix represents the *Lie algebra* generated by $\mathcal{B}$. ⌟

In the sequel, the matrix of scaling is used to deduce a new coordinate set in which the system can be rewritten in a simpler form (see § 2.1.2).

2.1.2 Change of Coordinates associated to Scalings

In this section we present a way of constructing a new coordinate set in a particular form just for scalings, not for other type Lie symmetries. This new coordinate set is used by the reduction and the reparametrization methods. It defines an invertible change of coordinates which transforms the system of interest to its reduced or reparametrized form. Moreover, the differential operators of interest are transformed into semi-rectified symmetries when they are rewritten in these new coordinates.

Remark 2.1.10. The new coordinates presented in this section are based on the notion of *invariants* and *semi-invariants* defined in § 2.3.3 page 54. ⌟

Change of Coordinates Associated to One Scaling

In this paragraph, we are looking for a change of coordinates deduced from one scaling. One can write a scaling $\delta_{\mathcal{S}}$ as given in (2.14) page 36 i.e.:

$$\delta_{\mathcal{S}} = \alpha_1\, z_1 \frac{\partial}{\partial z_1} + \alpha_2\, z_2 \frac{\partial}{\partial z_2} + \cdots + \alpha_{n-1}\, z_{n-1} \frac{\partial}{\partial z_{n-1}} + \alpha_n\, z_n \frac{\partial}{\partial z_n} \tag{2.18}$$

but with the coefficients α_i in $\mathbb{Z}$.

Remark 2.1.11. In practice, if the coefficients α_i are rational fractions then one can multiply the whole scaling by the least common multiple (lcm) of their denominators. This operation (multiplication by a constant in $\mathbb{R}$) does not modify its algebraic structure (see remark 2.3.6 page 54). ⌟

Suppose that the first coordinate z_1 is the parameter w.r.t. which the simplifications need to be performed. Thus α_1 is assumed not to vanish. We want to avoid, in the change of coordinates, powers with rational fractions for coordinates that can be negative. While using $\delta_{\mathcal{S}}$, we suppose that z_1 is strictly positive and only this parameter is allowed to have a power with rational fraction.

If α_1 is not equal to 1 then because of forthcoming new coordinates form, one can get expressions with rational powers. Thus introducing a coordinate $\widetilde{z}_1 = z_1^{1/\alpha_1}$ helps to avoid non integer power problems for other coordinates. Remark that $\widetilde{z}_1$ verifies $\delta_{\mathcal{S}}\widetilde{z}_1 = \widetilde{z}_1$. As a consequence, the infinitesimal generator (2.18) can be rewritten in the form:

$$\delta_{\mathcal{S}} = \widetilde{z}_1 \frac{\partial}{\partial \widetilde{z}_1} + \alpha_2\, z_2 \frac{\partial}{\partial z_2} + \cdots + \alpha_{n-1}\, z_{n-1} \frac{\partial}{\partial z_{n-1}} + \alpha_n\, z_n \frac{\partial}{\partial z_n}\cdot \tag{2.19}$$

In this special case, one can choose $n-1$ supplementary new coordinates as follow:

$$\widetilde{z}_1 = z_1^{1/\alpha_1}, \quad \widetilde{z}_i = \frac{z_i}{\widetilde{z}_1^{\alpha_i}} \quad \forall i \in \{2, \ldots, n\} \tag{2.20}$$

where $\delta_{\mathcal{S}}\widetilde{z}_i = 0$ for all i in $\{2, \ldots, n\}$ by construction.

In these conditions, the inverse of the change of coordinates always exists and one does not see non integer powers of variables appear, except for z_1. Only z_1 must be taken on $\mathbb{R}_+^\star$, remaining coordinates belong to $\mathbb{R}$.

Let us recall that, in these new coordinates, the differential operator $\delta_{\mathcal{S}}$ is written as a semi-rectified symmetry equal to $\widetilde{z}_1 \partial/\partial \widetilde{z}_1$.

Example 2.1.12. Let us find these particular new coordinates for the differential operator $\delta_{\mathcal{S}_1}$ given in (2.8) page 35 defined on $Z = (t, x, y, k_1, k_2, a, b)$. Let us assume that the parameter a is strictly positive. The new coordinates $\widetilde{Z} = \left(\widetilde{t}, \widetilde{x}, \widetilde{y}, \widetilde{k}_1, \widetilde{k}_2, \widetilde{b}, \widetilde{a}\right)$ are given by:

$$\widetilde{t} = t, \quad \widetilde{x} = \frac{x}{a}, \quad \widetilde{y} = \frac{y}{a}, \quad \widetilde{k}_1 = k_1, \quad \widetilde{k}_2 = k_2\, a^2, \quad \widetilde{b} = \frac{b}{a}, \quad \widetilde{a} = a. \tag{2.21}$$

They verify $\delta_{\mathcal{S}_1}\widetilde{t} = \delta_{\mathcal{S}_1}\widetilde{x} = \delta_{\mathcal{S}_1}\widetilde{y} = \delta_{\mathcal{S}_1}\widetilde{k}_1 = \delta_{\mathcal{S}_1}\widetilde{k}_2 = \delta_{\mathcal{S}_1}\widetilde{b} = 0$ and $\delta_{\mathcal{S}_1}\widetilde{a} = \widetilde{a}$ by definition. The differential operator $\delta_{\mathcal{S}_1}$ in these coordinates is equal to:

$$\delta_{\mathcal{S}_1} = \widetilde{a}\frac{\partial}{\partial \widetilde{a}} \tag{2.22}$$

meaning that it becomes a semi-rectified symmetry. Since $\delta_{\mathcal{S}_1}$ is a symmetry of (2.1) page 33, rewriting this system in the new coordinate set (2.21) page 38 eliminates the parameter a (see example 2.2.2 page 41 for the reduction of the parameters a and k_1 at the same time). ⌟

Change of Coordinates Associated to Many Scalings

In this paragraph we present, through an example, how to find such new coordinates if many scalings are present. The formal version of this algorithm can be found in § 2.3 page 49.

Example 2.1.13. Let us consider the system of ODEs:

$$\begin{cases} \frac{\mathrm{d}x}{\mathrm{d}t} = b\,y - a, \\ \frac{\mathrm{d}y}{\mathrm{d}t} = a\,x + b \end{cases} \tag{2.23}$$

defined on $Z = (t, x, y, a, b)$. The goal of this example is to find the new coordinate set $\widetilde{Z} = \left(\widetilde{t}, \widetilde{x}, \widetilde{y}, \widetilde{a}, \widetilde{b}\right)$ associated to the following two scalings:

$$\delta_1 = x\frac{\partial}{\partial x} - y\frac{\partial}{\partial y} - a\frac{\partial}{\partial a}, \quad \delta_2 = a\frac{\partial}{\partial a} + b\frac{\partial}{\partial b} \tag{2.24}$$

in order to reparametrize the system (2.23). Let us suppose that a and b are strictly positive parameters. The coordinate set Z must be reorganized by putting these parameters at the beginning of the list i.e. $Z = (a, b, t, x, y)$. The matrix of scaling

$$M = \begin{pmatrix} -1 & 0 & 0 & 1 & -1 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix} \tag{2.25}$$

written w.r.t. this new order, helps to handle the two scalings (2.24) at the same time. The unique reduced row echelon form

$$R = \begin{pmatrix} 1 & 0 & 0 & -1 & 1 \\ 0 & 1 & 0 & 1 & -1 \end{pmatrix} \tag{2.26}$$

of the matrix M represents the same vector space of scalings as M.

The change of coordinates (thus the new coordinate set) that we are looking for transforms the scalings represented by R into semi-rectified symmetries. Because the parameters a and b are strictly positive, we are looking for the semi-rectified symmetries of the form:

$$\delta_1 = \widetilde{a}\frac{\partial}{\partial \widetilde{a}}, \quad \delta_2 = \widetilde{b}\frac{\partial}{\partial \widetilde{b}}. \tag{2.27}$$

These scalings are represented by the following matrix of scaling:

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} \tag{2.28}$$

written w.r.t. the new coordinates $\widetilde{Z} = \left(\widetilde{a}, \widetilde{b}, \widetilde{t}, \widetilde{x}, \widetilde{y}\right)$.

Recall that a right multiplication on a matrix of scaling performs a change of coordinates (see § 2.3.1 page 50). Thus looking for a new coordinate set is equivalent to searching an invertible matrix C such that:

$$R\,C = D. \tag{2.29}$$

This implies $R = D\,C^{-1}$. Because of the special form of the matrix D (1s at the diagonal and 0s elsewhere), one can deduce the matrix C^{-1} easily:

$$C^{-1} = \begin{pmatrix} 1 & 0 & 0 & -1 & 1 \\ 0 & 1 & 0 & 1 & -1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{2.30}$$

Its inverse

$$C = \begin{pmatrix} 1 & 0 & 0 & 1 & -1 \\ 0 & 1 & 0 & -1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.31}$$

encodes the new coordinates that we are looking for. Its elements indicate the powers of the old coordinates in the new coordinates expressions. Every column of the matrix defines one new coordinate. For example, consider the 4^{th} column from which one can deduce the expression of the new coordinate $\widetilde{x}$:

$$\begin{aligned} \widetilde{x} &= a^{C_{1,4}}\, b^{C_{2,4}}\, t^{C_{3,4}}\, x^{C_{4,4}}\, y^{C_{5,4}} \\ &= a^1\, b^{-1}\, t^0\, x^1\, y^0 \\ &= \frac{a\,x}{b}\cdot \end{aligned} \tag{2.32}$$

The complete change of coordinates follows:

$$\widetilde{a} = a, \quad \widetilde{b} = b, \quad \widetilde{t} = t, \quad \widetilde{x} = \frac{a\,x}{b}, \quad \widetilde{y} = \frac{b\,y}{a}\cdot \tag{2.33}$$

Rewriting the system (2.23) page 39 in this new coordinate set $\widetilde{Z}$ leads to the following system of ODEs:

$$\begin{cases} \frac{\mathrm{d}\widetilde{x}}{\mathrm{d}\widetilde{t}} = \frac{\widetilde{a}^2(\widetilde{y}-1)}{\widetilde{b}}, \\ \frac{\mathrm{d}\widetilde{y}}{\mathrm{d}\widetilde{t}} = \frac{\widetilde{b}^2(\widetilde{x}+1)}{\widetilde{a}}\cdot \end{cases} \tag{2.34}$$

Remark that both parameters $\widetilde{a}$ and $\widetilde{b}$, strictly positive, are in factor in the right-hand side of the differential equations and do not appear in the algebraic system that defines associated steady points i.e. in:

$$\begin{cases} \widetilde{y} - 1 = 0, \\ \widetilde{x} + 1 = 0. \end{cases} \tag{2.35}$$

These expressions do not depend on parameters. ⌟

Remark 2.1.14. This process of getting a change of coordinates from a matrix of scaling is used for both exact simplification methods (the reduction and the reparametrization) presented in the following section. ⌟

2.2 Exact Simplification Method Algorithms

This section is devoted to two closely related exact model simplification methods for systems of ODEs, *reduction* and *reparametrization*. In the sequel, one can find details and illustrations of associated algorithms. The generalizations of some of these simplification methods along with their advantages and disadvantages are discussed in the part III page 57 of this document.

Remark 2.2.1. In `MABSys`, only scaling type Lie symmetries are considered. Their structure permits to easily get the exact relations associated to model simplifications and to respect the positivity property of biological quantities when the system models a biological phenomenon. Moreover, biological models created by using the mass-action law possess very frequently scaling type Lie symmetries. More complicated symmetries are also useful for the model simplifications and can help to eliminate more coordinates but finding the associated exact relations require the resolution of systems of PDEs (or to use moving frame based reduction process as in the ch. 5 page 103). However, such symmetries do not guarantee systematically the positivity assumption of the system coordinates. ⌟

2.2.1 Reduction of Parameters

Here, we give the idea of the reduction of parameters from a system of ODEs through an example. Then we detail the reduction algorithm implemented in `MABSys`.

Idea of Reduction

The reduction algorithm of this section aims to eliminate some parameters from a system of ODEs by rewriting it in a new coordinate set. This new coordinate set is deduced from the scalings of the same system. There exists then a bijection between positive solutions of the system of ODEs and these of the reduced one.

Example 2.2.2. Let us illustrate the reduction process on the two-species oscillator that models the dynamics of a system of biochemical reactions:

$$\begin{cases} \frac{\mathrm{d}x}{\mathrm{d}t} = a - k_1\, x + k_2\, x^2\, y, \\ \frac{\mathrm{d}y}{\mathrm{d}t} = b - k_2\, x^2\, y. \end{cases} \tag{2.36}$$

This system has two state variables x, y and depends on 4 parameters: a, b, k_1 and k_2. The idea is to look for the system properties that allow the system to be rewritten with less parameters. For this issue, one can search for the scalings of (2.36) page 41. The scalings (according to the algorithms of ch. 4 page 87) of the system are $\delta_{\mathcal{S}_1}$ and $\delta_{\mathcal{S}_2}$ (given also in the equations (2.8) page 35 and (2.10) page 36):

$$\begin{aligned}\delta_{\mathcal{S}_1} &= x\frac{\partial}{\partial x} + y\frac{\partial}{\partial y} - 2\,k_2\frac{\partial}{\partial k_2} + b\frac{\partial}{\partial b} + a\frac{\partial}{\partial a},\\ \delta_{\mathcal{S}_2} &= -2\,t\frac{\partial}{\partial t} + x\frac{\partial}{\partial x} + y\frac{\partial}{\partial y} + 3\,a\frac{\partial}{\partial a} + 3\,b\frac{\partial}{\partial b} + 2\,k_1\frac{\partial}{\partial k_1}\cdot\end{aligned} \tag{2.37}$$

Assuming that the system parameters are strictly positive, these scalings permit to rewrite the system in a new coordinate set where the system depend on only 2 parameters instead of 4. The necessary change of coordinates can be found by applying the idea given in § 2.1.2 page 37 (see example 2.3.3 page 52 for detailed computations). The new coordinate set $\widehat{Z} = \left(\widehat{t}, \widehat{x}, \widehat{y}, \widehat{k}_1, \widehat{k}_2, \widehat{a}, \widehat{b}\right)$ is defined by (the parameters a and k_1 are strictly positive):

$$\widehat{t} = t\,k_1, \quad \widehat{x} = \frac{x\,k_1}{a}, \quad \widehat{y} = \frac{y\,k_1}{a}, \quad \widehat{k}_1 = k_1, \quad \widehat{k}_2 = \frac{k_2\,a^2}{k_1^3}, \quad \widehat{a} = a, \quad \widehat{b} = \frac{b}{a}\cdot \tag{2.38}$$

It is sufficient to inverse this change of coordinates and apply it to the model (2.36) page 41 to reduce its parameters. The inverse of (2.38) is given by:

$$t = \frac{\widehat{t}}{k_1}, \quad x = \frac{\widehat{x}\,a}{k_1}, \quad y = \frac{\widehat{y}\,a}{k_1}, \quad k_1 = \widehat{k}_1, \quad k_2 = \frac{\widehat{k}_2\,k_1^3}{a^2}, \quad a = \widehat{a}, \quad b = \widehat{b}\,a. \tag{2.39}$$

The application of (2.39) gives:

$$\begin{cases} \dfrac{\mathrm{d}\left(\frac{\widehat{x}\,a}{k_1}\right)}{\mathrm{d}\left(\frac{\widehat{t}}{k_1}\right)} = \widehat{a} - \widehat{k}_1\,\dfrac{\widehat{x}\,a}{k_1} + \dfrac{\widehat{k}_2\,k_1^3}{a^2}\,\dfrac{\widehat{x}^2\,a^2}{k_1^2}\,\dfrac{\widehat{y}\,a}{k_1}, \\ \dfrac{\mathrm{d}\left(\frac{\widehat{y}\,a}{k_1}\right)}{\mathrm{d}\left(\frac{\widehat{t}}{k_1}\right)} = \widehat{b}\,a - \dfrac{\widehat{k}_2\,k_1^3}{a^2}\,\dfrac{\widehat{x}^2\,a^2}{k_1^2}\,\dfrac{\widehat{y}\,a}{k_1}\cdot \end{cases} \tag{2.40}$$

Thus, in these new coordinates, the system (2.36) page 41 can be rewritten as:

$$\begin{cases} \dfrac{\mathrm{d}\widehat{x}}{\mathrm{d}\widehat{t}} = 1 - \widehat{x} + \widehat{k}_2\,\widehat{x}^2\,\widehat{y}, \\ \dfrac{\mathrm{d}\widehat{y}}{\mathrm{d}\widehat{t}} = \widehat{b} - \widehat{k}_2\,\widehat{x}^2\,\widehat{y}. \end{cases} \tag{2.41}$$

Remark that the parameters a and k_1 are eliminated from the system. However, this new system (2.41) is *equivalent* to the system (2.36) page 41 thanks to the exact relations (2.38). There is a bijection between positive solutions of these two systems. ⌟

This method can be generalized to more complicated symmetries, to the reduction of state variables and also to systems of ordinary recurrence equations i.e. to discrete dynamical systems (see § 5.4 page 114) by using moving frame based methods. These generalizations are implemented in `ExpandedLiePointSymmetry` package (see ch. 5 page 103).

Reduction Algorithm used in `MABSys`

The algorithm of the reduction procedure is summarized in the algorithm 1. The function `InvariantizeByScalings` of `MABSys` implements this algorithm. It aims to facilitate the usage of Lie symmetries for the model reduction without any particular knowledge of them.

Proposition 2.2.3. *The reduction of a system of ODEs detailed in the algorithm 1 has a polynomial complexity in the input size.* ⌟

The algorithm 1 follows the classical reduction process (see § 5.2 page 105). One also computes explicitly the new coordinates in function of the old ones. Thanks to the restriction of the method to scalings and some computational strategies (see § 4.2 page 93) its complexity is polynomial in the input size.

The algorithm 1 has 4 inputs. The first one is a parametric system of first-order ODEs. The second one is a list of parameters, assumed to be strictly positive, that one would like to eliminate by priority order. The third input is a list of coordinates to keep unchanged. One computes scalings that do not act on these coordinates. Thus,

Algorithm 1 Algorithm of the function `InvariantizeByScalings`.

Input: The parametric system of first-order ODEs Σ defined on $Z = (z_1, \ldots, z_n)$.
The list of parameters $E = (\theta_1, \ldots, \theta_e)$ that one would like to eliminate by priority order. Remark that for all i in $\{1, \ldots, e\}$, there exists j in $\{1, \ldots, n\}$ such that $\theta_i = z_j$. These parameters are assumed to be strictly positive.
The list of remaining coordinates $P = (p_1, \ldots, p_{n-e-1})$ except the time variable.
The optional list of coordinates $U = (u_1, \ldots, u_f)$ to keep unchanged. Remark that for all i in $\{1, \ldots, f\}$, there exists j in $\{1, \ldots, n\}$ such that $u_i = z_j$.
The optional boolean named "scaletime" that indicates if the time variable can be changed or not during the reduction. The default value is true.

Output: The reduced system of ODEs, denoted by $\widetilde{\Sigma}$, written in a new coordinate set $\widetilde{Z} = (\widetilde{z}_1, \ldots, \widetilde{z}_n)$.
The change of coordinates defining the new coordinate set $\widetilde{Z}$ thanks to the scalings of Σ.
The list of parameters that were eliminated from the system of ODEs Σ.

1: #*Computation of scalings*
2: $S :=$ ELPSymmetries(Σ, sym = scaling, fixedcoord = if(scaletime, U, $[t, u_1, \ldots, u_f]$));
3: #*Computation of change of coordinates.*
4: $Z := [\theta_1, \ldots, \theta_e, t, p_1, \ldots, p_{n-e-1}]$;
5: $M :=$ MatrixOfScaling (S, Z) ;
6: C, pivots := GetChangeOfCoordinates (M, E, P) ;
7: #*Computation of the reduced system*
8: $\widetilde{\Sigma} :=$ ApplyChangeOfCoordinates (Σ, C) ;
9: **return** $\left[\widetilde{\Sigma}, C, \text{pivots}\right]$;

in the final change of coordinates, these coordinates are not modified. The last input is an optional boolean that indicates if the time variable can be changed or not during the reduction. The default value is true. The output is composed of three objects: the reduced system of ODEs, the change of coordinates deduced from used scalings and the list of parameters that were eliminated.

The first step of the algorithm 1 page 43 is the computation of the scalings of the system of ODEs by respecting given conditions on the coordinates. One can use the function `ELPSymmetries` of the `ExpandedLiePointSymmetry` package for this issue. The second step aims to compute a change of coordinates associated to the computed scalings. In order to find a change of coordinates (thus a new coordinate set) which allows to eliminate required parameters by respecting the priority order, the coordinates to be eliminated are put at the beginning of Z. The matrix of scaling (see definition 2.1.8 page 37) is created to handle scalings all together. Then one deduces the change of coordinates and the list of parameters that are going to be eliminated by it. The detailed algorithm of the function `GetChangeOfCoordinates` is given in § 2.3 page 49. The third step of the algorithm consists in applying the found change of coordinates to the system of ODEs in order to get the reduced one.

Example 2.2.4. Let us reproduce the reduction of the parameters a and k_1 of the example 2.2.2 page 41 by following the algorithm 1 page 43. Here are the inputs and the outputs of the function `InvariantizeByScalings` of `MABSys`.

```
> # Model
> ODEs := [diff(x(t),t)=a-k1*x(t)+k2*x(t)^2*y(t), diff(y(t),t)=b-k2*x(t)^2*y(t)];


            d                                    2        d                        2
   ODEs := [-- x(t) = a - k1 x(t) + k2 x(t)  y(t), -- y(t) = b - k2 x(t)  y(t)]
            dt                                           dt

> # Reduction
> out := InvariantizeByScalings(ODEs,[a,b,k1,k2],[x,y]):
> ReducedODEs := out[1];
                d                                  2        d                        2
ReducedODEs := [-- x(t) = 1 - x(t) + k2 x(t)  y(t), -- y(t) = b - k2 x(t)  y(t)]
                dt                                           dt

> ChangeOfCoord1 := out[2];
                                                 2
                                            k2 a                    x k1       y k1
       ChangeOfCoord1 := [b = b/a, k2 = -----, t = t k1, x = ----, y = ----]
                                              3                      a          a
                                             k1

> EliminatedParams := out[3];
       EliminatedParams := [a, k1]
```

Remark that the output notation in the code example does not differentiate the new coordinates from the old ones for the sake of computational simplicity. The outputs must be interpreted exactly as in the example 2.2.2 page 41.

The system of ODEs (2.36) page 41 is the first input. The user asks for the elimination of the parameters a, b, k_1 and k_2, in this order, which are thus supposed to be strictly

positive. Every coordinate of the system is allowed to be modified so the third argument is an empty list.

The function `ELPSymmetries` of the `ExpandedLiePointSymmetry` package (see ch. 4 page 87) computes the scalings given in (2.37) page 42. Because the parameters are assumed to be strictly positive, the coordinates need to be reorganized i.e. one must take $Z = (a, b, k_1, k_2, t, x, y)$. The associated matrix of scaling follows:

$$M = \begin{pmatrix} 1 & 1 & 0 & -2 & 0 & 1 & 1 \\ 3 & 3 & 2 & 0 & -2 & 1 & 1 \end{pmatrix}. \tag{2.42}$$

The change of coordinates (2.38) page 42 can be deduced from the matrix M by following the idea given in § 2.1.2 page 37. The application of this change of coordinates to the system (2.36) page 41 leads to the reduced system given in (2.41) page 42. Remark that the parameters a and k_1 are eliminated. The structure of the scalings of the considered system of ODEs prevents the elimination of b at the same time of a. ⌟

2.2.2 Reparametrization

Here, we give the idea of the reparametrization of a system of ODEs through an example. Then we detail the reparametrization algorithm implemented in `MABSys`.

Idea of Reparametrization

The reparametrization algorithm of this section aims to eliminate some parameters from the algebraic system that defines the steady points of a system of ODEs. As for the reduction algorithm, the system of ODEs must be rewritten in a new coordinate set. The original idea is to tackle the scalings of the algebraic system that defines the steady points of the system of ODEs of interest. Such scalings are independent from the scalings of the system itself. Indeed, every scaling of a system of ODEs is also a scaling of the algebraic system that defines its steady points but the contrary is false. Their complementary information allows to improve the exact simplification done by the reduction algorithm. There exists also a bijection between positive solutions of the system of ODEs and these of the reparametrized one.

Example 2.2.5. Let us illustrate the reparametrization process on the reduced two-species oscillator given in (2.41) page 42. The algebraic system that defines its steady points is:

$$\begin{cases} 1 - \widehat{x} + \widehat{k}_2\, \widehat{x}^2\, \widehat{y} = 0, \\ \widehat{b} - \widehat{k}_2\, \widehat{x}^2\, \widehat{y} = 0 \end{cases} \tag{2.43}$$

and it depends on two parameters $\widehat{b}$ and $\widehat{k}_2$. The system of ODEs (2.41) page 42 does not have any more scalings but this algebraic system does. The scaling in question can be represented by the differential operator $\delta_{\mathcal{S}_3}$ (given also by the equation (2.12) page 36):

$$\delta_{\mathcal{S}_3} = \widehat{y}\frac{\partial}{\partial \widehat{y}} - \widehat{k}_2 \frac{\partial}{\partial \widehat{k}_2}\cdot \tag{2.44}$$

This scaling permits to put in factor the parameter $\widehat{k}_2$, which is assumed to be strictly positive, in the right-hand side of the differential equations of (2.41) page 42. The new coordinate set $\widetilde{Z} = \left(\widetilde{t}, \widetilde{x}, \widetilde{y}, \widetilde{k}_1, \widetilde{k}_2, \widetilde{a}, \widetilde{b}\right)$ is defined by the relation

$$\widetilde{y} = \widehat{y}\,\widehat{k}_2 \tag{2.45}$$

and all other coordinates remain the same (see example 2.3.4 page 53 for detailed computations). Introducing the inverse of these new coordinates i.e.:

$$\widehat{t} = \widetilde{t}, \quad \widehat{x} = \widetilde{x}, \quad \widehat{y} = \frac{\widetilde{y}}{\widehat{k}_2}, \quad \widehat{k}_1 = \widetilde{k}_1, \quad \widehat{k}_2 = \widetilde{k}_2, \quad \widehat{a} = \widetilde{a}, \quad \widehat{b} = \widetilde{b} \tag{2.46}$$

to the model (2.41) page 42 completes the reparametrization process. The system of ODEs can finally be rewritten as:

$$\begin{cases} \frac{\mathrm{d}\widetilde{x}}{\mathrm{d}\widetilde{t}} = 1 - \widetilde{x} + \widetilde{x}^2\,\widetilde{y}, \\ \frac{\mathrm{d}\widetilde{y}}{\mathrm{d}\widetilde{t}} = \left(\widetilde{b} - \widetilde{x}^2\,\widetilde{y}\right)\widetilde{k}_2 \end{cases} \tag{2.47}$$

and the algebraic system that defines its steady points as:

$$\begin{cases} 1 - \widetilde{x} + \widetilde{x}^2\,\widetilde{y} = 0, \\ \widetilde{b} - \widetilde{x}^2\,\widetilde{y} = 0 \end{cases} \tag{2.48}$$

by supposing that $\widetilde{k}_2$ is strictly positive. Remark that the steady points of the reduced system defined by (2.43) page 45 depend on 2 parameters $\widehat{b}$ and $\widehat{k}_2$ but the steady points defined by (2.48) depend on just 1 parameter $\widetilde{b}$. Also, the new system (2.47) is equivalent to the reduced system (2.41) page 42 and thus to the original system (2.36) page 41 thanks to the exact relations (2.45) and (2.38) page 42. ⌟

I think that this method can be generalized to more complicated symmetries. This thought is among my current research subjects.

Reparametrization Algorithm used in `MABSys`

The algorithm of the reparametrization procedure is summarized in the algorithm 2 page 47. The function `CylindrifySteadyPoints` of `MABSys` (see remark 5.5.8 page 132 for nomenclature choice) implements this algorithm. The user does not need any particular knowledge about the Lie symmetries but some assumptions on the coordinates positivity are required.

Proposition 2.2.6. *The reparametrization of a system of ODEs detailed in the algorithm 2 page 47 has a polynomial complexity in the input size.* ⌟

The algorithm 2 page 47 reparametrizes a given system of ODEs and computes explicitly the new coordinates in function of old ones. Thanks to the restriction of

the reparametrization method to scalings and some computational strategies (see 4.2 page 93) its complexity is polynomial in the input size.

The algorithm 2 has 4 inputs. The first one is a parametric system of first-order ODEs. The second one is a list of parameters that are supposed to be strictly positive by priority order. The algorithm tries to put these parameters in factor in the right-hand side of the differential equations. The third input is a list of coordinates that can also be zero or negative. The last input is an optional list of coordinates to keep unchanged. One computes scalings that do not act on these coordinates. Thus, in the final change of coordinates, these coordinates are not modified. Each coordinate of the system of ODEs, except for the time variable t, must appear in one of these lists. The output is composed of 4 elements: the reparametrized system of ODEs, the algebraic system that defines its steady points, the change of coordinates deduced from used scalings and the list of parameters that were eliminated from the expressions of the steady points.

Algorithm 2 Algorithm of the function `CylindrifySteadyPoints`.

Input: The parametric system of first-order ODEs Σ defined on $Z = (z_1, \ldots, z_n)$.
The list of parameters $E = (\theta_1, \ldots, \theta_e)$ that are supposed to be strictly positive. Remark that for all i in $\{1, \ldots, e\}$, there exists j in $\{1, \ldots, n\}$ such that $\theta_i = z_j$.
The list of remaining coordinates $P = (p_1, \ldots, p_{n-e-1})$ except the time variable.
The optional list of coordinates U to keep unchanged.

Output: The reparametrized system of ODEs, denoted by $\widetilde{\Sigma}$, written in a new coordinate set $\widetilde{Z} = (\widetilde{z}_1, \ldots, \widetilde{z}_n)$.
The algebraic system, written in $\widetilde{Z}$, that defines the steady points of $\widetilde{\Sigma}$.
The change of coordinates defining the new coordinate set $\widetilde{Z}$ thanks to the scalings of the algebraic system that defines the steady points of Σ.
The list of parameters that were eliminated from the expressions of the steady points.

1: #*Computation of scalings*
2: $\Omega :=$ SteadyPointSystem (Σ) ;
3: $S :=$ ELPSymmetries $(\Omega, \text{sym} = \text{scaling}, \text{fixedcoord} = U)$;
4: #*Computation of change of coordinates*
5: $\bar{Z} := [\theta_1, \ldots, \theta_e, p_1, \ldots, p_{n-e-1}]$;
6: $M :=$ MatrixOfScaling $\left(S, \bar{Z}\right)$;
7: C, pivots $:=$ GetChangeOfCoordinates (M, E, P) ;
8: #*Computation of reparametrized system*
9: $\widetilde{\Sigma} :=$ ApplyChangeOfCoordinates (Σ, C) ;
10: $\widetilde{\Omega} :=$ ApplyChangeOfCoordinates (Ω, C) ;
11: **return** $\left[\widetilde{\Sigma}, \widetilde{\Omega}, C, \text{pivots}\right]$;

The first step of the algorithm 2 is the computation of the scalings of the algebraic system that defines the steady points of the given system of ODEs. One can use the function `ELPSymmetries` of the `ExpandedLiePointSymmetry` package. In the second step, one computes a change of coordinates associated to the computed scalings. In order to find a change of coordinates (thus a new coordinate set) which allows to put in factor these parameters in the right-hand side of differential equations by respecting the priority order, the coordinates which are strictly positive are put at the beginning

of Z. The matrix of scaling (see definition 2.1.8 page 37) is created to handle scalings all together. The change of coordinates and the list of parameters that are going to be put in factor are computed. The detailed algorithm of the function `GetChangeOfCoordinates` is given in § 2.3 page 49. The third step of the algorithm consists in applying the found change of coordinates to the system of ODEs and the associated algebraic system in order to get reparametrized ones.

Example 2.2.7. Let us reproduce the reparametrization of the system of ODEs (2.41) page 42 done in the example 2.2.5 page 45 by following the algorithm 2 page 47. Here are the inputs and the outputs of the function `CylindrifySteadyPoints` of `MABSys`.

```
> ReducedODEs;
        d                                2        d                        2
       [-- x(t) = 1 - x(t) + k2 x(t)  y(t), -- y(t) = b - k2 x(t)  y(t)]
        dt                                      dt

> out := CylindrifySteadyPoints(ReducedODEs, [b,k2], [x,y]):
> out := out[1]:
> ReparametrizedODEs := out[1];


                         d                           2        d                  2
ReparametrizedODEs := [-- x(t)=1 - x(t) + x(t)  y(t), -- y(t)=(b - x(t)  y(t)) k2]
                        dt                                  dt

> ReparametrizedAlgSys := out[2];
                                          2          2
       ReparametrizedAlgSys := [1 - x + x  y, b - x  y]

> ChangeOfCoord2 :=out[3];
       ChangeOfCoord2 := [y = y k2]

> FactorizedParams := out[4];
       FactorizedParams := [k2]
```

Remark that, as for the reduction process, the output notation in the code example does not differentiate the new coordinates from the old ones for the sake of computational simplicity. The outputs must be interpreted exactly as in the example 2.2.5 page 45.

The system of ODEs (2.41) page 42 is the first input. The user indicates that the two parameters $\widehat{b}$ and $\widehat{k}_2$ are supposed to be strictly positive thus the algorithm tries to put them both in factor in the right-hand side of differential equations. In the new coordinates expressions, they can appear with rational powers and in denominators. In our case, the state variables are just assumed to be positive i.e. in $\mathbb{R}^+$.

The function `ELPSymmetries` of the `ExpandedLiePointSymmetry` package (see ch. 4 page 87) computes the scalings of the algebraic system (2.43) page 45 that defines the steady points of the model (2.41) page 42. There is one scaling, it is given in (2.44) page 45. In order to take into account the positivity assumption of the arguments, one needs to organize the coordinates and take $\widehat{Z} = \left(\widehat{b}, \widehat{k}_2, \widehat{x}, \widehat{y}\right)$. Remark that the time variable t does not appear in the coordinates because we are interested in scalings of an *algebraic* system. The associated matrix of scaling follows:

$$M = \begin{pmatrix} 0 & -1 & 0 & 1 \end{pmatrix}. \tag{2.49}$$

The change of coordinates (2.46) page 46 can be deduced from M by following the idea given in § 2.1.2 page 37. The application of this change of coordinates to the algebraic system (2.43) page 45 leads to the reparametrized algebraic system (2.48) page 46. Its application to the system (2.41) page 42 leads to the reparametrized system (2.47) page 46. The steady points of the system of ODEs (2.41) page 42 can be freed from the parameter k_2 only. Since there is no scaling that acts on b, it is impossible to free the steady points from it. ⌟

Triangularization

For the reparametrization, another boolean option, namely `triangularization`, is also available. It calls the `Triangularize` function of the `RegularChains` package (see [67]) of `Maple`. This option improves the reparametrization method by finding more scalings useful to simplify the expressions of steady points. The disadvantage is that the complexity of the associated computations is not polynomial in the worst case.

Example 2.2.8. Let us consider an algebraic system defined by:

$$\begin{cases} (x-a)^2 = 0, \\ (x-a) + (x-a)^3 = 0. \end{cases} \tag{2.50}$$

The algorithms used in the `ExpandedLiePointSymmetry` package cannot find a scaling for this system (see ch. 4 page 87). Thus the reparametrization algorithm cannot remove x or a from the solutions of (2.50). However, in the polynomial ring composed by x and a, the solutions of (2.50) are equivalent to the solutions of:

$$x - a = 0. \tag{2.51}$$

This equivalence is found thanks to the triangularization. Using the symmetries of (2.51), one can simplify also the solutions of (2.50). Indeed, the differential operator

$$\delta = x\frac{\partial}{\partial x} + a\frac{\partial}{\partial a} \tag{2.52}$$

is a symmetry of (2.51). According to § 2.1.2 page 37, if a is assumed strictly positive, one can deduce the change of coordinates $\widetilde{x} = x/a$ and $\widetilde{a} = a$. Its application on (2.50) leads to the following simplified algebraic system:

$$\begin{cases} (\widetilde{x}-1)^2 = 0, \\ (\widetilde{x}-1) + (\widetilde{x}-1)^3 = 0. \end{cases} \tag{2.53}$$

Remark that the solutions of (2.53) depend on only $\widetilde{x}$. ⌟

2.3 Change of Coordinates Algorithm associated to Scalings

This section presents a systematic way to find new coordinates to perform the previous exact simplification methods for many scalings at the same time. In these new coordinates, the system of interest or some expressions derived from it can be rewritten with less parameters.

First, let us recall the left and the right multiplication on a matrix of scaling thus on the set of scalings it represents. Second, we see the algorithm of deducing a new coordinate set from a matrix of scaling. Finally, we define the *invariants* and the *semi-invariants* that are, in fact, main objects of these new coordinates.

2.3.1 Left and Right Multiplication on a Matrix of Scaling

The left multiplication of a matrix of scaling performs a linear combination of the associated scalings. The right multiplication of a matrix of scaling rewrites the associated scalings in a new coordinate set defined by the second factor.

Lemma 2.3.1. *Let us denote by $\mathcal{B}_1 = \{\delta_1, \ldots, \delta_r\}$ a set of scalings acting on the coordinate set $Z = (z_1, \ldots, z_n)$ and M associated matrix of scaling of dimension $(r \times n)$. Let also P be any invertible matrix of dimension $(r \times r)$ in $\mathbb{R}$. Then the matrix $P\,M$ is associated to a set of scaling $\mathcal{B}_2$ that generates the same vector space as $\mathcal{B}_1$.* ⌟

Proof. The scalings of $\mathcal{B}_1$ form a vector space thus performing left multiplication on M amounts to perform linear combination on the elements of $\mathcal{B}_1$. Since P is invertible, the vector space defined by $\mathcal{B}_1$ and that defined by $\mathcal{B}_2$ are equal. ⌟

Lemma 2.3.2. *Let $\mathcal{B} = \{\delta_1, \ldots, \delta_r\}$ be a set of scalings of a system Σ defined on a coordinate set $Z = (z_1, \ldots, z_n)$ with*

$$\delta_i = \sum_{j=1}^{n} \alpha_j^i \, z_j \frac{\partial}{\partial z_j} \quad \forall i \in \{1, \ldots, r\}\,,\ \alpha_j^i \in \mathbb{R} \tag{2.54}$$

and M associated matrix of scaling. Let C be a $(n \times n)$ invertible matrix, with coefficients in $\mathbb{Q}$, that defines a change of coordinates on Z in order to obtain a new coordinate set $\widetilde{Z} = (\widetilde{z}_1, \ldots, \widetilde{z}_n)$ where

$$\widetilde{z}_j = \prod_{k=1}^{n} z_k^{C_{k,j}} \quad \forall j \in \{1, \ldots, n\}\,. \tag{2.55}$$

The matrix $M\,C$ is a matrix of scaling of the system Σ rewritten in $\widetilde{Z}$, denoted by $\widetilde{\Sigma}$. ⌟

Proof. Let us denote by

$$\widetilde{\delta}_i = \sum_{j=1}^{n} \widetilde{\xi}_j^i \frac{\partial}{\partial \widetilde{z}_j} \quad \forall i \in \{1, \ldots, r\} \tag{2.56}$$

the scaling δ_i rewritten in the new coordinate chart $\widetilde{Z}$. According to paragraph "Vector Fields" in ch. 1 of [87], one has:

$$\widetilde{\xi}_j^i \;=\; \sum_{k=1}^{n} \alpha_k^i \, z_k \frac{\partial \widetilde{z}_j}{\partial z_k} \;=\; \sum_{k=1}^{n} \alpha_k^i \, C_{k,j} \, \widetilde{z}_j \;=\; \left(\sum_{k=1}^{n} \alpha_k^i \, C_{k,j} \right) \widetilde{z}_j \;=\; \beta_j^i \, \widetilde{z}_j. \tag{2.57}$$

By definition β_j^i are in $\mathbb{R}$. So the coefficients of the scalings obtained from the change of coordinates (2.55) are given by $M\,C$ and in addition they correspond to scalings of $\widetilde{\Sigma}$ by definition. ⌟

2.3.2 Deducing a Change of Coordinates from a Matrix of Scaling

The algorithm that deduces a change of coordinates, thus a new coordinate set from a set of scalings, in which the system of interest can be rewritten in a simpler form, is summarized in the algorithm 3.

The algorithm 3 has 3 inputs. The matrix of scaling of dimension $(r \times n)$ represents r scalings that act on the n coordinates in Z. Each column of M is indexed by a coordinate. The second input is the list of parameters that are supposed to be strictly positive given by priority order. By convention, they are supposed to be at the beginning of Z. The final argument corresponds to the list of remaining coordinates. The output has 2

Algorithm 3 Change of coordinates deduced from a matrix of scaling. Algorithm of the function `GetChangeOfCoordinates`.

Input: The matrix of scaling M of dimension $(r \times n)$ constructed w.r.t. the coordinate set $Z = (z_1, \ldots, z_n)$ in which the associated scalings act. Every column of M is indexed by a coordinate.
The list of parameters E that are supposed to be strictly positive given by priority order. By convention, they are supposed to be at the beginning of Z.
The list of remaining coordinates.

Output: The change of coordinates that defines a new coordinate set $\widetilde{Z} = (\widetilde{z}_1, \ldots, \widetilde{z}_n)$ on which some given scalings are rewritten as semi-rectified scalings.
The list of parameters on which these semi-rectified scalings act.

$R :=$ ReducedRowEchelonForm (M);
#Removing unnecessary symmetries
$Q \leftarrow$ the first e lines of R where e is the number of symmetries that act at least on one strictly positive parameter;
#Getting the parameters w.r.t. which the exact simplification is done.
pivots $:= [p_1, \ldots, p_e] \leftarrow$ the list of column indices (in $\mathbb{N}$) of the first not zero elements in each line of Q;
#Construction of the inverse of the matrix C that encodes new coordinates.
#$C^{-1}_{i,j}$ is $(i,j)^{th}$ element of C^{-1}.
$C^{-1} \leftarrow$ matrix of size $(n \times n)$ defined by:
    $C^{-1}_{p_i,j} = Q_{i,j}\ \forall i \in \{1, \ldots, e\}, \forall j \in \{1, \ldots, n\}$;
    $\forall i \in \{1, \ldots, n\}$ such that $i \neq p_j\ \forall j \in \{1, \ldots, e\}$
        $C^{-1}_{i,j} = 0$ with $i \neq j$
        $C^{-1}_{i,i} = 1$;
$\ell_1, \ldots, \ell_n \leftarrow$ lcm of denominators in each row of C^{-1};
$C^{-1}_{i,j} := C^{-1}_{i,j} * \ell_i \quad \forall i \in \{1, \ldots, n\}, \forall j \in \{1, \ldots, n\}$;
#Computing new coordinates expressions.
return $\left[\text{seq}\left(\widetilde{z}_j = \prod_{k=1}^{n} z_k^{C_{k,j}}, j = 1..n\right)\right], [z_{p_1}, \ldots, z_{p_e}]$;

elements: the change of coordinates that defines a new coordinate set $\widetilde{Z}$ in which some given scalings are rewritten as semi-rectified symmetries and the list of parameters on which these semi-rectified symmetries act (these parameters are put in factor in the right-hand side of differential equations).

The instructions that permit to deduce in a unique way the change of coordinates thus the new coordinate set for the exact simplification methods (see § 2.2.1 page 41 and § 2.2.2 page 45) follow.

The matrix M of dimension $(r \times n)$ encodes the vector space of the scalings of interest. In order to obtain a unique representation of it, one can use the modified LU decomposition. One has:

$$M = P\,L\,U_1\,R \tag{2.58}$$

where R is the unique reduced row echelon form (see [27]) of dimension $(r \times n)$. By definition, the first non-zero entries in each row of R is equal to 1. Thanks to lemma 2.3.1 page 50, one can conclude that M and R represent the same vector space of scalings.

The reduction or the reparametrization methods must be performed w.r.t. the list of parameters that are supposed to be strictly positive. Thus the symmetries that do not act on one of these parameters are useless. Removing these unnecessary symmetries correspond to keeping the first e lines of R where e is the number of scalings that act at least on one strictly positive parameters. Let us denote this matrix by Q. The first not zero elements in each line of Q are indexed by the parameters, called pivots, w.r.t. which the exact simplification is done.

