Theoretical and Numerical Topics in the Invariant Calculus of Variations

Michele Zadra

School of Mathematics, Statistics and Actuarial Science

University of Kent

This dissertation is submitted for the degree of

Doctor of Philosophy

March 2020

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work, done in collaboration with my supervisor Prof. Elizabeth Mansfield, and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. In particular, the content of Chapter 4 has been published in [56].

Michele Zadra

March 2020

Acknowledgements

First of all I would like to thank my supervisor, Prof. Elizabeth Mansfield, for introducing me to this beautiful field of Mathematics, for her continuous support throughout these years, and for transmitting me the passion of conducting mathematical research. Your guidance has been instrumental in the accomplishment of this project.

I would like to acknowledge the EPSRC and the SMSAS department of the University of Kent for the funding that made this project possible through the grant EP/M506540/1.

I am indebted to John Pearson and Kuan Xu for introducing me to Chebfun and kindly assisting me when I had doubts on numerical topics.

I am grateful to Steffen Krusch, who has been the first one to introduce me to the field of Geometric Integration and also gave a great deal of help during the final stages of this project.

I am obliged to Claire Carter for the support throughout this project, and for the effort she always puts in order to create a good atmosphere in the Department. I am also thankful to Derek Baldwin for solving many IT issues, always with a very appreciated light–hearted attitude.

Furthermore, I want to thank my colleagues, officemates, housemates and, most of all, friends, Alan, Aniketh, Christos, Dimitris, Josè, Larry, and Nikitas. You have taught me so many things in these years, that I will always be indebted to you.

My family has always been supporting me during this time abroad. To my parents, Loredana and Camillo, and my brothers, Luca and Matteo, I want to express my gratitude.

The physical distance that has separated us was hidden by your visits, calls and all the times we were thinking of each other.

Finally, let me say how much I feel lucky to have met you, Dácil. Your unconditional support, and love, have given me a purpose even in the most difficult times.

Table of contents

1. Introduction

In the mathematical field of the Calculus of Variations, one of the most important results is 1918 Emmy Noether's paper "Invariante Variationsprobleme", [38]. The article contains two theorems that are commonly known in the community as "first" and "second" Noether's theorems. Both statements focus on the consequences of the presence of symmetries in a variational problem. The first theorem is about what Noether and her contemporaries used to call "finite continuous groups", nowadays known as Lie groups. In layman's terms, the theorem states that if the Lagrangian that defines the variational problem is invariant under a n–dimensional Lie group, then there will be n quantities that will be preserved by the solution to the variational problem. Important examples come from classical mechanics, where it is often the case that a solution (a trajectory of a rigid body for instance) is sought such that it conserves quantities like the total energy, the angular momentum or the momentum itself. The second theorem considers the case of "infinite continuous groups", which is when the Lagrangian invariance is expressed with respect to some set of functions rather than parameters. In this text we will make extensive use of the conservation laws arising from a Lie group invariance. We will use them both to derive the solutions of the variational problem and as checks on qualitative features of numerical solutions. In that sense, our focus will be on the first of Noether's theorems, rather than on the second.

Theorem 1.0.1 (Noether's First Theorem, [38])**.** *Suppose that the one–dimensional Lagrangian*

$$\mathscr{L} = \int L(x, \mathbf{u}, \mathbf{u}_x, ..)\, dx \tag{1.1}$$

is invariant under the action of a one–parameter Lie group. Then, there is a first integral, i.e. a quantity that is conserved by a minimiser of (1.1)*.*