Let us recall that rewriting the scalings of a system in a new coordinate set, as explained in § 2.1.2 page 37, transforms these scalings into semi-rectified symmetries. Let us denote by D the matrix of scaling of these semi-rectified symmetries, written in a new coordinate set. It is composed only of the first non-zero entry of each line of Q and 0s elsewhere. We are looking for a matrix C (see lemma 2.3.2 page 50) that encodes the change of coordinates between the matrix Q and D i.e.:

$$Q\,C = D \quad \Rightarrow \quad Q = D\,C^{-1}. \tag{2.59}$$

Thanks to the special form of D, one can deduce the elements of C^{-1} by using Q. C^{-1} is a square matrix of dimension n which is constructed by inserting lines with one not zero element into Q so that its diagonal has only 1s. Every pivot must stay on the diagonal. One must also get rid of the denominators in C^{-1} so that the change of coordinates does not contain any non integer power. Each line must be multiplied by the least common multiple of its elements. These scalars can be kept in the matrix D.

Finally, the inverse of this matrix C^{-1} i.e. the matrix C encodes the change of coordinates necessary to transform the scalings represented by the matrix Q into semi-rectified symmetries (represented by the matrix D).

Example 2.3.3. Let us apply the algorithm 3 page 51 on the reduction of the two-species oscillator example modeled by (2.36) page 41. The scalings to use, $\delta_{\mathcal{S}_1}$ and $\delta_{\mathcal{S}_2}$, are given in (2.37) page 42. One assumes that the parameters a, b, k_1 and k_2 are strictly positive. Thus the coordinates are organized as follow: $Z = (a, b, k_1, k_2, t, x, y)$. The associated matrix of scaling M is defined in (2.42) page 45. The reduced row echelon form of M is:

$$R = \begin{pmatrix} 1 & 1 & 0 & -2 & 0 & 1 & 1 \\ 0 & 0 & 1 & 3 & -1 & -1 & -1 \end{pmatrix}. \tag{2.60}$$

In this case, the matrix Q is equal to R because the two scalings represented by R act at least on one of the parameters. The first not zero elements of each line of R correspond to parameters a and k_1 so the reparametrization will be done w.r.t. them. The semi-rectified symmetries that we are looking for thus have the form:

$$\delta_{\mathcal{S}_1} = \widehat{a}\frac{\partial}{\partial \widehat{a}} \quad \text{and} \quad \delta_{\mathcal{S}_2} = \widehat{k}_1\frac{\partial}{\partial \widehat{k}_1}\cdot \tag{2.61}$$

The associated matrix of scaling

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.62}$$

is written in the new coordinate set $\widehat{Z} = \left(\widehat{a}, \widehat{b}, \widehat{k}_1, \widehat{k}_2, \widehat{t}, \widehat{x}, \widehat{y}\right)$. Let us construct now the matrix C^{-1} by using the matrices Q and D:

$$C^{-1} = \begin{pmatrix} 1 & 1 & 0 & -2 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 3 & -1 & -1 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{2.63}$$

The inverse of C^{-1} encodes the change of coordinates that transforms the scalings given in (2.37) page 42 into these given in (2.61):

$$C = \begin{pmatrix} 1 & -1 & 0 & 2 & 0 & -1 & -1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -3 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{2.64}$$

The elements of C correspond to powers of the coordinates Z in the new coordinates expressions. The resulting change of coordinates is defined in (2.38) page 42. ⌟

Example 2.3.4. Let us apply now the algorithm 3 page 51 on the reparametrization of the reduced two-species oscillator example modeled by (2.41) page 42. The only scaling to use is $\delta_{\mathcal{S}_3}$ given in (2.44) page 45. The parameters $\widehat{b}$ and $\widehat{k}_2$ are assumed to be strictly positive thus the coordinates are organized as follow: $\widehat{Z} = \left(\widehat{b}, \widehat{k}_2, \widehat{x}, \widehat{y}\right)$ (the symmetry belongs to an algebraic system so the time parameter $\widehat{t}$ is not among the coordinates). The associated matrix of scaling M is defined in (2.49) page 48. The reduced row echelon form of M is:

$$R = \begin{pmatrix} 0 & 1 & 0 & -1 \end{pmatrix}. \tag{2.65}$$

In this case, the matrix Q is equal to R because the associated scaling acts on the parameter $\widehat{k}_2$. The first not zero elements of R correspond to $\widehat{k}_2$ meaning that it can be

removed from the equations of the steady points of (2.41) page 42. The semi-rectified symmetry that we are looking for has the form:

$$\delta_{\mathcal{S}_1} = \widetilde{k}_2 \frac{\partial}{\partial \widetilde{k}_2}. \tag{2.66}$$

The associated matrix of scaling D written in the new coordinate set $\widetilde{Z} = \left(\widetilde{b}, \widetilde{k}_2, \widetilde{x}, \widetilde{y}\right)$ follows:

$$D = \begin{pmatrix} 0 & 1 & 0 & 0 \end{pmatrix}. \tag{2.67}$$

Let us construct now the matrix C^{-1} by using the matrices Q and D:

$$C^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{2.68}$$

The inverse of C^{-1} encodes the change of coordinates that transforms the scaling given in (2.44) page 45 into that given in (2.66):

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{2.69}$$

The elements of C correspond to powers of the coordinates Z in the new coordinates expressions. The resulting change of coordinates is defined in (2.45) page 46. ⌟

2.3.3 Invariants and Semi-Invariants

The reduction (see § 2.2.1 page 41) and the reparametrization (see § 2.2.2 page 45) methods consist in looking for a new coordinate set in which the system can be rewritten in a simpler form. In fact, these new coordinates are *invariants* and *semi-invariants* (see [92, 86, 59, 58]) of the scalings of the system.

Definition 2.3.5 (Invariants and Semi-Invariants)**.** Let $\delta_{\mathcal{S}}$ be a differential operator representing a Lie symmetry with coefficients in $\mathbb{R}(Z)$. A smooth real-valued function $\zeta : \mathbb{R}(Z) \to \mathbb{R}$ is an *invariant* (resp. a *semi-invariant*) of $\delta_{\mathcal{S}}$ if, and only if, $\delta_{\mathcal{S}}\zeta = 0$ (resp. $\delta_{\mathcal{S}}\zeta = \alpha\,\zeta$ with α in $\mathbb{R}$) for all Z in $\mathbb{R}^n$. ⌟

Remark 2.3.6. The invariants of a symmetry form a field and the semi-invariants of a symmetry form an algebra. Thus, there are infinitely many invariants and semi-invariants for a given Lie symmetry. ⌟

Example 2.3.7. Let us reconsider the differential operator $\delta_{\mathcal{S}_1}$ given in (2.8) page 35. Among all of its invariants, assuming that a does not vanish, one can cite:

$$\zeta_1 = t, \quad \zeta_2 = \frac{x}{a}, \quad \zeta_3 = \frac{y}{a}, \quad \zeta_4 = k_1, \quad \zeta_5 = k_2\,a^2, \quad \zeta_6 = \frac{b}{a} \tag{2.70}$$

because $\delta_{\mathcal{S}_1}\zeta_i = 0$ for all i in $\{1,\ldots,6\}$. Also remark that each coordinate on which $\delta_{\mathcal{S}_1}$ acts is a semi-invariant of it because of the definition of scalings. One has:

$$\delta_{\mathcal{S}_1} x = x, \quad \delta_{\mathcal{S}_1} y = y, \quad \delta_{\mathcal{S}_1} k_2 = -2\,k_2, \quad \delta_{\mathcal{S}_1} b = b, \quad \delta_{\mathcal{S}_1} a = a. \tag{2.71}$$

⌟

2.4 Comparison with Existing Works

There are lots of work done about the model simplification. This section is devoted to present some of them and to point out the differences of my work w.r.t. the existing ones.

I propose two exact simplification methods, meaning that the exact relationships between the original and the simplified systems are available. There exist widely used general strategies for the model simplification (see [85, 6]), including inexact methods. For instance, the lumping (see [116]), the sensitivity analysis (see [108, 94, 71]) and the time-scale analysis (see [16, 113]) rely on different methods but they all decrease the number of variables. The lumping transforms the vector of the original variables to a lower dimensional one by bringing some variables together. The sensitivity analysis seeks to determine and to eliminate the insignificant variables on the basis of their impact on designated important variables; only a subset of the original variables remain in the reduced model. The time-scale analysis is used to determine the dynamic behavior of the reaction systems with multiple time-scales; some fast reactions are considered nearly instantaneous relative to the remaining slow ones. These three methods cause a loss of information about individual original variables.

The reduction and the reparametrization methods presented in this document have polynomial time complexity in the input size. Medium size nonlinear systems of ODEs are thus relatively easily tractable. In addition, there is no need to specify the dimension of the variables, which is not always obvious to define. In modeling domain, it is common and practical to introduce some parameters with undefined physical or biological meaning in the models. For the simplifications of this chapter, only the assumption about the positivity of some variables is required. The dimensional analysis (see [21, 17]) is a classical reduction method based on the units of variables. It simplifies large-scale problems by using dimensionless parameters (see [76] for its applications in biology). There is no complexity result for such an analysis (see [64]). Also, in general cases, the complexity of inexact reduction methods is more complicated than the polynomial time complexity in the input size. For example, the time-scale analysis is based on the quasi-steady state approximation. In [10], authors present a new algorithm (see § 1.3.3 page 14 and B.2.3 page 158) for this approximation that is based on the differential elimination (see [118, 8]). This last method has an exponential complexity in general cases. Indeed, the dimensional analysis is based on the scaling type Lie symmetries (see § 1.2 of [5] and th. 3.22 of [86]) on which I am working. The reduction of the variables of a differential system through the dimensional analysis is a special case of the reduction using scalings (see ch. 1 of [6]) so a special case of my work.

The reduction and the reparametrization methods that I present in this document are based on the classical Lie symmetry theory with new algorithmic ideas. In my

work, I propose a computation of Lie symmetries with polynomial time complexity in the input size (see [101]) thanks to our computational approaches. In particular, I restrict myself to special kind Lie symmetries (scalings and more complicated ones). The required operations are thus restricted to linear algebra over a number field and univariate polynomial factorization. The classical (see [24]) and the moving frame based (see [37, 38]) reduction methods are mostly applied to differential equations (see also [86, 88]). Various applications of invariants (deduced from Lie symmetries) in the study of dynamical systems are available (see [42]). In [59], authors show how to compute a complete set of invariants given a rational group action. All these works consider Lie symmetries mostly with their general form. Their computation is based on Gröbner bases which has exponential complexity in the general cases. Moreover, the reduction and the reparametrization algorithms are also of polynomial time complexity in the input size thanks to some computational strategies. There is no need of solving any partial differential equation contrarily, for example, the classical reduction method.

In this document, one of the newness is that the exact simplification methods are adapted to the context of modeling in biology. Their polynomial complexity is essential to treat realistic examples. Practical examples showed that the positivity assumption of the variables is very important in our algorithms. Fortunately, this information is usually available.

The reparametrization method is an original way of using the Lie symmetries which leads to a new algorithm that improves the preliminary analysis of models and facilitates their qualitative analysis. We extend the classical reduction based on the Lie symmetries of a differential system. We use also the Lie symmetries of the associated algebraic system that defines its steady points. As a result, the steady points of the reparametrized system depend on less parameters. This phenomena eases the computation and all further analysis of steady points. Another new point of this method is the possibility of combining Lie symmetries and regular chains (see [67]). Clearly, if this option is chosen, the complexity of the algorithm increases. However, this triangularization helps to find some extra Lie symmetries that can be used for the exact simplification.

"Great things are not done by impluse, but by a series of small things brought together."

Letter to Theo van Gogh
The Hague, October 22^{th} 1882
Vincent van Gogh

III Generalizations of Exact Model Simplification Methods

3 Geometrical Framework and Notations

This chapter presents the geometrical framework for understanding the generalizations of the exact simplifications algorithms. Algebraic systems, systems of first-order ordinary differential equations (ODEs), systems of first-order ordinary recurrence equations (OREs) and Lie algebras are the main mathematical objects used in this document. Along this chapter, first I detail the descriptions and the syntax of the these objects implementation in my work. Then I explain the associated geometry useful in the forthcoming chapters. Finally, I give the corresponding data structures, manipulated in the `MABSys` and/or in the `ExpandedLiePointSymmetry` packages.

3.1 Algebraic Systems

In this document, an *algebraic system* is described in the n-dimensional affine Euclidean space $\mathbb{R}^n$ where n is a fixed integer. In this section, we present the definition of such systems, the associated geometrical objects and finally their implementation in the package `ExpandedLiePointSymmetry`.

3.1.1 Representation

This subsection is devoted to the definition of *algebraic systems* as they are considered in this document.

Definition 3.1.1 (Algebraic System)**.** Let $F = (f_1, \ldots, f_k) = (p_1/q_1, \ldots, p_k/q_k)$ be a finite set of rational functions over the field $\mathbb{R}$ where p_i and q_i are polynomials in $\mathbb{R}[Z]$ i.e. in variables $Z = (z_1, \ldots, z_n)$ with coefficients in $\mathbb{R}$. An *algebraic system* associated to F is defined by the following equalities and inequalities:

$$\begin{cases} p_1(z_1, \ldots, z_n) = 0, \\ \quad\vdots \\ p_k(z_1, \ldots, z_n) = 0 \end{cases} \quad \text{and} \quad \begin{cases} q_1(z_1, \ldots, z_n) \neq 0, \\ \quad\vdots \\ q_k(z_1, \ldots, z_n) \neq 0. \end{cases} \tag{3.1}$$

⌟

Example 3.1.2. Below system of equations is an algebraic system defined on a space of 6 variables, namely $Z = (P, Q, a, b, c, \ell)$:

$$\begin{cases} (1 - c\,P) + b\,Q + 1 = 0, \\ a\,P - \ell\,Q + 1 = 0. \end{cases} \tag{3.2}$$

⌟

3.1.2 Semi-Algebraic Set

Here, we give the definition of a *semi-algebraic set* (see [28]) generated by an algebraic system (see definition 3.1.1 page 59) and the associated *regularity* property along with examples.

Definition 3.1.3 (Semi-Algebraic Set)**.** A *semi-algebraic set* $\mathcal{V}$ is defined by a generating set of finite number of rational functions $F = (f_1, \ldots, f_k) = (p_1/q_1, \ldots, p_k/q_k)$ where p_i and q_i are polynomials in $\mathbb{Q}[Z]$:

$$\mathcal{V} := \{Z \in \mathbb{R}^n \mid p_i(Z) = 0 \quad \text{and} \quad q_i(Z) \neq 0 \quad \forall i \in \{1, \ldots, k\}\}\,. \tag{3.3}$$

⌟

Hypothesis 3.1.4. *For the sake of simplicity, instead of working with semi-algebraic sets, we restrict ourselves to* semi-algebraic varieties *in this document. Indeed, a semi-algebraic set is not necessarily irreducible contrarily to a semi-algebraic variety (see [28]). For instance, the algebraic system defined by the equation* $x^3 - p\,x = 0$ *is a union of two varieties* $x = 0$ *and* $x^2 - p = 0$*.* ⌟

Definition 3.1.5 (Regular Algebraic System)**.** An algebraic system $F = (f_1, \ldots, f_k)$ and the associated semi-algebraic variety $\mathcal{V}$ are called *regular* (a.k.a. *smooth*) if the system F is of maximal rank k, meaning that the Jacobian matrix $(\partial f_i/\partial z_j)$ is of rank k at every solution Z in $\mathcal{V}$ of the system (see [34]). ⌟

Hypothesis 3.1.6. *I restrict the algebraic systems of this document, thus the semi-algebraic varieties they define, to regular ones.*

The hypothesis 3.1.6 is essential, for instance, when one wants to look for the *expanded Lie point symmetries* of an algebraic system as in the theorem 4.1.1 page 88. The following two examples illustrate regular and non-regular algebraic systems.

Example 3.1.7. The algebraic system defined by $f(x, y) = x^4 + x^2\,y^2 + y^2 - 1$ is regular. One can verify this property by computing the associated Jacobian matrix:

$$\left(4\,x^3 + 2\,x\,y^2, 2\,x^2\,y + 2\,y\right). \tag{3.4}$$

This matrix has rank 0 if, and only if, $x = y = 0$. This is not a solution of $f(x, y) = 0$ hence the maximal rank condition is satisfied. ⌟

Example 3.1.8. Contrarily to the previous example, the algebraic system defined by $g(x) = x^2 - 2\,x + 1$ is not regular. The solution of $g(x) = 0$ is a point $x = 1$. The associated Jacobian matrix:

$$\left(2\,x - 2\right) \tag{3.5}$$

vanishes on this solution, hence the maximal rank condition is not satisfied. ⌟

3.1.3 Implementation of an Algebraic System

This section is devoted to the data structure `AlgebraicSystem` that represents an algebraic system in the `ExpandedLiePointSymmetry` package.

This object created by the `NewAlgebraicSystem` constructor is identified by the key word `AlgebraicSystem` and the following elements:

- a list of rational functions $(f_1, \ldots, f_k)$; if one of these rational functions is a constant then the system is reduced to a list of unique element 1 i.e. to the simplest inconsistent system $1 = 0$;
- a list of coordinates $(z_1, \ldots, z_n)$ defining the space on which the corresponding semi-algebraic set is considered;
- a list of inequalities $(q_1, \ldots, q_k)$ i.e. not null polynomials issue from the rational functions denominators (see definition 3.1.1 page 59), if they exist.

Remark 3.1.9. Gröbner basis computations are very frequent in computer algebra systems. In the future, if such algorithms are needed to be used effectively on algebraic systems, one can add fields to this data structure that corresponds to an ordering and Gröbner basis. ⌟

Remark 3.1.10. Our implementation does not display as output the real object representation but a simpler notation based only on the associated equations for the sake of simplicity. ⌟

The following example constructs an object of type `AlgebraicSystem` and details its components.

Example 3.1.11. The algebraic system given in the example 3.1.2 page 59 may be handled in our framework by the following commands. An object of type `AlgebraicSystem` is defined by the `NewAlgebraicSystem` constructor.

```
> # Creation of an algebraic system
> AlgSys := NewAlgebraicSystem([(1-c*P)*P+b*Q+1,a*P-l*Q+1]);
       AlgSys := [(1 - c P) P + b Q + 1, a P - Q l + 1]

> # Details of the AlgebraicSystem data structure
> lprint(AlgSys);
AlgebraicSystem([(1-c*P)*P+b*Q+1, a*P-Q*l+1],[P, Q, a, b, c, l],[])

> # Type check
> type(AlgSys, AlgebraicSystem);
       true
```

Remark 3.1.12. `Maple` procedures use `Maple` types to direct the flow of control in the algorithms or to decide whether an expression is a valid input. There is a new way of defining the types since its version 12 but it is not used in the implementations presented in this document. ⌟

3.2 Systems of ODEs and OREs

This section is devoted to *systems of ODEs* and *OREs* that are frequently used in modeling domains. We define them explicitly, discuss the geometrical objects they define and show their implementation.

3.2.1 Algebraic Representations and Pseudo-derivations

We define separately *systems of ODEs* and *systems of OREs* and then recall a unified framework, *pseudo-derivations*. In this document, this last is used to apply the Lie symmetries theory almost in the same manner for these two kinds of systems.

Systems of ODEs

In this paragraph, we define *systems of first-order ODEs* as used in this document.

Definition 3.2.1 (System of ODEs)**.** Let $\mathrm{d}\cdot/\mathrm{d}t$ be a derivation w.r.t. the continuous independent variable t. We consider two sets $X := (x_1, \ldots, x_k)$ and $\Theta := (\theta_1, \ldots, \theta_\ell)$. The associated *coordinate set* is defined by $Z := (z_1, \ldots, z_n) = (t, x_1, \ldots, x_k, \theta_1, \ldots, \theta_\ell)$ and its cardinal is $n = 1 + k + \ell$. With these notations, a *system of first-order ordinary differential equations (ODEs)* is a system where:

$$\begin{cases} \frac{\mathrm{d}t}{\mathrm{d}t} = 1, \\ \frac{\mathrm{d}x_i}{\mathrm{d}t} = f_i(z_1, \ldots, z_n) \text{ with } f_i \in \mathbb{R}(Z) \quad \forall i \in \{1, \ldots, k\}, \\ \frac{\mathrm{d}\theta_j}{\mathrm{d}t} = 0 \quad \forall j \in \{1, \ldots, \ell\} \end{cases} \tag{3.6}$$

and the set $F := (f_1, \ldots, f_k)$ specifies the evolution of the state variables of ODEs. The elements of the set $X := (x_1, \ldots, x_k)$ are called *state variables*, these of $\Theta := (\theta_1, \ldots, \theta_\ell)$ *parameters*. ⌟

Remark 3.2.2. The definition of the systems of ODEs are given in $\mathbb{R}(Z)$ but the transcendent elements of $\mathbb{R}$ are difficult to handle in the computations. Thus, in the sequel, the computations are done in the effective field $\mathbb{Q}$. ⌟

Remark 3.2.3. The first differential equation $\mathrm{d}t/\mathrm{d}t = 1$ of (3.6) defines the independent variable t. By definition, a system of ODEs has just one independent variable. That is why, in the sequel, this equation is omitted in the explicit systems definitions. ⌟

Remark 3.2.4. Every system of ODEs may be written as a system of first-order ODEs by introducing new indeterminates and equations. As a classical example, let us look at the Van der Pol oscillator (see [111]). It involves a second-order ODE that verifies:

$$\frac{\mathrm{d}^2 y}{\mathrm{d}t^2} - \mu\left(1 - y^2\right)\frac{\mathrm{d}y}{\mathrm{d}t} - y = 0, \quad \frac{\mathrm{d}\mu}{\mathrm{d}t} = 0. \tag{3.7}$$

An equivalent but a first-order system can be constructed by introducing a new variable x equal to the first-derivative of y i.e. $x = \mathrm{d}y/\mathrm{d}t$. With this new variable, one can write

the Van der Pol oscillator in the following form:

$$\begin{cases} \frac{\mathrm{d}x}{\mathrm{d}t} = \mu\left(1 - y^2\right) x - y, \\ \frac{\mathrm{d}y}{\mathrm{d}t} = x, \\ \frac{\mathrm{d}\mu}{\mathrm{d}t} = 0. \end{cases} \tag{3.8}$$

⌟

Remark 3.2.5. The `ExpandedLiePointSymmetry` package includes also a state space representation type. This type is not treated in this document but for more information see associated help pages [102]. ⌟

Here follows an example of a system of ODEs that is used in the sequel.

Example 3.2.6. Let us consider the Verhulst's logistic growth model with a linear predation (see § 1.1 in [83]). This is a simple model where the state variable x represents a population. The parameter a is the difference between its growth and predation rate and the parameter b corresponds to the receive capacity of the environment:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = (a - b\,x)\,x, \quad \frac{\mathrm{d}a}{\mathrm{d}t} = \frac{\mathrm{d}b}{\mathrm{d}t} = 0. \tag{3.9}$$

⌟

Systems of OREs

Now, let us look at *systems of first-order OREs* that are defined in a similar way of the systems of first-order ODEs. The main difference is that systems of OREs encode discrete evolutions.

Definition 3.2.7 (System of OREs)**.** Let τ be a discrete independent variable. We consider two sets $X := (x_1, \ldots, x_k)$ and $\Theta := (\theta_1, \ldots, \theta_\ell)$. The cardinal of the coordinate set $Z := (z_1, \ldots, z_n) = (\tau, x_1, \ldots, x_k, \theta_1, \ldots, \theta_\ell)$ is equal to $n = 1 + k + \ell$. In addition, ${z_i}^\tau$ represents the expression of z_i at step τ. With these notations, a *system of first-order ordinary recurrence equations (OREs)* is a system where:

$$\begin{cases} {x_i}^{\tau+1} = f_i({z_1}^\tau, \ldots, {z_n}^\tau) \text{ with } f_i \in \mathbb{R}(Z) \quad \forall i \in \{1, \ldots, k\}, \\ {\theta_j}^{\tau+1} = {\theta_j}^\tau \quad \forall j \in \{1, \ldots, \ell\} \end{cases} \tag{3.10}$$

and the set $F := (f_1, \ldots, f_k)$ specifies the evolution of the state variables of OREs. The elements of the set $X := (x_1, \ldots, x_k)$ are called *state variables*, these of $\Theta := (\theta_1, \ldots, \theta_\ell)$ *parameters*. ⌟

Remark 3.2.8. The definition of system of OREs (see definition 3.2.7) is similar with the system of difference equations with a fixed and regular lattice defined in [117]. ⌟

Remark 3.2.9. As for the systems of ODEs, the definition of the systems of OREs are given in $\mathbb{R}(Z)$. For the same reasons (see remark 3.2.2 page 62), in the sequel, the computations are done in the effective field $\mathbb{Q}$. ⌟

Here is an academic example of systems of OREs used in the sequel.

Example 3.2.10. The following system is a simple example of systems of OREs with two state variables x, y and one parameter c:

$$x^{\tau+1} = c^\tau\, x^\tau, \quad y^{\tau+1} = y^\tau + x^\tau, \quad c^{\tau+1} = c^\tau. \tag{3.11}$$

⌟

Pseudo-derivations

The goal of this paragraph is to recall a unified framework for the systems of ODEs and OREs, *pseudo-derivations* (see ch. 1 of [26] and references therein). This framework gives a generic standpoint for the forthcoming algorithms, more precisely for the computation of their *Lie point symmetries*.

Definition 3.2.11 (Pseudo-derivation)**.** A *pseudo-derivation* δ in the field $\mathbb{R}(Z)$ is a linear operator $\delta : \mathbb{R}(Z) \to \mathbb{R}(Z)$ that has $\mathbb{R}$ in its kernel and that satisfies the twisted Leibniz rule:

$$\forall\, (f_1, f_2) \in \mathbb{R}(Z)^2\,, \quad \delta f_1 f_2 = f_1 \delta f_2 + \sigma(f_2)\, \delta f_1 \tag{3.12}$$

where σ is a $\mathbb{R}(Z)$-endomorphism. ⌟

Remark 3.2.12. The pseudo-derivations can be called *Ore operator* of an *Ore algebra* (see [26]) denoted by Ore$\mathbb{R}(Z)$. ⌟

An *infinitesimal generator* is a derivation (a pseudo-derivation where σ is equal to the identity function) that is closely related to the systems of ODEs (more precisely to *continuous dynamical systems*). Following definitions describe an infinitesimal generator and show how one can associate such a differential operator to a system of ODEs. They are followed by an illustration example. For the link between a system of ODEs, the associated vector field and the infinitesimal generator see § 1.3 of [86] and § 3.4 page 79 of this document.

Definition 3.2.13 (Infinitesimal Generator)**.** An *infinitesimal generator* δ is a derivation i.e. a pseudo-derivation where the endomorphism σ is the identity map (see ch. 16 of [34]). In the canonical basis of elementary derivations $\{\partial/\partial z_1, \ldots, \partial/\partial z_n\}$, it is written as:

$$\delta = \sum_{i=1}^{n} \xi_{z_i}(Z)\, \frac{\partial}{\partial z_i} \tag{3.13}$$

where ξ_{z_i} is in $\mathbb{R}(Z)$ for all i in $\{1, \ldots, n\}$. ⌟

Definition 3.2.14 (Infinitesimal Generator associated to a System of ODEs)**.** The *infinitesimal generator associated to a system of ODEs*, described in the definition 3.2.1 page 62, is defined with the same notations as follows:

$$\delta = \frac{\partial}{\partial t} + \sum_{i=1}^{k} f_i(Z)\, \frac{\partial}{\partial x_i}\cdot \tag{3.14}$$

⌟

Example 3.2.15. The infinitesimal generator associated to the system of ODEs given in the example 3.2.6 page 63 is:

$$\delta = \frac{\partial}{\partial t} + ((a - b\,x)\,x)\,\frac{\partial}{\partial x}\cdot \tag{3.15}$$

⌟

Actually, a *recurrence operator* defines a system of OREs to which one can associate a pseudo-derivation. The following definition shows how one can describe such operators.

Definition 3.2.16 (Recurrence Operator and its Pseudo-derivation)**.** A *recurrence operator* σ is a $\mathbb{R}(Z)$-endomorphism that maps the expression of a coordinate z_i at step τ to its expression at step $\tau + 1$. The recurrence operator associated to a system of OREs described in the definition 3.2.7 page 63 follows:

$$\left\{ \begin{array}{l} \sigma\tau = \tau + 1, \quad \sigma x_i = f_i(Z) \quad \forall i \in \{1, \ldots, k\}\,, \\ \sigma\theta_j = \theta_j \quad \forall j \in \{1, \ldots, \ell\}\,, \quad \sigma\kappa = \kappa \quad \forall \kappa \in \mathbb{R}. \end{array} \right. \tag{3.16}$$

A pseudo-derivation δ may be associated to every system of OREs defined in the terms of its recurrence operator σ as follows:

$$\forall f \in \mathbb{R}(Z)\,, \quad \delta f = \sigma f - f. \tag{3.17}$$

⌟

Remark 3.2.17. The recurrence operator σ of (3.16) is the same $\mathbb{R}(Z)$-endomorphism used in (3.12) page 64. ⌟

The next example illustrates the pseudo-derivations associated to recurrence operators, with a notation as close as possible to that of systems of ODEs. Remark that in this discrete case, one can not mention a basis of elementary derivations unlike the continuous case. Thus the action of the pseudo-derivation must be given explicitly for every coordinate.

Example 3.2.18. The recurrence operator σ and the pseudo-derivation δ associated to the example 3.2.10 page 64 verify the following equalities:

$$\left\{ \begin{array}{ll} \sigma\tau = \tau + 1, & \sigma y = y + x, \\ \sigma x = c\,x, & \sigma c = c \end{array} \right. \quad \text{and} \quad \left\{ \begin{array}{ll} \delta\tau = 1, & \delta y = x, \\ \delta x = (c - 1)\,x, & \delta c = 0. \end{array} \right. \tag{3.18}$$

⌟

Remark 3.2.19. Some more general systems of OREs may be written as a system of first-order OREs by introducing new indeterminates and equations. For example, let us consider the system of second-order OREs:

$$\left\{ \begin{array}{l} \sigma^2 x = \sigma x + x - y, \\ \sigma y = \sigma x - y. \end{array} \right. \tag{3.19}$$

An equivalent but first-order system can be constructed by introducing a new variable z equal to σx. By the help of this new variable, one can write the system (3.19) in the following form:

$$\left\{ \begin{array}{l} \sigma x = z, \\ \sigma z = z + x - y, \\ \sigma y = z - y. \end{array} \right. \tag{3.20}$$

⌟

3.2.2 Implementation of Systems of ODEs and OREs

The `InfinitesimalGenerator` and the `OrdinaryDifferentialEquations` data structures correspond to the implicit representations of the systems of ODEs. Whereas the `RecurrenceOperator` data structure corresponds to the systems of OREs. In this subsection, we detail these data structures and their implementation.

Systems of ODEs

The data structure `InfinitesimalGenerator` represents the systems of ODEs and this notation is adapted especially to the Lie symmetries computations. The creation of this object by the `NewInfinitesimalGenerator` constructor requires the coefficients of the canonical basis of the elementary derivations and the list of the associated coordinates. The result is defined by the key word `InfinitesimalGenerator` and the following elements:

- a list of coefficients $(\xi_{z_1}, \ldots, \xi_{z_n})$ of the elementary derivations as described in (3.13) page 64);
- a list of coordinates $(z_1, \ldots, z_n)$ on which these elementary derivations are defined;
- a list of input coordinates i.e. variables used in the expression of coefficients but not given in the list of coordinates. The default value is an empty list;
- the associated Lie derivation operator created by the `NewLieDerivationOperator` constructor (see [102]).

Remark 3.2.20. Our implementation does not display as output the real object representation for the `InfinitesimalGenerator` data structure. Instead of this, a functional notation that corresponds to the Lie derivation operator is given. ⌟

The construction of an object of type `InfinitesimalGenerator` and the details of its components are illustrated in the following example.

Example 3.2.21. The example 3.2.6 page 63 may be translated in our framework with the following commands. An object of type `InfinitesimalGenerator` is created by the constructor `NewInfinitesimalGenerator`. The associated derivation operator can be used directly via this object.

```
> # Creation of a system of ODEs
> InfGen := NewInfinitesimalGenerator([1, (a-b*x)*x, 0, 0],[t,x,a,b]);
                      /d   \                /d   \
       InfGen := F -> |-- F| + (a - b x) x |-- F|
                      \dt  /                \dx  /

> lprint(InfGen);
InfinitesimalGenerator([1, (a-b*x)*x, 0, 0],[t, x, a, b],[],
F -> diff(F,t)+(a-b*x)*x*diff(F,x))

> # Type check
> type(InfGen, InfinitesimalGenerator);
       true
```

```
> # Application of this infinitesimal generator
> InfGen(x);
       (a - b x) x
```

Systems of ODEs are also represented by the `OrdinaryDifferentialEquations` data structure in order to create the link with other packages and to have a classical notation when necessary. This type requires no knowledge about the infinitesimal generator notion. There is no special constructor function, it is enough to write the ODEs in a list with "diff" operator of `Maple`. This helps also the usage of the package `MABSys` by the non specialists of the domain.

The main difference between the encoding of an ODE by `InfinitesimalGenerator` and `OrdinaryDifferentialEquations` data structures is related to the fact that the parameters can be encoded in two different ways:

- The first one includes one differential equation per parameter (considered as constant functions). This is equivalent to the definition 3.2.1 page 62 and it is used through the data structure `InfinitesimalGenerator`.
- The second one does not put any constraint on the parameters but *they are supposed to have values in* $\mathbb{R}$. The data structure `OrdinaryDifferentialEquations` adopts the second option for the sake of coherence with the classical notations in `Maple` and simplicity of its syntax.

The following code shows these two encodings associated to the example 3.2.6 page 63.

```
> ODEs_1 := [diff(x(t),t)=(a(t)-b(t)*x(t))*x(t),diff(a(t),t)=0,diff(b(t),t)=0];
                 d                                          d             d
       ODEs_1 := [-- x(t) = (a(t) - b(t) x(t)) x(t), -- a(t) = 0, -- b(t) = 0]
                 dt                                         dt            dt

> # Type check
> type(ODEs_1, OrdinaryDifferentialEquations);
       false

> ODEs_2 := [diff(x(t),t)=(a-b*x(t))*x(t)];
                 d
       ODEs_2 := [-- x(t) = (a - b x(t)) x(t)]
                 dt

> # Type check
> type(ODEs_2, OrdinaryDifferentialEquations);
       true
```

Remark 3.2.22. This nuance does not affect the forthcoming computations. ⌟

In addition of these types and their constructors, some auxiliary functions are available such as the conversion between the data structures `InfinitesimalGenerator` and `OrdinaryDifferentialEquations`, the manipulation of single infinitesimal generators (for example, getting the parameters or the state variables) and some arithmetic functions implementing the Lie algebra of infinitesimal generators (multiplication by a scalar, addition of two infinitesimal generators, etc.; see § 3.3 page 68). For more information, see associated help pages [102].

Systems of OREs

A system of OREs (see definition 3.2.16 page 65) is represented by the data structure `RecurrenceOperator`. Its creation requires the coordinates of the system and the image of these coordinates by the endomorphism associated to the recurrence operator. The `NewRecurrenceOperator` constructor defines an object by the key word `RecurrenceOperator` and the following elements:

- a list of the images of the coordinates $(\xi_{z_1}, \ldots, \xi_{z_n})$;
- a list of these coordinates $(z_1, \ldots, z_n)$;
- a list of input coordinates i.e. the variables that are used in the expression of the images but not given in the list of the coordinates. The default value is an empty list;
- the associated endomorphism operator created by the `NewEndomorphism` constructor.

Remark 3.2.23. Our framework does not display the real object representation for the type `RecurrenceOperator`. A functional notation that corresponds to the associated endomorphism is given in the output. ⌟

The following example illustrates the construction and the components of an object of type `RecurrenceOperator`.

Example 3.2.24. The example 3.2.10 page 64 may be translated in our framework with the following commands. An object of type `RecurrenceOperator` is created by the constructor `NewRecurrenceOperator`. The associated endomorphism operator can be used directly via this object.

```
> # Creation of a system of OREs
> RecurOp := NewRecurrenceOperator([tau+1,c*x,y+x,c],[tau,x,y,c]);
       RecurOp := (tau, x, y, c) -> [tau + 1, c x, y + x, c]

> lprint(RecurOp);
RecurrenceOperator([tau+1, c*x, y+x, c],[tau, x, y, c],[],
(tau, x, y, c) -> [tau+1, c*x, y+x, c])

> # Type check
> type(RecurOp, RecurrenceOperator);
       true

> # Application of this recurrence operator
> RecurOp(x);
       c x
```

3.3 Lie Algebras of Infinitesimal Generators

In this document, *Lie algebras* are represented by a generating set of infinitesimal generators. First, we give the formal definition of a Lie algebra and discuss its base field representation because it has an important role in our implementations. We define the

structure constants associated to a Lie algebra. Then we focus on the corresponding Lie algebra classifications. We survey also the Lie algebras of scalings that are frequently used in the `MABSys` package. Finally, we close the section by explaining the implementation of a Lie algebra in the `ExpandedLiePointSymmetry` package.

3.3.1 Definition of Lie Groups and Lie Algebras

This subsection is devoted to the definition of *Lie groups*, *Lie algebras* and naturally to the discussion Lie algebras base field.

Definition 3.3.1 (Lie Group)**.** A *Lie Group* G is a differentiable manifold with group structure where the applications

$$\begin{array}{rcl} G \times G & \rightarrow & G \\ (g, h) & \rightarrow & g \cdot h \end{array} \tag{3.21}$$

and

$$\begin{array}{rcl} G & \rightarrow & G \\ g & \rightarrow & g^{-1} \end{array} \tag{3.22}$$

are also differentiable i.e. compatible with the smooth structure. ⌟

Before defining the Lie algebras, let us present the *Lie bracket* operation.

Definition 3.3.2 (Lie Bracket)**.** The operation *Lie bracket* is defined by a bilinear map on a set of infinitesimal generators $\mathfrak{g}$:

$$\begin{array}{rccl} [\,,\,]: & \mathfrak{g} \times \mathfrak{g} & \rightarrow & \mathfrak{g}, \\ & (\delta_1, \delta_2) & \rightarrow & \delta_1 \delta_2 - \delta_2 \delta_1. \end{array} \tag{3.23}$$

This map is skew-symmetric and satisfies the relations:

$$\forall\, (\delta_1, \delta_2, \delta_3) \in \mathfrak{g}^3, \quad [\delta_1, [\delta_2, \delta_3]] + [\delta_2, [\delta_3, \delta_1]] + [\delta_3, [\delta_1, \delta_2]] = 0 \tag{3.24}$$

known as the Jacobi identity. ⌟

Definition 3.3.3 (Lie Algebra)**.** A *Lie algebra* $\mathfrak{g}$ is an algebra constituted by a vector space equipped with Lie bracket as additional operation. It is defined on an extended field of $\mathbb{R}$ (see below). ⌟

Hypothesis 3.3.4. *In this document, unless stated otherwise, all considered Lie algebras are supposed to be* finite *dimensional.* ⌟

Remark 3.3.5. To every Lie group, one can associate a Lie algebra (see theorem 1.54 of [86]) ⌟

In our framework, the base field of a Lie algebra depends on the concept of *constants*.

Definition 3.3.6 (Constant)**.** A *constant* ζ (a.k.a. *invariant*) in $\mathbb{R}(Z)$ of a pseudo-derivation δ is an expression that verifies $\delta\zeta = 0$. ⌟

Remark 3.3.7. In this document, while computing a Lie algebra associated to the Lie symmetries of a system, the base field is composed of numerical constants However, in theory, one must extend this base field in order to take into account the informations given by the system of study: algebraic relations for algebraic systems, constants for systems of ODEs and OREs. The explicit computation of such a base field requires to perform operations such as integration and elimination that have an exponential complexity in the worst cases. An implicit definition of the extended base field of such a Lie algebra is provided by the system itself. For example, if the system of study is an algebraic system then the base field of the symmetries Lie algebra must verify also the associated algebraic relations. If the system is a system of ODEs or OREs then the constants of the associated pseudo-derivations must be considered. ⌟

The utility of this extension on the computations can be seen in the following example.

Example 3.3.8. Let us look at the following system of ODEs:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = a\,x, \quad \frac{\mathrm{d}a}{\mathrm{d}t} = 0 \tag{3.25}$$

and the associated infinitesimal generator $\delta_{\mathcal{D}} = \partial/\partial t + ax\partial/\partial x$. Consider now, the Lie algebra $\mathfrak{g}$ generated by the infinitesimal generators $\delta_{\mathcal{D}}, \partial/\partial t, x\partial/\partial x$. Remark that the infinitesimal generators $\partial/\partial t$ and $a\partial/\partial t$ are both in $\mathfrak{g}$. They are independent w.r.t. $\mathbb{R}$ but dependent w.r.t. the extended base field $\mathbb{R}(a)$ meaning that they define the same vector field because a is a constant i.e. $\delta_{\mathcal{D}} a = \partial a/\partial t = x\partial a/\partial x = 0$. ⌟

Definition 3.3.9 (Lie Subalgebra)**.** A vector space $\tilde{\mathfrak{g}} \subset \mathfrak{g}$ is called a *Lie subalgebra* of $\mathfrak{g}$ if for any (δ_1, δ_2) in $\tilde{\mathfrak{g}}^2$, $[\delta_1, \delta_2]$ is also in $\tilde{\mathfrak{g}}$. ⌟

Definition 3.3.10 (Commutative Lie Algebra)**.** A Lie algebra generated by the basis of infinitesimal generators $\mathcal{B} = (\delta_1, \ldots, \delta_r)$ is *commutative* (a.k.a. *abelian*) if, and only if, for all i, j in $\{1, \ldots, r\}$ the relation $[\delta_i, \delta_j] = 0$ holds. ⌟

3.3.2 Structure Constants of Lie algebras

The *structure constants* of a Lie algebra specify its structure (see ch. 9 of [39]). In the sequel, we give their definition and we present their computation in our framework for a given Lie algebra of infinitesimal generator.

Definition 3.3.11 (Structure Constants – Second Fundamental Theorem of Lie, see th. 2.4.2-1 in § 2.4.2 of [6])**.** Let $\mathfrak{g}$ be a finite-dimensional Lie algebra of infinitesimal generator and $\mathcal{B} = \{\delta_1, \ldots, \delta_r\}$ be one of its basis. The Lie bracket of any two infinitesimal generators of $\mathcal{B}$ is also an infinitesimal generator in $\mathfrak{g}$, in particular

$$[\delta_i, \delta_j] = \sum_{k=1}^{r} c_{ij}^k \delta_k \tag{3.26}$$

where the coefficients c_{ij}^k, that are in the base field of $\mathfrak{g}$, are called *structure constants*. According to the third fundamental theorem of Lie (see th. 2.4.2-2 in § 2.4.2 of [6]), the following constraints exist on these structure constants, for all i, j and k in $\{1, \ldots, r\}$:

- the skew-symmetry implies $c_{ij}^k = -c_{ji}^k$ and in particular $c_{ii}^k = 0$;
- the Jacobi identity implies for all m in $\{1, \ldots, r\}$:

$$\sum_{k=1}^{r} \left(c_{ij}^k c_{kl}^m + c_{li}^k c_{kj}^m + c_{jl}^k c_{ki}^m \right) = 0. \tag{3.27}$$

⌟

Remark 3.3.12. The structure constants of a Lie algebra $\mathfrak{g}$ permit to construct the associated *commutator series* from which its decomposition can be guessed (see § 3.3.3 page 73). ⌟

Any set of constants c_{ij}^k that satisfy above properties are the structure constants for some Lie algebra (see § 1.4 of [86]). Furthermore, if one chooses a new basis for a Lie algebra then, in general, the structure constants change but they always remain related to the old ones (see equation (1.45) in [86]). Let us denote this new basis by $\bar{\mathcal{B}} = \left(\bar{\delta}_1, \ldots, \bar{\delta}_r\right)$. If $\bar{\delta}_i = \sum_{j=1}^r a_{ij}\delta_j$ then

$$\bar{c}_{ij}^k = \sum_{l=1}^{r} \sum_{m=1}^{r} \sum_{n=1}^{r} a_{il}\, a_{jm}\, b_{nk}\, c_{lm}^n \tag{3.28}$$

where (b_{ij}) is the inverse matrix to (a_{ij}).