The main tool we will be using to study Lie group actions on Lagrangians is what is called a *moving frame*. This is a generalisation of the theory started by Darboux and Cartan, and developed in the context of differential geometry. The theory of moving frames that grew from the studies of Fels and Olver, [19; 20], has encountered many applications in different areas from the study of equivalence problems to the Calculus of Variations, [50]. The latter is the application we will be mostly interested in this research. The name "Invariant Calculus of Variations", first introduced in [42], comes from the fact that it makes use of a moving frame in order to express the key objects needed to study variational problems in terms of the generating differential invariants of the Lie group action. Also the conservation laws can be expressed in a more insightful way, as a linear action of the Adjoint representation of the frame on an invariant vector, [22; 23; 24; 42]. Deriving the Euler–Lagrange equations and conservation laws via moving frames, rather than in the original variables, can sometimes be the difference between a problem that is computationally tractable and one that is not. So far, moving frames have been used to completely solve one dimensional variational problems that are invariant under actions of $SL(2)$ or $SE(2)$ and $SE(3)$, [22; 23; 24; 42]. In this text we consider first the case of a linear action of the Lie group $SU(2)$, a group that is relevant in the field of quantum mechanics as a way to model particles that have non–integer spin. We will characterise the algebra of differential invariants of the action and use this to compute the Euler–Lagrange equations and the conservation laws. Finally, our contribution in this chapter is to show how the computations needed to solve conservation laws (expressed in

terms of the Adjoint representation of the moving frame and a vector of invariants) can be simplified making extensive use of the geometrical setting of the problem.

The study of the conservation laws arising from variational systems has been gaining a lot of interest also among the numerical analysis community in the past thirty years, with the birth of the field of Geometric Integration, [25]. Starting from the second half of 1900, numerical analysts have been more and more involved in the investigation of qualitative features of the numerical solutions. This led to a more geometric approach in the study of the properties of the approximated solution, compared to the ones of the exact flow. As an example, consider the following case, inspired by [25]. The Hamiltonian formulation of the small–angle approximation to the classical harmonic oscillator can be described by the following initial value problem

$$\begin{cases} \frac{d}{dt}q = v \\ \frac{d}{dt}v = -q \\ q(t_0) = q_0 \\ v(t_0) = v_0 \end{cases} \tag{1.2}$$

where q and p represent the position of the pendulum and its momentum respectively. If

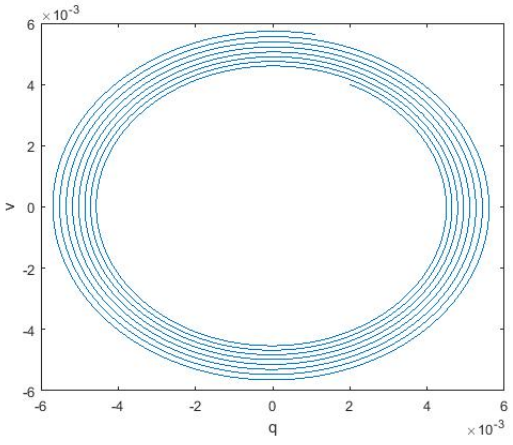


Fig. 1.1 The orbit in the phase using an explicit Euler method with a step size $\Delta t = 10^{-2}$ is not a closed curve: the total energy is not conserved

we try to approximate system (1.2) with an explicit Euler method, we would obtain that the total energy of the system is not preserved. The fact that the trajectories in the phase space are closed curves is equivalent to the total energy of the system being conserved. This is the behaviour we would expect in the absence of dissipative forces. However, with an explicit Euler method, the trajectories in the phase space are not closed curves, as we can see clearly in Figure 1.1. Namely, the trajectory is spiralling outwards, meaning that the energy of the pendulum is increasing, even in the absence of external forces. It is clear that a pendulum that swings faster and faster is not a good approximations to the physical system we started with. If we instead apply a semi–implicit Euler method to the same equations, then we will have that the trajectories are closed curves and the total energy is preserved, see Figure 1.2.