The most convenient way to display the structure constants of a given Lie algebra is to write them in a tabular form. A *commutator table* is a $r \times r$ table for a Lie algebra basis $\mathcal{B} = \{\delta_1, \ldots, \delta_r\}$ of dimension r where (i,j)th entry expresses the Lie bracket $[\delta_i, \delta_j]$. The coefficient of the infinitesimal generator δ_k in the (i,j)th table entry is the structure constant c_{ij}^k.

The following examples illustrate on the one hand some Lie algebras and on the other hand their commutator table thus their structure constants.

Example 3.3.13. Let us consider the Lie algebra composed by the 4 infinitesimal generators:

$$\begin{cases} \delta_1 = \dfrac{\partial}{\partial t}, \quad \delta_2 = -x\dfrac{\partial}{\partial x} + b\dfrac{\partial}{\partial b}, \\ \delta_3 = t\dfrac{\partial}{\partial t} - x\dfrac{\partial}{\partial x} - a\dfrac{\partial}{\partial a}, \quad \delta_4 = -x^2\dfrac{\partial}{\partial x} + a\dfrac{\partial}{\partial b}. \end{cases} \tag{3.29}$$

This algebra corresponds to some Lie point symmetries (see lemma 4.1.14 page 92) of the system of ODEs given in the example 3.2.6 page 63. The commutator table of this Lie algebra is shown below.

	$\boldsymbol{\delta_1}$	$\boldsymbol{\delta_2}$	$\boldsymbol{\delta_3}$	$\boldsymbol{\delta_4}$
$\boldsymbol{\delta_1}$	0	0	δ_1	0
$\boldsymbol{\delta_2}$	0	0	0	$-\delta_4$
$\boldsymbol{\delta_3}$	$-\delta_1$	0	0	$-\delta_4$
$\boldsymbol{\delta_4}$	0	δ_4	δ_4	0

(3.30)

The structure constants are $c_{13}^1 = c_{42}^4 = c_{43}^4 = 1$ and $c_{31}^1 = c_{24}^4 = c_{34}^4 = -1$ with all other elements c_{ij}^k are being zero. ⌟

Example 3.3.14. Now, let us see a second example of a commutator table of a Lie algebra composed by the 4 infinitesimal generators enumerated below:

$$\begin{cases} \delta_1 = (b-y)\,\dfrac{\partial}{\partial a} + (x-a)\,\dfrac{\partial}{\partial b} - y\dfrac{\partial}{\partial x} + x\dfrac{\partial}{\partial y}, \\ \delta_2 = (-2\,b+y)\,\dfrac{\partial}{\partial b} + (2\,a-x)\,\dfrac{\partial}{\partial x} + (-2\,b+y)\,\dfrac{\partial}{\partial y}, \\ \delta_3 = (-a+2\,x)\,\dfrac{\partial}{\partial a} + 3\,b\dfrac{\partial}{\partial b} + 3\,\ell\dfrac{\partial}{\partial \ell} + (-4\,a+5\,x)\,\dfrac{\partial}{\partial x} + (4\,b+y)\,\dfrac{\partial}{\partial y}, \\ \delta_4 = -a\dfrac{\partial}{\partial a} - b\dfrac{\partial}{\partial b} - \ell\dfrac{\partial}{\partial \ell} - x\dfrac{\partial}{\partial x} - y\dfrac{\partial}{\partial y}\cdot \end{cases} \tag{3.31}$$

The commutator table of this Lie algebra follows.

	$\boldsymbol{\delta_1}$	$\boldsymbol{\delta_2}$	$\boldsymbol{\delta_3}$	$\boldsymbol{\delta_4}$
$\boldsymbol{\delta_1}$	0	0	0	0
$\boldsymbol{\delta_2}$	0	0	$2\delta_2 - \delta_3 - 3\delta_4$	0
$\boldsymbol{\delta_3}$	0	$-2\delta_2 + \delta_3 + 3\delta_4$	0	0
$\boldsymbol{\delta_4}$	0	0	0	0

(3.32)

The structure constants are $c_{23}^2 = -c_{32}^2 = 2, c_{23}^3 = -c_{32}^3 = 1$ and $c_{23}^4 = -c_{32}^4 = 3$ with all other c_{ij}^k are being zero. ⌟

Remark 3.3.15. Under some conditions, structure constants are invariants (see prop. 4.1 in [114]). ⌟

Computation of Structure Constants of a Given Lie Algebra of Infinitesimal Generator

Here, some classical computational strategies concerning the structure constants are presented. The goal is to find explicitly the values of the structure constants of a given Lie algebra of infinitesimal generator knowing its basis.

Thanks to the definition of the structure constants (3.26) page 70 and to the fact that the Lie bracket of two infinitesimal generators of a Lie algebra is again an infinitesimal generator in the same algebra, one can write:

$$\delta_k = \sum_{h=1}^{n} \xi_{z_h}^k \frac{\partial}{\partial z_h}, \qquad [\delta_i, \delta_j] = \sum_{k=1}^{r} c_{ij}^k \delta_k = \sum_{k=1}^{r} c_{ij}^k \left(\sum_{h=1}^{n} \xi_{z_h}^k \frac{\partial}{\partial z_h} \right) = \sum_{h=1}^{n} \xi_{z_h}^{ij} \frac{\partial}{\partial z_h}\cdot \tag{3.33}$$

All the coefficients $\xi_{z_h}^k$ and $\xi_{z_h}^{ij}$ in the above expressions are in $\mathbb{R}(Z)$ and they are either known or can easily be computed. As a result, the following matrix form is deduced:

$$\begin{pmatrix} \xi_{z_1}^1 & \cdots & \xi_{z_1}^r \\ \vdots & & \vdots \\ \xi_{z_n}^1 & \cdots & \xi_{z_n}^r \end{pmatrix} \begin{pmatrix} c_{ij}^1 \\ \vdots \\ c_{ij}^r \end{pmatrix} = \begin{pmatrix} \xi_{z_1}^{ij} \\ \vdots \\ \xi_{z_n}^{ij} \end{pmatrix} \quad \Leftrightarrow \quad M\,C_{ij} = N_{ij}. \tag{3.34}$$

The resolution of this linear system leads to the structure constants associated to δ_i and δ_j (details of a similar computation can be found in the algorithm 4 page 99 where the indeterminates must be replaced by the structure constants).

Remark 3.3.16. Since the infinitesimal generators used in the above expressions form a basis, they are considered to be linearly independent. ⌟

Remark 3.3.17. Computing the structure constants of a Lie algebra in $\mathbb{R}(Z)$ requires the explicit knowledge of the associated base field. In practice such computation is unavailable due to the classical computer algebra limitations. That is why in the `ExpandedLiePointSymmetry` package, at first the structure constants are computed in $\mathbb{R}$ but not in $\mathbb{R}(Z)$ (see § 3.3.5 page 76). If a structure constant is not in $\mathbb{R}$ then the associated algorithm tries to compute them in $\mathbb{R}(Z)$ by using formal computations with a higher computational complexity. ⌟

3.3.3 Lie Algebra Structure

This subsection is devoted to the Lie algebra structure, especially to the notion of *solvability*. This notion is introduced to ensure the usage of as much as possible infinitesimal generators of a Lie algebra by the exact simplification algorithms (see ch. 5 page 103). First, let us define *commutator series* and then *solvable* Lie algebras.

Definition 3.3.18 (Commutator Series)**.** The *commutator series* (also called *derived series*) of a Lie algebra $\mathfrak{g}$ is the sequence of the ideals $\mathfrak{g}^{(i)}$ (also called *derived* ideals) inductively defined by:

$$\begin{cases} \mathfrak{g}^{(0)} = \mathfrak{g}, \\ \mathfrak{g}^{(i+1)} = \left[\mathfrak{g}^{(i)}, \mathfrak{g}^{(i)}\right] \end{cases} \tag{3.35}$$

where $[\mathfrak{g}_1, \mathfrak{g}_2]$ means the linear span of the elements of the form $[\delta_1, \delta_2]$ in the base field of $\mathfrak{g}$ where δ_1 is in $\mathfrak{g}_1$ and δ_2 is in $\mathfrak{g}_2$. This sequence of subspaces is always decreasing w.r.t. the inclusion and the dimension. Also, there exists κ in $\mathbb{N}$ such that $\mathfrak{g}^{(\kappa)} = \mathfrak{g}^{(\kappa+1)}$ (see § 2.5 of [86]):

$$\mathfrak{g} = \mathfrak{g}^{(0)} \supset \mathfrak{g}^{(1)} \supset \cdots \supset \mathfrak{g}^{(\kappa-1)} \supset \mathfrak{g}^{(\kappa)}. \tag{3.36}$$

⌟

These commutator series are related to the non-commutativity of Lie algebras as shown by the following proposition.

Proposition 3.3.19. *With the above notations, if* $\mathfrak{g}^{(i+1)} = \left[\mathfrak{g}^{(i)}, \mathfrak{g}^{(i)}\right]$ *for all* i *then the quotient algebra* $\mathfrak{g}^{(i)}/\mathfrak{g}^{(i+1)}$ *is commutative.* ⌟

Proof. This proof is due to [90]. Let us consider the projection $p : \mathfrak{g}^{(i)} \to \mathfrak{g}^{(i)}/\mathfrak{g}^{(i+1)}$ which is a morphism of Lie algebras. By definition, p commutes with the Lie bracket. For δ_1 and δ_2 in $\mathfrak{g}^{(i)}$, $p(\delta_1)$ and $p(\delta_2)$ are two elements of $\mathfrak{g}^{(i)}/\mathfrak{g}^{(i+1)}$ that satisfy:

$$[p(\delta_1), p(\delta_2)] = p([\delta_1, \delta_2]). \tag{3.37}$$

In addition, by definition, $[\delta_1, \delta_2]$ is in $\mathfrak{g}^{(i+1)}$ so $p([\delta_1, \delta_2]) = 0$. That implies also that one has $[p(\delta_1), p(\delta_2)] = 0$. So the quotient algebra $\mathfrak{g}^{(i)}/\mathfrak{g}^{(i+1)}$ is commutative. ⌟

Definition 3.3.20 (Solvable Lie Algebra)**.** A Lie algebra $\mathfrak{g}$ is called *solvable* when its commutator series vanishes for some κ in $\mathbb{N}$. In other words, $\mathfrak{g}$ is solvable if there exists κ in $\mathbb{N}$ and a chain of subalgebras such that:

$$\mathfrak{g} = \mathfrak{g}^{(0)} \supset \mathfrak{g}^{(1)} \supset \cdots \supset \mathfrak{g}^{(\kappa-1)} \supset \mathfrak{g}^{(\kappa)} = \{0\}\,. \tag{3.38}$$

⌟

Remark 3.3.21. The solvability is equivalent to the existence of a basis $\mathcal{B} = \{\delta_1, \ldots, \delta_r\}$ of $\mathfrak{g}$ (see § 2.5/Solvable Groups of [86]) such that:

$$[\delta_i, \delta_j] = \sum_{k=1}^{j-1} c_{ij}^k \delta_k \quad \text{whenever} \quad i < j. \tag{3.39}$$

⌟

Remark 3.3.22. Every commutative Lie algebra is solvable. ⌟

Example 3.3.23. The Lie algebra $\mathfrak{g}$ composed by the 4 infinitesimal generators given in (3.29) page 71 is a solvable Lie algebra because one has:

$$\mathfrak{g}^{(0)} = \operatorname{span}(\delta_1, \delta_2, \delta_3, \delta_4) \supset \mathfrak{g}^{(1)} = \operatorname{span}(\delta_1, \delta_4) \supset \mathfrak{g}^{(2)} = \{0\}\,. \tag{3.40}$$

It is obvious that $\mathfrak{g}^{(0)} = \mathfrak{g}$ is generated by the infinitesimal generators $\delta_1, \delta_2, \delta_3$ and δ_4. One can see from the commutator table that $\mathfrak{g}^{(1)} = \left[\mathfrak{g}^{(0)}, \mathfrak{g}^{(0)}\right]$ is generated only by δ_1 and δ_4. Again the same commutator table let us deduce that $\mathfrak{g}^{(2)}$ is an empty set, thus $\mathfrak{g}$ is solvable. ⌟

Example 3.3.24. The Lie algebra $\mathfrak{g}$ composed by the 4 infinitesimal generators given in (3.31) page 72 is not a solvable Lie algebra because one has:

$$\mathfrak{g}^{(0)} = \operatorname{span}(\delta_1, \delta_2, \delta_3, \delta_4) \supset \mathfrak{g}^{(1)} = \operatorname{span}(\delta_2, \delta_3, \delta_4)\,. \tag{3.41}$$

It is obvious that $\mathfrak{g}^{(0)} = \mathfrak{g}$ is generated by $\delta_1, \delta_2, \delta_3$ and δ_4. From the commutator table, one can deduce that $\mathfrak{g}^{(1)} = \left[\mathfrak{g}^{(0)}, \mathfrak{g}^{(0)}\right]$ is generated by δ_2, δ_3 and δ_4. Again the same commutator table tells that for any κ in $\mathbb{N}$ greater then 1, $\mathfrak{g}^{(\kappa)}$ is equal to $\mathfrak{g}^{(1)}$. Thus, there is no an empty subalgebra at the end of the chain of derived subalgebras given in (3.38). ⌟

Knowing the basis of a solvable Lie algebra and its structure constants lead to the associated commutator series. For the reduction algorithm given in ch. 5 page 103 to work efficiently, one actually may need this decomposition. In fact, the required order of the reduction process can be found using these subalgebras. In the case of a solvable Lie algebra, one can construct the commutator series to find an appropriate order of its infinitesimal generators. If the Lie algebra is not solvable, then the Levi decomposition (see [31, 30] and see references therein for more detailed information) may be used.

Remark 3.3.25. The solvability of Lie algebras yields the required order of infinitesimal generators for the reduction process if the infinitesimal generators of $\mathfrak{g}^{(i)}$ are Lie point

symmetries of the infinitesimal generators of $\mathfrak{g}^{(i+1)}$ (see ch. 4 page 87 for more details on symmetries).

Sometimes, this property may not be satisfied. In this case, one can try to compute a new basis in order to restore this property. For example, let us consider a Lie algebra $\mathfrak{g}$ generated by 3 infinitesimal generators δ_1, δ_2 and δ_3 with the following commutator table:

	$\boldsymbol{\delta_1}$	$\boldsymbol{\delta_2}$	$\boldsymbol{\delta_3}$
$\boldsymbol{\delta_1}$	0	0	δ_2
$\boldsymbol{\delta_2}$	0	0	δ_1
$\boldsymbol{\delta_3}$	$-\delta_2$	$-\delta_1$	0

(3.42)

One can deduce the associated commutator series:

$$\mathfrak{g}^{(0)} = \text{span}\,(\delta_1, \delta_2, \delta_3) \supset \mathfrak{g}^{(1)} = \text{span}\,(\delta_1, \delta_2) \supset \mathfrak{g}^{(2)} = \{0\}\,. \tag{3.43}$$

Remark that δ_3 is neither a symmetry of δ_1 nor δ_2 (i.e. $[\delta_3, \delta_1] = \lambda\,\delta_1$ or $[\delta_3, \delta_2] = \lambda\,\delta_2$ with any function λ are not satisfied). The following new basis remedies this situation:

$$\bar{\delta}_1 = \delta_1 + \delta_2, \quad \bar{\delta}_2 = \delta_1 - \delta_2, \quad \bar{\delta}_3 = \delta_3. \tag{3.44}$$

In this case, the associated commutator table is of the form:

	$\bar{\boldsymbol{\delta}}_{\boldsymbol{1}}$	$\bar{\boldsymbol{\delta}}_{\boldsymbol{2}}$	$\bar{\boldsymbol{\delta}}_{\boldsymbol{3}}$
$\bar{\boldsymbol{\delta}}_{\boldsymbol{1}}$	0	0	$\bar{\delta}_1$
$\bar{\boldsymbol{\delta}}_{\boldsymbol{2}}$	0	0	$-\bar{\delta}_2$
$\bar{\boldsymbol{\delta}}_{\boldsymbol{3}}$	$-\bar{\delta}_1$	$\bar{\delta}_2$	0

(3.45)

The associated commutator series follows:

$$\mathfrak{g}^{(0)} = \text{span}\left(\bar{\delta}_1, \bar{\delta}_2, \bar{\delta}_3\right) \supset \mathfrak{g}^{(1)} = \text{span}\left(\bar{\delta}_1, \bar{\delta}_2\right) \supset \mathfrak{g}^{(2)} = \{0\}\,. \tag{3.46}$$

In addition, $\bar{\delta}_3$ is a symmetry of $\bar{\delta}_1$ and $\bar{\delta}_2$. ⌟

3.3.4 Properties of Lie Algebras of Scalings

In the `MABSys` package, the simplification algorithms are performed by using Lie algebras of scalings. One of the advantages of this type of algebras is their solvability properties that facilitate the computations.

Lemma 3.3.26. *If $\mathfrak{g}$ is a Lie algebra generated by a basis of scalings defined on a unique coordinate set $Z = (z_1, \ldots, z_n)$ then $\mathfrak{g}$ is a* commutative *thus* solvable *Lie algebra.* ⌟

Proof. Let δ_1 and δ_2 be two scalings of $\mathfrak{g}$ defined on Z as follows:

$$\delta_1 = \sum_{i=1}^{n} \alpha_i\, z_i \frac{\partial}{\partial z_i} \quad \text{and} \quad \delta_2 = \sum_{i=1}^{n} \beta_i\, z_i \frac{\partial}{\partial z_i} \tag{3.47}$$

where α_i and β_i are in $\mathbb{R}$. By definition, the Lie bracket of such two infinitesimal generators vanishes as shown below:

$$\begin{aligned}[\delta_1, \delta_2] &= \delta_1 \delta_2 - \delta_2 \delta_1, \\ &= \left(\alpha_1\beta_1 z_1 \frac{\partial}{\partial z_1} + \cdots + \alpha_n\beta_n z_n \frac{\partial}{\partial z_n}\right) - \left(\beta_1\alpha_1 z_1 \frac{\partial}{\partial z_1} + \cdots + \beta_n\alpha_n z_n \frac{\partial}{\partial z_n}\right), \\ &= 0.\end{aligned} \tag{3.48}$$

That implies that $\mathfrak{g}$ is commutative. Also, according to definition 3.3.20 page 74, $\mathfrak{g}$ is a solvable Lie algebra. ⌟

Remark 3.3.27. Indeed, the lemma 3.3.26 page 75 tells that in the reduction and the reparametrization algorithms presented in chapter 2 page 33, the order in which the scalings were used does not influence the results. That is why, we treat all the scalings at the same time using the associated matrix of scaling. ⌟

3.3.5 Implementation of Lie Algebras

Lie algebras, treated in this document, are represented by the `LieAlgebra` data structure. Here we detail this data structure and its computation.

The constructor `NewLieAlgebra` creates such an object that is defined by the key word `LieAlgebra` and the following elements:

- a basis given by a list of infinitesimal generators defined on the same coordinate set;
- associated structure constants stored in a table that can be printed by the function `PrintCommutationTable` (see § 3.3.2 page 70);
- associated base field of the Lie algebra (see § 3.3.1 page 69). The default value is an empty list defining the numerical constant field;
- a short description of the Lie algebra structure encoded by a composition of unevaluated functions. By default, the description is equal to `LieAlgebraOfDim`(d) where d is the dimension of the considered Lie algebra. It can be computed more precisely by the `LieAlgebraStructure` function (see § 3.3.3 page 73).

Example 3.3.28. Let us illustrate how to construct a Lie algebra data structure from the 4 independent infinitesimal generators given in the example 3.3.13 page 71.

```
> # Infinitesimal generators
> Generators := [NewInfinitesimalGenerator([1,0,0,0],[t,x,a,b]),
>                NewInfinitesimalGenerator([0,-x,0,b],[t,x,a,b]),
>                NewInfinitesimalGenerator([-t,x,a,0],[t,x,a,b]),
>                NewInfinitesimalGenerator([0,-x^2,0,a],[t,x,a,b])];
                          d              /d   \     /d   \
      Generators := [F -> -- F, F -> -x |-- F| + b |-- F|,
                          dt             \dx  /     \db  /
                             /d   \     /d   \     /d   \
                     F -> -t |-- F| + x |-- F| + a |-- F|,
                             \dt  /     \dx  /     \da  /
```

```
                              2 /d   \     /d   \
                      F -> -x  |-- F| + a |-- F|]
                               \dx  /     \db  /

> # Creation of a Lie algebra
> LieAlg := NewLieAlgebra(Generators);
       LieAlg := LieAlgebraOfDim(4)

> # Type check
> type(LieAlg,LieAlgebra);
       true
```

By definition, a Lie algebra $\mathfrak{g}$ is stable w.r.t. the Lie bracket which implies that the Lie bracket of any two infinitesimal generators in $\mathfrak{g}$ is also an infinitesimal generator in $\mathfrak{g}$. For example, the Lie bracket of δ_2 and δ_4 is equal to $-\delta_4$.

```
> LieBracket(Generators[2],Generators[4]);
             2 /d   \     /d   \
       F -> x  |-- F| - a |-- F|
               \dx  /     \db  /
```

Some arithmetic functions are also available for infinitesimal generators.

```
> AddInfinitesimalGenerator(Generators[1],Generators[2]);
            /d   \     /d   \     /d   \
       F -> |-- F| - x |-- F| + b |-- F|
            \dt  /     \dx  /     \db  /

> MultiplyInfinitesimalGenerator(-1,Generators[4]);
             2 /d   \     /d   \
       F -> x  |-- F| - a |-- F|
               \dx  /     \db  /
```

The associated commutator table can be computed as follows where $\_n$ stands for n^{th} infinitesimal generator:

```
> # Computation of commutator table
> PrintCommutatorTable(LieAlg);
       [0     0     -_1     0 ]
       [                      ]
       [0     0      0     -_4]
       [                      ]
       [_1    0      0     _4 ]
       [                      ]
       [0     _4    -_4     0 ]
```

This Lie algebra is solvable (see example 3.3.23 page 74).

```
> # Computation of Lie algebra structure
> LieAlgBis := LieAlgebraStructure(LieAlg);
       LieAlgBis := SemiDirectSum(AbelianLieAlgebraOfDim(2),
                                  AbelianLieAlgebraOfDim(2))

> # Infinitesimal generators of the Lie algebra
> GeneratorsOf(LieAlgBis);
             d            2 /d   \     /d   \          /d   \     /d   \
       [F -> -- F, F -> -x  |-- F| + a |-- F|, F -> -x |-- F| + b |-- F|,
             dt             \dx  /     \db  /          \dx  /     \db  /
                /d   \     /d   \     /d   \
        F -> -t |-- F| + x |-- F| + a |-- F|]
                \dt  /     \dx  /     \da  /
```

As one can see, the Lie algebra is composed of semi-direct sum of two commutative (abelian) subalgebras of dimension 2. The function `GeneratorsOf` is used to display infinitesimal generators of a Lie algebra. Remark also that the order of the infinitesimal generators has changed according to the associated commutator series. The infinitesimal generators δ_1 and δ_4 are put at first place because they construct the last subalgebra not null in the commutator series; thus they must be used at the first place in the reduction process. ⌟

Changing the Base Field of a Lie Algebra

In our framework, while computing a Lie algebra associated to the Lie symmetries of a system, first its base field is considered equal to $\mathbb{R}$. Then one must extend this base field according to the system of study (see § 3.3.1 page 69). Unfortunately, because of the limitations of the computer algebra techniques (exponential complexity in the worst cases of the integration and the elimination methods), one can not hope to express explicitly such an extended base field. However, one may do some additional computations in order to take into account, as much as possible, this extended base field as shown below for systems of ODEs.

Let $\delta_{\mathcal{D}}$ be the infinitesimal generator associated to a system of ODEs. Let δ_i and δ_j be two infinitesimal generators in a set $\mathcal{B} := \{\delta_1, \ldots, \delta_r\}$ associated to the Lie algebra composed by the expanded Lie point symmetries of $\delta_{\mathcal{D}}$. If there exists a constant ζ in $\mathbb{R}(Z)$ such that $\delta_i = \zeta\,\delta_j$ then one of the infinitesimal generators can be discarded from $\mathcal{B}$. Similarly if there exists a linear relation in $\mathbb{R}$ between some infinitesimal generators then one can discard one of them. Remark that this method let us remove many dependency relations in the extended base field but it is incomplete. For example, a relation with two no numeric constants ζ_1, ζ_2 such that $\delta_i = \zeta_1\delta_j + \zeta_2\delta_k$ is ignored.

Remark that this approach, or similar ones used for algebraic systems or systems of OREs, can miss some dependency relations. This missing relations produce in our framework a Lie algebra that is generated by a list of infinitesimal generators that is not a basis. This fact does not change the correctness of the further algorithms but it can effect their efficiency. This point must be improved in the future. The following example shows one of the cases where a dependency relation couldn't be taken into account by the present implementation.

Example 3.3.29. The following infinitesimal generators are the Lie symmetries of the algebraic system $f(x, y, r) = x^2 + y^2 - r^2$ computed by the `ExpandedLiePointSymmetry` package:

$$\delta_1 = r\frac{\partial}{\partial y} + y\frac{\partial}{\partial r}, \quad \delta_2 = r\frac{\partial}{\partial x} + x\frac{\partial}{\partial r}, \quad \delta_3 = -x\frac{\partial}{\partial x} - y\frac{\partial}{\partial y} - r\frac{\partial}{\partial r}. \tag{3.49}$$

These three infinitesimal generators are independent in $\mathbb{R}$ that is assumed to be the base field of the Lie algebra generated by them. Considering the information given by the algebraic equation requires to extend this base field. Indeed, for non zero r values, one has:

$$-\frac{y}{r}\delta_1 - \frac{x}{r}\delta_2 = -x\frac{\partial}{\partial x} - y\frac{\partial}{\partial y} - \frac{x^2+y^2}{r}\frac{\partial}{\partial r} \equiv \delta_3 \bmod f \tag{3.50}$$

because $x^2 + y^2 = r^2$. This problem can be remarked by looking at the associated commutator table:

	$\boldsymbol{\delta_1}$	$\boldsymbol{\delta_2}$	$\boldsymbol{\delta_3}$
$\boldsymbol{\delta_1}$	0	$-\frac{x}{r}\delta_1 + \frac{y}{r}\delta_2$	0
$\boldsymbol{\delta_2}$	$\frac{x}{r}\delta_1 - \frac{y}{r}\delta_2$	0	0
$\boldsymbol{\delta_3}$	0	0	0

(3.51)

where the structure constants are not numerical. ⌟

3.4 Group Actions

This subsection aims at discussing geometrical objects defined by systems of ODEs and OREs that are going to be necessary in the next chapters. The Lie symmetry theory deals with "*infinitesimal transformation*". In order to present this notion, we develop first the *group action* and then the concept of a *vector field* on a manifold. We present finally continuous and discrete *dynamical systems*.

3.4.1 Definition of a Group Action

Here, we define a *group action* and the associated *orbits*.

Definition 3.4.1 (Group Action)**.** A *(left-)action of a group* $(G, +)$ on a manifold M (see § 1.1, § 1.2 of [86] and ch. 2 of [75]) is an evolution function:

$$\begin{array}{rccc} \mathcal{D}: & G \times M & \rightarrow & M, \\ & \nu \times Z & \rightarrow & \mathcal{D}(\nu, Z) \end{array} \tag{3.52}$$

such that for all point Z in M:

- $\mathcal{D}(e, Z) = Z$ where e is the neutral element of G;
- for all $(\nu, \hat{\nu})$ in G^2, $\mathcal{D}(\nu, \mathcal{D}(\hat{\nu}, Z)) = \mathcal{D}(\nu + \hat{\nu}, Z)$. ⌟

An orbit is a collection of points formed by a group action.

Definition 3.4.2 (Orbit)**.** An *orbit*, given a group action $\mathcal{D}$ is a set of points passing through a point Z and defined by:

$$\mathrm{Orb}_{(\mathcal{D},Z)} = \{\mathcal{D}(\nu, Z) \mid \nu \in G\}. \tag{3.53}$$

⌟

Definition 3.4.3. Let G be a r-dimensional local group of transformation acting on $\mathbb{R}^n$. The group action $\mathcal{D}$ is called *regular* if for all points Z and Z' in the same orbit, there exists exactly one ν in G such that the relation $\mathcal{D}(\nu, Z) = Z'$ holds. Remark that in that case, G acts locally freely on $\mathbb{R}^n$ and the orbits of $\mathcal{D}$ are all of dimension r (see definition 1.26 in § 1.2 of [86]). ⌟

Hypothesis 3.4.4. *In the sequel, every treated group action is supposed to be regular in order to apply easily the forthcoming theorems about the exact simplification methods (see ch. 5 page 103).* ⌟

3.4.2 Vector Fields on a Manifold

Let us begin by defining *tangent vectors* in a *tangent space* which are used in the *vector field* definition. Then we show how to associate a vector field to a system of ODEs. For more detailed explications on vector fields, see § 1.3 of [86].

Definition 3.4.5 (Tangent Vector)**.** Let $\mathcal{C} : \mathbb{R} \to M$ be a smooth C^∞-curve on a manifold M (identified to $\mathbb{R}^n$) passing through the point $\mathcal{C}(t_0) := Z$. A *tangent vector* w to $\mathcal{C}$, namely a *derivative*, is a vector such that:

$$\mathrm{w} = \frac{\mathrm{d}\mathcal{C}(t)}{\mathrm{d}t}\Big|_{t=t_0}. \tag{3.54}$$

⌟

Definition 3.4.6 (Tangent Space and Bundle)**.** Let M be a manifold. The *tangent space* TM_Z is the set of all tangent vectors to M at Z. The *tangent bundle* TM is the disjoint union of all the tangent spaces i.e. $TM := \coprod_{Z\in M} TM_Z$. ⌟

Definition 3.4.7 (Vector Field)**.** Let M be a manifold. A *vector field* $\mathbf{v}$ on M is a map that assigns to every point Z of M a tangent vector in the tangent space of M at Z i.e. $\mathbf{v} : M \to \mathbb{R}^n$ and $\mathbf{v}(Z)$ is in TM_Z. In local coordinates, we consider vector fields of the form:

$$\begin{array}{rccc} \mathbf{v}: & M & \to & \mathbb{R}^n, \\ & Z = (z_1, \ldots, z_n) & \to & (\xi_{z_1}(Z), \ldots, \xi_{z_n}(Z)) \end{array} \tag{3.55}$$

where ξ_{z_i} is in $\mathbb{R}(Z)$. ⌟

The images ξ_{z_i} show changes regarding to coordinates when one wants to move on a manifold along a given trajectory. Note that the vector field does not depend on the chosen local coordinates, here $Z = (z_1, \ldots, z_n)$.

This vector field may be regarded as a derivation thus it can be defined for systems of ODEs but not OREs (one can associate a pseudo-derivation to a system of OREs but not a derivation). With the notations of definition 3.2.1 page 62, a vector field of a system of ODEs is defined as follows:

$$\begin{array}{rccc} \mathbf{v}: & \mathbb{R}^n & \to & \mathbb{R}^n, \\ & Z = (t, x_1, \ldots, x_k, \theta_1, \ldots, \theta_\ell) & \to & (1, f_1(Z), \ldots, f_k(Z), 0, \ldots, 0) \end{array} \tag{3.56}$$

where f_i are in $\mathbb{R}(Z)$. In this continuous case, a vector field shows the modifications due to the continuous independent variable t in $\mathbb{R}$ by one infinitesimal step. The following example illustrates such a vector field.

Example 3.4.8. The vector field of the system of ODEs given in the example 3.2.6 page 63 may be written as follows:

$$\begin{array}{rccc} \mathbf{v}: & \mathbb{R}^4 & \to & \mathbb{R}^4, \\ & (t, x, a, b) & \to & (1, (a - b\,x)\,x, 0, 0)\,. \end{array} \tag{3.57}$$

⌟

Remark 3.4.9. Vector fields are closely related to infinitesimal generators. ⌟

3.4.3 Dynamical Systems and their Properties

This section is devoted to *continuous* and *discrete dynamical systems* as defined in this document and to their properties.

Definition 3.4.10 (Dynamical System)**.** A *dynamical system* (a.k.a. *flow*) is a one-parameter group action (see definition 3.4.1 page 79). ⌟

Remark 3.4.11. In the sequel, the notation $\mathcal{D}$ designates a dynamical system. ⌟

Remark 3.4.12. A dynamical system associated to a system of ODEs (resp. OREs) is called *continuous* (resp. *discrete*) because its independent variable belongs to the group G that can be identified to $\mathbb{R}$ (resp. is isomorphic to $\mathbb{Z}$). ⌟

The following examples of dynamical systems are obtained by resolving associated systems of ODEs and OREs.

Example 3.4.13. The dynamical system associated to the system of ODEs given in the example 3.2.6 page 63 follows:

$$\begin{array}{rccc} \mathcal{D}: & (\mathbb{R},+)\times\mathbb{R}^4 & \to & \mathbb{R}^4, \\ & \big(\hat{t},(t,x,a,b)\big) & \to & \left(t+\hat{t},\frac{a\,x\,e^{a\,\hat{t}}}{a-\left(1-e^{a\,\hat{t}}\right)b\,x},a,b\right). \end{array} \tag{3.58}$$

The independent variable $\hat{t}$ varies continuously thus the associated group can be identified to $\mathbb{R}$. ⌟

Example 3.4.14. The dynamical system associated to the system of OREs given in the example 3.2.10 page 64 follows:

$$\begin{array}{rccc} \mathcal{D}: & (\mathbb{Z},+)\times\big(\mathbb{Z}\times\mathbb{R}^3\big) & \to & \mathbb{R}^4, \\ & (\hat{\tau},(\tau,x,y,c)) & \to & \left(\tau+\hat{\tau},x\,c^{\hat{\tau}},\frac{c^{\hat{\tau}}-1}{c-1}x+y,c\right). \end{array} \tag{3.59}$$

The time variable $\hat{\tau}$ is discrete thus the associated group is isomorphic to $\mathbb{Z}$. ⌟

Dynamical System Properties

Hypothesis 3.4.15. *Evolution functions of dynamical systems presented in this document are supposed to be at least of differentiability class C^1 w.r.t. the initial conditions Z in $\mathbb{R}^n$. Thus, the following relation holds for ϵ in the neighborhood of 0 in $\mathbb{R}$ and for all H in $\mathbb{R}^n$:*

$$\mathcal{D}(\nu,Z+\epsilon\,H)=\mathcal{D}(\nu,Z)+\epsilon\,\frac{\partial\mathcal{D}(\nu,Z)}{\partial Z}\,H+\mathrm{O}\big(\epsilon^2\big). \tag{3.60}$$

⌟

The following paragraphs treat continuous and discrete dynamical systems properties separately. All the points denoted by Z belong to a smooth manifold M on which the dynamical system is defined.

Continuous Dynamical Systems. Let $\mathcal{D}$ be a continuous dynamical system based on a local Lie group G identified to $\mathbb{R}$ and associated to a system of ODEs. Let $\delta_{\mathcal{D}}$ be the infinitesimal generator associated to this system of ODEs (see § 2.1 of [104]). $\mathcal{D}$ and $\delta_{\mathcal{D}}$ verify the following relation for all ν in G and ϵ in a neighborhood of neutral element e of G:

$$\mathcal{D}(\nu+\epsilon, Z) = \mathcal{D}(\nu, Z) + \epsilon\, \delta_{\mathcal{D}} \mathcal{D}(\nu, Z) + \mathrm{O}\big(\epsilon^2\big)\,. \tag{3.61}$$

In particular, for the neutral element e of G, one can write:

$$\mathcal{D}(e+\epsilon, Z) = Z + \epsilon\, \delta_{\mathcal{D}} Z + \mathrm{O}\big(\epsilon^2\big)\,. \tag{3.62}$$

The derivative of a continuous dynamical system $\mathcal{D}$ w.r.t. the group element ν is the tangent vector at the point $\mathcal{D}(\nu, Z)$ in the associated vector field $\mathbf{v}$ (see first fundamental theorem of Lie, § 2.2.1 of [6]) i.e.:

$$\frac{\mathrm{d}\mathcal{D}}{\mathrm{d}\nu}(\nu, Z) = \mathbf{v}(\mathcal{D}(\nu, Z))\,. \tag{3.63}$$

This relation can be written also in the form:

$$\mathbf{v}(Z) = \delta_{\mathcal{D}} Z = \frac{\partial \mathcal{D}(\nu, Z)}{\partial \nu}\Big|_{\nu=e}. \tag{3.64}$$

These analytic remarks lead to the following algebraic relation in the framework of formal power series:

$$\mathcal{D}(\nu, Z) = e^{\nu \delta_{\mathcal{D}}} Z := \sum_{i\in\mathbb{N}} \frac{\nu^i\, \delta_{\mathcal{D}}^i\, Z}{i!}. \tag{3.65}$$

Remark 3.4.16. A one-dimensional Lie point symmetry is a continuous dynamical system. ⌟

In theory, the action of a one-dimensional group can be calculated explicitly by the exponentiation of the associated infinitesimal generator (see equation (3.65)). Inversely, given a one-parameter group action, one can deduce the associated infinitesimal generator (see equation (3.64)). The following examples illustrate these conversions.

Example 3.4.17. Consider a continuous dynamical system $\mathcal{S}$ generated by the following infinitesimal generator:

$$\delta_{\mathcal{S}} = x\frac{\partial}{\partial x} + y\frac{\partial}{\partial y} \tag{3.66}$$

acting on a coordinate set $Z = (t, x, y, a)$. The group action of $\delta_{\mathcal{S}}$ i.e. $\mathcal{S}(\nu, Z)$ is computed by the following formula:

$$\mathcal{S}(\nu, Z) = \begin{pmatrix} \sum_{i\in\mathbb{N}} \frac{\nu^i\, \delta_{\mathcal{S}}^i t}{i!} \\ \sum_{i\in\mathbb{N}} \frac{\nu^i\, \delta_{\mathcal{S}}^i x}{i!} \\ \sum_{i\in\mathbb{N}} \frac{\nu^i\, \delta_{\mathcal{S}}^i y}{i!} \\ \sum_{i\in\mathbb{N}} \frac{\nu^i\, \delta_{\mathcal{S}}^i a}{i!} \end{pmatrix} = \begin{pmatrix} t + \frac{\nu\, \delta_{\mathcal{S}} t}{1!} + \frac{\nu^2\, \delta_{\mathcal{S}}^2 t}{2!} + \cdots \\ x + \frac{\nu\, \delta_{\mathcal{S}} x}{1!} + \frac{\nu^2\, \delta_{\mathcal{S}}^2 x}{2!} + \cdots \\ y + \frac{\nu\, \delta_{\mathcal{S}} y}{1!} + \frac{\nu^2\, \delta_{\mathcal{S}}^2 y}{2!} + \cdots \\ a + \frac{\nu\, \delta_{\mathcal{S}} a}{1!} + \frac{\nu^2\, \delta_{\mathcal{S}}^2 a}{2!} + \cdots \end{pmatrix} = \begin{pmatrix} t + 0 + 0 + \cdots \\ x + \frac{\nu x}{1!} + \frac{\nu^2 x}{2!} + \cdots \\ y + \frac{\nu y}{1!} + \frac{\nu^2 y}{2!} + \cdots \\ a + 0 + 0 + \cdots \end{pmatrix} = \begin{pmatrix} t \\ x\, e^{\nu} \\ y\, e^{\nu} \\ a \end{pmatrix}. \tag{3.67}$$

Thus, the action of $\mathcal{S}$ on the coordinates Z is:

$$t \to t, \quad x \to \lambda\, x, \quad y \to \lambda\, y, \quad a \to a \tag{3.68}$$

where λ is equal to e^{ν}. As one can see, the infinitesimal generator does not act on t and a so these coordinates are not affected by the associated action. ⌟

Example 3.4.18. Let us consider the system of ODEs:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = t, \quad \frac{\mathrm{d}y}{\mathrm{d}t} = y - 2, \quad \frac{\mathrm{d}a}{\mathrm{d}t} = 0 \tag{3.69}$$

which has following one-parameter ν local Lie group action $\mathcal{S}(\nu, Z)$:

$$t \to t, \quad x \to x\, e^{\nu}, \quad y \to y, \quad a \to a + \nu \tag{3.70}$$

acting on a coordinate set $Z = (t, x, y, a)$. The infinitesimal generator $\delta_{\mathcal{S}}$ associated to this action is computed by the following formula:

$$\delta_{\mathcal{S}} Z = \frac{\partial \mathcal{S}(\nu, Z)}{\partial \nu}\Big|_{\nu=0} = \begin{pmatrix} \frac{\partial t}{\partial \nu}\big|_{\nu=0} \\ \frac{\partial (x\, e^{\nu})}{\partial \nu}\big|_{\nu=0} \\ \frac{\partial y}{\partial \nu}\big|_{\nu=0} \\ \frac{\partial (a+\nu)}{\partial \nu}\big|_{\nu=0} \end{pmatrix} = \begin{pmatrix} 0 \\ x \\ 0 \\ 1 \end{pmatrix} \tag{3.71}$$

which implies that:

$$\delta_{\mathcal{S}} = x\frac{\partial}{\partial x} + a\frac{\partial}{\partial a}\cdot \tag{3.72}$$

⌟

The problem of these conversions is their complexity. When the group action of interest has a simple form as in these examples, the exponentiation or the derivation do not show any difficulties. But one can not compute the group action as easily or even can not compute at all if infinitesimal generator becomes more complicated. This is why the *classical reduction method* (see § 5.2 page 105), that is based on these operations, is not practical. Instead of trying to compute local group actions, the *moving frame based reduction method* uses informations encoded by infinitesimal generators in an implicit way (see § 5.4 page 114). This is the reason why we use systematically implicit representation in our work (see § 3.4.4 page 84).

Discrete Dynamical Systems. Let $\mathcal{D}$ be a discrete dynamical system based on a local group G identified to $\mathbb{Z}$ and associated to a system of OREs with σ as recurrence operator. Let $\delta_{\mathcal{D}}$ be the pseudo-derivation associated to this system of OREs.

The pseudo-derivation $\delta_{\mathcal{D}}$ is related to the discrete dynamical system $\mathcal{D}$ for all ν in G with the neutral element e by:

$$\delta_{\mathcal{D}} Z = (\mathcal{D}(\nu + 1, Z) - \mathcal{D}(\nu, Z))\Big|_{\nu=e}. \tag{3.73}$$

Furthermore, the discrete exponential map is defined, for all ν in G, by iteration of the recurrence operator σ:

$$\mathcal{D}(\nu, Z) = \sigma^{[\nu]} Z \quad \text{with} \quad \sigma^{[\nu]} = \sigma^{[\nu-1]} \circ \sigma, \quad \sigma^{[0]} = \mathrm{I} \tag{3.74}$$

where I is the identity function.

3.4.4 Dynamical Systems: Pseudo-Derivation versus Evolution Function

Until now we talked about two different ways of representing systems of ODEs and OREs. The first one is an *implicit* form based on the associated pseudo-derivations (see definitions 3.2.14 page 64 and 3.2.16 page 65). The second one is an *explicit* form based on the associated evolution function (see definition 3.4.10 page 81).

Remark 3.4.19. In theory, the integration of a pseudo-derivation expressions gives the evolution functions of dynamical systems. Unfortunately, there is no general recipe for integrating such systems. Furthermore, in my case, the implicit representations are already the input of our algorithms. For these reasons, we prefer not to make extra computations to reach the explicit representations. All the results that we are looking for may be found by the implicit version. In the sequel, the use of the explicit form is avoided for the computations. So, instead of evolution functions, the pseudo-derivations are chosen. ⌟

3.4.5 Invariants Definitions and Properties

This section is devoted to *invariants* that are important for understanding the forthcoming exact simplification processes. We are not interested in their computation but one needs to know them to seize main ideas.

An *invariant*, roughly speaking, is an element that does not change under a transformation. They can be seen as constants (see definition 3.3.6 page 69). There exist effective computations of invariants in polynomial or differential cases (see [59, 73]). These computations (see [57, 74]) do not have polynomial complexities.

Proposition 3.4.20 (Invariant Function, see prop. 2.6 of § 2.1 in [86])**.** *Let G be an r-dimensional Lie group acting on $\mathbb{R}^n$ and $\{\delta_1, \ldots, \delta_r\}$ a basis of infinitesimal generators of the associated Lie algebra $\mathfrak{g}$. A smooth real-valued function ζ is an invariant function for G if, and only if, the relations $\delta_j \zeta = 0$ hold for all Z in $\mathbb{R}^n$ and j in $\{1, \ldots, r\}$. In other words, if it is a solution to the following first-order system of PDEs (using the notation of the equation (3.13) page 64):*

$$\delta_j \zeta = \sum_{i=1}^{n} \xi_{z_i}^j(Z) \, \frac{\partial \zeta(Z)}{\partial z_i} = 0 \quad \forall j \in \{1, \ldots, r\} \tag{3.75}$$

where $\xi_{z_i}^j$ and ζ are in $\mathbb{Q}(Z)$. ⌟

The invariants $\zeta_1, \ldots, \zeta_s$ are *functionally independent* if there is no a smooth real-valued function F such that $F(\zeta_1, \ldots, \zeta_s) = 0$. The classical necessary and sufficient condition for these invariants to be functionally independent is that their $s \times n$ Jacobian matrix $(\partial\zeta_j/\partial z_i)$ is of rank s everywhere (see paragraph "Invariants and Functional Dependence" in § 2.1 of [86]). The following theorem states the exact number of functionally independent local invariants associated to a local group action.

Theorem 3.4.21 (see th. 2.17 of § 2.1 in [86])**.** *Let a local transformation group G act regularly on the n-dimensional manifold $\mathbb{R}^n$ with r-dimensional orbits. If $Z \in \mathbb{R}^n$, then there exists precisely $s := n - r$ functionally independent local invariants $\zeta_1, \ldots, \zeta_s$ defined in a neighborhood of Z.* ⌟

Remark 3.4.22. These invariants constitute a new coordinate chart. The procedure of writing a system in such a new coordinate chart is called "reduction" (see § 5.2 page 105).

Remark 3.4.23. In particular, when one manipulates a one-dimensional Lie algebra associated to a symmetry group acting on a n-dimensional manifold, according to the theorem 3.4.21, there exist $n - 1$ independent invariants. In practice, these invariants can be deduced from the associated group action. They are not unique and they form a field. ⌟

Example 3.4.24. Let us find three invariants of the group action (3.68) of the example 3.4.17 page 82. The associated infinitesimal generator is given in the equation (3.66) page 82. Because t and a are not affected at all by the group action, they are two easy invariant candidates. Indeed, $\zeta_1 = t, \zeta_2 = a$ are invariants because $\delta_{\mathcal{S}} t = \delta_{\mathcal{S}} a = 0$. The third one, for this simple case, is also straightforward; one can choose $\zeta_3 = x/y$ because $\delta_{\mathcal{S}}(x/y) = 0$. As said before, these invariants are not unique; one could, for example, also choose x^2/y^2. ⌟

4 Lie Point Symmetry

This chapter is entirely devoted to *expanded Lie point symmetries*. First, we define such continuous symmetries for algebraic and dynamical systems. General approaches to compute Lie symmetries require resolution of systems of PDEs. Here, we present some computational strategies to reach an algorithm of polynomial complexity in the input size for computation of a large set of symmetries.

Roughly speaking, a *Lie point symmetry* of a system is a transformation that maps every solution of the system to another solution of the same system. In other words, it maps the solution set of the system to itself. The properties of these symmetries are at the heart of the reduction and the reparametrization algorithms (see ch. 5 page 103). The literature for this theory can be found, among others, in [86, 87, 6, 104, 117, 70, 101].

Definition 4.0.25 (Symmetry Group of Geometrical Objects)**.** Let $\mathcal{S}$ be a continuous dynamical system defined by a local Lie group G acting on a manifold M. A subset $\mathcal{U} \subset M$ is called G-invariant and G is called a *symmetry group* of $\mathcal{U}$, if whenever $Z \in \mathcal{U}$ and $\nu \in G$ such that $\mathcal{S}(\nu, Z)$ is defined, then $\mathcal{S}(\nu, Z) \in \mathcal{U}$. ⌟

There are many kinds of symmetries that are not treated in this document. For example, *contact transformation* let coefficients of the transformations infinitesimal generator depend also on first derivatives of the coordinates or *Lie-Bäcklund transformation* let them involve derivatives up to an arbitrary order. For Lie point symmetries, the coefficients of the infinitesimal generators depend only on the coordinates Z and, in this document, they are assumed to be in $\mathbb{Q}[Z]$.

There exist many softwares on this subject (see ch. 17 of [53]). These procedures do not respond to challenges of this document. For example, the package `liesymm` of `Maple` provides some Lie symmetry methods for PDEs (see [23]). It manipulates integration of determining systems and also differential forms. Despite its success on little systems, it must be remarked that its integration capabilities for solving determining systems automatically are limited by complexity issues. The `DEtools` package uses the prolongation of vector fields for searching Lie symmetries of ODEs. Indeed, the determining system to be solved is a system of PDEs. So finding Lie symmetries for ODEs, in this general case, could be as complicated as solving the original system. We recall that Lie symmetries are used to simplify these systems of ODEs and thus we are going to avoid these methods.

4.1 Definition of a Lie Point Symmetry

This section presents the definitions of *expanded Lie point symmetries* of algebraic systems as well as dynamical ones. We work in an *expanded* space meaning that we avoid, as much as possible, the distinction between independent variable, state variables and parameters (see [22] for another application of this standpoint). Nevertheless, for dynamical systems, we analyze Lie point symmetries acting and not acting on the independent variable separately. Lie point symmetries not acting on the independent variable can be handled by a unique formula (see lemma 4.1.6 page 91) for both continuous and discrete dynamical systems. Lie point symmetries acting on the independent variable are just studied for continuous dynamical systems (see remark 4.1.10 page 91).

4.1.1 Symmetries of Algebraic Systems

In this subsection, we give the definition of *Lie symmetries of algebraic systems* along with necessary and sufficient conditions to compute them.

A *symmetry group of an algebraic system* is a continuous dynamical system defined on a local Lie group G acting on a manifold M. G transforms solutions of this algebraic system to its other solutions (see § 2.1 of [86]).

The following theorem (see th. 2.8 in ch. 2, § "Infinitesimal Invariance" of [87]) gives necessary and sufficient conditions so that a local Lie group G is a symmetry group of an algebraic system.

Theorem 4.1.1. *Let G be a connected local Lie group of a continuous dynamical system acting on the n-dimensional manifold $M = \mathbb{R}^n$. Let $F : \mathbb{R}^n \to \mathbb{R}^k$ with $k \leq n$ define a regular system of algebraic equations (see definition 3.1.1 page 59):*

$$f_i(Z) = 0 \quad \forall i \in \{1, \ldots, k\}\,.$$

Then G is a symmetry group *of this algebraic system if, and only if,*

$$\delta f_i(Z) = 0 \quad \forall i \in \{1, \ldots, k\} \text{ whenever } f_1(Z) = \cdots = f_k(Z) = 0 \tag{4.1}$$

for every infinitesimal generator δ in the Lie algebra $\mathfrak{g}$ of G. ⌟

Example 4.1.2. Let us consider the algebraic system (see example 3.1.2 page 59) defined on $Z = (P, Q, a, b, c, \ell)$ with:

$$\begin{cases} f_1(Z) = (1 - c\,P) + b\,Q + 1, \\ f_1(Z) = a\,P - \ell\,Q + 1. \end{cases} \tag{4.2}$$

The infinitesimal generator

$$\delta_{\mathcal{S}} = a\,(a-1)\,\frac{\partial}{\partial a} + (\ell + b)\,\frac{\partial}{\partial b} + (2\,a\,c - c)\,\frac{\partial}{\partial c} + (-a\,P + P)\,\frac{\partial}{\partial P} \tag{4.3}$$

is associated to one of its symmetries. It acts on 4 variables, namely a, b, c and P. One can easily verify that $\delta f_1 = f_1 - f_2$ and $\delta f_2 = 0$. Thus the relations $\delta f_1 = \delta f_2 = 0$ are satisfied for any Z in $\mathbb{R}^n$ that vanishes the algebraic system. ⌟

Remark 4.1.3. In the sequel we restrict the set of symmetries defined by this theorem in order to avoid the resolution of systems of PDEs while computing Lie symmetries (see § 4.2 page 93 for details).

4.1.2 Symmetries of Dynamical Systems

In this section, *Lie symmetries of dynamical systems* are considered. Let M be a manifold. A unique notation, namely $\mathcal{D}$, is used to express continuous and discrete dynamical systems with:

$$\begin{array}{rccc} \mathcal{D}: & G \times M & \rightarrow & M, \\ & \nu \times Z & \rightarrow & \mathcal{D}(\nu, Z)\,. \end{array} \tag{4.4}$$

The existence of a symmetry group is independent of the choice of the coordinate set, only its explicit form (the components of its infinitesimal generators) depend on this choice. Thus, let us denote by $\mathcal{S}$ a continuous dynamical system and by $\delta_{\mathcal{S}}$ the associated infinitesimal generator with:

$$\begin{array}{rccc} \mathcal{S}: & \hat{G} \times M & \rightarrow & M, \\ & \hat{\nu} \times Z & \rightarrow & \mathcal{S}(\hat{\nu}, Z)\,. \end{array} \tag{4.5}$$

Lie Point Symmetries *not Acting* on the Independent Variable

Let us give the definition of *Lie point symmetries of a dynamical system* preserving the independent variable and some of its properties. First, let us see that the linearization of *defining systems* leads to *determining systems* composed of PDEs.

Definition 4.1.4 (Lie Point Symmetry of a Dynamical System)**.** Let $\mathcal{D}$ be a dynamical system. A continuous dynamical system $\mathcal{S}$ is a *Lie point symmetry* of $\mathcal{D}$, if and only if, $\mathcal{S}$ sends an orbit of $\mathcal{D}$ on another orbit of $\mathcal{D}$. Hence, a Lie symmetry $\mathcal{S}$ of a dynamical system $\mathcal{D}$ satisfies the following *defining system* for all ν in G and $\hat{\nu}$ in $\hat{G}$:

$$\mathcal{D}(\nu, \mathcal{S}(\hat{\nu}, Z)) = \mathcal{S}(\hat{\nu}, \mathcal{D}(\nu, Z)) \tag{4.6}$$

that is presented in figure 4.1 page 90. ⌟

Figure 4.1 page 90 is drawn to illustrate the relation (4.6). The difference between the continuous and the discrete systems would be the form of the orbits of $\mathcal{D}$. In the continuous case, the orbits are composed of complete horizontal (violet) lines. In the discrete case, the orbits are a discrete set of points on these (violet) lines at each intersection with an orbit of $\mathcal{S}$ drawn by vertical (green) lines. The order of application of the dynamical systems $\mathcal{D}$ and $\mathcal{S}$ does not matter. Two ways reach the same point.

The differentiability hypothesis (see equation (3.60) page 81) and the continuity of the group element (see equation (3.62) page 82) of the symmetry let us write the following relations for all ν in G and ϵ in the neighborhood of the neutral element e of $\hat{G}$ (below $e = 0$):

$$\begin{array}{rcl} \mathcal{D}(\nu, \mathcal{S}(\epsilon, Z)) & = & \mathcal{D}\big(\nu, Z + \epsilon\, \delta_{\mathcal{S}} Z + \mathrm{O}\big(\epsilon^2\big)\big)\,, \\ & = & \mathcal{D}(\nu, Z) + \epsilon\, \frac{\partial \mathcal{D}(\nu, Z)}{\partial Z}\, \delta_{\mathcal{S}} Z + \mathrm{O}\big(\epsilon^2\big)\,. \end{array} \tag{4.7a}$$

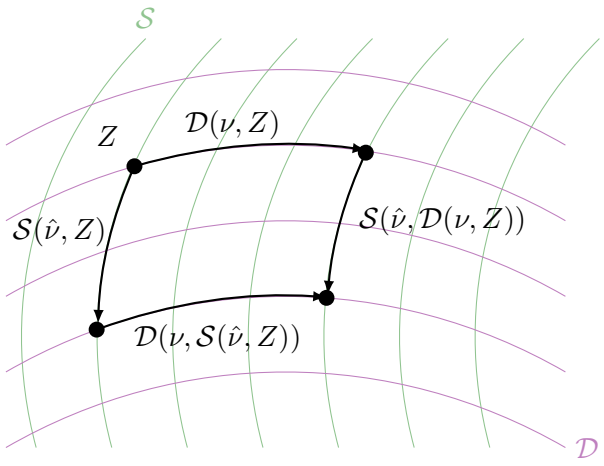


Figure 4.1: Diagram involving a dynamical system $\mathcal{D}$ and one of its Lie symmetries $\mathcal{S}$ preserving the independent variable of $\mathcal{D}$. ⌟

$$\mathcal{S}(\epsilon, \mathcal{D}(\nu, Z)) \quad = \quad \mathcal{D}(\nu, Z) + \epsilon\, \delta_{\mathcal{S}} \mathcal{D}(\nu, Z) + \mathrm{O}\left(\epsilon^2\right). \tag{4.7b}$$

Hence, the relation (4.6) page 89 implies for all ν in G the following *determining system*:

$$\frac{\partial \mathcal{D}(\nu, Z)}{\partial Z}\, \delta_{\mathcal{S}} Z = \delta_{\mathcal{S}} \mathcal{D}(\nu, Z). \tag{4.8}$$

Now, using pseudo-derivations, let us see the unified computational algebraic definition that corresponds to this determining system for continuous and discrete dynamical systems.

Computational Algebraic Definition of Determining Systems. Lie point symmetries of continuous and discrete dynamical systems must verify the system (4.8). This last can be expressed by a simple algebraic relation involving the associated pseudo-derivations. This relation requires the use of *bracket* of two pseudo-derivations defined in the Ore algebra $\mathrm{Ore}\mathbb{R}(Z)$ (see remark 3.2.12 page 64).

Definition 4.1.5 (Bracket on Ore Algebras)**.** The operation *bracket* of two pseudo-derivations is defined by the $\mathbb{R}$-bilinear skew-symmetric map on $\mathrm{Ore}\mathbb{R}(Z)$:

$$\begin{array}{rccc} [\,,\,]: & \mathrm{Ore}\mathbb{R}(Z) \times \mathrm{Ore}\mathbb{R}(Z) & \to & \mathrm{Ore}\mathbb{R}(Z), \\ & (\delta_1, \delta_2) & \to & \delta_1\, \delta_2 - \delta_2\, \delta_1 \end{array} \tag{4.9}$$

with same properties as the Lie bracket defined in definition 3.3.2 page 69. ⌟

The following lemma is a result that allows to unify the symmetry (preserving the independent variable) definitions and implementations of systems of ODEs and OREs. It summarizes below computations of a Lie point symmetry with one simple algebraic relation.

Lemma 4.1.6. *Let $\mathcal{D}$ be a dynamical system and $\delta_{\mathcal{D}}$ its associated pseudo-derivation. A continuous dynamical system $\mathcal{S}$ is a symmetry of $\mathcal{D}$ with its infinitesimal generator $\delta_{\mathcal{S}}$ if the relation $[\delta_{\mathcal{D}}, \delta_{\mathcal{S}}] = 0$ holds.* ⌟

At this step, one needs to discuss the continuous and the discrete cases separately in order to specify associated determining systems.

First-order Ordinary Recurrence Equations. A discrete dynamical system $\mathcal{D}$, associated to a system of OREs, is defined on a group G that is supposed to be isomorphic to a subgroup of $(\mathbb{Z}, +)$. The associated recurrence operator is denoted by σ and the pseudo-derivation by $\delta_{\mathcal{D}}$.

Proposition 4.1.7. *The recurrence operator σ associated to a system of OREs is commutative ($\sigma\delta_{\mathcal{S}} \equiv \delta_{\mathcal{S}}\sigma$) with the infinitesimal generator $\delta_{\mathcal{S}}$ of its symmetry.* ⌟

Proof. The lemma 4.1.6 page 91 and the definition of pseudo-derivations (see 3.2.16 page 65) imply for all Z in $\mathbb{R}^n$:

$$\begin{array}{rcl} [\delta_{\mathcal{D}}, \delta_{\mathcal{S}}](Z) & = & 0, \\ \delta_{\mathcal{D}}\delta_{\mathcal{S}}Z & = & \delta_{\mathcal{S}}\delta_{\mathcal{D}}Z, \\ \sigma\delta_{\mathcal{S}}Z - \delta_{\mathcal{S}}Z & = & \delta_{\mathcal{S}}\sigma Z - \delta_{\mathcal{S}}Z, \\ \sigma\delta_{\mathcal{S}}Z & = & \delta_{\mathcal{S}}\sigma Z. \end{array} \tag{4.10}$$

Thus, the recurrence operator σ and the infinitesimal generator $\delta_{\mathcal{S}}$ commute. ⌟

For a system of OREs, the relation (4.8) page 90 is equivalent to the following one for all τ in G:

$$\frac{\partial \sigma^{[\tau]} Z}{\partial Z} \delta_{\mathcal{S}} Z = \delta_{\mathcal{S}} \sigma^{[\tau]} Z. \tag{4.11}$$

Definition 4.1.8. Thanks to the proposition 4.1.7, the *determining system of a Lie point symmetry of a system of OREs* is given for $\tau = 1$ in the equation (4.11) i.e. by the relation:

$$\frac{\partial \sigma Z}{\partial Z} \delta_{\mathcal{S}} Z = \sigma \delta_{\mathcal{S}} Z. \tag{4.12}$$

⌟

Remark 4.1.9. The formula (4.12) could be obtained in various framework; it appears also in the equation (1.4) of [62] for analytic first-order discrete dynamical systems and in the equation (2.21) of [60] for systems of ordinary difference equations. ⌟

Remark 4.1.10. In [62], authors show that symmetries of discrete dynamical systems always verify the lemma 4.1.6. Thus there is no need to treat Lie symmetries acting on the independent variable for such systems. ⌟

Example 4.1.11. The system of OREs given in the example 3.2.10 page 64 has its associated recurrence operator in (3.18) page 65. A Lie point symmetry of this system has:

$$\delta_{\mathcal{S}} = x\frac{\partial}{\partial x} + y\frac{\partial}{\partial y} \tag{4.13}$$

as infinitesimal generator. One can easily verify that the determining system (4.12) is satisfied by these objects. ⌟

First-order Ordinary Differential Equations. A continuous dynamical system $\mathcal{D}$, associated to a system of ODEs, is defined on the group G that is supposed to be isomorphic to a continuous subgroup of $(\mathbb{R}, +)$. The associated infinitesimal generator is denoted by $\delta_{\mathcal{D}}$. If a continuous dynamical system $\mathcal{S}$ is the symmetry of such a system, then they verify the relation (4.8) page 90. In addition, the continuity of the dynamical system $\mathcal{D}$ let us make a second linearization, thus one can deduce following equalities for ϵ in the neighborhood of the neutral element 0 of G:

$$\begin{aligned} \tfrac{\partial \mathcal{D}(\epsilon,Z)}{\partial Z}\,\delta_{\mathcal{S}} Z &= \left(\tfrac{\partial\left(Z+\epsilon\,\delta_{\mathcal{D}} Z+\mathrm{O}\left(\epsilon^2\right)\right)}{\partial Z}\right)\delta_{\mathcal{S}} Z, \\ &= \delta_{\mathcal{S}} Z+\epsilon\,\tfrac{\partial \delta_{\mathcal{D}} Z}{\partial Z}\,\delta_{\mathcal{S}} Z+\mathrm{O}\left(\epsilon^2\right). \end{aligned} \tag{4.14a}$$

$$\begin{aligned} \delta_{\mathcal{S}}\mathcal{D}(\epsilon, Z) &= \delta_{\mathcal{S}}\left(Z+\epsilon\,\delta_{\mathcal{D}} Z+\mathrm{O}\left(\epsilon^2\right)\right), \\ &= \delta_{\mathcal{S}} Z+\epsilon\,\tfrac{\partial \delta_{\mathcal{S}} Z}{\partial Z}\,\delta_{\mathcal{D}} Z+\mathrm{O}\left(\epsilon^2\right). \end{aligned} \tag{4.14b}$$

Definition 4.1.12. The relations given in the system (4.14) establish the *determining system of a Lie point symmetry of a system of ODEs*:

$$\frac{\partial \delta_{\mathcal{D}} Z}{\partial Z}\,\delta_{\mathcal{S}} Z=\frac{\partial \delta_{\mathcal{S}} Z}{\partial Z}\,\delta_{\mathcal{D}} Z. \tag{4.15}$$

⌟

Example 4.1.13. The system of ODEs given in the example 3.2.6 page 63 has its associated infinitesimal generator in (3.15) page 65. A Lie point symmetry of this system has:

$$\delta_{\mathcal{S}}=-x\frac{\partial}{\partial x}+b\frac{\partial}{\partial b} \tag{4.16}$$

as infinitesimal generator. One can easily verify that the determining system (4.15) is satisfied by these objects. ⌟

Lie Point Symmetries *Acting* on the Independent Variable

This case is only studied for continuous dynamical systems so, in this section, $\mathcal{D}$ stands for such systems. Contrarily to the previous case, for symmetries acting also on the independent variable of $\mathcal{D}$, the order of application of $\mathcal{D}$ and $\mathcal{S}$ does matter. The associated geometrical definition is illustrated in figure 4.2 page 93. Remark that this is a generalization of figure 4.1 page 90. This geometrical definition leads to the following lemma.

Lemma 4.1.14. *Let $\mathcal{D}$ be a continuous dynamical system and $\delta_{\mathcal{D}}$ its infinitesimal generator. A continuous dynamical system $\mathcal{S}$ is a non-trivial symmetry of $\mathcal{D}$ with its infinitesimal generator $\delta_{\mathcal{S}}$, if the following relation (see § 3.3 of [104]) holds:*

$$\left[\delta_{\mathcal{D}}, \delta_{\mathcal{S}}\right]=\lambda\,\delta_{\mathcal{D}} \tag{4.17}$$

where λ is any constant of $\delta_{\mathcal{D}}$ and $\delta_{\mathcal{S}}$. These generators are linearly independent. ⌟

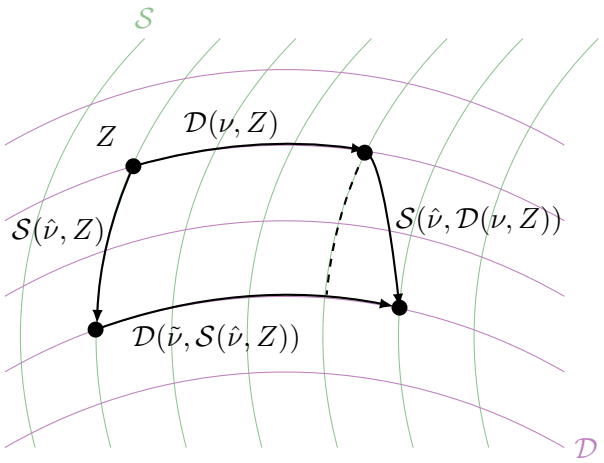


Figure 4.2: Diagram involving a continuous dynamical system $\mathcal{D}$ and one of its Lie symmetries $\mathcal{S}$ acting also on the independent variable of $\mathcal{D}$. ⌟

Remark 4.1.15. For continuous dynamical systems, specializing λ to 0 in (4.17) page 92 leads to the formula that computes Lie symmetries not acting on the independent variable (see lemma 4.1.6 page 91). ⌟

Remark 4.1.16. Following the lemma 2.3 in [114], with above notations, assume that one has $[\delta_{\mathcal{D}}, \delta_{\mathcal{S}}] = \lambda\, \delta_{\mathcal{D}}$ where λ is any function and let ψ be a first integral of $\delta_{\mathcal{S}}$ i.e. $\delta_{\mathcal{S}}\psi = 0$ with $\delta_{\mathcal{D}}\psi \neq 0$. Then

$$\left[\frac{1}{\delta_{\mathcal{D}}\psi}\delta_{\mathcal{D}}, \delta_{\mathcal{S}}\right] = 0. \tag{4.18}$$

This statement allows to work with a symmetry $\mathcal{S}$ acting on the independent variable of $\mathcal{D}$ as a symmetry preserving an independent variable if one can determine a first integral of $\mathcal{D}$. ⌟

Remark 4.1.17. There is also another way of defining a Lie point symmetry of a continuous dynamical system similar to that given for algebraic systems. This involves the notion of prolongation in a jet space where the studied differential system is considered as an algebraic one (see th. 2.31 in § 2.3 of [86]). ⌟

4.2 Computation of a Restricted Set of Lie Point Symmetries

In this section, we present the computation of Lie point symmetries as they were conceived in the package `ExpandedLiePointSymmetry`. First, we detail the restrictions done on the determining systems and on the set of symmetries in order to have an efficient implementation. Then we show a probabilistic algorithm to solve resulting determining systems. Thanks to our computational strategies, the complexity of this algorithm is polynomial in the input size. But the complete set of symmetries are not obtained. Nevertheless, in practice, almost all encountered symmetries could be obtained using these strategies. Finally, associated implementation choices are given.

In order to apply the reduction and the reparametrization methods to a given system (see ch. 5 page 103), the first task to perform is to determine its symmetries. The general approaches to compute Lie symmetries require the resolution of systems of PDEs (see for example [86, 104]). Unfortunately there is no a general algorithm for solving such systems. Our framework proposes an efficient implementation with polynomial complexity in the input size to compute Lie symmetries. It uses the linear algebra over a number field. In return, we need to perform the following restrictions:

- determining systems of Lie symmetries are restricted;
- set of symmetries of interest is adapted, according to needs, to classical geometric transformations. The available symmetries (in October, 2009) are translations, scalings, affine and quadratic type symmetries (see § 4.2.2 page 96).

4.2.1 Restriction of Determining Systems

In this subsection we show how to restrict the definitions of Lie symmetries in order to have an efficient implementation.

Let us recall that, the general form of infinitesimal generators $\delta_{\mathcal{S}}$ of symmetries is:

$$\delta_{\mathcal{S}} = \sum_{i=1}^{n} \xi_{z_i}(Z) \frac{\partial}{\partial z_i} \tag{4.19}$$

where ξ_{z_i} are functions in Z. In the sequel, $Z = (z_1, \ldots, z_n)$ stands for the coordinate set and $\xi = (\xi_{z_1}, \ldots, \xi_{z_n})$ for the symmetry coefficients.

Determining Systems for Symmetries of Algebraic Systems

Let $F = (f_1, \ldots, f_k)$ be an algebraic system (see definition 3.1.1 page 59) and $\mathcal{S}$ a continuous dynamical system associated to the infinitesimal generator $\delta_{\mathcal{S}}$. According to the theorem 4.1.1 page 88, $\mathcal{S}$ is a symmetry of F if $\delta_{\mathcal{S}}$ leaves invariant the ideal spanned by $\{f_1, \ldots, f_k\}$. This definition requires the usage of elimination methods (see [20] for computational methods and [46] for more geometrical cases) that we want to avoid because of their complexity problems in general cases (exponential complexity in the input size, see [25]). In the `ExpandedLiePointSymmetry` package, we use the determining system given in the following corollary to compute expanded Lie point symmetries of an algebraic system.

Corollary 4.2.1. *Let $F = (f_1, \ldots, f_k)$ be a regular algebraic system. A continuous dynamical system associated to the infinitesimal generator $\delta_{\mathcal{S}}$ is a symmetry of F if the following relation, in vector-valued notations, holds:*

$$\delta_{\mathcal{S}} F = \Lambda F \tag{4.20}$$

for a matrix Λ with elements in $\mathbb{Q}$. ⌟

Let us rewrite the determining system (4.20) page 94 by considering the explicit form (4.19) page 94 of the infinitesimal generator $\delta_{\mathcal{S}}$. The relation to verify so that $\delta_{\mathcal{S}}$ can be associated to a symmetry of an algebraic system is:

$$\begin{pmatrix} \frac{\partial f_1}{\partial z_1} & \cdots & \frac{\partial f_1}{\partial z_n} \\ \vdots & & \vdots \\ \frac{\partial f_k}{\partial z_1} & \cdots & \frac{\partial f_k}{\partial z_n} \end{pmatrix} \begin{pmatrix} \xi_{z_1} \\ \vdots \\ \xi_{z_n} \end{pmatrix} - \begin{pmatrix} \lambda_{11} & \cdots & \lambda_{1k} \\ \vdots & & \vdots \\ \lambda_{k1} & \cdots & \lambda_{kk} \end{pmatrix} \begin{pmatrix} f_1 \\ \vdots \\ f_k \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \tag{4.21}$$

with λ_{ij} in $\mathbb{Q}$.

Corollary 4.2.1 page 94 provides a sufficient condition for the theorem 4.1.1 page 88. Since for all i in $\{1, \ldots, k\}$, $\delta_{\mathcal{S}} f_i$ is a linear combination of $\{f_1, \ldots, f_k\}$, whenever these functions vanish, $\delta_{\mathcal{S}} f_i$ vanishes also. On the other hand, remark that this is not a necessary condition. The following example shows limitations of this corollary compared with the theorem 4.1.1 page 88.

Example 4.2.2. For the regular algebraic system $f(x, y) = x^4 + x^2\, y^2 + y^2 - 1$ (see example 3.1.7 page 60), the `ExpandedLiePointSymmetry` package does not find any symmetry. However the infinitesimal generator $\delta_{\mathcal{S}} = -y\partial/\partial x + x\partial/\partial y$ is associated to a rotation symmetry of it. One can compute $\delta_{\mathcal{S}} f = -\left(2\, x\, y/x^2 + 1\right) f$. This is not a multiple of f in $\mathbb{Q}$, which is why it is not detected by the corollary 4.2.1 page 94. But whenever f vanishes $\delta_{\mathcal{S}} f$ vanishes also thus the theorem 4.1.1 page 88 is satisfied. ⌟

Determining Systems for Symmetries of Continuous Dynamical Systems

In the `ExpandedLiePointSymmetry` package, the computations of expanded Lie point symmetries of a system of ODEs rely on the lemma 4.1.14 page 92 but λ is restricted to values in $\mathbb{Q}$. Let us rewrite the determining system (4.17) page 92 by considering the explicit form (4.19) page 94 of the infinitesimal generator $\delta_{\mathcal{S}}$ and the definition of a system of ODEs (see definition 3.2.1 page 62). The relation to verify so that $\delta_{\mathcal{S}}$ can be associated to a symmetry of a system of ODEs is:

$$\begin{pmatrix} 0 & \cdots & 0 \\ \frac{\partial f_1}{\partial z_1} & \cdots & \frac{\partial f_1}{\partial z_n} \\ \vdots & & \vdots \\ \frac{\partial f_k}{\partial z_1} & \cdots & \frac{\partial f_k}{\partial z_n} \\ 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{pmatrix} \begin{pmatrix} \xi_{z_1} \\ \vdots \\ \xi_{z_n} \end{pmatrix} - \begin{pmatrix} \frac{\partial \xi_{z_1}}{\partial z_1} & \cdots & \frac{\partial \xi_{z_1}}{\partial z_n} \\ \vdots & & \vdots \\ \frac{\partial \xi_{z_n}}{\partial z_1} & \cdots & \frac{\partial \xi_{z_n}}{\partial z_n} \end{pmatrix} \begin{pmatrix} 1 \\ f_1 \\ \vdots \\ f_k \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \lambda \begin{pmatrix} 1 \\ f_1 \\ \vdots \\ f_k \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \tag{4.22}$$

with λ in $\mathbb{Q}$.

Remark 4.2.3. If one wants to work with the determining system given in (4.15) page 92, it is enough to substitute $\lambda = 0$ in (4.22). ⌟

Determining Systems for Symmetries of Discrete Dynamical Systems

In the `ExpandedLiePointSymmetry` package, for systems of OREs, only expanded Lie point symmetries not acting on the independent variable are considered. That is why, the computations rely on the formula (4.12) page 91. Let us rewrite this formula by considering the explicit form (4.19) page 94 of the infinitesimal generator $\delta_{\mathcal{S}}$ and the definition of a system of OREs (see definition 3.2.7 page 63). The relation to verify so that $\delta_{\mathcal{S}}$ can be associated to a symmetry of a system of OREs is:

$$\begin{pmatrix} 0 & \cdots & 0 \\ \frac{\partial f_1}{\partial z_1} & \cdots & \frac{\partial f_1}{\partial z_n} \\ \vdots & & \vdots \\ \frac{\partial f_k}{\partial z_1} & \cdots & \frac{\partial f_k}{\partial z_n} \\ 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{pmatrix} \begin{pmatrix} \xi_{z_1} \\ \vdots \\ \xi_{z_n} \end{pmatrix} - \begin{pmatrix} \sigma\xi_{z_1} \\ \vdots \\ \sigma\xi_{z_n} \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{4.23}$$

4.2.2 Restriction of Set of Symmetries

Even with all the restrictions done until now, the computation of a symmetry requires integration of systems of PDEs. We need to focus on special kinds of symmetries. The available set of symmetries in the `ExpandedLiePointSymmetry` package are *translations*, *scalings*, *affine* and *quadratic* type symmetries as defined in the following definitions (see also [101, 68] for the same point of view).

Definition 4.2.4 (Translation)**.** A *translation* is a Lie point symmetry associated to an infinitesimal generator of the form:

$$\delta_{\mathcal{S}} := \sum_{i=1}^{n} \alpha_i \frac{\partial}{\partial z_i} \tag{4.24}$$

where α_i are in $\mathbb{Q}$. ⌟

Definition 4.2.5 (Scaling)**.** A *scaling* is a Lie point symmetry associated to an infinitesimal generator of the form:

$$\delta_{\mathcal{S}} := \sum_{i=1}^{n} \alpha_i \, z_i \frac{\partial}{\partial z_i} \tag{4.25}$$

where α_i are in $\mathbb{Q}$. ⌟

Definition 4.2.6 (Affine Type Symmetry)**.** An *affine type symmetry* is a Lie point symmetry associated to an infinitesimal generator of the form:

$$\delta_{\mathcal{S}} := \sum_{i=1}^{n} \left(\alpha_i + \sum_{j=1}^{n} \beta_{ij} \, z_j \right) \frac{\partial}{\partial z_i} \tag{4.26}$$

where α_i and β_{ij} are in $\mathbb{Q}$. ⌟

Definition 4.2.7 (Quadratic Type Symmetry)**.** A *quadratic type symmetry* is a Lie point symmetry associated to an infinitesimal generator of the form:

$$\delta_{\mathcal{S}} := \sum_{i=1}^{n} \left(\alpha_i + \sum_{j=1}^{n} \beta_{ij}\, z_j + \sum_{k=j}^{n} \sum_{j=1}^{n} \gamma_{ijk} z_j z_k \right) \frac{\partial}{\partial z_i} \tag{4.27}$$

where α_i, β_{ij} and γ_{ijk} are in $\mathbb{Q}$. ⌟

With these types of restrictions, the PDEs systems (4.21) page 95, (4.22) page 95 and (4.23) page 96 reduce to linear algebraic systems that could be solved efficiently. Let us adapt them now w.r.t. these types of symmetries. First, let us introduce the following notations of indeterminates:

$$\begin{aligned} A &:= (\alpha_i)_{1\leq i\leq n}\,, \quad B := (\beta_{ij})_{1\leq i\leq n,\, 1\leq j\leq n}\,, \quad Z := (z_i)_{1\leq i\leq n}\,, \\ \Gamma_i &:= \left((\gamma_{ijk})_{1\leq j\leq n,\, j\leq k\leq n}\right) \quad \forall i \in \{1,\ldots,n\}\,. \end{aligned} \tag{4.28}$$

Remark that A and Z are vectors of indeterminates, B is a matrix of indeterminates and Γ_i are upper triangular matrices of indeterminates. One must mainly adapt the vector of coefficients ξ of the searched symmetry and the gradient matrix ∇ of it that appear in the determining systems:

$$\xi := (\xi_i)_{1\leq i\leq n}\,, \quad \nabla := \left(\frac{\partial \xi_i}{\partial z_j}\right)_{1\leq i\leq n,\, 1\leq j\leq n}. \tag{4.29}$$

The recapitulation of the notations for these specific kind of symmetries follows:

- for a translation

$$\xi = A, \quad \nabla = \begin{pmatrix} 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{pmatrix}; \tag{4.30}$$

- for a scaling

$$\xi = \begin{pmatrix} \alpha_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \alpha_n \end{pmatrix} Z, \quad \nabla = \begin{pmatrix} \alpha_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \alpha_n \end{pmatrix}; \tag{4.31}$$

- for an affine type symmetry

$$\xi = A + B\,Z, \quad \nabla = B; \tag{4.32}$$

- for a quadratic type symmetry

$$\begin{aligned} &\xi = A + B\,Z + \left({}^{t}Z\,\Gamma_i\,Z\right)_{1\leq i\leq n}\,, \\ &\nabla = B + (\nabla_{ij})_{1\leq j\leq n,\, 1\leq i\leq n} \text{ where } \nabla_{ij} := \sum_{l=1}^{i} \gamma_{jli} z_l + \sum_{l=i}^{n} \gamma_{jil} z_l. \end{aligned} \tag{4.33}$$

The following three equations are respectively final determining systems used, in the `ExpandedLiePointSymmetry` package, to compute symmetries of algebraic systems, continuous and discrete dynamical systems where ξ and ∇ must be replaced by their above expressions.

$$\begin{pmatrix} \frac{\partial f_1}{\partial z_1} & \cdots & \frac{\partial f_1}{\partial z_n} \\ \vdots & & \vdots \\ \frac{\partial f_k}{\partial z_1} & \cdots & \frac{\partial f_k}{\partial z_n} \end{pmatrix} \xi - \begin{pmatrix} \lambda_{11} & \cdots & \lambda_{1k} \\ \vdots & & \vdots \\ \lambda_{k1} & \cdots & \lambda_{kk} \end{pmatrix} \begin{pmatrix} f_1 \\ \vdots \\ f_k \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}, \tag{4.34a}$$

$$\begin{pmatrix} 0 & \cdots & 0 \\ \frac{\partial f_1}{\partial z_1} & \cdots & \frac{\partial f_1}{\partial z_n} \\ \vdots & & \vdots \\ \frac{\partial f_k}{\partial z_1} & \cdots & \frac{\partial f_k}{\partial z_n} \\ 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{pmatrix} \xi - \nabla \begin{pmatrix} 1 \\ f_1 \\ \vdots \\ f_k \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \lambda \begin{pmatrix} 1 \\ f_1 \\ \vdots \\ f_k \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}, \tag{4.34b}$$

$$\begin{pmatrix} 0 & \cdots & 0 \\ \frac{\partial f_1}{\partial z_1} & \cdots & \frac{\partial f_1}{\partial z_n} \\ \vdots & & \vdots \\ \frac{\partial f_k}{\partial z_1} & \cdots & \frac{\partial f_k}{\partial z_n} \\ 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{pmatrix} \xi - \sigma\xi = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{4.34c}$$

Let us see now how these resulting determining systems are solved in the package `ExpandedLiePointSymmetry`.

4.2.3 Solving Considered Determining Systems

The aim of this subsection is to detail the algorithmic choices made in order to solve linear determining systems (equations (4.34)).

The algorithm 4 page 99 summarizes, using a `Maple` like syntax, the probabilistic resolution of these determining systems defining the quadratic infinitesimal generators. Solving determining systems for translations, scalings or affine type symmetries can easily be deduced from this algorithm by ignoring unnecessary indeterminates (the Γ_i's and B's for translations and scalings, the Γ_i's for affine type symmetries).

The inputs of the algorithm 4 page 99 are the list of equations that correspond to each line of the associated determining system, the vector of coordinates on which the dynamical system of interest is defined and the list of indeterminates of Lie point symmetries to compute. The output is a list of symmetry coefficients that defines the associated infinitesimal generators.

Algorithm 4 Solving determining systems (see equations (4.34) page 98) for quadratic type symmetries

Input: A list L of equations which are linear w.r.t. indeterminates of A, B and $(\Gamma_i)_{1\leq i\leq n}$.
A vector of coordinates Z.
A list of indeterminates $U := (\alpha_i, \beta_{ij}, \gamma_{ijk})$ (see (4.28) page 97 for definitions).

Output: A list of quadratic infinitesimal generators defined as in (4.27) associated to solutions of L.

1: #*Rewrite* L *in the form* $N\,{}^tU = 0$
2: $N := \text{Matrix}\left(\left[\text{seq}\left(\left[\text{seq}\left(\text{coeff}\left(\ell, u\right), \ell \in L\right)\right], u \in U\right)\right]\right)$;
3: $r := -1$; $M := \emptyset$;
4: **repeat**
5: $\quad C := \left\{\text{seq}\left(\text{RandomInteger}\left(\right), i = 1 \ldots n\right)\right\}$;
6: $\quad s := r$;
7: $\quad$ #*Concatenate the matrix* M *with a specialization of the matrix* N
8: $\quad M := \text{StackMatrix}\left(M, \text{subs}\left(Z = C, N\right)\right)$;
9: $\quad r := \text{Rank}\left(M\right)$;
10: **until** $r \neq s$;
11: #*Kernel computation of* M
12: $V := \text{NullSpace}\left(M\right)$;
13: **return** $\left[\text{seq}\left(\text{subs}\left(u = v, \xi = A + B\,Z + \left({}^tZ\,\Gamma_i\,Z\right)_{1\leq i\leq n}\right), v \in V\right)\right]$;

Let $U := (\alpha_i, \beta_{ij}, \gamma_{ijk})$ be a list of $n + n^2 + n^3$ indeterminates composing A, B and Γ (see notations in (4.28) page 97). One can write the determining systems (4.34) page 98 in the matrix form $N\,{}^tU = 0$ where N is a matrix of n rows (for the determining system (4.21) page 95 associated to algebraic systems, k rows) and $n + n^2 + n^3$ columns with coefficients in $\mathbb{Q}(Z)$. As this system is under-determined, several specializations of the coordinates Z to random values in $\mathbb{Q}$ are necessary to obtain at most $n + n^2 + n^3$ linear equations. The kernel in $\mathbb{Q}$ of the resulting purely numerical system gives the $\mathbb{Q}$-vector-space V of indeterminates in U. These indeterminates are solutions of the considered determining system and specify the quadratic infinitesimal generators of the symmetries. This kernel may be computed by any classical numerical method. The specialization set for which this process fails to find a correct solution is a zero-dimensional algebraic variety defined by the solution set of the polynomial $\det(N\left(Z\right))$ (see [100, 119, 120] for probabilistic aspect). Thus, the failure probability of such method is almost zero.

Remark that the base field of the Lie algebra (see definition 2.4.2-3 in § 2.4.2 of [6]) returned by this algorithm is $\mathbb{Q}$ (see § 3.3.1 page 69). As a result, some infinitesimal generators may define the same vector field (see example 3.3.8 page 70). The redundancy should be discarded, the base field must be extended as much as possible (see § 3.3.5 page 78 for the algorithm used in the `ExpandedLiePointSymmetry` package).

All these computational strategies with the restrictions on the determining systems and the symmetries yield a polynomial complexity as stated in the following proposition.

Proposition 4.2.8. *The probabilistic resolution of the above determining systems (equations (4.34) page 98) defining the infinitesimal generators of expanded Lie point sym-*

metries with polynomial coefficients in $\mathbb{Q}[Z]$, detailed in the algorithm 4 page 99, has an arithmetic complexity $\mathrm{O}\big(5\,L\,n^d + n^{(d+1)\omega}\big)$ where L is the complexity for the evaluation of the set $F = (f_1, \ldots, f_k)$, n is the cardinal of the coordinate set, d is the maximum degree of polynomials used in the coefficients ξ of the symmetries and ω is the linear algebra constant. ⌟

Proof. Let us see the complexity of the algorithm 4 page 99 in two steps: the construction of the matrix M and its kernel computation. Let L be the complexity for the evaluation of the set $F = (f_1, \ldots, f_k)$. The evaluation cost for determining systems is equal to $5\,L$ (see [80]). The cost of the construction of a matrix is supposed to be linear w.r.t. its size. The matrix N is of order $\big(n \times n^{(d+1)}\big)$ where n is the cardinal of the coordinate set and d is the maximum degree of polynomials used in the coefficients of symmetries. The construction of the matrix M needs n^d times specialization of N because its size is of order $\big(n^{(d+1)} \times n^{(d+1)}\big)$. Thus the kernel computation of M costs $n^{(d+1)\omega}$ operations where ω is the linear algebra constant. The probabilistic resolution of determining systems of interest (see algorithm 4 page 99) has a complexity $\mathrm{O}\big(5\,L\,n^d + n^{(d+1)\omega}\big)$. For the computation model see § 2 in [45] or § 3.2 in [44]. ⌟

Remark 4.2.9. The complexity of the algorithm 4 page 99 is linear w.r.t. the cardinal of the coordinate set but exponential in the maximum degree of polynomials of the coefficients of symmetries (see proposition 4.2.8). In this document, the maximum degree considered for such polynomials is 2 (quadratic type symmetries). In this worst case, the complexity is equal to $\mathrm{O}\big(5\,L\,n^2 + n^{3\omega}\big)$. Thus the complexity remains polynomial in the input size for the most common encountered symmetries. ⌟

4.2.4 Structure Computation of Lie Point Symmetries Algebras

In this subsection we first explicit the algorithm of computing expanded Lie point symmetries of a system of ODEs in the `ExpandedLiePointSymmetry` package using above restrictions. The algorithms used for algebraic systems and systems of OREs are similar to this one. Then we detail the usage of the `ELPSymmetries` function.