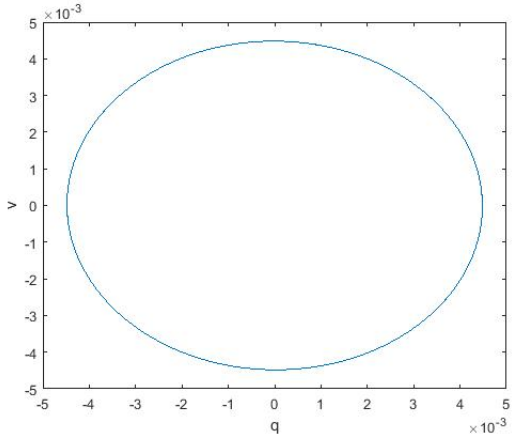


Fig. 1.2 The orbit in the phase using a semi–implicit Euler scheme with a step size $\Delta t = 10^{-2}$ is a closed curve: the total energy is conserved

One of the aspects that has been studied in the geometric integration setting, is to better understand the way numerical schemes for ODEs had been devised. An example of this is to study the Butcher theory for the Runge–Kutta coefficients, [5], from an algebraic perspective, [26]. It was noted that the "classical" integrators were built to solve equations evolving in $\mathbb{R}^n$. However, many important ODEs evolve on Lie groups, or even on generic smooth manifolds. The difference between working in $\mathbb{R}^n$ or on a manifold, is that $\mathbb{R}^n$ is a linear space, while a smooth manifold, in general, is not. This means that in general, we may not be

able to do vector operations with vectors belonging to tangent spaces at two different points in a manifold. For this reason, the "classical" numerical schemes struggle in retaining the geometrical features of the solution when applied to manifolds that are not $\mathbb{R}^n$. Therefore, even if the initial datum of an initial value problem is taken to be on a specific manifold, the solution might evolve outside the starting domain. Important results about the limits of the classical Runge–Kutta schemes in preserving non–linear structures can be found in [6; 13; 33; 34]. In order to solve this issue, the work of Crouch, [14] and Lewis [39] has been studied and taken as inspiration for the next steps. Some modifications of the Runge–Kutta with enhanced preservation features were derived, see [8; 52], but they required some non–trivial order conditions on the coefficients. The approach we will focus in this thesis is the one that maps the equation to the tangent space (the associated Lie algebra as our focus is on Lie groups). Being the Lie algebra a vector space, hence linear, an adaptation of the "classical" numerical schemes can now be used to solve the ODE. Once the solution has been found, it is possible to map it to the Lie group smoothly via the exponential map (which would give rise to a Magnus expansion), a composition of exponentials (Fer expansion) or the Cayley map. In this research, we will give a closer look at how is it possible to extend the theory of Lie group integrators based on the Magnus expansion, [40], from the context of ODEs to the one of a class of compatible systems of PDEs. An important motivation for this is in finding the solution of higher order variational problems in the presence of a Lie group invariance. In this setting, it is possible to obtain coupled PDEs for the moving frame in terms of the curvature matrices. Being a moving frame a map from the manifold to the Lie group, these differential equations evolve on a Lie group. Hence, the need for a numerical integrator that preserves the geometrical properties of the solution. Our original contribution in this chapter is to show that the solution to some compatible systems of PDEs evolving on a Lie group can be numerically approximated up to order 5 using the Magnus expansion, [56]. Aside from the Magnus expansion, another successful approach to differential equations

evolving on Lie group is the one through Fer expansion. Broadly speaking, the difference between this and the Magnus expansion is that the former "corrects" the solution in the Lie group, while the latter does it in the Lie algebra. This topic will not be discussed in this work, but the interested reader can find some of the fundamental results related to this approach in [1; 32; 57; 58; 59].

The last topic we will discuss in this thesis is related to discrete variational problems that are invariant under a Lie group action. As we have already mentioned above, the presence of a symmetry can greatly simplify the process of finding a solution to the variational problem. Moreover, it is guaranteed the existence of a set of conservation laws that can be used either to compute the solution or as a solid check when devising a numerical scheme to solve the problem at hand. One issue that could arise is how the symmetry is affected by the process of discretising a smooth variational problem. It is not guaranteed in fact that starting from a smooth variational problem that is invariant under a Lie group action, any discretisation of it would lead to a discrete variational problem invariant under the same action. The discretisation process can alter the geometrical features of the solution to the variational problem, generating an outcome that is not consistent with the initial smooth problem anymore. In order to overcome this problem, lattice–based multispaces, [44], have been ideated as a setting where it is possible to continuously pass from a smooth problem to a discrete one, retaining all the most–needed geometrical features of the smooth problem. Research has been done on discrete variational problems with symmetry, [31; 43], and we extend that focusing on how to discretise generating differential invariants and their derivatives up to any order of accuracy and on how to compute solutions to some variational problem in a more efficient way. Our contributions in this chapter are a formula for the infinitesimal action on the multispace approximation of the directional derivative, a way to construct discrete approximations of any order to smooth generating differential invariants and their derivatives, the derivation of the Euler–Lagrange equations for higher order multispace

variational problems and the introduction of a constraint in the $SE(2)$–invariant multispace variational problems.

In Chapter 2 we will give an introduction to the theory of moving frames and their application to the Calculus of Variations. Definitions, examples and main results from both fields will be given, with special focus on how a moving frame can be used to study the generating differential invariants of a prolonged Lie group action and how this makes for a simplified treatment of the Euler–Lagrange equations and conservation laws.

The class of smooth one– and two–dimensional variational problems that are invariant with respect to a linear action of $SU(2)$ is the subject of Chapter 3. We will apply the theory developed in the first chapter and compute the Euler–Lagrange and conservation laws up to dimension 2. We will show how the conservation laws can be used, in the one–dimensional case, in order to recover the solution of the variational problems only through quadratures. Finally we will introduce the issues that arise in the two–dimensional (and in general n–dimensional) case when trying to integrate and solve the conservation laws.

These issues will be addressed in Chapter 4, where we present an exposition of the theory of the Lie group integrators based on the Magnus expansion. We will show how it is possible to extend these methods to a class of two–dimensional (and in general n–dimensional) compatible systems of PDEs arising in the theory of moving frames. The key to this extension is to prove that these compatible systems of PDEs can be solved taking different paths of integration. The proof of this result is given up to order 5 in the discretisation variables, but it is conjectured that it actually holds up to any order. An important application of this is to the solution of n–dimensional invariant variational problems.

Chapter 5 will treat the case of multispace variational problems. After a brief introduction to the subject, variational problems that are invariant under the affine action of SE(2) are considered. This case is also used as a running example in the subsequent section on a way to use curvature matrices to discretise the generating differential invariants of a Lie group

action. Higher order and constrained variational problems are then considered. The chapter ends with some numerical examples.

Finally, in Chapter 6, we will provide a summary of the main topics that have been studied and reflect on how the ideas presented in this work could be progressed in the future.

2. Preliminaries

2.1 Introduction

The case where the Lagrangian is invariant under a Lie group action, it is one of great importance in the study of variational problems. There are at least two good reasons why this is the case. The first one is that, as a consequence of Noether's first theorem, the presence of a Lie group invariance is related to the existence of a set of conservation laws. The latter are quantities that are preserved on solutions of the Euler–Lagrange equations. The importance of conservation laws cannot be overstated: they can be used, as we will see later on in the following chapters, to solve for the Lagrangian minimisers. From a numerical point of view, conservation laws are also crucial in providing quantitative and qualitative checks on the approximated solution, as they usually correspond to some relevant features of physical systems.

Another reason why Lie group symmetries are so important in the field of Calculus of Variations, is that they allow to rewrite the variational problem in terms of the invariants of the action. This has a major impact in the way the minimisers are found: the new variables often simplify the computations and make problems that used to be not tractable in the original variables, solvable.

In this chapter we introduce the theory of moving frames and its applications to the Calculus of Variations. These two disciplines combined form what is called the Invariant Calculus of Variations, [42]. We will begin presenting the basic theory of Lie group actions

and moving frames. We continue the expositions providing definitions and examples of differential invariants, curvature matrices and the adjoint representation. The chapter ends with some important results regarding the application of moving frames to variational problems. Comprehensive sources on moving frames and symmetries, and their applications, are [19; 20; 42; 50; 51].