The algorithm 5 page 101 summarizes the computation of a Lie algebra composed by expanded Lie point symmetries of a continuous dynamical system $\mathcal{D}$. The input system of ODEs is represented by the associated infinitesimal generator $\delta_{\mathcal{D}}$. One needs to indicate the type of required symmetries among translations, scalings, affine and quadratic type symmetries (see § 4.2.2 page 96). One can also specify a list of coordinates if the resulting symmetries must not act on them. This option does not appear in the algorithm 5 page 101 for the sake of clarity. The output is the Lie algebra defined by the computed symmetries.

The steps of the algorithm 5 page 101 along with associated line numbers follow:

- lines 1-2, initialization of the vector of coordinates Z and the vector of polynomials F associated to the state variables of the given system of ODEs;
- lines 4-7, construction of the determining system thanks to the equation (4.34b) page 98 for the chosen type of symmetry;

- lines 9-11, computation of the Lie algebra that corresponds to the Lie point symmetries of $\delta_{\mathcal{D}}$ by using the algorithm 4 page 99;
- line 13, extension of the base field. The base field of this Lie algebra is supposed to be $\mathbb{Q}$ (see § 3.3.1 page 69). In order to discard surplus of the infinitesimal generators, one needs to extend this base field (see § 3.3.5 page 78).

Algorithm 5 Computing expanded Lie point symmetries of a continuous dynamical system $\mathcal{D}$.

Input: A system of ODEs represented by the associated infinitesimal generator $\delta_{\mathcal{D}}$. A variable `sym` that indicates the type of the required symmetries (translation, scaling, affine or quadratic)

Output: A Lie algebra composed by infinitesimal generators of the symmetries of $\delta_{\mathcal{D}}$ given in the input.

1: $Z := (z_1, \ldots, z_n)$;
2: $F := (f_1, \ldots, f_k)$;
3: #*The vector of coefficients ξ and the gradient matrix ∇ of the searched symmetries (see equation (4.29) page 97)*
4: $\xi :=$ VectorOfCoefficients (sym, Z) ;
5: $\nabla :=$ GradientMatrix (sym, Z) ;
6: #*Computation of the determining system thanks to the equation (4.34b) page 98*
7: DetSys := DeterminingSystem (F, ξ, ∇) ;
8: #*The vector of unknowns is composed of the indeterminates A, B, Γ of the equation (4.28) page 97 used to define symmetries and the unknown λ used in the determining system*
9: $U := (A, B, \Gamma, \lambda)$;
10: #*Resolution of the determining system using the algorithm 4* page 99
11: LieAlg := SolveDeterminingSystem (DetSys, Z, U) ;
12: #*Extension of the base field as in the paragraph "Changing the Base Field of a Lie Algebra" in § 3.3.5 page 78*
13: **return** ChangeBaseField (LieAlg, $\delta_{\mathcal{D}}$) ;

In the `ExpandedLiePointSymmetry` package, expanded Lie point symmetries of an algebraic or a dynamical system are computed by calling the `ELPSymmetries` function that follows the algorithm 5. The first argument that represents the system of interest must be of type `AlgebraicSystem`, `InfinitesimalGenerator` or `RecurrenceOperator`. The second argument is used to specify the kind of symmetries that one is looking for. This is an optional argument with a default value devoted to quadratic type symmetries but one can ask for translations, scalings and also affine type symmetries (see § 4.2.2 page 96). The output is a Lie algebra generated by all the infinitesimal generators that have been computed.

Example 4.2.10. This example shows the usage of the function `ELPSymmetries` with different data structures. The first code example computes the translations, the scalings and the affine type symmetries (see § 4.2.2 page 96) of the algebraic system given in the example 3.1.2 page 59. The system does not have any translation (see 4.2.4 page 96)

but one scaling (see 4.2.5 page 96). Furthermore, the system possesses two affine type symmetries. Remark that a scaling, by definition, is also an affine type symmetry so the scaling of `LieAlg2` appears again in `LieAlg3`.

```
> # Symmetries of an algebraic system
> AlgSys := NewAlgebraicSystem([(1-c*P)*P+b*Q+1,a*P-l*Q+1]);
       AlgSys := [(1 - c P) P + b Q + 1, a P - Q l + 1]

> LieAlg1 := ELPSymmetries(AlgSys,sym=translation);
       LieAlg1 := LieAlgebraOfDim(0)

> GeneratorsOf(LieAlg1);
       []

> LieAlg2 := ELPSymmetries(AlgSys,sym=scaling);
       LieAlg2 := LieAlgebraOfDim(1)

> GeneratorsOf(LieAlg2);
                 /d   \     /d   \     /d   \
       [F -> -Q |-- F| + b |-- F| + l |-- F|]
                 \dQ  /     \db  /     \dl  /

> LieAlg3 := ELPSymmetries(AlgSys,sym=affine);
       LieAlg3 := LieAlgebraOfDim(2)

> GeneratorsOf(LieAlg3);
                /d   \     /d   \          /d   \     /d   \     /d   \
       [F -> Q |-- F| + P |-- F|, F -> -Q |-- F| + b |-- F| + l |-- F|]
                \da  /     \dl  /          \dQ  /     \db  /     \dl  /
```

The following code example computes affine type symmetries of the system of ODEs given in the example 3.2.6 page 63. It is represented by the associated infinitesimal generator. One finds three such symmetries.

```
> # Symmetries of an infinitesimal generator
> InfGen := NewInfinitesimalGenerator([1, (a-b*x)*x, 0, 0],[t,x,a,b]);
                      /d   \               /d   \
       InfGen := F -> |-- F| + (a - x b) x |-- F|
                      \dt  /               \dx  /

> LieAlg4 := ELPSymmetries(InfGen, sym=affine);
       LieAlg4 := LieAlgebraOfDim(3)

> GeneratorsOf(LieAlg4);
          d             /d   \     /d   \          /d   \     /d   \     /d   \
    [F -> -- F, F -> -x |-- F| + b |-- F|, F -> t |-- F| - x |-- F| - a |-- F|]
          dt            \dx  /     \db  /          \dt  /     \dx  /     \da  /
```

The final code example computes three affine type symmetries of the system of OREs given in the example 3.2.18 page 65 and represented by its recurrence operator.

```
> # Symmetries of a recurrence operator
> RecurOp := NewRecurrenceOperator([tau+1,c*x,y+x,c],[tau,x,y,c]);
       RecurOp := (tau, x, y, c) -> [tau + 1, c x, y + x, c]

> LieAlg5 := ELPSymmetries(RecurOp, sym=affine);
       LieAlg5 := LieAlgebraOfDim(3)

> GeneratorsOf(LieAlg5);
             d           d            /d   \     /d   \
       [F -> ---- F, F -> -- F, F -> x |-- F| + y |-- F|]
             dtau        dy           \dx  /     \dy  /
```

5 Exact Simplification Processes

This chapter shows how one can, given an algebraic or a dynamical system, construct an equivalent one using Lie point symmetries. In fact, the original system could be described by a lower dimensional system and fibers. During the *reduction process*, Lie point symmetries of the given system are used to obtain these lower dimensional system and fibers. During the *reparametrization process*, we consider a continuous dynamical system and the algebraic system that defines its steady points. Lie point symmetries of this algebraic system are used to perform a change of coordinates on the dynamical system.

We begin by giving the *geometry* of the reduction process. After illustration of *classical reduction process* (see [86]), we reach the *moving frame based reduction process*, main subject of this chapter. The general form of this algorithm may be found in [37, 38]. We show the reduction of algebraic and dynamical systems by a special case of this method. Then, we explain some practical problems and the implementation of moving frame based reduction process in `ExpandedLiePointSymmetry` package. Finally we address the *reparametrization process*. We detail its main idea and its implementation.

5.1 Geometry of the Reduction Process

Geometrically, to *reduce* a system means to consider it in a lower dimensional space. The following theorem gives formal definition of this process in our framework.

Theorem 5.1.1 (see § 3.4 of [86])**.** *Let G be a r-dimensional regular local Lie group of a Lie point transformation $\mathcal{S}$ with again r-dimensional orbits defined in $\mathbb{R}^n$. Then there exists a $(n-r)$-dimensional quotient space of $\mathbb{R}^n$ by the action of $\mathcal{S}$, denoted $\mathbb{R}^n/G$, together with a smooth projection $\pi : \mathbb{R}^n \to \mathbb{R}^n/G$, that satisfies the following properties:*

- *The points Z_1 and Z_2 lie in the same orbit of $\mathcal{S}$ in $\mathbb{R}^n$, if and only if, $\pi(Z_1)$ is equal to $\pi(Z_2)$;*
- *If $\mathfrak{g}$ denotes the Lie algebra of infinitesimal generators $\mathcal{B} = (\delta_1, \ldots, \delta_r)$ associated to the action of $\mathcal{S}$, then for every point Z in $\mathbb{R}^n$, the linear map*

$$\mathrm{d}\pi : T\mathbb{R}^n_Z \to T(\mathbb{R}^n/G)_{\pi(Z)} \tag{5.1}$$

between tangent spaces is onto, with $\mathcal{B}$ as kernel. ⌟

Hence, the projection of any object in $\mathbb{R}^n$, that is invariant under the action of the local Lie group G, leads to a quotient manifold. This last has r fewer dimensions where r is the dimension of the orbits of the symmetry $\mathcal{S}$ of the object. Inversely, the fiber composed of this quotient manifold and the orbits of $\mathcal{S}$ allows to reach the original object from the reduced one.

Example 5.1.2. Consider the algebraic system defined by the equation $x^2 + y^2 - r^2 = 0$ that corresponds to a cone in 3 dimensional space. Let us take $Z = (x, y, r)$. The group action associated to one of its one-dimensional symmetry $\mathcal{S}$ is given below:

$$x \to x\,\lambda, \quad y \to y\,\lambda, \quad r \to r\,\lambda \tag{5.2}$$

where λ is the group parameter. The reduction of the variable r from this algebraic

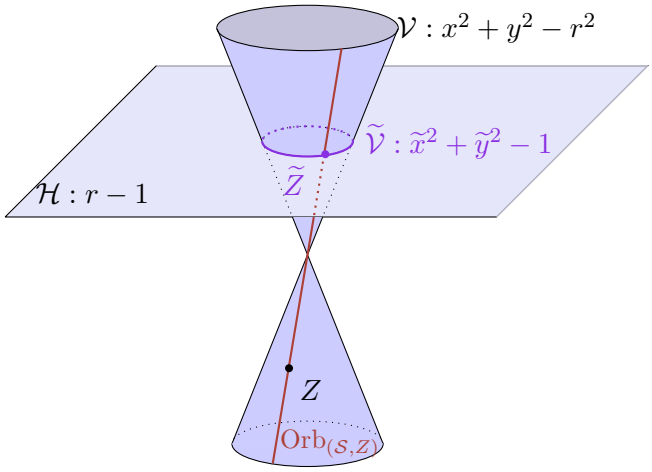


Figure 5.1: Reduction of the algebraic system defined by $x^2 + y^2 - r^2 = 0$.

system by using $\mathcal{S}$ is illustrated in figure 5.2. The variety $\mathcal{V}$, defined by this algebraic system, is projected onto the variety $\mathcal{H}$ defined by the equation $r - 1 = 0$ along the orbits $\mathrm{Orb}_{(\mathcal{S},Z)}$ of $\mathcal{S}$. Each orbit of $\mathcal{S}$ is represented by a single point in $\mathcal{H}$. The reduced variety $\widetilde{\mathcal{V}}$ of equation $\widetilde{x}^2 + \widetilde{y}^2 - 1 = 0$ (the circle of radius 1), is at the intersection of $\mathcal{V}$ and $\mathcal{H}$. Remark that this reduced variety $\widetilde{\mathcal{V}}$ is in a 2-dimensional space. The fiber, i.e. the reduced variety $\widetilde{\mathcal{V}}$ and the orbits $\mathrm{Orb}_{(\mathcal{S},Z)}$, contains enough information to recover the original variety $\mathcal{V}$. The explicit relations between the original coordinates $Z = (x, y, r)$ and the new ones $\mathcal{I} = (\widetilde{x}, \widetilde{y})$ are given by:

$$\widetilde{x} = \frac{x}{r} \quad \widetilde{y} = \frac{y}{r}. \tag{5.3}$$

⌟

Remark 5.1.3. The same geometrical process is used for the *reparametrization* of systems of ODEs (see § 5.5 page 129). However, one does not use the symmetries of differential systems but these of the algebraic systems that define their steady points. ⌟

5.2 Classical Reduction Process

In this section we illustrate the *classical reduction process* that is partially based on the *rectification* of a symmetry. Nevertheless, in this document we are *not* using this type of process that requires computation of invariants. However we survey this process in order to approach the *moving frame based reduction process*.

The classical reduction process requires two main steps. The first one is the computation of new coordinates for the rectification of Lie point symmetries in classical framework. This necessitates resolution of systems of PDEs. The second one is the rewriting the studied system in these new coordinates (see [59]). This change could be performed using elimination algorithms such as Gröbner basis or differential elimination (see [13]). According also to the chosen set of coordinates, one can obtain different but equivalent reduced systems. Even if the reduced system changes, its main geometrical properties such as the number of equations remain the same.

5.2.1 Rectification of a Symmetry

In this subsection, we define and illustrate the *rectification* of a symmetry.

Definition 5.2.1 (Rectified Symmetries)**.** A *rectified* (a.k.a. *normal form* in [104]) symmetry is represented by an infinitesimal generator δ of the form:

$$\delta = \frac{\partial}{\partial z} \tag{5.4}$$

where z is a coordinate. Hence the action is just a translation. ⌟

Definition 5.2.2 (Principal Element)**.** Given an infinitesimal generator δ, a function p such that $\delta p = 1$ is called a *principal element*. ⌟

Theorem 5.2.3 (see § 2.2 of [104])**.** *Let δ be an infinitesimal generator acting on the coordinate set $Z = (z_1, \ldots, z_n)$. The rectification of a symmetry corresponds to find new coordinates $\mathcal{I} = (p, \zeta_1, \ldots, \zeta_{n-1})$ such that:*

$$\delta p = 1 \quad \textit{and} \quad \delta\zeta_i = 0 \quad \forall i \in \{1, \ldots, n-1\}\,. \tag{5.5}$$

In these new coordinates, by definition, δ can be written in its rectified form. There always exist such coordinates. ⌟

Remark 5.2.4. The inverse of a *moving frame* $\tilde{\rho} = -\rho$ associated to an infinitesimal generator δ always verifies the equality $\delta\tilde{\rho} = 1$ (see lemma 5.3.17 page 113). ⌟

Example 5.2.5. Let us look at a scaling on a two dimensional space $Z = (x, y)$:

$$\delta = x\frac{\partial}{\partial x} + y\frac{\partial}{\partial y}\cdot \tag{5.6}$$

In order to write this infinitesimal generator in its rectified form, one needs to compute new coordinates (z, ζ) by solving the partial differential equations $\delta z = 1$ and $\delta\zeta = 0$.

For example, $z = \log(y)$ (supposing that y is strictly positive) and $\zeta = x/y$ verify these equations. Indeed, in these new coordinates, δ is rectified:

$$\delta = \frac{\partial}{\partial z}\cdot \tag{5.7}$$

⌟

Remark 5.2.6. The rectified form of an infinitesimal generator is not unique, it depends on the chosen new coordinates. ⌟

5.2.2 Illustrations

In the following examples, we illustrate the two main steps of the classical reduction process: the invariants computation and the elimination.

Example 5.2.7. Let us reconsider the algebraic system defined by $x^2 + y^2 - r^2 = 0$ (see example 5.1.2 page 104). The infinitesimal generator $\delta_{\mathcal{S}}$ associated to the one-dimensional symmetry $\mathcal{S}$ given in (5.2) page 104 follows:

$$\delta_{\mathcal{S}} = x\frac{\partial}{\partial x} + y\frac{\partial}{\partial y} + r\frac{\partial}{\partial r}\cdot \tag{5.8}$$

1. **Invariants Computation.** According to remark 3.4.23 page 85 there exist two independent invariants for the symmetry $\mathcal{S}$. Let us denote them by $\widetilde{x}$ and $\widetilde{y}$. They verify the partial differential equations $\delta_{\mathcal{S}}\widetilde{x} = 0$ and $\delta_{\mathcal{S}}\widetilde{y} = 0$. By solving these equations, one can take as new coordinates, for example, $\widetilde{x} = x/r, \widetilde{y} = y/r$ (given in (5.3) page 104). Remark that these invariants exist locally where the variable r is different than 0.

2. **Elimination.** Now, the goal is to rewrite the algebraic system $x^2 + y^2 - r^2 = 0$ in the new coordinate chart $\mathcal{I} = (\widetilde{x}, \widetilde{y})$. The elimination procedure applied on the system:
$$\begin{cases} x^2 + y^2 - r^2 = 0, \\ \widetilde{x}\, r - x = 0, \\ \widetilde{y}\, r - y = 0 \end{cases} \tag{5.9}$$
leads to the reduced algebraic system $\widetilde{x}^2 + \widetilde{y}^2 - 1 = 0$.

By this reduction process, as expected, the number of coordinates of the original algebraic system is decreased by one. ⌟

Remark 5.2.8. The Gröbner basis of the system (5.9) w.r.t. the lexicographic order given by $(x, y, r, \widetilde{x}, \widetilde{y})$ is:
$$\begin{cases} r^2\left(\widetilde{x}^2 + \widetilde{y}^2 - 1\right) = 0, \\ -\widetilde{y}\, r + y = 0, \\ -\widetilde{x}\, r + x = 0. \end{cases} \tag{5.10}$$

⌟

Example 5.2.9. Let us reduce the system of ODEs (3.9) given in the example 3.2.6 page 63. It is defined on the coordinate set $Z = (t, x, a, b)$. The infinitesimal generator associated to one of its one-dimensional symmetries follows:

$$\delta_{\mathcal{S}} = -x\frac{\partial}{\partial x} + b\frac{\partial}{\partial b}\cdot \tag{5.11}$$

1. **Invariants Computation.** We are looking for three invariants $\left(\widetilde{t}, \widetilde{x}, \widetilde{y}\right)$ of $\delta_{\mathcal{S}}$ that can be found by solving the partial differential equations $\delta_{\mathcal{S}}\widetilde{t} = 0, \delta_{\mathcal{S}}\widetilde{x} = 0$ and $\delta_{\mathcal{S}}\widetilde{a} = 0$. One can take:

$$\widetilde{t} = t, \quad \widetilde{x} = x\,b, \quad \widetilde{a} = a. \tag{5.12}$$

2. **Elimination.** Now, the goal is to rewrite the dynamical system (3.9) page 63 in the new coordinate chart of invariants $\mathcal{I} = \left(\widetilde{t}, \widetilde{x}, \widetilde{a}\right)$:

$$\left\{\begin{array}{rcl} \dfrac{\mathrm{d}\widetilde{x}}{\mathrm{d}\widetilde{t}} & = & \dfrac{\mathrm{d}\,(x\,b)}{\mathrm{d}t} = \dfrac{b\,\mathrm{d}x + x\mathrm{d}b}{\mathrm{d}t} = b\dfrac{\mathrm{d}x}{\mathrm{d}b}, \\ & = & (a - b\,x)\,b\,x = (\widetilde{a} - \widetilde{x})\,\widetilde{x}, \\ \dfrac{\mathrm{d}\widetilde{a}}{\mathrm{d}\widetilde{t}} & = & \dfrac{\mathrm{d}a}{\mathrm{d}t} = 0, \end{array}\right. \quad \Rightarrow \quad \left\{\begin{array}{rcl} \dfrac{\mathrm{d}\widetilde{x}}{\mathrm{d}\widetilde{t}} & = & \widetilde{x}\,(\widetilde{a} - \widetilde{x})\,, \\ \dfrac{\mathrm{d}\widetilde{a}}{\mathrm{d}\widetilde{t}} & = & 0. \end{array}\right. \tag{5.13}$$

The reduction process, as expected, eliminated the parameter b and thus the equation $\mathrm{d}b/\mathrm{d}t = 0$ from the resulting system. ⌟

The same method can be applied to discrete dynamical systems as in the following example.

Example 5.2.10. Let us illustrate the reduction of a system of OREs by using the recurrence operator σ in (3.2.18) page 65. It is defined on the coordinate set $Z = (\tau, x, y, c)$. The infinitesimal generator associated to one of its one-dimensional symmetries follows:

$$\delta_{\mathcal{S}} = x\frac{\partial}{\partial x} + y\frac{\partial}{\partial y}\cdot \tag{5.14}$$

1. **Invariants Computation.** We are looking for three invariants $(\widetilde{\tau}, \widetilde{y}, \widetilde{c})$ of $\delta_{\mathcal{S}}$ that can be found by solving the partial differential equations $\delta_{\mathcal{S}}\widetilde{\tau} = 0, \delta_{\mathcal{S}}\widetilde{y} = 0$ and $\delta_{\mathcal{S}}\widetilde{c} = 0$. One can take:

$$\widetilde{\tau} = \tau, \quad \widetilde{y} = \frac{y}{x}, \quad \widetilde{c} = c. \tag{5.15}$$

   Remark that these invariants exist locally where x is different than 0.

2. **Elimination.** Now, the goal is to rewrite the dynamical system (3.11) page 64 in the new coordinate chart of invariants $\mathcal{I} = (\widetilde{\tau}, \widetilde{y}, \widetilde{c})$. The recurrence operator σ is an endomorphism thus we obtain (or for example using Gröbner bases):

$$\sigma\widetilde{\tau} = \widetilde{\tau} + 1, \quad \sigma\widetilde{y} = \frac{\sigma y}{\sigma x} = \frac{\left(\frac{y}{x}\right) + 1}{c\,x}x = \frac{\widetilde{y} + 1}{\widetilde{c}}, \quad \sigma\widetilde{c} = \widetilde{c}. \tag{5.16}$$

The reduction process, as expected, eliminated the state variable x and thus the equation $\sigma x = c\,x$ from the resulting system. ⌟

Remark that the classical reduction method is an exact reduction method meaning that there exist relations (fibers) between the original and the reduced systems coordinates given by the invariants. Its limitations can be seen as systems and infinitesimal generators of their symmetries complicate. In the previous examples, we use scaling type symmetries so that the invariants can be computed easily. The next example illustrates a more difficult case.

Example 5.2.11. Let us consider the following algebraic system (we are not interested in the point defined by $x = y = a = b = c = 0$):

$$\begin{cases} (y-b)^2 + a^2 - \frac{c^2}{4} = 0 \\ (x-a)^2 + b^2 - \frac{c^2}{4} = 0 \\ x^2 + y^2 - c^2 = 0. \end{cases} \tag{5.17}$$

The following infinitesimal generator is associated to one of its one-dimensional symmetry groups:

$$\delta_{\mathcal{S}} = 4\,c\,(b-y)\,\frac{\partial}{\partial x} + 4\,c\,a\frac{\partial}{\partial y} + c\,(4\,b - 3\,y)\,\frac{\partial}{\partial a} + x\,c\frac{\partial}{\partial b} + 4\,(y\,a + b\,x - y\,x)\,\frac{\partial}{\partial c}\cdot \tag{5.18}$$

This quadratic type symmetry acts on the parameter c ($\delta_{\mathcal{S}} c \neq 0$) so it could be used to eliminate it from the studied system. What are the associated invariants? One needs to solve partial differential equations of the form $\delta_{\mathcal{S}}\zeta = 0$. Perhaps, some special techniques could give solutions to this example but there is no a general algorithm for such computations. ⌟

The main idea of all these examples is the projection (see § 5.1 page 103) of the system variety onto an appropriate submanifold. The fiber i.e. the reduced system and the orbits of the associated symmetry keep necessary informations to reconstruct solutions of the original system. So the explicit expressions of the invariants are not essential in the reduction process. In the next section, let us describe roughly the notion of *moving frames*. In the sequel, the *moving frame based reduction process* shows how one can reduce a system without computing associated invariants and avoiding as much as possible nonlinear elimination steps.

5.3 Around Moving Frames

This section provides the preliminaries for the moving frame based reduction process and some related notions. We begin by defining *cross-sections* that are used for the projection of the reduction process. Then we define *moving frames* as they are used in this document and some of their properties. Remark that we employ a consequence of the *recurrence formula* given in § 13 of [38]. Finally we show how to use these moving frames for the invariants computation and for the rectification of symmetries.

5.3.1 Cross-Section

In moving frame based reduction process, a *cross-section* is one of the key objects. It corresponds to the submanifold on which the original system is projected in order to obtain the reduced system.

Definition 5.3.1 (Cross-Section)**.** Let G be a r-dimensional local Lie group of transformation acting regularly on $\mathbb{R}^n$ where Z is the coordinate set. A *cross-section* is a $(n-r)$-dimensional submanifold $\mathcal{H} \subset \mathbb{R}^n$ of equations:

$$h_1(Z) = \cdots = h_r(Z) = 0 \tag{5.19}$$

such that

- $\mathcal{H}$ intersects locally the orbits of G at least in one point;
- $\mathcal{H}$ intersects these orbits transversely i.e. for each infinitesimal generator δ associated to G, the relation $\delta h_i \neq 0$ holds for all i in $\{1, \ldots, r\}$.

The cross-section is said to be *regular* if $\mathcal{H}$ intersects each orbit at most once. ⌟

Remark 5.3.2. If G acts (semi-)regularly, then the Implicit Function Theorem guarantees the existence of regular local cross-sections at any point of M (see § 2 of [37]). ⌟

Remark 5.3.3. There are infinitely many cross-sections that can be used for a reduction process. Thus it must be chosen carefully in order to facilitate computations. However, even if two different cross-sections lead to two different reduced systems, their main geometrical properties, as the dimension of the system, remain the same. ⌟

Hypothesis 5.3.4. *In the following algorithms, one focuses on* local *cross-sections (even if it is not mentioned explicitly) and chosen cross-sections are taken linear w.r.t. the coordinates that we want to eliminate.* ⌟

5.3.2 Moving Frame

Let us give formal definition of a *moving frame* as it is used along this document, in a restricted framework. The goal of defining moving frames is to use them to reduce an algebraic or dynamical system without any computation of invariants or need to integrate any differential equation.

Definition 5.3.5 (Moving Frame)**.** Let G be a r-dimensional local Lie group acting regularly on n-dimensional manifold $\mathbb{R}^n$ with $r \leq n$. A *moving frame* is a smooth local G-equivariant map (for group elements near the neutral element e of G):

$$\rho : \mathbb{R}^n \to G. \tag{5.20}$$

⌟

Remark 5.3.6. Each moving frame of this document is a *right* moving frame. Although, there is an elementary correspondence between right and left moving frames (see th. 4.2 in § 4 of [38]). ⌟

The notion of *G-equivariance* indicates the commutativity between the moving frame and the group action.

Definition 5.3.7 (*G*-Equivariance of a Moving Frame)**.** Let G be a r-dimensional local Lie symmetry group associated to a Lie symmetry $\mathcal{S}$. The G-equivariance of a moving frame $\rho : \mathbb{R}^n \to G$ implies the commutativity property through the following relation for all ν in G and Z in $\mathbb{R}^n$:

$$\rho(\mathcal{S}(\nu, Z)) = \rho(Z) + (-\nu) \quad \text{with} \quad (-\nu) + \nu = e \tag{5.21}$$

where $+$ is the group operation. ⌟

Hypothesis 5.3.8. *In the context of this document, such a moving frame always exists in a neighborhood of a point Z of $\mathbb{R}^n$ because G is supposed to act regularly near Z (see th. 4.4 in § 4 of [38]).* ⌟

In our framework, the determination of a moving frame relies on the choice of a local cross-section as one can see in the following theorem.

Theorem 5.3.9 (Moving Frame, see th. 2.3 in § 2 of [89])**.** *Let G, a r-dimensional local Lie symmetry group associated to a Lie symmetry $\mathcal{S}$, act regularly (and thus freely) on $\mathbb{R}^n$. Let $\mathcal{H} \subset \mathbb{R}^n$ be a local cross-section. Given Z in $\mathbb{R}^n$, let $\rho(Z)$ be the unique group element that maps Z to the cross-section i.e.:*

$$\mathcal{S}(\rho(Z), Z) \in \mathcal{H}. \tag{5.22}$$

Then $\rho : \mathbb{R}^n \to G$ is a right moving frame. ⌟

Geometrically, the group element $\rho(Z)$ given by a moving frame is characterized as the unique group transformation that moves the point Z onto the cross-section $\mathcal{H}$. Moreover, a point $\mathcal{S}(\rho(Z), Z)$ lies on $\mathcal{H}$ and on the orbit $\mathrm{Orb}_{(\mathcal{S},Z)}$ passing through Z.

Remark 5.3.10. The moving frame ρ is related to the classical reduction process because it verifies $\delta_{\mathcal{S}}\rho = -1$ where $\delta_{\mathcal{S}}$ is the infinitesimal generator of the associated one-dimensional symmetry group (see proof of the proposition 5.3.17 page 113). Thus it is a part of the computation summarized in theorem 5.2.3 page 105. ⌟

The following lemma states how to choose a local moving frame from a given cross-section by using theorem 5.3.9.

Lemma 5.3.11 (see § 2 of [89])**.** *Let G be a r-dimensional local symmetry group acting on $\mathbb{R}^n$ and $\mathcal{H} \subset \mathbb{R}^n$ a local cross-section to the orbits of G. The* normalization equations *associated to $\mathcal{H}$ are the system of equations:*

$$h_i(\mathcal{S}(\rho(Z), Z)) = 0 \quad \forall i \in \{1, \ldots, r\}. \tag{5.23}$$

Assuming that G acts freely and $\mathcal{H}$ is a regular cross-section, there is a unique local solution $\rho(Z)$ to these algebraic equations determining the moving frame ρ associated to $\mathcal{H}$. ⌟

The next example illustrates all these notions on a one-dimensional Lie symmetry group.

Example 5.3.12. Consider the following continuous dynamical system of two state variables where $Z = (t, x, y)$:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = y, \quad \frac{\mathrm{d}y}{\mathrm{d}t} = \frac{y}{x}. \tag{5.24}$$

Let $\mathcal{S}$ be a one-dimensional Lie symmetry of this system with:

$$\begin{array}{rccl} \mathcal{S}: & G \times \mathbb{R}^3 & \to & \mathbb{R}^3, \\ & \nu \times Z & \to & \mathcal{S}(\nu, Z) = \begin{pmatrix} t\, e^{\nu} \\ x\, e^{\nu} \\ y \end{pmatrix}. \end{array} \tag{5.25}$$

This group action is associated to the following infinitesimal generator:

$$\delta_{\mathcal{S}} = t\frac{\partial}{\partial t} + x\frac{\partial}{\partial x}. \tag{5.26}$$

One can correspond a 2-dimensional local cross-section to this one-dimensional symmetry group. Let us choose the cross-section $\mathcal{H}$ by respecting the conditions of definition 5.3.1 page 109 but quite arbitrarily $h(Z) = x - 1$. The associated normalization equation is of the form:

$$h(\mathcal{S}(\rho(Z)\,, Z)) = x\, e^{\rho(Z)} - 1 = 0. \tag{5.27}$$

The normalization formula issue from this normalization equation states the expression of the moving frame to $\rho(Z) = -\log(x)$. ⌟

Remark 5.3.13. The moving frame reduction process detailed in the forthcoming sections does *not* require the computation of moving frames; it relies on the *recurrence formula* (see § 13 of [38]). ⌟

5.3.3 Invariants Computation using Moving Frames

We recall here the moving frame based invariant computation to show that this method provides tools to compute the invariants *if they are asked*. In my work, I do *not* compute the invariants. They require the associated explicit group action that is given by the resolution of PDEs systems. Again, there is no a general algorithm for such problems. Considering that one already has the group action, moving frame based invariant computation is a fully algorithmic method and complete presentation can be found in § 4 of [38] and in [59].

Theorem 5.3.14 (see th. 8.25 of [88])**.** *The substitution of the normalization formula $\rho(Z)$ obtained from resolution of normalization equations (5.23) page 110 into the group action, leads to $n - r$ functionally independent invariants. This* invariantization *of coordinates yields fundamental invariants denoted by $\mathcal{I} := \mathcal{S}(\rho(Z)\,, Z)$ (one needs to remove numerical projections). The point Z is projected on $\mathcal{H}$ and the invariants $\mathcal{I}$ of $\mathcal{S}$ are the coordinates induced by this projection.* ⌟

The following examples illustrate this computation in simple cases of scalings.

Example 5.3.15. Reconsider the example 5.3.12 page 111. A set of independent invariants is found by substituting the moving frame $\rho(Z) = -\log(x)$ into the parameter ν of the group action (5.25) page 111:

$$\mathcal{S}(\rho(Z), Z) = \begin{pmatrix} t\, e^{\rho(Z)} \\ x\, e^{\rho(Z)} \\ y \end{pmatrix} = \begin{pmatrix} t\, e^{-\log(x)} \\ x\, e^{-log(x)} \\ y \end{pmatrix} = \begin{pmatrix} \frac{t}{x} \\ 1 \\ y \end{pmatrix}. \tag{5.28}$$

Discarding the numerical constant in above projection (it corresponds to the coordinate to be eliminated), one finds the invariants associated to the Lie symmetry $\mathcal{S}$ i.e. the set $\mathcal{I} = (t/x, y)$. ⌟

Example 5.3.16. Consider now, the discrete dynamical system defined on $Z = (\tau, x, y)$:

$$\sigma\tau = \tau + 1, \quad \sigma x = 2\,x - y, \quad \sigma y = x. \tag{5.29}$$

Let $\mathcal{S}$ be a one-dimensional Lie symmetry of this system with:

$$\begin{array}{rccc} \mathcal{S}: & G \times \mathbb{R}^3 & \to & \mathbb{R}^3, \\ & \nu \times Z & \to & \mathcal{S}(\nu, Z) = \begin{pmatrix} \tau \\ x\, e^{\nu} \\ y\, e^{\nu} \end{pmatrix}. \end{array} \tag{5.30}$$

This group action is associated to the following infinitesimal generator:

$$\delta_{\mathcal{S}} = t\frac{\partial}{\partial t} + x\frac{\partial}{\partial x}\cdot \tag{5.31}$$

One can correspond a 2-dimensional local cross-section to this one-dimensional symmetry group. Let us choose the cross-section $\mathcal{H}$ by respecting the conditions of definition 5.3.1 page 109 but quite arbitrarily $h(Z) = x - 1$. The associated normalization equation is of the form:

$$h(\mathcal{S}(\rho(Z), Z)) = x\, e^{\rho(Z)} - 1 = 0. \tag{5.32}$$

The normalization formula issue from this normalization equation states the expression of the moving frame to $\rho(Z) = -\log(x)$. A set of independent invariants is found by substituting this moving frame into the parameter ν of the group action (5.30):

$$\mathcal{S}(\rho(Z), Z) = \begin{pmatrix} \tau \\ x\, e^{\rho(Z)} \\ y\, e^{\rho(Z)} \end{pmatrix} = \begin{pmatrix} \tau \\ x\, e^{-log(x)} \\ y\, e^{-\log(x)} \end{pmatrix} = \begin{pmatrix} \tau \\ 1 \\ \frac{y}{x} \end{pmatrix}. \tag{5.33}$$

Discarding the numerical result, one finds the invariants associated to this symmetry i.e. $\mathcal{I} = (\tau, y/x)$. ⌟

5.3.4 Rectification of a Symmetry using Moving Frames

The following proposition states a way of choosing a new coordinate set so that an infinitesimal generator associated to a Lie symmetry can be rewritten in its rectified or semi-rectified form.

Proposition 5.3.17. *Let $\delta_{\mathcal{S}}$ be an infinitesimal generator acting on the coordinate set Z of dimension n with associated $n-1$ invariants $\mathcal{I} = (\zeta_1, \ldots, \zeta_{n-1})$.*

1. *In the new coordinate set composed by these invariants and the inverse of the moving frame i.e. $\tilde{\rho} = -\rho$ (when it exists), this infinitesimal generator can be rewritten in its rectified form $\delta_{\mathcal{S}} = \partial/\partial\tilde{\rho}$.*
2. *In the new coordinate set composed by these invariants and a semi-invariant μ, this infinitesimal generator can be rewritten in its semi-rectified form $\delta_{\mathcal{S}} = \alpha\,\mu\partial/\partial\mu$ with α in $\mathbb{R}$.* ⌟

Proof. A one-dimensional symmetry acting on n coordinates has $n-1$ independent invariants (see remark 3.4.23 page 85), denoted by $\mathcal{I} = (\zeta_1, \ldots, \zeta_{n-1})$. They verify $\delta_{\mathcal{S}}\zeta_i = 0$ (see § 3.4.5 page 84) for all i in $\{1, \ldots, n-1\}$.

1. Let us show that the inverse of a moving frame $\tilde{\rho} = -\rho$ verifies $\delta_{\mathcal{S}}\tilde{\rho} = 1$. This can be deduced from the G-equivariance property of moving frames given in definition 5.3.7 page 110. The equality (5.21) page 110 induces the following one:

$$\mathcal{S}(\rho(\mathcal{S}(\epsilon, Z)), Z) = \mathcal{S}(\rho(Z) - \epsilon, Z) \tag{5.34}$$

for all ϵ in the neighborhood of the neutral element 0 of G. The equation (5.34) leads to the following relations by considering the continuity properties (see equation (3.60) page 81 and (3.61) page 82) of the symmetry $\mathcal{S}$:

$$\begin{aligned}
\mathcal{S}(\epsilon, \mathcal{S}(\rho(\mathcal{S}(\epsilon, Z)), Z)) &= \mathcal{S}(\rho(Z), Z), \\
\mathcal{S}\big(\epsilon, \mathcal{S}\big(\rho\big(Z + \epsilon\,\delta_{\mathcal{S}}Z + \mathrm{O}\big(\epsilon^2\big)\big), Z\big)\big) &= \mathcal{S}(\rho(Z), Z), \\
\mathcal{S}\big(\epsilon, \mathcal{S}\big(\rho(Z) + \epsilon\tfrac{\partial\rho}{\partial Z}(Z)\,\delta_{\mathcal{S}}Z + \mathrm{O}\big(\epsilon^2\big), Z\big)\big) &= \mathcal{S}(\rho(Z), Z), \\
\mathcal{S}\big(\epsilon, \mathcal{S}(\rho(Z), Z) + \epsilon\tfrac{\partial\rho}{\partial Z}(Z)\,\delta_{\mathcal{S}}Z\delta_{\mathcal{S}}\mathcal{S}(\rho(Z), Z) + \mathrm{O}\big(\epsilon^2\big)\big) &= \mathcal{S}(\rho(Z), Z), \\
\mathcal{S}(\rho(Z), Z) + \epsilon\tfrac{\partial\rho}{\partial Z}(Z)\,\delta_{\mathcal{S}}Z\delta_{\mathcal{S}}\mathcal{S}(\rho(Z), Z) + \epsilon\delta_{\mathcal{S}}\mathcal{S}(\rho(Z), Z) + \mathrm{O}\big(\epsilon^2\big) &= \mathcal{S}(\rho(Z), Z).
\end{aligned} \tag{5.35}$$

This latest relation implies the following one:

$$\frac{\partial\rho}{\partial Z}(Z)\,\delta_{\mathcal{S}}Z\delta_{\mathcal{S}}\mathcal{S}(\rho(Z), Z) = -\delta_{\mathcal{S}}\mathcal{S}(\rho(Z), Z) \tag{5.36}$$

which leads to the equality $\partial\rho/\partial Z(Z)\,\delta_{\mathcal{S}}Z = -1$. Hence, one can conclude that for a right moving frame ρ, $\delta_{\mathcal{S}}\rho = -1$ and for its inverse $\tilde{\rho}$, $\delta_{\mathcal{S}}\tilde{\rho} = 1$ hold.

2. By definition, a semi-invariant μ verifies the relation $\delta_{\mathcal{S}}\mu = \alpha\,\mu$ with α in $\mathbb{R}$.

Thus, in the given new coordinate set, the infinitesimal generator can be rewritten in its rectified (resp. semi-rectified) form in the first (resp. second) case. ⌟

Remark 5.3.18. The sign of the expression $\delta_S\rho$ depends whether we choose right or left moving frames. ⌟

5.4 Moving Frame Based Reduction Process

The moving frame method is first introduced by DARBOUX (see [29]) in the XIX$^{\text{th}}$ century, then developed by CARTAN (see [24]). In [37], the authors FELS and OLVER say: "*The theory of moving frame is acknowledged to be a powerful tool for studying the geometric properties of submanifolds under the action of a transformation group*". The method of moving frames can be directly applied to practical problems arising in geometry, invariant theory or differential equations.

This section is devoted to moving frame based reduction process also known as *invariantization process*, not in its whole generality but as it is implemented in the package `ExpandedLiePointSymmetry`. This is a special case of the *recurrence formula* given in § 13 of [38] or in § 5 of [65]. The goal of recalling it here is the simplicity of its adaptation in our framework. We present the reduction of an algebraic and a dynamical system w.r.t. a single symmetry i.e. one-dimensional Lie algebra. Then we discuss the reduction w.r.t. symmetries with an associated Lie algebra of more than one dimension. Finally, the implementation is detailed.

Here, the aim of the moving frame based reduction process is to represent a given algebraic or dynamical system by a lower dimensional system and fibers. The standpoint of this section implies polynomial complexity in the input size for presented computations. In fact, the moving frame method leads to a reduction process for which neither the moving frame nor the invariants need to be computed (see § 13 of [38] for the *recurrence formula*).

In this document, the inputs of this process are an algebraic or a dynamical system (see ch. 3 page 59 for their representation) and its Lie point symmetries (see ch. 4 page 87 for their computation). The output is the representation of the input system by a lower dimensional system and the implicit fibers. These fibers constitute the link between the original and the reduced system. When these fibers are explicitly known, they correspond to invariants of the used symmetries. In this document, we do not compute invariants explicitly. Instead we propose to compute numerically the corresponding fibers. Given a cross-section, the moving frame application gives the unique element of the used symmetry group that maps a point of the variety defined by the input system to another point in this cross-section.

Hypothesis 5.4.1. *In this section, used local Lie symmetry groups are of dimension one thus they can be represented by a single infinitesimal generator. For higher dimensional symmetry groups, the structure of Lie algebras permits to perform an incremental reduction.* ⌟

Hypothesis 5.4.2. *In this section, computations rely upon the chosen local cross-section $\mathcal{H}$ and associated moving frame ρ. In our framework, cross-sections depend*

on the coordinate z to eliminate (see hyp. 5.3.4 page 109) and they verify the relation $\partial\mathcal{H}/\partial z \neq 0$. Moreover, as the used local Lie symmetry group is of dimension one, the dimension of the variety defined by $\mathcal{H}$ is $n-1$ where n is the dimension of the original system. ⌟

5.4.1 Reduction of an Algebraic System

This subsection explains and illustrates the reduction of an algebraic system with a one dimensional local Lie symmetry group using moving frames.

Let us sketch by the following example the moving frame based reduction process of an algebraic system without computing neither the moving frame nor the invariants.