2.2 Lie group actions and moving frames

This section is devoted to present the basics of Lie group actions on smooth manifolds and moving frames.

2.2.1 Smooth manifolds and Lie group actions

We will start giving the main definitions regarding Lie group actions on smooth manifolds. These concepts will be useful later when introducing moving frames. For more details regarding smooth manifolds and matrix Lie groups, see [28; 29].

Definition 2.2.1 ([29])**.** A manifold M of dimension n is a topological space that is locally homeomorphic to $\mathbb{R}^n$.

The naive idea behind a manifold is that of a space that can be more "complicated" in some sense than $\mathbb{R}^n$, but locally we could treat it as we were in $\mathbb{R}^n$, at least from a topological point of view. Well–known examples of manifolds are the n-dimensional sphere, the torus and of course $\mathbb{R}^n$. As we will see later on, a Lie group is also a manifold. Definition 2.2.1 implies that there is an open cover $\{\mathscr{U}_i\}_i$ of M such that for every $\mathscr{U}_i$ there is an homeomorphism ϕ_i to an open set in $\mathbb{R}^n$. The couple $(\mathscr{U}_i, \phi_i)$ is called a *chart* and the set of all charts $\alpha = \{(\mathscr{U}_i), \phi_i\}_i$ is an *atlas*. Consider two charts, $(\mathscr{U}_i, \phi_i), (\mathscr{U}_j, \phi_j)$ and their coordinate change map given by

$$\psi_{ij} = \phi_i \phi_j^{-1} : \phi_j\left(\mathscr{U}_i \cap \mathscr{U}_j\right) \to \phi_i\left(\mathscr{U}_i \cap \mathscr{U}_j\right).$$

Note that the coordinate change map is defined between two open sets in $\mathbb{R}^n$, so it does make sense to study its differentiability. Suppose that all the ψ_{ij}, for every i, j in the atlas α are of class C^r for some $r \geq 1$. Then the couple (M, α) is said to be a *smooth* manifold. In the following we will be referring to a smooth manifold only with the name of the topological space, M in this case. As we mentioned above, a notable example of smooth manifold is given by a Lie group.

Definition 2.2.2. A group G is a set equipped with an operation $\cdot : G \times G \to G$, such that the following axioms are satisfied:

G1 $\forall a, b, c \in G$, $(a \cdot b) \cdot c = a \cdot (b \cdot c)$,

G2 $\forall a, b \in G$, $a \cdot b \in G$,

G3 $\exists e \in G$ s.t. $\forall a \in G$, $a \cdot e = e \cdot a = a$. The element e is called the *identity element*,

G4 $\forall a \in G \; \exists b \in G$ s.t. $a \cdot b = b \cdot a = e$. The element b is called *the inverse of a* and is denoted as a^{-1}.

Throughout this text we will be dealing with a special kind of groups that can be represented as matrices with coefficients in $\mathbb{C}$. The most general group of this type is given by $GL(n, \mathbb{C})$.

Lemma 2.2.3. *The set of all $n \times n$ complex invertible matrices equipped with the matrix product operation is a group. This group is denoted as $GL(n, \mathbb{C})$ and is called the general linear group over the complex numbers.*

All the groups that we will look at hereafter are subgroups of $GL(n, \mathbb{C})$. As our main application will regard the Calculus of Variations, more structure on the group is needed.

Definition 2.2.4. A Lie group is a smooth manifold M and a group, such that both the product $\cdot : M \times M \to M$ and the inverse map $a \mapsto a^{-1}$ are smooth.

Lie groups take advantage of the smooth manifold structure in order to allow operations as differentiation and integration, whereas these are not well–defined in a general group.

Example 2.2.5. Here are some examples of Lie groups that we will be studying in the next chapters. They are all subgroups of $GL(n,\mathbb{C})$ for some n.