Example 5.4.3. Reconsider the algebraic system defined by $x^2+y^2-r^2=0$ and its variety $\mathcal{V}$ (see example 5.1.2 page 104). Let us reduce the coordinate r by using moving frame based reduction process. The infinitesimal generator associated to the symmetry group action given in (5.2) page 104 follows:

$$\delta_{\mathcal{S}} = x\frac{\partial}{\partial x} + y\frac{\partial}{\partial y} + r\frac{\partial}{\partial r}\cdot \tag{5.37}$$

The reduction of r is possible because $\mathcal{S}$ acts on r i.e. $\delta_{\mathcal{S}} r \neq 0$ and the orbits of $\mathcal{S}$ are transversal to $\mathcal{V}$. The variety $\mathcal{H}$ defined by $r-1=0$ can be chosen as a cross-section. Remark that it is not included in $\mathcal{V}$ but intersects it and is linear w.r.t. the coordinate r. The moving frame based reduction process consists of specializing the coordinate r to 1 as imposed by the cross-section. So the reduced algebraic system $\widetilde{x}^2+\widetilde{y}^2-1=0$ is in the cross-section $\mathcal{H}$ where $\widetilde{x}$ and $\widetilde{y}$ are new coordinates (invariants). Remark that the reduced variety $\widetilde{\mathcal{V}}$ is the projection of $\mathcal{V}$ in $\mathcal{H}$ along the orbits $\mathrm{Orb}_{(\mathcal{S},Z)}$ of $\mathcal{S}$. The fiber, i.e. this reduced variety and the orbits of $\mathcal{S}$, is enough to represent the original variety $\mathcal{V}$. ⌟

The following result summarizes this reduction process.

Proposition 5.4.4 (see th. 4.3 in § 4.1 of [59])**.** *Let G be a group action and $\mathcal{V}$ the variety defined by an algebraic system in $\mathbb{R}(Z)^G$ i.e. an expression invariant under the action of G. Let $\mathcal{H}$ be a local cross-section linear w.r.t. a coordinate z, on which G acts. We assume that $\mathcal{H} \not\subset \mathcal{V}$ and $\mathcal{H}\cap\mathcal{V} \neq \emptyset$. $\mathcal{H}$ defines the invariants $\zeta_1,\ldots,\zeta_{n-1}$ (see § 5.3.3 page 111). The rewriting of the algebraic system in $\mathbb{R}(\zeta_1,\ldots,\zeta_{n-1})$ is given by the projection of $\mathcal{V}$ onto the cross-section $\mathcal{H}$ i.e., in our case, by the specialization of z w.r.t. the cross-section $\mathcal{H}$.* ⌟

Remark 5.4.5. The reduced algebraic system obtained by the proposition 5.4.4 is equal to the result of the classical reduction process and the chosen invariants are associated to moving frame (see § 5.3.3 page 111) when the principal element in (5.5) page 105 is this moving frame. ⌟

5.4.2 Reduction of a Dynamical System

The *recurrence formula* given in § 13 of [38] uses r-dimensional symmetry groups in order to reduce continuous dynamical systems. This formula is expressed in another formalism by theorem 3.6 in [58]. In our work we apply this formula to one-dimensional symmetry groups. Thus, for greater dimensional symmetries, the link between the original and the reduced systems are expressed by a succession of one-parameter fibers (the succession is structured by the properties of the symmetries Lie algebra).

This subsection deals with the reduction of a dynamical system in two cases (as the reduction process is a projection, we call it *elimination* in the sequel). The first one is the elimination of a state variable. In this document, we present only continuous dynamical systems for this case. For discrete dynamical systems a similar method can be applied under some conditions. This research subject is left to future work. The second case is the elimination of a parameter where the used method is common for both, continuous and discrete, dynamical systems.

State Variable Elimination

In this paragraph, we deal with the *elimination* of a state variable from continuous dynamical systems. Such a reduction can be done by moving frame based methods without computing any explicit group action, moving frame or invariant. The following proposition states a simple case of the *recurrence formula* (see § 13 of [38] for a presentation in an extern calculus framework, [59] for a presentation in an algebraic geometric framework and [58] for a presentation with derivation in a jet space).

Proposition 5.4.6. *Let $\mathcal{D}$ be a continuous dynamical system with G as transformation group, $\delta_{\mathcal{D}}$ associated infinitesimal generator and x a state variable ($\delta_{\mathcal{D}} x \neq 0$). Let $\mathcal{S}$ be a one-dimensional local Lie symmetry of $\mathcal{D}$ and $\delta_{\mathcal{S}}$ associated infinitesimal generator. We assume that $\mathcal{S}$ acts on x i.e. $\delta_{\mathcal{S}} x \neq 0$. Let also $\mathcal{H} : h(Z) = 0$ be a cross-section associated to $\mathcal{S}$. In this case, the orbits of $\mathcal{D}$ do not have to stay in this cross-section i.e. $\delta_{\mathcal{D}} h(Z) \neq 0$. The following integral relation, illustrated in figure 5.2 page 117 holds:*

$$\forall\, (\nu, Z) \in (G, \mathcal{H}), \quad \mathcal{S}(\rho(\mathcal{D}(\nu, Z)), \mathcal{D}(\nu, Z)) \in \mathcal{H}. \tag{5.38}$$

This proposition provides the infinitesimal version of this statement. In these notations, the moving frame based reduction of a state variable is a specialization of the state variable x w.r.t. the chosen cross-section (following hypothesis 5.3.4 page 109 the cross-section is linear w.r.t. x) in the following infinitesimal generator:

$$\delta_{\mathcal{D}} - \frac{\delta_{\mathcal{D}} x}{\delta_{\mathcal{S}} x} \delta_{\mathcal{S}}. \tag{5.39}$$

The resulting infinitesimal generator denoted by $\delta_{\widetilde{\mathcal{D}}}$ is associated to the reduced dynamical system $\widetilde{\mathcal{D}} = \mathcal{S}(\rho(\mathcal{D}(\nu, Z)), \mathcal{D}(\nu, Z))$. ⌟

Proof. The moving frame based method let work directly with infinitesimal expressions. The aim of this proof is to determine the infinitesimal generator $\delta_{\widetilde{\mathcal{D}}}$ encoding the reduced

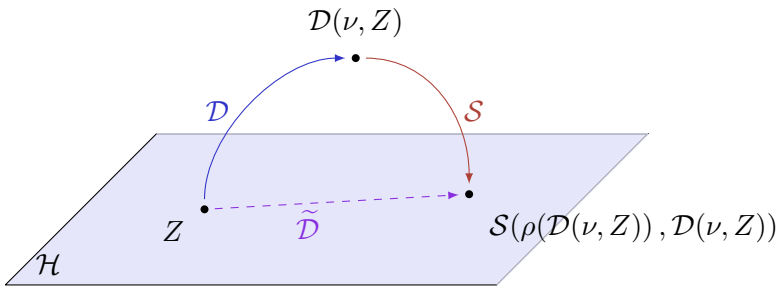


Figure 5.2: Reduction of a dynamical system.

dynamical system $\widetilde{\mathcal{D}}$ by using $\delta_{\mathcal{D}}$ and the infinitesimal generator $\delta_{\mathcal{S}}$ of the used Lie point symmetry. We use an analytic framework in order to explicit the *recurrence formula* presented in § 13 of [38] in our special case.

First, remark that any point Z in the chosen cross-section $\mathcal{H}$ satisfies the relation $h(Z) = 0$ and also one has $\mathcal{S}(\rho(Z), Z) = Z$. Furthermore, for such a point and any ϵ in the neighborhood of 0 in $\mathbb{R}$, the following relation holds:

$$h(\mathcal{S}(\rho(\mathcal{D}(\epsilon, Z)), \mathcal{D}(\epsilon, Z))) = 0. \tag{5.40}$$

Using the properties (3.60) page 81 and (3.61) page 82, one can realize the following series of computation from (5.40):

$$\begin{aligned}
&h(\mathcal{S}(\rho(\mathcal{D}(\epsilon, Z)), \mathcal{D}(\epsilon, Z))) = 0 \\
&\quad = h\big(\mathcal{S}\big(\rho(\mathcal{D}(\epsilon, Z)), Z + \epsilon\delta_{\mathcal{D}} Z + \mathrm{O}(\epsilon^2)\big)\big), \\
&\quad = h\Big(\mathcal{S}(\rho(\mathcal{D}(\epsilon, Z)), Z) + \epsilon \tfrac{\partial \mathcal{S}(\rho(\mathcal{D}(\epsilon, Z)), Z)}{\partial Z} \delta_{\mathcal{D}} Z + \mathrm{O}(\epsilon^2)\Big), \\
&\quad = h\Big(\mathcal{S}\Big(\rho(Z) + \epsilon \tfrac{\partial \rho}{\partial Z}(Z)\, \delta_{\mathcal{D}} Z + \mathrm{O}(\epsilon^2), Z\Big) + \epsilon \tfrac{\partial \mathcal{S}(\rho(Z), Z)}{\partial Z} \delta_{\mathcal{D}} Z + \mathrm{O}(\epsilon^2)\Big), \\
&\quad = h\Big(Z + \epsilon \tfrac{\partial \rho}{\partial Z}(Z)\, \delta_{\mathcal{D}} Z \delta_{\mathcal{S}} Z + \epsilon \delta_{\mathcal{D}} Z + \mathrm{O}(\epsilon^2)\Big), \\
&\quad = h(Z) + \epsilon \tfrac{\partial h}{\partial Z}(Z) \Big(\tfrac{\partial \rho}{\partial Z}(Z)\, \delta_{\mathcal{D}} Z \delta_{\mathcal{S}} Z + \delta_{\mathcal{D}} Z\Big) + \mathrm{O}(\epsilon^2), \\
&\quad = h(Z) + \epsilon \tfrac{\partial h}{\partial Z}(Z) \left(\delta_{\mathcal{D}} \rho(Z)\, \delta_{\mathcal{S}} Z + \delta_{\mathcal{D}} Z\right) + \mathrm{O}(\epsilon^2).
\end{aligned} \tag{5.41}$$

As already said, $h(Z) = 0$ so the coefficient of ϵ in above power series expansion is 0. Thus $(\delta_{\mathcal{D}} + \mu\, \delta_{\mathcal{S}})\, h = 0$ where $\mu = \delta_{\mathcal{D}} \rho(Z)$ and is unknown. The projection of the continuous dynamical system has $\delta_{\mathcal{D}} + \mu\, \delta_{\mathcal{S}}$ as infinitesimal generator (see 3.65 page 82). On the other hand, the new infinitesimal generator must not depend on the state variable x that one wants to eliminate meaning that the relation $(\delta_{\mathcal{D}} + \mu\, \delta_{\mathcal{S}})\, x = 0$ holds. Thus μ is equal to $-\delta_{\mathcal{D}} x / \delta_{\mathcal{S}} x$. Finally, as $\widetilde{\mathcal{D}}$ remains in the cross-section, the specialization of the infinitesimal generator given in (5.39) page 116 w.r.t. the chosen cross-section gives the infinitesimal generator $\delta_{\widetilde{\mathcal{D}}}$ of the reduced continuous dynamical system. ⌟

The following example shows the reduction of a state variable from a continuous dynamical system. First, we use moving frame based reduction method and reach the

reduced dynamical system without any integration or any differential elimination. Then we perform the same process by using the classical reduction method in order to illustrate the relationship with the first computations.

Example 5.4.7. The following continuous dynamical system $\mathcal{D}$ and associated infinitesimal generator has 2 state variables where $Z = (t, x, y)$:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = y,\ \frac{\mathrm{d}y}{\mathrm{d}t} = \frac{y}{x} \quad \text{and} \quad \delta_{\mathcal{D}} = \frac{\partial}{\partial t} + y\frac{\partial}{\partial x} + \frac{y}{x}\frac{\partial}{\partial y}\cdot \tag{5.42}$$

By using the local Lie point symmetry $\mathcal{S}$ associated to:

$$\delta_{\mathcal{S}} = t\frac{\partial}{\partial t} + x\frac{\partial}{\partial x} \tag{5.43}$$

one can eliminate the state variable x because $\delta_{\mathcal{S}} x \neq 0$. Let us choose, as always, a cross-section $\mathcal{H} : h(Z) = 0$ that locally and transversely intersects the orbits of $\mathcal{S}$. For example, let us take $\mathcal{H} : x - 1 = 0$ that verifies $\delta_{\mathcal{S}} h(Z) \neq 0$ and linear w.r.t. x. Remark that, in this case the orbits of $\mathcal{D}$ do not stay in the cross-section (see figure 5.2 page 117) i.e. $\delta_{\mathcal{D}}\,(x-1) \neq 0$. In order to apply the proposition 5.4.6 page 116, one must compute the infinitesimal generator associated to the projection of the original dynamical system in the cross-section by using the formula (5.39) page 116:

$$\delta_{\mathcal{D}} - \frac{\delta_{\mathcal{D}} x}{\delta_{\mathcal{S}} x}\delta_{\mathcal{S}} = \left(\frac{\partial}{\partial t} + y\frac{\partial}{\partial x} + \frac{y}{x}\frac{\partial}{\partial y}\right) - \frac{y}{x}\left(t\frac{\partial}{\partial t} + x\frac{\partial}{\partial x}\right) = \left(1 - \frac{y\,t}{x}\right)\frac{\partial}{\partial t} + \frac{y}{x}\frac{\partial}{\partial y}\cdot \tag{5.44}$$

The specialization of the variable x to 1 as imposed by the chosen cross-section gives the infinitesimal generator of the reduced continuous dynamical system $\widetilde{\mathcal{D}}$ in the new coordinate chart $(\widetilde{t}, \widetilde{y})$:

$$\delta_{\widetilde{\mathcal{D}}} = \left(1 - \widetilde{y}\,\widetilde{t}\right)\frac{\partial}{\partial \widetilde{t}} + \widetilde{y}\frac{\partial}{\partial \widetilde{y}}\cdot \tag{5.45}$$

Now, let us use the classical reduction method to reach the same result.

1. **Invariants Computation.** Let us denote the invariants that we are looking for by $\widetilde{x}$ and $\widetilde{y}$. They verify the differential equations $\delta_{\mathcal{S}}\widetilde{x} = 0$ and $\delta_{\mathcal{S}}\widetilde{y} = 0$. The moving frame allows to compute these invariants. Thus, following the theorem 5.3.14 page 111, one has the explicit expressions of some invariants:

$$\widetilde{t} = \frac{t}{x}, \quad \widetilde{y} = y. \tag{5.46}$$

2. **Elimination.** For this step, one can compute $\delta_{\mathcal{D}}\widetilde{t}$ and $\delta_{\mathcal{D}}\widetilde{y}$ w.r.t. $\widetilde{t}$ and $\widetilde{y}$ and perform a differential elimination using [13]. But one can also use the dual form of the infinitesimal generator $\delta_{\mathcal{D}}$:

$$\mathrm{d}x = y\,\mathrm{d}t, \quad x\,\mathrm{d}y = y\,\mathrm{d}t \tag{5.47}$$

as follows. The exterior derivation of $\widetilde{t}$ implies that the relation $x\,\mathrm{d}\widetilde{t} + \widetilde{t}\,\mathrm{d}x = \mathrm{d}t$ holds; using (5.47), one can eliminate $\mathrm{d}x$ in this relation to obtain:

$$x\,\mathrm{d}\widetilde{t} + \widetilde{t}\,x\,\mathrm{d}y = \frac{x\,\mathrm{d}y}{y}\cdot \tag{5.48}$$

Using explicit definition of the invariants $\widetilde{t}$ and $\widetilde{y}$, other variables can be eliminated from (5.48) page 118 to obtain:

$$\widetilde{y}\,\mathrm{d}\widetilde{t} = \left(1 - \widetilde{t}\,\widetilde{y}\right)\mathrm{d}\widetilde{y}. \tag{5.49}$$

Thus the associated reduced dynamical system is associated to the infinitesimal generator given in (5.45) page 118. ⌟

Remark 5.4.8. As one can see, neither the invariants nor the moving frame is computed by this moving frame based reduction method, only implicit representations are used. The original system is represented by the reduced system and the orbits of symmetry $\mathcal{S}$. This fiber is implicitly encoded by differential equations associated to $\delta_{\mathcal{S}}$ and could be numerically computed. ⌟

Remark 5.4.9. Until now, in order to be clear as much as possible, considered examples were based on scalings. The importance of moving frame based reduction process can be seen if symmetries are more complicated. Imagine that one needs to eliminate the parameter b from the following continuous dynamical system defined on $Z = (x, y, p, q, a, b)$:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = p,\ \frac{\mathrm{d}y}{\mathrm{d}t} = q,\ \frac{\mathrm{d}p}{\mathrm{d}t} = -2\,a\,x,\ \frac{\mathrm{d}q}{\mathrm{d}t} = b - 2\,a\,y,\ \frac{\mathrm{d}a}{\mathrm{d}t} = 0,\ \frac{\mathrm{d}b}{\mathrm{d}t} = 0 \tag{5.50}$$

by using the quadratic infinitesimal generator:

$$\delta_{\mathcal{S}} = p\frac{\partial}{\partial x} + \frac{\partial}{\partial y} - 2\,a\,x\frac{\partial}{\partial p} + 2\,a\frac{\partial}{\partial b} \tag{5.51}$$

associated to a symmetry of it. One needs to choose a cross-section that verifies required conditions; for example $\mathcal{H} : b - 1 = 0$. In this case, computing associated invariants is not obvious. But using moving frame based reduction process, the specialization of b to 1 leads directly to the reduced system:

$$\frac{\mathrm{d}\widetilde{x}}{\mathrm{d}\widetilde{t}} = \widetilde{p},\ \frac{\mathrm{d}\widetilde{y}}{\mathrm{d}\widetilde{t}} = \widetilde{q},\ \frac{\mathrm{d}\widetilde{p}}{\mathrm{d}\widetilde{t}} = -2\,\widetilde{a}\,\widetilde{x},\ \frac{\mathrm{d}\widetilde{q}}{\mathrm{d}\widetilde{t}} = 1 - 2\,\widetilde{a}\,\widetilde{y},\ \frac{\mathrm{d}\widetilde{a}}{\mathrm{d}\widetilde{t}} = 0. \tag{5.52}$$

As one can see, there is no need to compute invariants nor the moving frame. This reduction method ensures that there exists a new coordinate chart of invariants $\left(\widetilde{t}, \widetilde{x}, \widetilde{y}, \widetilde{p}, \widetilde{q}, \widetilde{a}\right)$ in which the system (5.50) can be represented by (5.52) and the fibers of (5.51). ⌟

Remark 5.4.10. For a continuous dynamical system, the process followed for the reduction of a parameter is a special case of the reduction of a state variable, more precisely one has $\mu = 0$. ⌟

Parameter Elimination

In this paragraph, the moving frame based elimination of a parameter is presented for continuous and discrete dynamical systems. Because the method doesn't involve any continuity property, these two cases can be handled in the same way.

Let $\delta_{\mathcal{D}}$ be an infinitesimal generator associated to a continuous dynamical system. Remark that for a parameter θ, one has $\delta_{\mathcal{D}}\theta = 0$. Thus, according to the proposition 5.4.6 page 116, the moving frame based reduction of a parameter is a simple specialization of this parameter w.r.t. the chosen cross-section. The following example illustrates this case.

Example 5.4.11. Let us consider the following continuous dynamical system $\mathcal{D}$ defined on the coordinate set $Z = (t, x, y, a, b)$:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = x - a\,y, \quad \frac{\mathrm{d}y}{\mathrm{d}t} = b - a\,x, \quad \frac{\mathrm{d}a}{\mathrm{d}t} = \frac{\mathrm{d}b}{\mathrm{d}t} = 0 \tag{5.53}$$

and its associated infinitesimal generator:

$$\delta_{\mathcal{D}} = \frac{\partial}{\partial t} + (x - a\,y)\,\frac{\partial}{\partial x} + (b - a\,x)\,\frac{\partial}{\partial y}\cdot \tag{5.54}$$

This dynamical system possesses a one-dimensional local Lie point symmetry $\mathcal{S}$ represented by the infinitesimal generator:

$$\delta_{\mathcal{S}} = x\frac{\partial}{\partial x} + y\frac{\partial}{\partial y} + b\frac{\partial}{\partial b}\cdot \tag{5.55}$$

One can eliminate the parameter b from this dynamical system by using $\delta_{\mathcal{S}}$ because it acts on this parameter i.e. $\delta_{\mathcal{S}} b \neq 0$. Let us choose a local cross-section $\mathcal{H} : h(Z) = 0$ that locally and transversely intersects the orbits of $\mathcal{S}$ i.e. $\delta_{\mathcal{S}} h(Z) \neq 0$, verifying $\partial h(Z)/\partial b \neq 0$ and linear in b; for example $h = b - 1$. The moving frame based reduction process tells that, in this case, the elimination of the parameter b is equivalent to the specialization of the dynamical system $\mathcal{D}$ by $b = 1$ i.e. to:

$$\frac{\mathrm{d}\widetilde{x}}{\mathrm{d}\widetilde{t}} = \widetilde{x} - \widetilde{a}\,\widetilde{y}, \quad \frac{\mathrm{d}\widetilde{y}}{\mathrm{d}\widetilde{t}} = 1 - \widetilde{a}\,\widetilde{x}, \quad \frac{\mathrm{d}\widetilde{a}}{\mathrm{d}\widetilde{t}} = 0 \tag{5.56}$$

where $\widetilde{t}, \widetilde{x}, \widetilde{y}$ and $\widetilde{a}$ are new coordinates (invariants). Remark that the system (5.53) can be represented by the lower dimensional system (5.56) and the orbits of $\mathcal{S}$. ⌟

The following example shows that the same reduction process is likely valid for discrete dynamical systems. This subject will be part of a future work for formal demonstration. Its implementation is already available in [102].

Example 5.4.12. The discrete dynamical system defined on $Z = (\tau, X, Y, R, X_0, C, r)$ by the recurrence operator σ:

$$\begin{cases} \sigma\tau = \tau + 1, \\ \sigma X = R\,X - \frac{(R-1)X^2}{X_0} - C\,X\,Y, \quad \sigma Y = \frac{r\,X\,Y}{X_0}, \\ \sigma R = R, \quad \sigma X_0 = X_0, \quad \sigma C = C, \quad \sigma r = r \end{cases} \tag{5.57}$$

is a discrete time predator-prey model (see [81]). Here, X and Y represent the prey and predator numbers in year τ whose maximum reproductive rates are, respectively, R and r and X_0 is the equilibrium value of X in the absence of Y. One of its one-dimensional local Lie point symmetries $\mathcal{S}$ associated to the infinitesimal generator:

$$\delta_{\mathcal{S}} = X\frac{\partial}{\partial x} + X_0\frac{\partial}{\partial X_0} \tag{5.58}$$

can be used to eliminate the parameter X_0 because one has $\delta_{\mathcal{S}} X_0 \neq 0$. As for the continuous case, one can choose a local cross-section defined by $\mathcal{H} : X_0 - 1 = 0$ that

verifies required conditions; $\delta_{\mathcal{S}}(X_0 - 1) \neq 0$ and $\partial(X_0 - 1)/\partial X_0 \neq 0$. According to the moving frame based reduction process, the reduced discrete dynamical system can be obtained by specializing the parameter X_0 to 1:

$$\begin{cases} \sigma\widetilde{\tau} = \widetilde{\tau} + 1, \\ \sigma\widetilde{X} = \widetilde{R}\,\widetilde{X} - \left(\widetilde{R} - 1\right)\widetilde{X}^2 - \widetilde{C}\,\widetilde{X}\,\widetilde{Y}, \quad \sigma\widetilde{Y} = \widetilde{r}\,\widetilde{X}\,\widetilde{Y}, \\ \sigma\widetilde{R} = \widetilde{R}, \quad \sigma\widetilde{C} = \widetilde{C}, \quad \sigma\widetilde{r} = \widetilde{r} \end{cases} \tag{5.59}$$

where $\widetilde{\tau}, \widetilde{X}, \widetilde{Y}, \widetilde{R}, \widetilde{C}$ and $\widetilde{r}$ are new coordinates (invariants). Remark that the system (5.57) page 120 can be represented by the lower dimensional system (5.59) and the orbits of $\mathcal{S}$.

Now, let us verify this result by rewriting the original system in the new coordinate set composed by the invariants $\widetilde{\tau} = \tau, \widetilde{X} = X/X_0, \widetilde{Y} = Y, \widetilde{R} = R, \widetilde{C} = C, \widetilde{r} = r$:

$$\begin{cases} \sigma\widetilde{\tau} = \sigma\tau = \tau + 1 = \widetilde{\tau} + 1, \\ \sigma\widetilde{X} = \frac{\sigma X}{\sigma X_0} = R\frac{X}{X_0} - (R-1)\left(\frac{X}{X_0}\right)^2 - C\frac{X}{X_0}Y = \widetilde{R}\,\widetilde{X} - \left(\widetilde{R} - 1\right)\widetilde{X}^2 - \widetilde{C}\,\widetilde{X}\,\widetilde{Y}, \\ \sigma\widetilde{Y} = \sigma Y = r\frac{X}{X_0}Y = \widetilde{r}\,\widetilde{X}\,\widetilde{Y}, \\ \sigma\widetilde{R} = \sigma R = R = \widetilde{R}, \\ \sigma\widetilde{C} = \sigma C = C = \widetilde{C}, \\ \sigma\widetilde{r} = \sigma r = r = \widetilde{r}. \end{cases} \tag{5.60}$$

These eliminations show that the reduced discrete dynamical system found by the classical reduction process is given in (5.59). ⌟

5.4.3 Successive Reduction Process

In this document, a reduction process is performed by using one-dimensional symmetry groups. For r-dimensional local Lie symmetry groups, we perform successively r times this process (using the relation (5.1) page 103). Given such a Lie algebra, one must choose the order of its infinitesimal generators to perform these reductions. The following proposition gives this order in the case of a solvable Lie algebra.

Proposition 5.4.13. *[see th. 2.60 in § 2.5 of [86]] Consider a dynamical system $\mathcal{D}$ and the chain of Lie subalgebras $\mathfrak{g}^{(1)}, \ldots, \mathfrak{g}^{(\kappa)}$ of its solvable Lie algebra $\mathfrak{g}$ as described in (3.38) page 74. Suppose that the Lie subalgebra $\mathfrak{g}^{(i+1)}$ is used to reduce $\mathcal{D}$ and that a new dynamical system $\widetilde{\mathcal{D}}$ is deduced. In this context, $\widetilde{\mathcal{D}}$ admits the quotient algebra $\mathfrak{g}^{(i)}/\mathfrak{g}^{(i+1)}$ as the Lie algebra of its symmetries.* ⌟

Remark 5.4.14. If the dimension difference between two successive Lie subalgebras $\mathfrak{g}^{(i)}$ and $\mathfrak{g}^{(i+1)}$ is greater than 1, then one can use associated infinitesimal generators in any order because the quotient algebra $\mathfrak{g}^{(i)}/\mathfrak{g}^{(i+1)}$ is commutative by construction (see proposition 3.3.19 page 73). ⌟

The reduction process remain correct, if for some reasons, the infinitesimal generators are not used in the right order. However, their effectiveness may be affected. Given a r-dimensional Lie algebra, one can be sure to reduce the associated system *at most* by r

but not exactly by r. Recall that, on the other hand, these cases can be handled by applying the more general algorithm based on the *recurrence formula* given in [38, 65] without considering a special order of infinitesimal generators. The following examples illustrate the importance of this order in this document.

Example 5.4.15. The dynamical system (another type of Verhulst logistic growth model given in (3.9) page 63):

$$\frac{\mathrm{d}x}{\mathrm{d}t} = (a - b\,x)\,x - c\,x, \quad \frac{\mathrm{d}a}{\mathrm{d}t} = \frac{\mathrm{d}b}{\mathrm{d}t} = \frac{\mathrm{d}c}{\mathrm{d}t} = 0 \tag{5.61}$$

possesses a Lie symmetry group of dimension 3 that can be represented by a translation and two scalings:

$$\delta_1 = \frac{\partial}{\partial a} + \frac{\partial}{\partial c}, \quad \delta_2 = -t\frac{\partial}{\partial t} + a\frac{\partial}{\partial a} + b\frac{\partial}{\partial b} + c\frac{\partial}{\partial c}, \quad \delta_3 = x\frac{\partial}{\partial x} - b\frac{\partial}{\partial b}\cdot \tag{5.62}$$

If one uses at first the scalings in order to eliminate two parameters from the system, then the reduced system will no longer have a translation. Indeed, one can eliminate, for example, the parameters a and b using δ_2 and δ_3. In this case, the reduced system writes in a new coordinate set:

$$\frac{\mathrm{d}\widetilde{x}}{\mathrm{d}\widetilde{t}} = (1 - \widetilde{x})\,\widetilde{x} - \widetilde{c}\,\widetilde{x}, \quad \frac{\mathrm{d}\widetilde{c}}{\mathrm{d}\widetilde{t}} = 0\cdot \tag{5.63}$$

Remark that this system has no translation. On the other hand, if one uses at first the information of the translation to eliminate one parameter, let us say c, then the reduced system writes in a new coordinate set:

$$\frac{\mathrm{d}\widetilde{x}}{\mathrm{d}\widetilde{t}} = \left(\widetilde{a} - \widetilde{b}\,\widetilde{x}\right)\widetilde{x}, \quad \frac{\mathrm{d}\widetilde{a}}{\mathrm{d}\widetilde{t}} = \frac{\mathrm{d}\widetilde{b}}{\mathrm{d}\widetilde{t}} = 0\cdot \tag{5.64}$$

Remark that this reduced system keeps its two scalings which serve to eliminate two remaining parameters (see § 2.6 of [109]). This is because the scalings are symmetries of the translation but the opposite is not true. One can verify this result by constructing associated commutator table:

	$\boldsymbol{\delta_1}$	$\boldsymbol{\delta_2}$	$\boldsymbol{\delta_3}$
$\boldsymbol{\delta_1}$	0	δ_1	0
$\boldsymbol{\delta_2}$	$-\delta_1$	0	0
$\boldsymbol{\delta_3}$	0	0	0

(5.65)

In the associated commutator series, remark that $\mathfrak{g}^{(0)}$ contains δ_1, δ_2 and δ_3, $\mathfrak{g}^{(1)}$ contains just δ_1 and $\mathfrak{g}^{(2)}$ is empty. So, one must use δ_1 before others in the reduction process. The order in which the infinitesimal generators δ_2 and δ_3 are used is not important because the quotient algebra $\mathfrak{g}^{(0)}/\mathfrak{g}^{(1)}$ is commutative. ⌟

Example 5.4.16. Look at the solvable Lie algebra considered in the example 3.3.23 page 74. One has a commutator series as follows:

$$\mathfrak{g} = \mathfrak{g}^{(0)} \subset \mathfrak{g}^{(1)} \subset \mathfrak{g}^{(2)} = \{0\}\,. \tag{5.66}$$

According to the solvability theory, one must use the information given by the infinitesimal generators of $\mathfrak{g}^{(1)}$ i.e. δ_1 or δ_4 before these in $\mathfrak{g}^{(0)}$ but not in $\mathfrak{g}^{(1)}$ i.e. δ_2 or δ_3. ⌟

Another practical question is the rewriting of remaining Lie symmetries after using one. When one performs the reduction procedure with one infinitesimal generator $\delta_{\mathcal{S}}$, the coordinate set of the system changes. In fact, the system is rewritten in new coordinates so that it can have desired properties. In this case, remaining symmetries of the original system must be rewritten also in this new coordinate set. Knowing that a Lie symmetry is actually a continuous dynamical system, remaining symmetries must be also reduced by the same infinitesimal generator $\delta_{\mathcal{S}}$ (see proposition 5.4.13 page 121). After each reduction process, this adaptation is fundamental.

Remark 5.4.17. Let δ_1 be an infinitesimal generator in $\mathfrak{g}^{(i+1)}$ and δ_2 be infinitesimal generator in $\mathfrak{g}^{(i)}$ defined on the coordinate set Z. Let us suppose that δ_2 is a symmetry of δ_1 (see remark 3.3.25 page 74). Once δ_1 is used for the reduction process, the infinitesimal generators of symmetries must be rewritten in a new coordinates set $\widetilde{Z}$. δ_1 is transformed into its rectified form w.r.t. a new coordinate, let us say $\widetilde{z}_1$ i.e. $\delta_1\widetilde{z}_1 = 1$. Let us suppose that δ_2 is rewritten in $\widetilde{Z}$ and that it is used to eliminate the coordinate $\widetilde{z}_2$ (see proposition 5.4.13 page 121). After this second reduction, actually, one needs to reduce again δ_1 to see if it is affected. Because δ_2 is a symmetry of δ_1 and $\widetilde{z}_2$ is an invariant of δ_1 i.e. $\delta_1\widetilde{z}_2 = 0$ (see equation (5.39) page 116), there is no need to come back to the used infinitesimal generators during a successive reduction process. ⌟

Let us illustrate this rewriting of infinitesimal generators during a successive reduction process.

Example 5.4.18. The continuous dynamical system of one state variable and two parameters:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = a\,b\,x, \quad \frac{\mathrm{d}a}{\mathrm{d}t} = \frac{\mathrm{d}b}{\mathrm{d}t} = 0 \tag{5.67}$$

possesses a Lie symmetry of dimension 2 that can be represented by the following infinitesimal generators:

$$\delta_1 = -t\frac{\partial}{\partial t} - x\frac{\partial}{\partial x} + b\frac{\partial}{\partial b} \quad \text{and} \quad \delta_2 = -2\,t\frac{\partial}{\partial t} + a\frac{\partial}{\partial a} + b\frac{\partial}{\partial b}\cdot \tag{5.68}$$

The goal is to reduce the parameters a and b from the system (5.67) by using these infinitesimal generators. The first one can be used to eliminate the parameter b by introducing new coordinates such as:

$$\widetilde{t} = t\,b, \quad \widetilde{x} = x\,b, \quad \widetilde{a} = a. \tag{5.69}$$

Remark that these are invariants of δ_1. In this new coordinate set, the system can be rewritten as:

$$\frac{\mathrm{d}\widetilde{x}}{\mathrm{d}\widetilde{t}} = \widetilde{a}\,\widetilde{x}, \quad \frac{\mathrm{d}\widetilde{a}}{\mathrm{d}\widetilde{t}} = 0. \tag{5.70}$$

Now, the information given by the second infinitesimal generator δ_2 can also be used in order to eliminate $\widetilde{a}$. But in (5.68), δ_2 is given in the original coordinates and must be rewritten in new ones given in (5.69). In these new coordinate set, one can reduce δ_2 into $\widetilde{\delta}_2$:

$$\widetilde{\delta}_2 = -\widetilde{t}\frac{\partial}{\partial\widetilde{t}} + \widetilde{x}\frac{\partial}{\partial\widetilde{x}} + \widetilde{a}\frac{\partial}{\partial\widetilde{a}} \tag{5.71}$$

that is a symmetry of (5.70) page 123 because δ_1 and δ_2 commutes. Finally, $\widetilde{\delta}_2$ can be used to introduce a third coordinate set:

$$T = \widetilde{t}\,\widetilde{a}, \quad X = \frac{\widetilde{x}}{\widetilde{a}}\cdot \tag{5.72}$$

In this last coordinate set, the system (5.70) page 123 is rewritten as:

$$\frac{\mathrm{d}X}{\mathrm{d}T} = X. \tag{5.73}$$

As expected, the final system, defined on the coordinate set (T, X), has no parameters. In order to reach the original system from this last equation, one may use associated fibers that are given explicitly in this example thanks to the new coordinates. ⌟

5.4.4 Implementation of Moving Frame Based Reduction Process

In the `ExpandedLiePointSymmetry` package, the moving frame based reduction method is implemented by the function `Invariantize`, a name that stands for rewriting a given system in a new coordinate set of invariants of associated symmetries. This subsection is devoted to clarify this function and its usage. In the sequel, its inputs and its outputs are illustrated in many examples.

The inputs of the function `Invariantize` are:

- a system to reduce; it can be an algebraic or a dynamical system represented by types `AlgebraicSystem`, `InfinitesimalGenerator` or `RecurrenceOperator`;
- an infinitesimal generator or a Lie algebra of its symmetries;
- a list of coordinates to eliminate. This argument is optional; if it is not specified, the algorithm chooses appropriate variables to eliminate arbitrarily.

The output of the function `Invariantize` involves 3 elements:

- the reduced system written in the new coordinate set, if it was possible to apply the moving frame based reduction; otherwise the original system;
- a list of used cross-sections during the reduction;
- a list of infinitesimal generators of symmetries used in the reduction.

Remark 5.4.19. There exists also another implementation for this output. After every reduction process, a dynamical system data structure is created. It involves the reduced system, the used cross-section and the infinitesimal generator of the used symmetry. In the case of successive reductions, these data structures are overlapped. In this document, we do not use this implementation to avoid the enlarging of expressions. ⌟

There are two points to clarify about the function `Invariantize` so that the user can exactly see what happens in the function and how to interpret the output.

The first remark concerns the elimination of a given list of variables. The reduction method can fail for some variables. For example if there is no symmetry acting on a given variable, then this variable cannot be eliminated. Or in the case of a Lie algebra input, if symmetries acting on a given variable are lost after first reductions then the variable can't be eliminated neither. For this second case, the order of given variables to eliminate may be important.

The second remark concerns the new coordinate set notation. Observe that in this document, invariantized coordinates are denoted with a tilde. In this way, it is easy to distinguish the old and new coordinates but this notation is not practical for the implementation, especially when several reductions must be done successively. That is why a unique notation is used for all coordinates. The only thing to remember is that every time that a reduction is performed, the system is rewritten in a *new* coordinate set. The used infinitesimal generator and associated cross-section of each reduction are given in a separated coordinate set. This point will be lighten up by the next examples.

Example 5.4.20. The following function calls realize the reduction of the algebraic system defined by $x^2+y^2-r^2=0$ explained in the example 5.1.2 page 104. The goal is to eliminate the variable r by using $\delta_{\mathcal{S}} = x\partial/\partial x + y\partial/\partial y + r\partial/\partial r$.

```
> # Reduction of an algebraic system with an infinitesimal generator
> AlgSys := NewAlgebraicSystem([x^2+y^2-r^2]);
                    2    2    2
       AlgSys := [x  + y  - r ]
> Sym := NewInfinitesimalGenerator([x,y,r],[x,y,r]);
                      /d   \     /d   \     /d   \
       Sym := F -> x |-- F| + y |-- F| + r |-- F|
                      \dx  /     \dy  /     \dr  /

> out := Invariantize(AlgSys, Sym, toeliminate=[r]):
> ReducedSystem := out[1];
                           2    2
       ReducedSystem := [x  + y  - 1]

> CrossSection := out[2];
       CrossSection := [1 - r]

> UsedSym := out[3];
                           /d   \     /d   \     /d   \
       UsedSym := [F -> x |-- F| + y |-- F| + r |-- F|]
                           \dx  /     \dy  /     \dr  /
```

Remark that even if the reduced system is given in the notations of the original system, their coordinates are different. The reduced system could be seen as $\widetilde{x}^2+\widetilde{y}^2-1=0$ while the cross-section as $1-r=0$.

Let use realize now the reduction of r from the same algebraic system without specifying the infinitesimal generator to use.

```
> # Reduction of an algebraic system with a Lie algebra
> AlgSys := NewAlgebraicSystem([x^2+y^2-r^2]);
                    2    2    2
       AlgSys := [x  + y  - r ]
```

```
> # Lie algebra of its affine type Lie symmetries
> LieAlg := ELPSymmetries(AlgSys, sym=affine);
       LieAlg := LieAlgebraOfDim(3)

> out := Invariantize(AlgSys, LieAlg, toeliminate=[r]):
> ReducedSystem := out[1];
                          2    2
       ReducedSystem := [x  + y  - 1]

> CrossSection := out[2];
       CrossSection := [1 - r]

> UsedSym := out[3];
                         /d   \     /d   \
       UsedSym := [F -> x |-- F| + r |-- F|]
                         \dr  /     \dx  /
```

Remark that the user is informed of the used infinitesimal generator taken from the given Lie algebra. ⌟

Example 5.4.21. The following code corresponds to the reduction of a parameter from the continuous dynamical system (5.53) as given in the example 5.4.11 page 120 and that of the discrete dynamical system (5.57) as given in the example 5.4.12 page 120.

```
> # Reduction of a parameter of a continuous dynamical system
> ContSystem := NewInfinitesimalGenerator([1,x-a*y,b-a*x,0,0],[t,x,y,a,b]);
                         /d   \             /d   \             /d   \
       ContSystem := F -> |-- F| + (x - a y) |-- F| + (b - a x) |-- F|
                         \dt  /             \dx  /             \dy  /

> Sym := NewInfinitesimalGenerator([0,x,y,0,b],[t,x,y,a,b]);
                      /d   \     /d   \     /d   \
       Sym := F -> x |-- F| + y |-- F| + b |-- F|
                      \dx  /     \dy  /     \db  /

> out := Invariantize(ContSystem, Sym, toeliminate=[b]):
> ReducedSystem := out[1];
                           /d   \             /d   \             /d   \
       ReducedSystem := F -> |-- F| + (x - a y) |-- F| + (1 - a x) |-- F|
                           \dt  /             \dx  /             \dy  /

> CrossSection := out[2];
       CrossSection := [1 - b]

> UsedSym := out[3];
                         /d   \     /d   \     /d   \
       UsedSym := [F -> x |-- F| + y |-- F| + b |-- F|]
                         \dx  /     \dy  /     \db  /
```

```
> # Reduction of a parameter of a discrete dynamical system
> DiscSystem := NewRecurrenceOperator(
>     [tau+1,R*X-((R-1)*X^2/X0)-C*X*Y,(r*X*Y)/X0,R,X0,C,r],
>     [tau,X,Y,R,X0,C,r]);
        DiscSystem := (tau, X, Y, R, X0, C, r) ->
                                  2
                        (R - 1) X           r X Y
        [tau + 1, R X - ---------- - C X Y, -----, R, X0, C, r]
                          X0                X0

> Sym := NewInfinitesimalGenerator([0,X,0,0,X0,0,0],[tau,X,Y,R,X0,C,r]);
                      /d   \       / d   \
       Sym := F -> X |-- F| + X0 |--- F|
```

```
                      \dX  /      \dX0  /

> out := Invariantize(DiscSystem, Sym, toeliminate=[X0]):
> ReducedSystem := out[1];
       ReducedSystem := (tau, X, Y, R, X0, C, r) ->
                                                    2
                          [tau + 1, R X - (R - 1) X  - C X Y, r X Y, R, 1, C, r]

> CrossSection := out[2];
       CrossSection := [1 - X0]

> UsedSym := out[3];
                             /d   \      / d   \
       UsedSym := [F -> X |-- F| + X0 |--- F|]
                             \dX  /      \dX0  /
```

Until now, there was no extra difficulties to interpret the output of the function `Invariantize` because previous examples are based on reduction w.r.t. a one-parameter local Lie symmetry group i.e. by using one infinitesimal generator. The following example shows in details how to read this when two such reductions are done successively.

Example 5.4.22. Let us detail the implementation of the example 5.4.18 page 123. We perform the reduction of the parameters a and b from the continuous dynamical system $\mathcal{D}$:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = a\,b\,x, \quad \frac{\mathrm{d}a}{\mathrm{d}t} = \frac{\mathrm{d}b}{\mathrm{d}t} = 0 \tag{5.74}$$

represented by the infinitesimal generator:

$$\delta_{\mathcal{D}} = \frac{\partial}{\partial t} + a\,b\,x\frac{\partial}{\partial x}\cdot \tag{5.75}$$

One can compute the symmetries of this system by using the function `ELPSymmetries`. Its output is a Lie algebra of 4 infinitesimal generators:

$$\delta_1 = \frac{\partial}{\partial t}, \quad \delta_2 = -a\frac{\partial}{\partial a} + b\frac{\partial}{\partial b}, \quad \delta_3 = x\frac{\partial}{\partial x}, \quad \delta_4 = t\frac{\partial}{\partial t} - a\frac{\partial}{\partial a}\cdot \tag{5.76}$$

As one can remark, the infinitesimal generator δ_1 acts just on the independent variable so it is discarded during the reduction process. The first infinitesimal generator to use is δ_2 in order to eliminate the first parameter a. As expected, one can use the cross-section $\mathcal{H}_1 : a - 1 = 0$ to obtain the intermediate reduced system associated to the infinitesimal generator:

$$\delta_{\widetilde{\mathcal{D}}_1} = \frac{\partial}{\partial \widetilde{t}} + \widetilde{b}\,\widetilde{x}\frac{\partial}{\partial \widetilde{x}} \tag{5.77}$$

where $\widetilde{t} = t, \widetilde{x} = x$ and $\widetilde{b} = a\,b$ are new coordinates. At this stage of the reduction, remaining symmetries δ_3 and δ_4 must be rewritten (see § 5.4.3 page 121). In the new coordinate set, one can write:

$$\widetilde{\delta}_3 = \widetilde{x}\frac{\partial}{\partial \widetilde{x}}, \quad \widetilde{\delta}_4 = \widetilde{t}\frac{\partial}{\partial \widetilde{t}} - \widetilde{b}\frac{\partial}{\partial \widetilde{b}}\cdot \tag{5.78}$$

Again, the infinitesimal generator $\widetilde{\delta}_3$ acts on just 1 coordinate so it is discarded during the reduction process. The last symmetry acts now on the coordinate $\widetilde{b}$, thus it may be

eliminated. The cross-section $\mathcal{H}_2 : \widetilde{b} - 1 = 0$ is used to obtain the final reduced system associated to the infinitesimal generator:

$$\delta_{\widetilde{\mathcal{D}}_2} = \frac{\partial}{\partial T} + X\frac{\partial}{\partial X} \tag{5.79}$$

written in a third coordinate set defined by $T = \widetilde{t}\,\widetilde{b}$ and $X = \widetilde{x}$. The following code performs exactly this example.

```
> # Several reductions
> ContSystem := NewInfinitesimalGenerator([1,a*b*x,0,0],[t,x,a,b]);
                          /d   \         /d   \
       ContSystem := F -> |-- F| + a b x |-- F|
                          \dt  /         \dx  /
> LieAlg := ELPSymmetries(ContSystem);
       LieAlg := LieAlgebraOfDim(4)

> GeneratorsOf(LieAlg);
      d             /d   \     /d   \         /d   \         /d   \     /d   \
[F -> -- F, F -> -a |-- F| + b |-- F|, F -> x |-- F|, F -> t |-- F| - a |-- F|]
      dt            \da  /     \db  /         \dx  /         \dt  /     \da  /

> out := Invariantize(ContSystem, LieAlg, toeliminate=[a,b]):
> ReducedSystem := out[1];
                             /d   \     /d   \
       ReducedSystem := F -> |-- F| + x |-- F|
                             \dt  /     \dx  /

> CrossSections := out[2];
       CrossSections := [1 - a, 1 - b]

> UsedSyms := out[3];
                           /d   \     /d   \         /d   \     /d   \
       UsedSyms := [F -> -a |-- F| + b |-- F|, F -> t |-- F| - b |-- F|]
                           \da  /     \db  /         \dt  /     \db  /
```

So remark that contrarily to the uniqueness of the coordinates notation on the output of the function `Invariantize`, different coordinate sets are hidden. The first cross-section and first used symmetry are defined on (t, x, a, b). The second cross-section and second symmetry are defined on $\left(\widetilde{t}, \widetilde{x}, \widetilde{b}\right)$. Finally, the resulting reduced system is defined on the third coordinate set (T, X). ⌟

As many reductions as desired may be done automatically (if the symmetries of the studied system allow). The same principal of notation must be used to take advantage of the output.

Example 5.4.23. The example 5.4.7 page 118 illustrates the reduction of a state variable from a continuous dynamical system. The following code performs this reduction automatically.

```
> # Reduction of a state variable of a continuous dynamical system
> ContSystem := NewInfinitesimalGenerator([1,y,y/x],[t,x,y]);
                                                /d   \
                                              y |-- F|
                          /d   \     /d   \     \dy  /
       ContSystem := F -> |-- F| + y |-- F| + --------
                          \dt  /     \dx  /       x
```

```
> Sym := NewInfinitesimalGenerator([t,x,0],[t,x,y]);
                      /d   \     /d   \
       Sym := F -> t |-- F| + x |-- F|
                      \dt  /     \dx  /

> out := Invariantize(ContSystem, Sym, toeliminate=[x]):
> ReducedSystem := out[1];
                                        /d   \     /d   \
       ReducedSystem := F -> (1 - t y) |-- F| + y |-- F|
                                        \dt  /     \dy  /

> CrossSection := out[2];
       CrossSection := [1 - x]


> UsedSym := out[3];
                            /d   \     /d   \
       UsedSym := [F -> t |-- F| + x |-- F|]
                            \dt  /     \dx  /
```

5.5 Reparametrization Process

In this section, we present the reparametrization of systems of ODEs (see also § 2.2.2 page 45 for associated algorithms). The goal of this process is to eliminate some parameters from the algebraic system that defines their steady points by rewriting the system in a new coordinate set. We begin by the reduction of a parameter to introduce the reparametrization. Then we state the main ideas of the reparametrization process. Finally, we show the usage of its implementation in the `MABSys` package.

In this document, the reparametrization is done only w.r.t. a parameter. Also, we consider only scalings, even if I think that this method can be extended to more general kind of symmetries. Contrarily to the previous section, I do not look to avoid the computations of invariants. Fortunately, the special form of scalings permits to obtain associated invariants without computing any differential equation (see § 2.1.2 page 37). In addition, because a Lie algebra of scaling is always commutative (see lemma 3.3.26 page 75), one can consider all scalings together, without caring about their order or rewriting (see § 5.4.3 page 121).

5.5.1 Reparametrization of a Dynamical System

The aim of the reparametrization process is to continue to prepare systems of ODEs to their qualitative analysis after the reduction process. The original idea is, once there is no more suitable symmetries to reduce systems of ODEs, to use these of associated algebraic systems that define their steady points. By these symmetries we can not reduce the system using the reduction process presented in previous sections. However, we can find a new coordinate set where the steady points of the system of interest depend on less parameters. This process permits also to distinguish the roles of parameters (see § 1.4.2 page 18).

Remark 5.5.1. Every considered symmetry of a system of ODEs is also a symmetry of the algebraic system defining its steady points. The opposite is not true. ⌟

The reparametrization method follows the idea of the classical reduction process i.e. one computes a set of invariants associated to the used symmetry and rewrites the original system in these coordinates. Remark that, as stated in proposition 2.2.6 page 46, in our work this algorithm has a polynomial complexity. First, we restrict ourselves to scalings meaning that the invariants that we are looking for may be found without any integration (see § 2.1.2 page 37). Second, the chosen invariants constitute an invertible change of coordinates meaning that rewriting the original system in the new coordinates requires only a substitution (see § 2.1.2 page 37).

The following example shows the utility of the reparametrization process.

Example 5.5.2. Let us consider the dynamical system $\mathcal{D}$ defined on the coordinate set $Z = (t, G, \alpha, \theta)$:

$$\frac{\mathrm{d}G}{\mathrm{d}t} = (1 - G)\,\theta - \alpha\,G, \quad \frac{\mathrm{d}\theta}{\mathrm{d}t} = 0, \quad \frac{\mathrm{d}\alpha}{\mathrm{d}t} = 0. \tag{5.80}$$

We show how one can use a scaling of the algebraic system $\Sigma : \theta\,(1 - G) - \alpha\,G = 0$ defining the steady points of $\mathcal{D}$ in order to simplify them. One can see that a steady point of the system (5.80) is defined by $G = \theta/\,(\theta + \alpha)$ where G depends on 2 parameters θ and α. Remark that the infinitesimal generator:

$$\delta_{\mathcal{S}} = \theta\frac{\partial}{\partial\theta} + \alpha\frac{\partial}{\partial\alpha} \tag{5.81}$$

is a symmetry of Σ but not of $\mathcal{D}$ i.e. $\delta_{\mathcal{S}}$ can not be used for reduction. By supposing that α does not vanish, one can transform $\delta_{\mathcal{S}}$ into a semi-rectified symmetry of the form $\widetilde{\alpha}\partial/\partial\widetilde{\alpha}$ by using the new coordinate set $\widetilde{Z} = \left(\widetilde{t}, \widetilde{G}, \widetilde{\alpha}, \widetilde{\theta}\right)$ where $\widetilde{t} = t, \widetilde{G} = G, \widetilde{\alpha} = \alpha$ and $\widetilde{\theta} = \theta/\alpha$. In these coordinates, the system (5.80) is transformed into $\widetilde{\mathcal{D}}$:

$$\frac{\mathrm{d}\widetilde{G}}{\mathrm{d}\widetilde{t}} = \left(\left(1 - \widetilde{G}\right)\widetilde{\theta} - \widetilde{G}\right)\widetilde{\alpha}, \quad \frac{\mathrm{d}\widetilde{\theta}}{\mathrm{d}\widetilde{t}} = 0, \quad \frac{\mathrm{d}\widetilde{\alpha}}{\mathrm{d}\widetilde{t}} = 0. \tag{5.82}$$

The number of coordinates of this system is the same as the system (5.80). Nonetheless the advantage of this notation can be seen by looking at its steady points. The algebraic system $\widetilde{\Sigma} : \widetilde{\theta}\left(1 - \widetilde{G}\right) - \widetilde{G} = 0$ defines the steady points of $\widetilde{\mathcal{D}}$ and one has $\widetilde{G} = \widetilde{\theta}/\left(\widetilde{\theta} + 1\right)$. Remark that the steady points depend now on just 1 parameter $\widetilde{\theta}$. This can be very useful for further qualitative analysis computations. The number of parameters of the system of interest does not change but this method distinguishes the role of parameters. The value of the parameter $\widetilde{\alpha}$ is not important to localize a steady point but it specifies its nature because it appears on the eigenvalues of the associated Jacobian. ⌟

Remark 5.5.3. If one uses a symmetry of the algebraic system that is also a symmetry of the dynamical system, this process has the same effect as the reduction. ⌟

Remark 5.5.4. The equation (5.80) is put into its singular perturbed form (see ch. 7 of [82]). ⌟

The special form of the change of coordinates computed during the reparametrization process (see § 2.1.2 page 37) transforms the infinitesimal generators of the used symmetries into their semi-rectified form. The following lemma states the contribution of the reparametrization for the steady point expressions of ODEs systems.

Lemma 5.5.5. *Let $\Sigma = (f_1, \ldots, f_k)$ be a regular algebraic system (see definition 3.1.1 page 59) defined on $Z = (z_1, \ldots, z_n)$. If a rectified symmetry $\partial/\partial z_i$ (resp. a semi-rectified symmetry $z_i \partial/\partial z_i$) acting on a coordinate z_i is an infinitesimal generator of one of the symmetries of Σ, then its solutions do not depend on z_i.* ⌟

Proof. Let $\delta_{\mathcal{S}}$ be a rectified (resp. semi-rectified) symmetry of Σ. Then, according to definition of an expanded Lie point symmetry given by the theorem 4.1.1 page 88, following statement is correct:

$$\delta_{\mathcal{S}} f_j(Z) = 0 \quad \text{whenever} \quad f_j(Z) = 0, \quad \forall j \in \{1, \ldots, k\}\,. \tag{5.83}$$

One can associate the following one-parameter ν group of transformation to the rectified (resp. semi-rectified) symmetry (see § 3.4 page 79):

$$\left\{ \begin{array}{l} z_j \to z_j \quad \forall j \neq i, \\ z_i \to z_i + \nu. \end{array} \right. \qquad \left(\text{resp.} \quad \left\{ \begin{array}{l} z_j \to z_j \quad \forall j \neq i, \\ z_i \to \nu\, z_i. \end{array} \right. \right) \tag{5.84}$$

This tells that if the point $Z = (z_1, \ldots, z_{i-1}, z_i, z_{i+1}, \ldots, z_n)$ is a solution of Σ then the point $(z_1, \ldots, z_{i-1}, z_i + \nu, z_{i+1}, \ldots, z_n)$ (resp. $(z_1, \ldots, z_{i-1}, \nu\, z_i, z_{i+1}, \ldots, z_n)$) is also a solution of Σ for all ν in $\mathbb{R}$. This fact shows that solutions of Σ do not depend on the value of the ith coordinate z_i. ⌟

Example 5.5.6. In this example, we reparametrize the system of ODEs (see [83]):

$$\left\{ \begin{array}{l} \frac{\mathrm{d}X}{\mathrm{d}t} = a\, k_1\, X - k_2\, X\, Y, \\ \frac{\mathrm{d}Y}{\mathrm{d}t} = k_2\, X\, Y - k_3\, Y. \end{array} \right. \tag{5.85}$$

The algebraic system that defines its steady points:

$$\left\{ \begin{array}{l} a\, k_1\, X - k_2\, X\, Y = 0, \\ k_2\, X\, Y - k_3\, Y = 0 \end{array} \right. \tag{5.86}$$

possesses a scaling type symmetry that acts on the parameter k_2:

$$\delta_{\mathcal{S}} = k_1 \frac{\partial}{\partial k_1} + k_3 \frac{\partial}{\partial k_3} + k_2 \frac{\partial}{\partial k_2} \tag{5.87}$$

and $\delta_{\mathcal{S}}$ is not a symmetry of (5.85). One can deduce the new coordinates:

$$\widetilde{t} = t, \quad \widetilde{X} = X, \quad \widetilde{Y} = Y, \quad \widetilde{a} = a, \quad \widetilde{k}_1 = \frac{k_1}{k_2}, \quad \widetilde{k}_2 = k_2, \quad \widetilde{k}_3 = \frac{k_3}{k_2}. \tag{5.88}$$

in which rewriting the dynamical system (5.85) eads to the following dynamical system:

$$\left\{ \begin{array}{l} \frac{\mathrm{d}\widetilde{X}}{\mathrm{d}\widetilde{t}} = \left(\widetilde{a}\, \widetilde{k}_1\, \widetilde{X} - \widetilde{X}\, \widetilde{Y} \right) \widetilde{k}_2, \\ \frac{\mathrm{d}\widetilde{Y}}{\mathrm{d}\widetilde{t}} = \left(\widetilde{X}\, \widetilde{Y} - \widetilde{k}_3\, \widetilde{Y} \right) \widetilde{k}_2. \end{array} \right. \tag{5.89}$$

Remark that, in this system, the parameter $\widetilde{k}_2$ does not influence the location of its steady points that are defined by:

$$\begin{cases} \widetilde{a}\,\widetilde{k}_1\,\widetilde{X} - \widetilde{X}\,\widetilde{Y} = 0, \\ \widetilde{X}\,\widetilde{Y} - \widetilde{k}_3\,\widetilde{Y} = 0. \end{cases} \tag{5.90}$$

Thus, the steady points of (5.85) page 131 depends on 4 parameters but these of (5.89) page 131 on 3 parameters. ⌟

Remark 5.5.7. All these computations work very easily when one considers a reorganization w.r.t. a parameter. Working with state variables requires differential computations. Anyway, let us see what can be expected from such a situation. Look at the following system of ODEs:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = x - y, \quad \frac{\mathrm{d}y}{\mathrm{d}t} = x^2. \tag{5.91}$$

This continuous dynamical system does not have a scaling but associated algebraic system of its steady points i.e. $\{x - y = 0, x^2 = 0\}$ has one. The scaling $x\partial/\partial x + y\partial/\partial y$ could be used in the same way of reparametrization process based on a parameter. The new coordinate set would be $\widetilde{t} = t, \widetilde{x} = x/y$ and $\widetilde{y} = y$. Rewriting the system (5.91) in these new coordinates yields following dynamical system after some derivation computations:

$$\frac{\mathrm{d}\widetilde{x}}{\mathrm{d}\widetilde{t}} = -\widetilde{x}^2 + \widetilde{x} - 1, \quad \frac{\mathrm{d}\widetilde{y}}{\mathrm{d}\widetilde{t}} = \widetilde{x}^2\,\widetilde{y}^2. \tag{5.92}$$

As one can see, the expressions of the steady points are not really simplified. But a decoupling between variables, a sort of decomposition of dependency between the equations is made. Meaning that the first equation involves now just the variable $\widetilde{x}$ but its degree increased. Furthermore, the system (5.92) is triangular. ⌟

5.5.2 Implementation of Reparametrization Process

The reparametrization procedure is implemented in the `MABSys` package. Recall that, it is possible to apply this implementation to systems coming from any scientific context.

Remark 5.5.8. The name of the function that perform the reparametrization involves the cylindrification notion. This is because the aim of this method is to decompose an algebraic system in order to free its solutions from some coordinates i.e. so that its solutions do not involve these coordinates. ⌟

The following example shows how one can perform this method automatically.

Example 5.5.9. This example recalls the examples 5.5.6 page 131. Let us reparametrize the system of ODEs (5.85) page 131 w.r.t. the parameter k_2 by calling the `CylindrifySteadyPoints` function.

```
> ODEs := [diff(X(t),t)=a*k1*X(t)-k2*X(t)*Y(t),diff(Y(t),t)=k2*X(t)*Y(t)-k3*Y(t)];
                d                                                d
   ODEs := [-- X(t) = a k1 X(t) - k2 X(t) Y(t), -- Y(t) = k2 X(t) Y(t) - k3 Y(t)]
                dt                                               dt
```

```
> out := CylindrifySteadyPoints(ODEs, [k2], [X,Y,a,k1,k3]):
> out := out[1]:
> out[1];
        d                                          d
       [-- X(t) = -k2 X(t) (-a k1 + Y(t)), -- Y(t) = Y(t) k2 (X(t) - k3)]
        dt                                         dt

> out[2];
       [a k1 X - X Y, X Y - k3 Y]

> out[3];
               k1          k3
       [k1 = ----, k3 = ----]
               k2          k2

> out[4];
       [k2]
```

Steady points of the resulting system of ODEs depend on one less parameters than the original system. The output is composed of the steady points rewritten in the new coordinates, the change of coordinates given in (5.88) page 131 and the parameter w.r.t. which the reparametrization is done. ⌟

"What would life be if we had no courage to attempt anything?"

Letter to Theo van Gogh
The Hague, December 29th 1881
Vincent van Gogh

IV Conclusion

During the last three years (four if I count my master) I worked mainly on the exact simplification of algebraic and dynamical systems using expanded Lie point symmetries. In this document, I present a practical usage of the moving frame based reduction process on a particular case. Moreover, I propose an original way of using Lie symmetries for the preliminary analysis of differential systems. This reparametrization improves the exact simplification done by the reduction. In fact, we eliminate some coordinates from the studied system. We keep the equivalence relations between the original and the simplified systems in an implicit (reduction) or an explicit (reparametrization) form. For now, these methods are mainly applied on biochemical reaction networks. But for any scientific domain where algebraic systems, systems of ODEs or systems of OREs are employed, these algorithms can be useful. I concretized these ideas with the articles [15, 12] and the packages `MABSys` and `ExpandedLiePointSymmetry`. These implementations are ready to use even if it remains much to do.

There are many possible future works on this domain. For example, it would be interesting to find more symmetries to help the qualitative analysis either by extending their definitions used in our framework or by introducing new algorithms. The moving frame based reduction applied to discrete dynamical systems needs to be formalized. The idea of reparametrization needs to be generalized to more complicated symmetries than scalings and to the case of state variables (for the decoupling of coordinates in a dynamical system). One of the perspectives is also to tackle different application domains than modeling in biology. Furthermore, the implementation part must follow all these improvements.

The most difficult part of my work was its interdisciplinary structure. My subject is at the intersection of computer science, mathematics and biology. I learned to adapt myself to domains where I am not that comfortable. I improved to organise and to extract the essential part of my ideas in order to be understood by a large number of people. In the meanwhile, partly thanks to these difficulties, I really enjoyed my work.

V Appendix

$\mathcal{A}$

Hopf Bifurcation Analysis

In this chapter, we are interested in Hopf bifurcation analysis and classical symbolic qualitative analysis of continuous dynamical systems performed by the `MABSys` package. There is a large literature for this kind of studies, among which one can cite [52, 51, 50] for continuous dynamical systems and [107, 52, 72] for discrete dynamical systems. The qualitative analysis functions of `MABSys` don't come with new ideas. Some of them already exist in some of the `Maple` packages or use `Maple` commands in their implementations. But let me recall that, even if `MABSys`' pilot implementation is in `Maple`, the objective remains to construct a module compatible with chosen data structures as complete as possible and independent of the computer algebra softwares. In the future, other programming languages as C can be preferred to reach more people.

In the sequel, we focus on the symbolic qualitative analysis about steady points, their nature and Poincaré-Andronov-Hopf bifurcations. Throughout these explanations, we present the implementation of related qualitative analysis tools of the `MABSys` package. First, we survey the linearization procedure and the bifurcation theory. Then we give a general idea about Poincaré-Andronov-Hopf bifurcations through the known computer algebra method on Routh-Hurwitz criterion.

The symbolic qualitative approach allows to make conclusions regardless whether one has the explicit expression or not of the solutions and this for all initial values. We tackle the symbolic qualitative analysis for medium size systems i.e. with about twenty coordinates. For two-dimensional systems due their simple structure there exist specific methods as phase portraits or Poincaré-Bendixson theorem (see § 12.1 of [52] or ch. I.16 of [51]). For medium size systems one can expect algebraic methods to conclude the analysis. My work situates in this case. Some problems as searching for a Poincaré-Andronov-Hopf bifurcation are reduced to systems of equalities and inequalities to solve. Solving such conditions is not obvious especially if one is looking for solutions in the real numbers (and not complex numbers). Several methods can be applied. Exploration of numerical values can be a solution but this numerical method is of exponential time in the number of parameters. If one has 8 parameters and wants to test 10 values for each one, then the space to explore contains 10^8 cases. Another method is to use computer algebra techniques based on CAD or critical point methods (see [19, 33, 97]). Even if these are of exponential complexity in the number of variables in the worst case, they can work very well in some cases. Or sometimes it is enough to use some basic heuristics (see § 1.5.3 page 24). For more complicated systems, in general case, there is no well-established theory but one can try to reduce the number of coordinates of the system by divers ways, for example by using Lie point symmetries (see ch. 5 page 103).

A.1 Classical Notations for Symbolic Qualitative Analysis

In this section we survey the *stability* of *steady points* and the *linearization* to introduce my notations for Poincaré-Andronov-Hopf bifurcation.

Now on, we restrict systems of ODEs to autonomous ones i.e. where the evolution of state variables does not depend on the independent variable. This restriction is conceivable because mathematical models in which we are interested are mostly autonomous.

A.1.1 Stability of Steady Points

This subsection recalls the definition of *steady points* along with their stability conditions. On a *steady point* of a system of ODEs all derivatives vanish. It can be defined formally thanks to *nullclines*.

Definition A.1.1 (Nullcline)**.** A x_i*-nullcline* $\mathcal{N}_{x_i}$ of a system of ODEs is a set of points Z that satisfy $f_i(Z) = 0$ i.e.:

$$\mathcal{N}_{x_i} = \{Z \in \mathbb{R}^n \mid f_i(Z) = 0\}\,. \tag{A.1}$$

⌟

Definition A.1.2 (Steady Point)**.** A *steady point* (also called an *equilibrium* or a *fixed point*) of a system of ODEs is a point Z_e in $\mathbb{R}^n$ which is located in the intersection of all x_i-nullcline $\mathcal{N}_{x_i}$ of a system of ODEs i.e.:

$$Z_e \in \bigcap_{i\in\{1,\ldots,k\}} \mathcal{N}_{x_i} \quad \Rightarrow \quad f_i(Z_e) = 0 \quad \forall i \in \{1,\ldots,k\}\,. \tag{A.2}$$

⌟

Example A.1.3. Let us find the algebraic system that defines the steady points of the Van der Pol oscillator using `MABSys`. The following commands define the system of ODEs of interest given in equation (3.8) page 63 with the `OrdinaryDifferentialEquations` data structure (see remark 3.2.22 page 67). We deduce then associated algebraic system of steady points and give explicitly steady point expressions.

```
> # The definition of the system of ODEs.
> ODEs := [diff(x(t),t)=mu*(1-y(t)^2)*x(t)-y(t),diff(y(t),t)=x(t)];
                d                          2                  d
       ODEs := [-- x(t) = mu (1 - y(t) ) x(t) - y(t), -- y(t) = x(t)]
                dt                                        dt

> # Algebraic equations defining its steady points.
> SteadyPoint := SteadyPointSystem(ODEs);
                                2
       SteadyPoint := [mu (1 - y ) x - y, x]

> # The steady point of the system.
> SP := [x=0, y=0];
       SP := [x = 0, y = 0]
```

Hypothesis A.1.4. *In the following definitions, a steady point is assumed to be at the origin because one may always translate any steady point to the origin by a simple change of coordinates.* ⌟

Here, we recall necessary definitions about the stability to follow the forthcoming sections (see § 1.3 of [52] and for more general overview see whole book).

Definition A.1.5 (Stability)**.** A continuous dynamical system $\mathcal{D}$ is called *stable* at a steady point Z_e if:

$$\forall \epsilon > 0,\ \exists \eta = \eta(\epsilon)\ \mid\ \|\mathcal{D}(0, Z) - Z_e\| < \eta \quad \Rightarrow \quad \forall t \geq 0,\ \|\mathcal{D}(t, Z) - Z_e\| < \epsilon \tag{A.3}$$

where $\|\cdot\|$ refers to Euclidean distance. ⌟

In other words, if all solutions of a system of ODEs that start out near a steady point Z_e remain close enough forever, then Z_e is stable. Remark that solutions do not need to approach to the steady point.

Definition A.1.6 (Asymptotically Stability)**.** A continuous dynamical system $\mathcal{D}$ is called *asymptotically stable* at a steady point Z_e if the system is stable at Z_e and

$$\exists \nu > 0\ \mid\ \|\mathcal{D}(0, Z) - Z_e\| < \nu \quad \Rightarrow \quad \lim_{t \to \infty} \|\mathcal{D}(t, Z) - Z_e\| = 0 \tag{A.4}$$

where $\|\cdot\|$ refers to Euclidean distance. ⌟

In other words, if all solutions of a system of ODEs that start out near a steady point Z_e not only remain close enough but also converge to this point, then Z_e is asymptotically stable.

Small perturbations are useful also to distinguish types of stability. If after small perturbations, the system returns always to the steady point, then this system is *stable* at this point. If small perturbations cause the system to move away, then this system is *unstable* at this point. Note that they are many geometric objects which may be classified according to their stability. Informally, an *attractor* is an object that nearby orbits converge towards it. At the same way, a *repellor* is an object that nearby orbits diverge away from it. For example, *limit cycles* (see definition A.2.2 page 146) are objects that may be attractor or repellor.

A.1.2 Linearization

This section is devoted to the *linearization* which is a method for predicting the local stability of a dynamical system near a steady point.

The stability of a dynamical system is characterized by a small perturbation of its state variables and the reaction of the system to that perturbation. Let me recall that $X = (x_1, \ldots, x_k)$ denotes the set of state variables of a dynamical system and $F = (f_1, \ldots, f_k)$ specifies the evolution of these state variables (see definition 3.2.1 page 62). Let Z_e be a steady point and X_e state variables of this steady point. A small

perturbation $\varepsilon = (\varepsilon_1, \ldots, \varepsilon_k)$ may be performed on the steady point i.e., in vector-valued notation, one can write $X = X_e + \varepsilon$. Taylor series on this point gives:

$$\frac{\mathrm{d}X}{\mathrm{d}t} = \frac{\mathrm{d}\varepsilon}{\mathrm{d}t} = F(Z) = F(Z_e) + \varepsilon \frac{\partial F}{\partial X}(Z_e) + \mathrm{O}\big(\varepsilon^2\big)\,. \tag{A.5}$$

Taking into account $F(Z_e) = 0$, for small enough $|\varepsilon|$, the first order term dominates this expression, thus one can conclude that associated linear system is:

$$\frac{\mathrm{d}\varepsilon}{\mathrm{d}t} = \varepsilon \frac{\partial F}{\partial X}(Z_e)\,. \tag{A.6}$$

Let us introduce some classical notations:

- The *Jacobian* matrix J of all first-order partial derivatives of F is defined explicitly by:
$$J = \left(\frac{\partial F}{\partial X}\right) = \left(\frac{\partial f_i}{\partial x_j}\right)_{1 \leq i,j \leq k} \tag{A.7}$$
where the (i, j)-th entry of J is the partial derivative of f_i w.r.t. x_j;
- The *characteristic polynomial* of J in the variable λ, denoted by $P_J(\lambda)$, is the determinant of the matrix $\lambda \mathrm{I}_k - J$ where I_k is the identity matrix of size k. It can be denoted by:
$$P_J(\lambda) = a_0\lambda^k + a_1\lambda^{k-1} + \cdots + a_{k-1}\lambda + a_k \tag{A.8}$$
where a_i for i in $\{1, \ldots, k\}$ are rational functions in $\mathbb{R}(Z)$;
- Roots of the characteristic polynomial $P_J(\lambda)$, given in (A.8), are called *eigenvalues* of the Jacobian J. The (right) *eigenvector* is a not null vector V such as $JV = \lambda V$ where λ is in $\mathbb{C}$, namely an eigenvalue.

Example A.1.7. Let us recall the Van der Pol oscillator (see equation (3.8) page 63) and compute its Jacobian matrix evaluated at its steady point (see example A.1.3 page 142) as well as the associated characteristic polynomial.

```
> # The associated Jacobian matrix.
> J := JacobianMatrix(ODEs, statevars=[x,y]);
             [           2              ]
       J := [-mu (-1 + y )    -2 mu y x - 1]
             [                           ]
             [      1                0   ]

> J0 := subs(SP, J);
              [mu    -1]
       J0 := [         ]
              [1      0]

> P := LinearAlgebra:-CharacteristicPolynomial(J0, lambda);
                    2
       P := lambda  - mu lambda + 1
```

According to the following theorem, the behavior of a system of ODEs, at the neighborhood of a given steady point, is classified by looking the signs of the eigenvalues of the Jacobian evaluated at this point. Remark also that, unfortunately, the linearization does not always suffice to conclude the stability (see example 9.9 of [52] which is one of the classical examples).

Theorem A.1.8 (Stability, see § 1.3 of [52]). *Let us consider a system of nonlinear ODEs. Let F be associated set of C^1 functions, Z_e one of its fixed points. Suppose that the Jacobian J_0 at the steady point does not vanish. Then the steady point is asymptotically stable if J_0 has all eigenvalues with negative real part and unstable if it has at least one eigenvalue with positive real part.* ⌟

The main problem of this stability analysis is that, in general case, symbolic expressions that must be handled grow as the dimension of the system increases. Moreover, one cannot always compute explicitly the eigenvalues from the characteristic polynomial. According to Abel's impossibility theorem (see [1]) there is no general solution in radicals to polynomial equations of degree strictly greater than 5. Meaning that it is impossible, in general, to write down the roots of such a polynomial equation by finite expressions in function of its coefficients, rational operations and root extractions.

A.2 Poincaré-Andronov-Hopf Bifurcation

In this section we are interested in oscillations due to *Poincaré-Andronov-Hopf bifurcations*. We present *Routh-Hurwitz criterion* that gives the conditions to have such bifurcation and associated implementations in `MABSys`. For a more general survey see ch. 11 of [52] or § I.16 of [51] and another symbolic method can be found in [50]. In the sequel, for brevity, this bifurcation is called only *Hopf* bifurcation.

The *bifurcation theory* is a systematic study of possible changes in the structure of the orbits of a dynamical system depending on their parameters values (see ch. 2 of [52]). The goal is to find out the small smooth change of parameters values which causes a sudden qualitative or topological change in the dynamical system's behavior: the appearance or the disappearance of steady points, stability changes on a given steady point etc.

Definition A.2.1 (Codimension of a Bifurcation). The *codimension* of a bifurcation is the number of parameters which must be varied for the bifurcation to occur. ⌟

The codimension of a bifurcation is very high for complicated systems. One can not know which parameters to vary, how much to vary them w.r.t. each other, which parameters to fix to have desired behaviors. The reduction and the reparametrization (see ch. 5 page 103) algorithms can help the user.

A.2.1 Seeking for Oscillations

My motivation of seeking for oscillations comes from the aim of modeling a gene regulatory network which controls the circadian clock of a unicellular green alga (see § 1.2.2

page 9). Given a system of parametric ODEs, I am interested in finding ranges of values for the system parameters and variables which produce oscillating trajectories i.e. *stable limit cycles*.

Definition A.2.2 (Limit Cycle)**.** A *limit cycle* is a periodic orbit attracting nearby orbits. In other words, it corresponds to oscillation trajectories. If a limit cycle is reached as time tends to $+\infty$ (resp. to $-\infty$) then it is called a *stable* (resp. *unstable*) limit cycle. Otherwise, it is neither stable nor unstable. ⌟

Under some general hypothesis, in the neighborhood of a Hopf bifurcation, a stable steady point of the system gives birth to a small stable limit cycle. Remark that looking for Hopf bifurcations is not equivalent to looking for stable limit cycles. First, some Hopf bifurcations (e.g. subcritical ones) do not imply the existence of stable limit cycles; second, there may exist limit cycles not related to Hopf bifurcations. But here, we focus on how one can search a Hopf bifurcation with symbolic computer algebra methods.

A.2.2 Definition of a Hopf Bifurcation

The appearance or the disappearance of a periodic orbit through a local change in the stability properties of a steady point is known as the *Hopf bifurcation*. In the sequel, I restrict this study to steady points with one conjugate nonzero purely imaginary eigenvalues. The following theorem tells the conditions under which this bifurcation phenomena occurs.

Theorem A.2.3 (Poincaré-Andronov-Hopf Bifurcation, see § 11.2 of [52])**.** *Let J_0 be the Jacobian of a continuous parametric dynamical system evaluate at a steady point Z_e of it. Suppose that all eigenvalues of J_0 have negative real parts except one conjugate nonzero purely imaginary pair* [1] $\pm\iota\beta$. *A* Hopf bifurcation *arises when these two eigenvalues crosses the imaginary axis because of a variation of the system parameters (see figure A.1 page 146).* ⌟

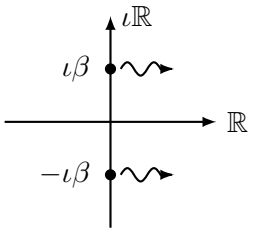


Figure A.1: Two conjugate nonzero purely imaginary eigenvalues cross the imaginary axis in the complex plan. A stable limit cycle may happen for the studied system.

Remark that if the real parameters values continuously vary then the steady points and their associated eigenvalues continuously vary also. For near-critical parameter values, limit cycles over steady points may be observed. A Hopf bifurcation is called *supercritical* if it results to stable limit cycles.

[1] ι stands for the imaginary unit equal to $\sqrt{-1}$.

A.2.3 Routh-Hurwitz Criterion

In this subsection, we are interested in giving necessary conditions so that a Hopf bifurcation occurs and their implementations in the `MABSys` package. The theorem A.2.3 page 146 tells that the linear part of a continuous dynamical system is important for the existence of nontrivial periodic orbits. The local dynamics near such points depend mainly on nonlinear terms of the associated system. The following paragraphs are about *Routh-Hurwitz criterion* (see § I.13 of [51]). First, let us begin by defining *Sturm series* which are very important for *Routh's theorem*.

Definition A.2.4 (Sturm Series, see [105])**.** Let p_0 and p_1 two univariate polynomials. Suppose that they do not have a common root and the degree of p_0 is greater then the degree of p_1. The *Sturm series* is constructed by:

$$p_i := p_{i+1}\, q_{i+1} - p_{i+2} \quad \text{for} \quad i \geq 0. \tag{A.9}$$

This is almost the same algorithm as Euclid's but the remainder p_{i+2} has negative sign. ⌟

Let us see now Sturm series $p_0, p_1, \ldots, p_k$ associated to the characteristic polynomial $P_J(\lambda)$. The series begins with two polynomials obtained by dividing $P_J(\iota\mu)$ by ι^k and separate real and imaginary parts:

$$\begin{aligned} p_0(\mu) &:= \Re\left(\tfrac{P_J(\iota\mu)}{\iota^k}\right) = a_0\,\mu^k - a_2\,\mu^{k-2} + a_4\,\mu^{k-4} \pm \ldots \\ p_1(\mu) &:= -\Im\left(\tfrac{P_J(\iota\mu)}{\iota^k}\right) = a_1\,\mu^{k-1} - a_3\,\mu^{k-3} + a_5\,\mu^{k-5} \pm \ldots \end{aligned} \tag{A.10}$$

The remaining terms are defined as in (A.9). Due to the special structure of these polynomials, they can be rewritten in the form:

$$p_i(\mu) := c_{i,0}\,\mu^{k-i} + c_{i,1}\,\mu^{k-i-2} + c_{i,2}\,\mu^{k-i-4} + \ldots \tag{A.11}$$

In these notations, the quotient q_i of (A.9) is equal to $(c_{i-1,0}/c_{i,0})\,\mu$ which provides the condition $c_{i,0} \neq 0$. Moreover, the replacement of (A.11) in (A.9) gives the following recursive formulas for computation of the coefficients $c_{i,j}$.

$$c_{i+1,j} = c_{i,j+1}\frac{c_{i-1,0}}{c_{i,0}} - c_{i-1,j+1} = \frac{1}{c_{i,0}}\det\begin{pmatrix} c_{i-1,0} & c_{i-1,j+1} \\ c_{i,0} & c_{i,j+1} \end{pmatrix}. \tag{A.12}$$

If $c_{i,0} = 0$ for some i, the quotient q_i is a higher degree polynomial and the sequence (A.9) stops at p_h with $h < k$.

The following *Routh's theorem* (see [96]) tells necessary conditions to have all the roots of a polynomial with negative real parts without computing them explicitly.

Theorem A.2.5 (Routh's theorem, see th. 13.4 of [51])**.** *With above notations, all roots λ of the real polynomial $P_J(\lambda)$ with $a_0 > 0$ lie in the negative half plane i.e. $\Re(\lambda) < 0$ if and only if $c_{i,0} > 0$ for i in $\{0, \ldots, k\}$.* ⌟

Routh theorem is also known as Routh-Hurwitz criterion because the coefficients $c_{i,0}$ for i in $\{1,\ldots,k\}$ correspond exactly to what is called *Hurwitz determinants* (see [63]); their definition related to *Hurwitz matrix* is given below.

Definition A.2.6 (Hurwitz Matrix)**.** The square *Hurwitz matrix* associated to a characteristic polynomial $P_J(\lambda)$ given as in (A.8) page 144 has the following form:

$$H = \begin{pmatrix} a_1 & a_3 & a_5 & \cdots & \cdots \\ a_0 & a_2 & a_4 & \cdots & \cdots \\ 0 & a_1 & a_3 & a_5 & \cdots \\ 0 & a_0 & a_2 & a_4 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}. \tag{A.13}$$

⌟

Definition A.2.7 (Hurwitz Determinants)**.** The *ith Hurwitz determinant* is the determinant of the ith principal minor of the above Hurwitz matrix H. There are k Hurwitz determinants for a characteristic polynomial of degree k. ⌟

Example A.2.8. We continue to apply key concepts on the Van der Pol oscillator (see equation (3.8) page 63) to show `MABSys` commands. `P` is the associated characteristic polynomial found in the example A.1.7 page 144. The following commands computes Hurwitz matrix and determinants of the same system from this characteristic polynomial.

```
> # The associated Hurwitz matrix
> H := HurwitzMatrix(P, lambda);
             [-mu    0]
       H := [         ]
             [ 1     1]

> # Associated Hurwitz determinants
> HDet := HurwitzDeterminants(P, lambda);
       HDet := [1, -mu, -mu]
```

The following two propositions are well known and facilitate the stability analysis of steady points. We give their proof (as in [12, 15]) because they give necessary Hopf bifurcation conditions that are used in § 1.5.3 page 24.

Proposition A.2.9. *If all the Hurwitz determinants $c_{i,0}$ are positive, apart perhaps $c_{k,0}$ then the associated Jacobian J has no pure imaginary eigenvalue.* ⌟

Proof. If all Hurwitz determinants $c_{i,0}$ are positive ($0 \leq i < k$) then they are a fortiori nonzero. Assume that J has pure imaginary eigenvalues $\pm\iota\tilde{\mu}$ (they are necessarily conjugate). These values $\pm\tilde{\mu}$ are then common zeros of p_0 and p_1. The gcd of p_0 and p_1 has thus degree greater than or equal to 2. This gcd is the last nonzero polynomial in the sequence $p_0,\ldots,p_{k-1}$. Thus one polynomial p_i with $0 \leq i < k$ must vanish identically. Therefore the corresponding Hurwitz determinant $c_{i,0}$ mush vanish also which is in contradiction with the positivity assumption. ⌟

Proposition A.2.10. *If all Hurwitz determinants $c_{i,0}$ (for all i in $\{0,\ldots,k-2\}$) are positive, $c_{k-1,0} = 0$ and $c_{k-2,1} < 0$ then all the eigenvalues of the associated Jacobian J have negative real parts except a purely imaginary conjugate pair.* ⌟

Proof. The polynomial p_{k-1} has the special form $p_{k-1} = c_{k,0}\,\mu$ and $c_{k,0} = 0$. Then p_{k-2} is the gcd of p_0 and p_1, it is of degree two and has the special form $p_{k-2} = c_{k-2,0}\,\mu^2 + c_{k-2,1}$. The conditions $c_{k-2,0} > 0$ and $c_{k-2,1} < 0$ let us know that the common roots $\pm\tilde{\mu}$ of p_0 and p_1 are real. Therefore J has one pair of purely imaginary conjugate eigenvalues $\pm\iota\tilde{\mu}$. Now, compute the Sturm series (see the equation (A.10) page 147) over the polynomial $\tilde{P}_J(\lambda) = P_J(\lambda) \,/\, \left(\lambda^2 + \tilde{\mu}^2\right)$. This Sturm series $\tilde{p}_0, \tilde{p}_1, \ldots, \tilde{p}_{\tilde{k}}$ can actually be derived from that of $P_J(\lambda)$:

$$\tilde{p}_0(\mu) = \frac{p_0}{(\lambda^2+\tilde{\mu}^2)},\ \tilde{p}_1(\mu) = \frac{p_1}{(\lambda^2+\tilde{\mu}^2)},\ \ldots,\ \tilde{p}_{\tilde{k}-1}(\mu) = \frac{p_{k-1}}{(\lambda^2+\tilde{\mu}^2)},\ \tilde{p}_{\tilde{k}}(\mu) = c_{k-2,0}. \tag{A.14}$$

All corresponding Hurwitz determinants are positive. According to the Routh theorem (see theorem A.2.5 page 147), all the roots of $\tilde{P}_J$ have negative real parts. This concludes the proof of the proposition. ⌟

The conditions that we are looking for so that a Hopf bifurcation occurs (see theorem A.2.3 page 146) for a parametric continuous dynamical system are given by the proposition A.2.10 page 148.

Example A.2.11. Let us reconsider the Van der Pol oscillator. This system could have a Hopf bifurcation if the necessary conditions given by the proposition A.2.10 page 148 are satisfied. These conditions can be computed automatically by using `MABSys` as follows.

```
> # Necessary conditions so that a Hopf bifurcation can happen
> pos, zero, neg := HopfBifurcationConditions(CP, lambda);
      pos, zero, neg := [1], -mu, -1
```

The result of `HopfBifurcationConditions` function corresponds to:

$$c_{0,0} = 1 > 0, \quad c_{1,0} = -\mu = 0, \quad c_{0,1} = -1 < 0. \tag{A.15}$$

Because $1 > 0$ and $-1 < 0$ are obvious, one can conclude that a Hopf bifurcation may occur for Van der Pol oscillator if $\mu = 0$. ⌟

$\mathcal{B}$

Biochemical Networks and their Modeling

The aim of this chapter is to present the first part of `MABSys` i.e. the modeling of biochemical networks by means of ODEs. In `MABSys` a biochemical network describes the interactions between macromolecules (genes, mRNAs, proteins) towards some process as binding, release, synthesis, degradation and transformation. First, we give a general survey of some biological phenomena to introduce their implementation. Then we tackle the modeling part. Two different ways of modeling are available in `MABSys`: by using directly rate laws of the reactions or by using quasi-steady state approximation (QSSA) algorithm as given in [10]. Last, we illustrate these modelings and show associated `MABSys` commands.

B.1 Cellular Biological Phenomena

In this section we glance at some intracellular phenomena and show their representation in `MABSys`.

B.1.1 Different Kinds of Reactions

Binding & Release

A *transcription* is a copy process of the coding part of a gene into an mRNA molecule. A *transcription factor* is a protein that binds to a specific part of the gene and thereby regulates the transcription. This transcription factor, along or in collaboration with other proteins, may be an *activator* by promoting the transcription or a *repressor* by blocking the transcription. Even transcription factors can be activated or deactivated by themselves or by other proteins. Binding the transcription factor to the gene is a complex mechanism that includes, for example, the opening of double helices by creating a single strand to which mRNA can bind while it is being built. There is a chain of phenomena that takes place while binding and release.

In this document, the modeling is made in its simplest way i.e. as an invertible phenomenon regulated by transcription rate constants α and θ as in figure B.1 page 152. The variable G represents the non-linked gene, P a protein which is a transcription factor and H the linked gene i.e. the gene G and the protein P together.

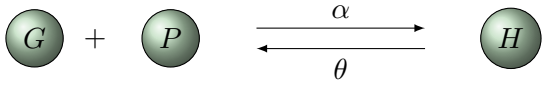


Figure B.1: Binding and release mechanisms.

Synthesis & Degradation

A *synthesis* is a phenomenon that produces a new molecule by using amino acids present in the cell. A *degradation* is the crumbling of a macromolecule into amino acids that return into the cell. For the sake of simplicity, in this document it is supposed that

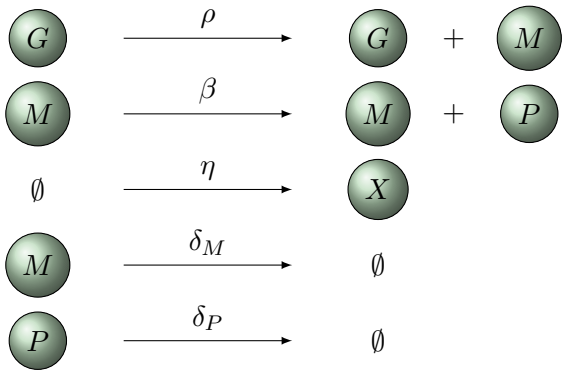


Figure B.2: Synthesis and degradation mechanisms.

cells possess enough amino acids each time that a synthesis is executed. An empty set represents these amino-acids that do not take place in the representations. Figure B.2 shows the synthesis of an mRNA denoted by M from a gene G (*transcription*), of a protein P from this mRNA (*translation*), of another protein X without considering its source and also the degradation of M and P. The letters above the arrows correspond to associated rate constants. Remark that after the synthesis of a mRNA or the protein P, the associated gene or mRNA remain unchanged. While these procedures, they serve just as an information board.

Transformation

A *transformation* corresponds to the conversion of species. Figure B.3 shows a classical transformation of two *reactants* R_1 and R_2 into two *products* P_1 and P_2 with a rate constant equal to k.

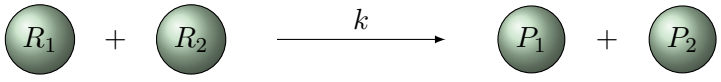


Figure B.3: Transformation mechanism.

B.1.2 Implementation of a Reaction

In this subsection, we detail and illustrate the implementation of a reaction in the package `MABSys`.

The implementation of a biological system is one of the first steps for its treatment. Graphical diagrams are useful for visualization and having a first thought about how these mechanisms work. However, computer algebra softwares need more formal structuring. SBML (see [121]) is a computer-readable format for representing biochemical reaction networks. There exist also structured diagram editors for drawing biochemical gene-regulatory networks as CellDesginer (see [40]) or JDesigner (see [98]) which permit to export the network in the form of SBML. In this document, we do not focus on SBML because the main subjects (simplification of models) can be used for dynamical systems coming from any kind of scientific context. Biology is one of the several application domains. For now, the description made in `MABSys` includes only the most important parts of a biochemical reaction in its simplest way. In the future, it can be interesting to integrate SBML format.

Here, every one-way reaction is represented by a simple `Reaction` data structure. It is created by the `NewReaction` constructor, defined by the key word `Reaction` and the following elements:

- reactants given as a linear combination of species names in the reaction;
- products given as a linear combination of species names in the reaction;
- the rate law of the system. A classical way of modeling reactions is to use the mass-action law indicated by the keyword `MassActionLaw` but one can also choose another reaction law in the form of rational fractions (for instance Hill functions) indicated by the keyword `CustomizedLaw`. For the mass-action law one must give associated rate constant and for a customized law the whole rate;
- a boolean that indicates the velocity of the reaction; it can be true for fast and false for slow reactions. The default value is false.

A system of reaction that corresponds to the `ReactionSystem` data structure is a list of reactions.

Example B.1.1. Let us see a basic enzymatic reaction system given in figure B.4 and its

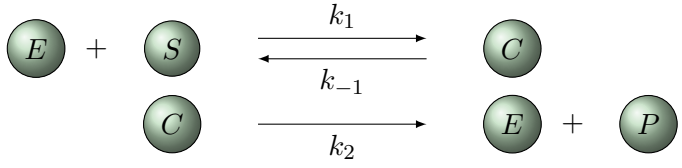


Figure B.4: Basic enzymatic biochemical reaction system

construction in `MABSys`. This system describes the transformation of a substrate S into

a product P under the action of the enzyme E. Meanwhile, an intermediate complex C is produced. Each reaction follows the mass-action law and associated rate constants are indicated above or below of the arrows. The system contains 3 one-way reactions defined in two timescales. The first two reactions are considered fast w.r.t. the third one meaning that their rate constants k_1, k_{-1} are supposed to be greater than the third one k_2.

```
> # The biochemical reactions
> R1 := NewReaction(E+S,C,MassActionLaw(k1),fast=true);
       R1 := Reaction([E, S], [C], MassActionLaw(k1), true)

> R2 := NewReaction(C,E+S,MassActionLaw(km1),fast=true);
       R2 := Reaction([C], [E, S], MassActionLaw(km1), true)

> R3 := NewReaction(C,E+P,MassActionLaw(k2));
       R3 := Reaction([C], [E, P], MassActionLaw(k2), false)

> type(R1, Reaction);
       true

> # The biochemical reaction system
> RS := [R1,R2,R3];
       RS := [Reaction([E, S], [C], MassActionLaw(k1), true),
              Reaction([C], [E, S], MassActionLaw(km1), true),
              Reaction([C], [E, P], MassActionLaw(k2), false)]

> type(RS, ReactionSystem);
       true
```

In `MABSys` there are also many auxiliary functions to manipulate the reactions such as to extract the name of products, the name of reactants, to get fast or slow reactions, to get rate constants etc. For all these functions see associated help pages [69].

B.2 Modeling by means of Nonlinear ODEs

This section tells how to transform a biological system composed of above phenomena into a system of ODEs and present associated implementations. First I give an overview about reactions that follows the classical mass-action law or a customized law, associated stoichiometric matrix and rate vectors. Then I show the basic modeling method and the quasi-steady state approximation (QSSA). This is given by an example using the new algorithm introduced by the authors of [10]. I illustrate each of these notions with their implementation in `MABSys`.

B.2.1 Background Knowledge

This subsection recalls some of the main notations and the notions about the modeling of biological systems which are useful to clarify § 1.3 page 11.

In this document, the name of a protein in the equations refer to its concentration in the medium where the reactions happen. If a reaction possesses a gene G within its species as in binding, release or synthesis, then one can not talk about its concentration. The derivative of G suppose the continuous variation of the gene which is meaningless.

However this dynamic may be seen as a rate of the gene transcription. In binding and release phenomena, G and H represent two different states of the same gene. They have complementary gene transcription rates i.e. $G + H = \gamma_0$ for some γ_0 in $\mathbb{R}$. A similar logic is valid also for mRNAs. The derivative of M may be seen as the variation in the rate of mRNA translation. In the sequel, the derivative of a species X w.r.t. time t is denoted by a dot $\dot{X}$.

Mass-action Law

The mass-action law is a classical way of defining the dynamic of a reaction in function of the concentration of the present species and the rate constant which quantifies its speed. Mainly, this law tells that, for an elementary reaction, the *reaction rate* is proportional to the reactant concentrations raised to a particular power depending on their ratio and to the rate constant.

Two biochemical reactions are given in figure B.5. The reactants A and B are

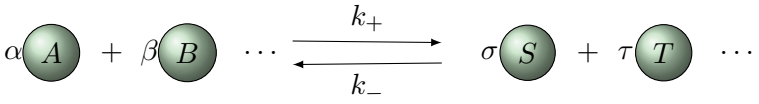


Figure B.5: Two one-way reaction

transformed into the products S and T by a rate constant k_+ and vice versa by a rate constant k_-. All these rate constants are considered to be strictly positive. The reactions tells how many molecules are needed to execute it. For example in figure B.5 α times A and β times B let the forward reaction (from left to right) be executed. The mass-action law indicates:

- the forward reaction rate $= k_+ A^\alpha B^\beta \dots$
- the backward reaction rate $= k_- S^\sigma T^\tau \dots$

When the concentration of a species increases, its velocity is taken positive; in the contrary case, it is taken negative. Here is the model that respects the mass-action law for the forward reaction:

$$\begin{cases} \dot{A} = \dot{B} = \dots = -k_+ A^\alpha B^\beta \dots, \\ \dot{S} = \dot{T} = \dots = k_+ A^\alpha B^\beta \dots \end{cases} \tag{B.1}$$

and that for the backward reaction:

$$\begin{cases} \dot{A} = \dot{B} = \dots = k_- S^\sigma T^\tau \dots, \\ \dot{S} = \dot{T} = \dots = -k_- S^\sigma T^\tau \dots. \end{cases} \tag{B.2}$$

Example B.2.1. In the following code example we create a reaction where two molecules of A and one molecule of B are transformed into the product C by following the mass-action law with k as the rate constant. By default, the reaction is considered as slow. Then we call some `MABSys` commands to illustrate some of its functionalities.

```
> # Mass-action law
> R := NewReaction(2*A+B,C,MassActionLaw(k));
       R := Reaction([2 A, B], [C], MassActionLaw(k), false)

> GetReactionReactants(R);
       [2 A, B]

> GetReactionProducts(R);
       [C]

> GetReactionRate(R);
            2
       k A  B
```

Customized Law

A customized law let the user to indicate any rate law in rational function for a given reaction. Again, names of species indicate associated concentrations and new letters are considered as parameters.

Example B.2.2. In this code example we create two reactions. The transformation consists of converting a protein P into another protein Q by respecting the customized law given just by some parameter k. We also create the synthesis of the protein P without considering the source and assuming that the reaction follows a customized law in the form of a Hill function. By default both are considered as slow.

```
> # Customized law
> R1 := NewReaction(P,Q,CustomizedLaw(k));
       R1 := Reaction([P], [Q], CustomizedLaw(k), false)

> GetReactionRate(R1);
       k

> R2 := NewReaction(0,P,CustomizedLaw(Q/(Q+theta)));
                                                Q
       R2 := Reaction([], [P], CustomizedLaw(---------), false)
                                             Q + theta

> GetReactionRate(R2);
           Q
       ---------
       Q + theta
```

Rate Vector and Stoichiometric Matrix

The modeling of a biological system composed of many reactions requires the combination of each reaction information. An algorithmic way to obtain these models necessitates the usage of the *rate vector* and the *stoichiometric matrix*.

Definition B.2.3 (Rate Vector)**.** Let us consider a system of r one-way reactions with $v_1, \ldots, v_r$ as their reaction rates. The vector $V = (v_1, \ldots, v_r)$ of dimension r is called its *rate vector*. ⌟

Definition B.2.4 (Stoichiometric Matrix)**.** Let us consider a system of r one-way biochemical reactions between the species denoted by $X = (x_1, \ldots, x_k)$ where k is their cardinal. The *stoichiometric matrix* is a matrix of dimension $k \times r$. Each column corresponds to a particular biochemical reaction and each row to a species. Its (i, j)th entry is the number of molecules needed for the ith species in the jth reaction i.e. its *stoichiometric coefficient*. If the species is formed by the reaction, the coefficient has a positive sign, if it is consumed, the stoichiometric coefficient appears with a negative sign. All the other entries of the column which correspond to species that do not participate to the reaction are zero. If a species appears on the left and right hand side of a reaction, then its stoichiometric matrix is the difference of the number of molecules that were formed and the number of molecules that were consumed. ⌟

Example B.2.5. Let us return to the example B.1.1 page 153 where a basic enzymatic reaction system of 3 one-way reactions, given in figure B.4 page 153, is encoded by `MABSys`. Remember that the variable `RS` corresponds to the system representation. The following `MABSys` commands compute associated rate vector of dimension 3 and stoichiometric matrix of dimension 4×3.

```
> # The rate vector
> RateVector(RS);
       [k1 E S]
       [       ]
       [km1 C ]
       [       ]
       [ k2 C ]

> # The stoichiometric matrix
> StoichiometricMatrix(RS, [E,S,C,P]);
       [-1      1      1]
       [                ]
       [-1      1      0]
       [                ]
       [ 1     -1     -1]
       [                ]
       [ 0      0      1]
```

B.2.2 Basic Modeling

One of the simplest classical way of modeling a biological system consists of constructing a model by following the rate laws (mass-action or customized law) of the reactions. The basic modeling of a biological system of r one-way reactions that involves k species denoted by $X = (x_1, \ldots, x_k)$ by means of ODEs requires associated rate vector V of dimension r and stoichiometric matrix $\mathcal{M}$ of dimension $k \times r$. In vector-valued notations, the basic model is given by the following formula:

$$\dot{X} = \mathcal{M}\, V. \tag{B.3}$$

This model can be obtained by `ReactionSystem2ODEs` function of `MABSys`.

Example B.2.6. The computation of the basic model that corresponds to the basic

enzymatic reaction system of figure B.4 page 153 follows.

$$\begin{pmatrix} \dot{E} \\ \dot{S} \\ \dot{C} \\ \dot{P} \end{pmatrix} = \begin{pmatrix} -1 & 1 & 1 \\ -1 & 1 & 0 \\ 1 & -1 & -1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} k_1\, E\, S \\ k_{-1}\, C \\ k_2\, C \end{pmatrix} \quad \Leftrightarrow \quad \left\{ \begin{array}{rcl} \dot{E} & = & -k_1\, E\, S + k_{-1}\, C + k_2\, C, \\ \dot{S} & = & -k_1\, E\, S + k_{-1}\, C, \\ \dot{C} & = & k_1\, E\, S - k_{-1}\, C - k_2\, C, \\ \dot{P} & = & k_2\, C. \end{array} \right. \tag{B.4}$$

This biological system is already encoded in the variable `RS` in example B.1.1 page 153. Its automatic modeling follows.

```
> # Basic modeling
> ReactionSystem2ODEs(RS, [E,S,C,P]);
        d
       [-- E(t) = -k1 E(t) S(t) + km1 C(t) + k2 C(t),
        dt
        d
        -- S(t) = -k1 E(t) S(t) + km1 C(t),
        dt
        d
        -- C(t) = k1 E(t) S(t) - km1 C(t) - k2 C(t),
        dt
        d
        -- P(t) = k2 C(t)]
        dt
```

B.2.3 Quasi-Steady State Approximation

There are different ways to perform quasi-steady state approximation (QSSA) which is an inexact simplification method. The classical one which consists to replace some variables with their steady state values is useful in many cases. However, for example when one is interested in timescales over which the system equilibrates or in period and amplitude of oscillations for an oscillating system, this classical QSSA must be adjusted. Indeed, the separation of timescales is the key for observing nontrivial behaviors. In biology, many processes like dimerization occur faster than others. In [10], QSSA method considered by [113, 4] is reformulated and made fully algorithmic. The algorithm can be expressed by means of either differential elimination methods (see [9, 56, 66, 95]) or regular chains using [67]. In [11] authors apply QSSA to a family of networks proposed in [12] and obtain a more precise model (see also § 1.3.3 page 14).

For biological systems, two classes of reactions are considered: slow and fast ones. The idea is to study the dynamics of slow reactions, assuming that the fast ones are at quasi-equilibrium, thereby removing from the system of ODEs, the differential equations which describe the evolution of the variables at quasi-equilibrium. Here, I present the main idea of this method through the basic enzymatic reaction system given in the example B.1.1 page 153. The main hypothesis which permits to perform QSSA is that the reversible reaction i.e. the first one in figure B.4 page 153 is much faster then the second reaction. In other words, one assumes that the parameters k_1 and k_{-1} are greater than k_2. In addition to this hypothesis, one uses the pre-equilibrium approximation on the fast reactions. That corresponds to say that the reaction rates are equal in two directions i.e. $k_1\, E\, S = k_{-1}\, C$.

As for the basic modeling, QSSA requires to know the contribution of each reaction. QSSA handles the fast reactions which are assumed to be at equilibrium in a different manner than the slow reactions. The contribution of each slow reaction is derived from the associated rate law as usual. The contribution of each fast reaction is represented by a new variable. The underlying idea is that the dynamic of the fast reactions is (for the moment) considered as unknown. This means that one adds a degree of freedom for each fast reaction. This freedom, in fact, allows the system to stay on the equilibrium conditions. Performing elimination on the extra unwanted variables, one gets a set of differential equations in the reactants only.

Over the basic enzymatic reaction system example, the contribution of the fast reaction is denoted by F (for fast reaction) and the contribution of the slow reaction is $k_2\,C$ (compare with the equations (B.4) page 158):

$$\left\{\begin{array}{rcl}\dot{E} &=& -F + k_2\,C,\\ \dot{S} &=& -F,\\ \dot{C} &=& F - k_2\,C,\\ \dot{P} &=& k_2\,C.\end{array}\right. \tag{B.5}$$

By differential elimination and introducing $K = k_{-1}/k_1$ one gets:

$$F = \frac{k_2\,E\,S\,(S+K)}{K\,(S+E+K)},\ \dot{E} = \frac{k_2\,E^2\,S}{K\,(S+E+K)},\ \dot{P} = \frac{k_2\,E\,S}{K},\ \dot{S} = -\frac{k_2\,E\,S\,(S+K)}{K\,(S+E+K)},\ C = \frac{E\,S}{K}\cdot \tag{B.6}$$

Using the conservation laws $E + C = E_0 + C_0$ and $S + C + P = S_0 + C_0 + P_0$ (that one can automatically deduce from the system (B.5) by means of linear algebra), assuming that $C_0 = P_0 = 0$ and introducing $V_m = k_2\,E_0$, computations yield:

$$\dot{S} = -\frac{V_m\,S\,(K+S)}{K\,E_0 + (K+S)^2} \tag{B.7}$$

where 0 subscript denotes the initial concentrations.

The classical formula given in the early XX$^{\text{th}}$ century for the same problem by Henri, Michaëlis and Menten (see [54, 77]) on the one hand, Briggs and Haldane (see [18]) on the other hand writes:

$$\dot{S} = -\frac{V_m\,S}{K+S} \tag{B.8}$$

where $V_m = k_2\,E_0$ and K are parameters. Both reductions rely on a few extra assumptions and has different values for the parameter K; k_{-1}/k_1 in Henri-Michaëlis-Menten case, $(k_{-1}+k_2)/k_1$ in Briggs-Haldane's.

Over this easy example, the benefits of (B.7) w.r.t. (B.8) are clear. The reduction is automatic and yields a formula which seems more accurate when S is not supposed to be greater than E_0. Observe that with this assumption the formula (B.8) is recovered from (B.7). For numerical simulations that verify these phenomena see [10, 11].

Example B.2.7. The formula (B.7) can be computed automatically by using the function `ModelReduce` of `MABSys`. In example B.1.1 page 153 the reactions are encoded in the variable `RS` with their velocity assumptions.

```
> # Modeling by QSSA
> QSSAModel := ModelReduce(RS, [E,C,P,S], useConservationLaws=true)[1,1]:
> # The further simplifications
> QSSAModel := simplify(subs({k1=km1/K,C_0=0,P_0=0,k2=Vm/E_0},QSSAModel)):
> # The last equation correspond to the substrate.
> QSSAModelS := QSSAModel[-1];
                      d                 Vm S(t) (K + S(t))
      QSSAModelS := -- S(t) = - ------------------------------
                      dt                          2            2
                                2 S(t) K + S(t)  + E_0 K + K
```

Bibliography

[1] N.-H. Abel. Beweis der Unmöglichkeit algebraische Gleichungen von höheren Graden als dem vierten allgemein aufzulösen. *J. reine u. Angew. Math.*, 1:65–84, 1826. Reprinted in Abel, N. H. (Ed. L. Sylow and S. Lie). Christiania [Oslo], Norway, 1881. Reprinted in New York: Johnson Reprint Corp., pp. 66-87, 1988.

[2] Observatoire Océanologique de Banyuls-Sur-Mer, Laboratoire Modèles en Biologie Cellulaire et Évolutive. `http://www.obs-banyuls.fr/UMR7628/`.

[3] G. Batt, D. Ropers, H. de Jong, J. Geiselmann, R. Mateescu, M. Page, and D. Schneider. Validation of qualitative models of genetic regulatory networks by model checking: analysis of the nutritional stress response in Escherichia Coli. *Bioinformatics*, 21(Suppl 1):i19–i28, 2005.

[4] M. R. Bennett, D. Volfson, L. Tsimring, and J. Hasty. Transient dynamics of genetic regulatory networks. *Biophysical Journal*, 92, issue 10:3501–3512, May 2007.

[5] G. Bluman and S. Anco. *Symmetry and Integration Methods for Differential Equations*, volume 154 of *Applied Mathematical Sciences Series*. Springer, 2002.

[6] G. Bluman and S. Kumei. *Symmetries and Differential Equations*, volume 81 of *Applied Mathematical Sciences Series*. Springer-Verlag, New York, 2 edition, August 1989.

[7] F. Boulier. *Réécriture algébrique dans les systèmes d'équations différentielles polynomiales en vue d'applications dans les Sciences du Vivant*. H497. Université de Lille 1, LIFL, 59655 Villeneve d'Ascq France, May 2006. Mémoire d'Habilitation à Diriger des Recherches.

[8] F. Boulier. Differential Elimination and Biological Modelling. In Markus Rosenkranz and Dongming Wang, editors, *Gröbner Bases in Symbolic Analysis Workshop D2.2 of the Special Semester on Gröbner Bases and Related Methods*, volume 2 of *Radon Series Comp. Appl. Math*, pages 111–139, Hagenberg Autriche, 2007. De Gruyter.

[9] F. Boulier, D. Lazard, F. Ollivier, and M. Petitot. Representation for the radical of a finitely generated differential ideal. In *ISSAC'95: Proceedings of the 1995 International Symposium on Symbolic and Algebraic Computation*, pages 158–166, New York, NY, USA, 1995. Montreal, Quebec, Canada, ACM Press. `http://hal.archives-ouvertes.fr/hal-00138020`.

[10] F. Boulier, M. Lefranc, F. Lemaire, and P.-E. Morant. Model Reduction of Chemical Reaction Systems using Elimination. *MACIS*, 2007. `http://hal.archives-ouvertes.fr/hal-00184558/fr`.

[11] F. Boulier, M. Lefranc, F. Lemaire, and P.-E. Morant. Applying a rigorous quasi-steady state approximation method for proving the absence of oscillations in models of genetic circuits. In *Proceedings of Algebraic Biology 2008*, pages 56–65, 2008. Available at the url `http://hal.archives-ouvertes.fr/hal-00213327/`.

[12] F. Boulier, M. Lefranc, F. Lemaire, P.-E. Morant, and A. Ürgüplü. On proving the absence of oscillations in models of genetic circuits. *Second International Conference Algebraic Biology*, 4545:66–80, September 2007. `http://hal.archives-ouvertes.fr/hal-00139667`.

[13] F. Boulier, F. Lemaire, and M. Moreno Maza. Computing differential characteristic sets by change of ordering. *Journal of Symbolic Computation*, 45(1):124–149, 2010. Available at `http://hal.archives-ouvertes.fr/hal-00141095`.

[14] F. Boulier, F. Lemaire, A. Sedoglavic, and A. Ürgüplü. Towards an Automated Reduction Method for Polynomial ODE Models in Cellular Biology - Supplementary Data, 2008. Available at `www.lifl.fr/~urguplu`.

[15] F. Boulier, F. Lemaire, A. Sedoglavic, and A. Ürgüplü. Towards an Automated Reduction Method for Polynomial ODE Models in Cellular Biology. *Mathematics in Computer Science, Special issue Symbolic Computation in Biology*, 2(3):443–464, March 2009. Available at `www.lifl.fr/~urguplu`.

[16] V. V. Breusegem and G. Bastin. Reduced Order Mathematical Modeling of Reaction Systems: A Singular Perturbation Approach. In *Proceedings of IEEE Conference on Decision and Control*, volume 2, pages 1049–1054, Piscataway, NJ, 1991.

[17] P. Bridgman. *Dimensional Analysis*. Yale University press, 1922.

[18] G. E. Briggs and J. B. S. Haldane. A note on the kinetics of enzyme action. *Biochemical Journal*, 19:338–339, 1925.

[19] C. W. Brown. QEPCAD B: A program for computing with semi-algebraic sets using CADs. Quantifier Elimination by Partial Cylindrical Algebraic Decomposition. *SIGSAM Bull.*, 37(4):97–108, 2003. `http://www.usna.edu/Users/cs/qepcad/B/QEPCAD.html`.

[20] B. Buchberger. Bruno Buchberger's PhD thesis 1965: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. *Journal of symbolic computation*, 41(3-4):475–511, 2006. English translation by Michael P. Abramson.

[21] E. Buckingham. On Physically Similar Systems; Illustrations of the Use of Dimensional Equations. *Phys. Rev.*, 4(4):345–376, Oct 1914.

[22] G. I. Burde. Expanded Lie group transformations and similarity reductions of differential equations. In A. G. Nikitin, V. M. Boyko, and R. O. Popovych, editors, *Symmetry in nonlinear mathematical physics Part I*, volume 43 of *Proceedings of Institute of Mathematics of NAS of Ukraine*, pages 93–101, Kiev, Ukraine, July 9–15 2002.

[23] J. Carminati, J. S. Devitt, and G. J. Fee. Isogroups of differential equations using algebraic computing. *Journal of Symbolic Computation*, 14(1):103–120, 1992.

[24] E. Cartan. *La méthode du repère mobile, la théorie des groups continus et les espaces généralisés*. Exposés de géométrie – 5. Hermann, Paris, 1935.

[25] D. Castro, M. Giusti, J. Heintz, G. Matera, and L. M. Pardo. The Hardness of Polynomial Equation Solving. *Foundations of Computational Mathematics*, 3:347–420, 2003.

[26] F. Chyzak. *Fonctions holonomes en calcul formel*. PhD thesis, École polytechnique, 1998. INRIA, TU 0531. 227 pages.

[27] R. M. Corless and D. J. Jeffrey. The turing factorization of a rectangular matrix. *Sigsam Bulletin*, 31:20–28, September 1997. University of Western Ontario. London, Ontario, Canada N6A 5B7. Available at `http://www.apmaths.uwo.ca/~rcorless/frames/PAPERS/SYMBOLIC/index.html`.

[28] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Algebraic Geometry and Commutative Algebra*. Springer Verlag, 2nd edition, 1996.

[29] G. Darboux. *Leçons sur la théorie générale des surfaces et les applications géométriques du calcul infinitésimal*. Imprimerie Gautier Villards et Fils, 1894. Digital version available at `http://name.umdl.umich.edu/ABV4153.0001.001` or at `http://gallica.bnf.fr/`.

[30] W. A. de Graaf. Classification of solvable lie algebras. *Experimental mathematics*, 1:15–25, 2005.

[31] W. A. de Graaf, G. Ivanyos, A. Küronya, and L. Rónyai. Computing Levi Decompositions of Lie Algebras. *Applicable Algebra in Engineering Communication and Computing*, 8:291–303, 1997.

[32] E. Doedel. AUTO Software for Continuation and Bifurcation Problems in ODEs, 1996. `http://indy.cs.concordia.ca/auto`.

[33] A. Dolzmann and T. Sturm. Redlog: computer algebra meets computer logic. *SIGSAM Bull.*, 31(2):2–9, 1997.

[34] D. Eisenbud. *Commutative Algebra with a View Toward Algebraic Geometry*, volume 150 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1995.

[35] B. Ermentrout. *Simulating, Analyzing, and Animating Dynamical Systems: A Guide to XPPAUT for Researchers and Students*, volume 14 of Software, Environments, and Tools. SIAM, 2002.

[36] C. P. Fall, E. S. Marland, J. M. Wagner, and J. J. Tyson, editors. *Computational Cell Biology*, volume 20 of *Interdisciplinary Applied Mathematics*. Springer-Verlag, 2002.

[37] M. Fels and P. J. Olver. Moving Coframes: I. A Practical Algorithm. *Acta Applicandae Mathematicae*, 51(2):161–213, April 1998.

[38] M. Fels and P. J. Olver. Moving coframes. II. Regularization and theoretical foundations. *Acta Applicandae Mathematicae*, 55(2):127–208, January 1999.

[39] W. Fulton and J. Harris. *Representation Theory*, volume 129 of *Graduate Texts in Mathematics*. Springer-Verlag, 1991.

[40] A. Funahashi, Y. Matsuoka, A. Jouraku, M. Morohashi, N. Kikuchi, and H. Kitano. Celldesigner 3.5: A versatile modeling tool for biochemical networks. In *Proceedings of the IEEE*, volume 96, pages 1254–1265, August 2008. Software available at `http://www.celldesigner.org/`.

[41] M. J. Gardner, K. E. Hubbard, C. T. Hotta, A. N. Dodd, and A. A. Webb. How plants tell the time. *Biochemical Journal*, 397:15–24, Jul 2006. London, England. Available at `http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1479754`.

[42] K. Gatermann. *Computer algebra methods for equivariant dynamical systems*, volume 1728 of *Lecture Notes in Mathematics*. Springer New York, 2000.

[43] K. Gatermann and S. Hosten. Computational algebra for bifurcation theory. *Journal of Symbolic Computation*, 40(4-5):1180–1207, October-November 2005.

[44] M. Giusti and J. Heintz. Kronecker's smart, little black boxes . *Foundations of Computational Mathematics eds. Ronald. A. DeVore, Arieh Iserles and Endre Süli*, 284:69–104, 2001.

[45] M. Giusti, J. Heintz, J. E. Morais, J. Morgenstern, and L. M. Pardo. Straight-line programs in geometric elimination theory. *Journal of Pure and Applied Algebra*, 124:101–146, 1998.

[46] M. Giusti, G. Lecerf, and B. Salvy. A Gröbner Free Alternative for Polynomial System Solving. *Journal of Complexity*, 17:154–211, 2001.

[47] B. C. Goodwin. *Temporal organization in cells*. Academic Press, London, 1963.

[48] B. C. Goodwin. *Advances in Enzyme Regulation*, volume 3. Pergamon Press, Oxford, 1965.

[49] J. S. Griffith. Mathematics of Cellular Control Processes. I. Negative Feedback to One Gene. *Journal of theoretical biology*, 20:202–208, 1968.

[50] J. Guckenheimer, M. Myers, and B. Sturmfels. Computing Hopf Bifurcations I. *SIAM Journal on Numerical Analysis*, 1997.

[51] E. Hairer, S. P. Norsett, and G. Wanner. *Solving ordinary differential equations I : nonstiff problems*. Springer-Verlag New York, Inc., New York, NY, USA, 2nd revised edition, 1993.

[52] J. Hale and H. Koçak. *Dynamics and Bifurcations*, volume 3 of *Texts in Applied Mathematics*. Springer-Verlag, New York, 1991.

[53] A. Heck. *Introduction to Maple*. ISBN 0-387-00230-8. Springer-Verlag, third edition, 2003.

[54] V. Henri. *Lois générales de l'Action des Diastases*. Hermann, Paris, 1903.

[55] F. Horn and R. Jackson. General mass action kinetics. *Archive for Rational Mechanics and Analysis*, 47:81–116, 1972.

[56] E. Hubert. Factorization-free Decomposition Algorithms in Differential Algebra. *Journal of Symbolic Computation*, 29(4-5):641–662, 2000.

[57] E. Hubert. AIDA Maple package: Algebraic Invariants and their Differential Algebras, 2007. available at `http://www-sop.inria.fr/cafe/Evelyne.Hubert/aida/`.

[58] E. Hubert. Differential invariants of a Lie group action: syzygies on a generating set. *Journal of Symbolic Computation*, 44(4):382–416, April 2009.

[59] E. Hubert and I. Kogan. Rational invariants of a group action. construction and rewriting. *Journal of Symbolic Computation*, 42:1-2:203–217, 2007.

[60] P. E. Hydon. Symmetries and first integrals of ordinary difference equations. *Proceedings of the Royal Society of London*, A(456):2835–2855, 2000. Available on author's webpage.

[61] H. D. Jong. Modeling and Simulation of Genetic Regulatory Systems: A Literature Review. *Journal of Computational Biology*, 9(1):67–103, January 2002.

[62] N. Joshi and P. Vassiliou. The existence of Lie symmetries for first order analytic discrete dynamical systems. *Journal of Mathematical Analysis and Applications*, 195:872–887, November 1995.

[63] M. E. Kahoui and A. Weber. Deciding Hopf bifurcations by quantifier elimination in a software-component architecture. *Journal of Symbolic Computation*, 30(2):161–179, 2000.

[64] R. Khanin. Dimensional Analysis in Computer Algebra. In B. Mourrain, editor, *Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation*, pages 201–208, London, Ontario, Canada, July 22–25 2001. ACM, ACM press.

[65] I. Kogan and P. Olver. Invariant Euler-Lagrange equations and the invariant variational bicomplex. *Acta Appl. Math*, 76:137–193, 2003.

[66] E. R. Kolchin. *Differential Algebra and Algebraic Groups*. Academic Press, New York, 1973.

[67] F. Lemaire, M. Moreno Maza, and Y. Xie. The RegularChains library in MAPLE 10. In I. S. Kotsireas, editor, *The MAPLE conference*, pages 355–368, 2005.

[68] F. Lemaire, A. Sedoglavic, and A. Ürgüplü. Moving Frame Based Strategies for Reduction of Ordinary Differential/Recurrence Systems Using their Expanded Lie Point Symmetries. Internal report, LIFL – CNRS, UMR 8022, University of Lille 1, January 2008. Available at the url `www.lifl.fr/~urguplu`.

[69] F. Lemaire and A. Ürgüplü. Modeling and Analysis of Biological Systems, 2008. Maple package (available at `www.lifl.fr/~urguplu`).

[70] D. Levi and P. Winternitz. Continuous symmetries of difference equations. *Journal of Physivs A: Mathematical and General*, 39:R1–R63, 2006.

[71] J. Liao and J. Delgado. Advances in Metabolic Control Analysis. *Biotechnology progress*, 9:221–233, 1993.

[72] D. G. Luenberger. *Introduction to Dynamic Systems. Theory, Models and Applications*. John Wiley and Sons, United States of America, Stanford University, 1979.

[73] E. Mansfield. Algorithms for symmetric differential systems. *Foundations of Computational Mathematics*, 1:335–383, April 2001.

[74] E. Mansfield. Indiff: a MAPLE package for over determined differential systems with Lie symmetry, 2001. Available at `http://www.kent.ac.uk/IMS/personal/elm2/`.

[75] T. Masson. Géométrie différentielle, groupes et algèbres de Lie, fibrés et connexions, 2001.

[76] T. A. McMahon and J. T. Bonner. *On Size and Life*. Scientific American Library, New York, 1983.

[77] L. Michaëlis and M. Menten. Die kinetik der invertinwirkung (the kinetics of invertase activity). *Biochemische Zeitschrift*, 49:333–369, 1973. Partial english translation available on `http://web.lemoyne.edu/~giunta/menten.html`.

[78] B. Mishra. Algebraic Systems Biology: Theses and Hypotheses. *Proceedings of Algebraic Biology* 2007 – *Second International Conference*, 4545:1–14, July 2007.

[79] P.-E. Morant, C. Vandermoere, Q. Thommen, B. Parent, F. Lemaire, F. Corellou, C. Schwartz, F.-Y. Bouget, and M. Lefranc. Oscillateurs génétiques simples. Applications à l'horloge circadienne d'une algue unicellulaire. In *proceedings of the* Rencontre du non linéaire, Paris, 2007. `http://nonlineaire.univ-lille1.fr`.

[80] J. Morgenstern. How to compute fast a function and all its derivatives: a variation on the theorem of Baur-Strassen. *SIGACT News*, 16(4):60–62, 1984.

[81] K. Murakami. Stability and bifurcation in a discrete time predator-prey model. *Journal of Difference Equations and Applications*, 13:911–925, October 2007. (Tokushima, Japan). Available at `http://dx.doi.org/10.1080/10236190701365888`.

[82] J. D. Murray. *Asymptotic Analysis*, volume 48 of *Applied Mathematical Sciences*. Springer, 1984.

[83] J. D. Murray. *Mathematical Biology*, volume 17 of *Interdisciplinary Applied Mathematics*. Springer, 2002.

[84] W. Niu and D. Wang. Algebraic approaches to stability analysis of biological systems. *Mathematics in Computer Science*, 1:507–539, 2008.

[85] M. S. Okino and M. L. Mavrovouniotis. Simplification of Mathematical Models of Chemical Reaction Systems. *Chemical Reviews*, 98(2):391–408, March/April 1998.

[86] P. J. Olver. *Applications of Lie Groups to Differential Equations*. Springer-Verlag, 2nd edition, 1993.

[87] P. J. Olver. *Equivalence, Invariants, and Symmetry*. Cambridge University Press, 1995.

[88] P. J. Olver. *Classical Invariant Theory*. Cambridge University Press, first edition, 1999.

[89] P. J. Olver. Lectures on moving frames. School of Mathematics, University of Minnesota, 2008. available at `http://www.math.umn.edu/~olver/paper.html`.

[90] P. J. Olver, M. Petitot, and A. Ürgüplü, 2009. Private Communication.

[91] B. Palenik, J. Grimwood, A. Aerts, P. Rouze, A. Salamov, N. Putnam, C. Dupont, R. Jorgensen, E. Derelle, S. Rombauts, K. Zhou, R. Otillar, S. S. Merchant, S. Podell, T. Gaasterland, C. Napoli, K. Gendler, A. Manuell, V. Tai, O. Vallon, G. Piganeau, S. Jancek, M. Heijde, K. Jabbari, C. Bowler, M. Lohr, S. Robbens, G. Werner, I. Dubchak, G. J. Pazour, Q. Ren, I. Paulsen, C. Delwiche, J. Schmutz, D. Rokhsar, Y. V. de Peer, H. Moreau, and I. V. Grigoriev. The tiny eukaryote ostreococcus provides genomic insights into the paradox of plankton speciation. *Proc Natl Acad Sci U S A*, 104:7705–7710, 2007. Available at `http://genome.jgi-psf.org/Ostta4/Ostta4.home.html`.

[92] T. Peyovitch. Sur les semi-invariants. *Bulletin de la Société Mathématique de France*, 53:208 – 225, 1925. Available at `http://www.numdam.org/item?id=BSMF_1925__53__208_0`.

[93] R. Porreca, S. Drulhe, H. de Jong, and G. Ferrari-Trecate. Structural Identification of Piecewise-Linear Models of Genetic Regulatory Networks. *Journal of Computational Biology*, 15(10):1365–1380, 2008.

[94] H. Rabitz, M. Kramer, and D. Dacol. Sensitivity Analysis of Chemical Kinetics. *Annual Review of Physical Chemistry*, 34:419–461, 1983.

[95] J. F. Ritt. *Differential Algebra*. American Mathematical Society Colloquium Publications, Vol. XXXIII. AMS, New York, N. Y., 1950. `http://www.ams.org/online_bks/coll33`.

[96] E. J. Routh. *A Treatise on the Stability of a Given State of Motion*. Macmillan, University of Cambridge, London, 1877. The Adams Prize.

[97] M. Safey El Din. RAGlib: A library for real solving polynomial systems of equations and inequalities. Real Algebraic Library Maple package. part of the SALSA library, 2003. Release 2.32. Available on `http://www-spiral.lip6.fr/~safey/RAGLib`.

[98] H. Sauro. An Introduction to Biochemical Modeling using JDesigner. Tutorial., October 2004. Software available at `http://www.sys-bio.org/software/jdesigner.htm`.

[99] J. Schnakenberg. Simple chemical reaction systems with limit cycle behaviour. *Journal of Theoretical Biology*, 81(3):389–400, December 1979.

[100] J. T. Schwartz. Probabilistic algorithms for verification of polynomial identites. *Journal of ACM*, 27:701 – 717, 1980.

[101] A. Sedoglavic. Reduction of Algebraic Parametric Systems by Rectification of their Affine Expanded Lie Symmetries. *Proceedings of Algebraic Biology* 2007 – *Second International Conference*, 4545:277–291, July 2007. `http://hal.inria.fr/inria-00120991`.

[102] A. Sedoglavic and A. Ürgüplü. Expanded Lie Point Symmetry, 2007. Maple package (available at `www.lifl.fr/~urguplu`).

[103] W. Selleck, R. Howley, Q. Fang, V. Podolny, M. G. Fried, S. Buratowski, and S. Tan. A histone fold TAF octamer within the yeast TFIID transcriptional coactivator. *Nature Structural Biology*, 8:695–700, 2001.

[104] H. Stephani. *Differential equations*. Cambridge University Press, 1st edition, 1989.

[105] C. F. Sturm. Résolution des équations algébriques. *Bulletin de Férussac*, 11:419–425, 1829.

[106] T. Sturm and A. Weber. Investigating generic methods to solve hopf bifurcation problems in algebraic biology. In K. H. et al., editor, *Proceedings of Algebraic Biology 2008*, number 5147 in LNCS, pages 200–215, Springer Verlag Berlin Heidelberg, 2008.

[107] G. Teschl. *Ordinary differential equations and Dynamical Systems*. Gerald Teschl, Fakultät für Mathematik Nordbergstrae 15 Universität Wien 1090 Wien, Austria, 2000. Version December 3, 2008.

[108] T. Turányi. Reduction of Large Reaction Mechanisms. *New Journal of Chemistry*, 14(11):795–803, 1990.

[109] A. Ürgüplü. Implantation d'une méthode de réduction du nombre de paramètres d'un système d'équations différentielles ordinaires. Master's thesis, Université des Sciences et Technologies de Lille (USTL), Laboratoire d'Informatique Fondamentale de Lille (LIFL) - Équipe Calcul Formel, 2006.

[110] A. Ürgüplü. Réduction d'un système d'équations différentielles ordinaires et son implémentation, October 2007. MajecSTIC'07, Presses universitaires de Caen. Prépublications de l'Université de Caen Basse-Normandie, Fascicule 2.

[111] B. van der Pol and J. van der Mark. The heartbeat considered as a relaxation oscillation, and an electrical model of the heart. *Philosophical Magazine*, 6(38):763 – 775, 1928.

[112] B. van der Waerden. *Algebra*, volume 2. Springer Verlag, 1991.

[113] N. Vora and P. Daoutidis. Nonlinear model reduction of chemical reaction systems. *AIChE (American Institute of Chemical Engineers) Journal*, 47, issue 10:2320 – 2332, April 2001.

[114] S. Walcher. Multi-parameter symmetries of first order ordinary differential equations. *Journal of Lie Theory*, 9(1):249–269, 1999.

[115] D. Wang and B. Xia. Stability analysis of biological systems with real solution classification. In M. Kauers, editor, *Proceedings of the 2005 International Symposium on Symbolic and Algebraic Computation*, pages 354–360, Bejing, China, July 24–27 2005. ACM, ACM press.

[116] J. Wei and J. C. Kuo. A lumping analysis in monomolecular reaction systems; analysis of the exactly lumpable system. *Industrial and Engineering Chemistry Fundamentals*, 8(1), 1969.

[117] P. Winternitz. Symmetries of discrete systems. Lecture presented at the CIMPA Winter School on Discrete Integrable Systems, September 2003.

[118] A. Wittkopf. *Algorithms and implementations for differential elimination*. PhD thesis, Simon Fraser University, Burnaby, BC, Canada, Canada, 2005.

[119] R. Zippel. Probabilistic algorithms for sparse polynomials. In E. W. Ng, editor, *Symbolic and Algebraic Computation, EUROSAM'79, An International Symposiumon Symbolic and Algebraic Computation - EUROSAM*, volume 72 of *Lecture Notes in Computer Science*, Marseille, France, June 1979. Springer.

[120] R. Zippel. An Explicit Separation of Relativised Random Polynomial Time and Relativised Deterministic Polynomial Time, February 1989. Cornell University.

[121] V. P. Zoltan Szallasi, Jörg Stelling, editor. *System Modeling in Cellular Biology: From Concepts to Nuts and Bolts*, chapter 17 - Software Infrastructure for Effective Communication and Reuse of Computational Models, pages 355–378. A Bradford Book, The MIT Press, London, England, 2006. `http://sbml.org`.

Author Index

T

U

V

W

X

Z