- The *special linear group* is a (n^2-1)–dimensional Lie group given by

$$SL(n) = \{A \in GL(n,\mathbb{R}) : \det(A) = 1\}.$$

A particularly interesting example is given by

$$SL(2) = \left\{ A \in GL(2,\mathbb{R}) : A = \begin{pmatrix} a & b \\ c & \frac{1+bc}{a} \end{pmatrix}, a,b,c \in \mathbb{R}, a \neq 0, \det(A) = 1 \right\}.$$

If we denote by $g(a,b,c) = \begin{pmatrix} a & b \\ c & \frac{1+bc}{a} \end{pmatrix}$, consider two elements $g1 = g(a_1,b_1,c_1)$ and $g_2 = g(a_2,b_2,c_2)$; the group product is the matrix product and, in terms of the group parameters can be expressed as

$$g_1 \cdot g_2 = g\left(a_1a_2 + b_1c_2, \frac{(a_1a_2+b_1c_2)b_2+b_1}{a_2}, \frac{a_1a_2c_1+b_1c_1c_2+c_2}{a_1}\right).$$

The inverse element is found via the matrix inverse, that is guaranteed to exist as the determinant is always equal to 1. The inverse function in terms of the group parameters is

$$g(a,b,c)^{-1} = g\left(\frac{1+bc}{a}, -b, -c\right).$$

The identity element is $e = g(1,0,0)$.

- The *special orthogonal group* is a $\frac{n(n-1)}{2}$–dimensional Lie group defined as

$$SO(n) = \left\{A \in GL(n,\mathbb{R}) : AA^T = A^T A = I\,, \det(A) = 1\right\}.$$

The group operation is the matrix multiplication and the group inverse is the matrix inverse (or transpose in this case).

The case where $n = 2$ can be represented as

$$SO(2) = \left\{ A \in GL(2,\mathbb{R}) : A = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}, \theta \in \mathbb{R} \right\}.$$

In terms of the group parameter, the group product is, using a notation analogous to the example above,

$$g(\theta_1) \cdot g(\theta_2) = g(\theta_1 + \theta_2).$$

The inverse map is given by

$$g(\theta)^{-1} = g(-\theta).$$

The Lie group $SO(2)$ is important in geometry as it is the group of rotations in $\mathbb{R}^2$.

- The Lie group $U(n)$ is defined as

$$U(n) = \left\{A \in GL(n,\mathbb{C})\,, AA^* = A^* A = I\right\},$$

where A^* is the conjugate transpose of A. Consider the case where $n = 2$. An element $g \in U(2)$ can be represented as

$$g = \begin{pmatrix} \alpha & \beta \\ -\bar{\beta} & \bar{\alpha} \end{pmatrix},$$

where $\alpha, \beta \in \mathbb{C}$. The group multiplication in terms of the parameters is

$$g(\alpha_1, \beta_1) \cdot g(\alpha_2, \beta_2) = g(\alpha_1\alpha_2 - \beta_1\bar{\beta}_2, \alpha_1\beta_2 + \beta_1\bar{\alpha}_2)$$

and the inverse element is $g(\alpha, \beta)^{-1} = (|\alpha|^2 + |\beta|^2)^{-1} g(\bar{\alpha}, -\beta)$.

- $\mathbb{R}^n$ with the addition as the group operation, is a simple yet important example of Lie group

It is also possible to combine Lie groups in order to create new ones. A fundamental example that lies at the basis of the Euclidean geometry is the *special Euclidean group*. This is defined in terms of the special orthogonal group and $\mathbb{R}^n$, namely

$$SE(n) = SO(n) \ltimes \mathbb{R}^n = \left\{ A \in GL(n+1, \mathbb{R}) : \begin{pmatrix} R_\theta & v \\ 0 & 1 \end{pmatrix}, R_\theta \in SO(n), v \in \mathbb{R}^n \right\}.$$

where $\ltimes$ stands for the semi–direct product. When $n = 2$, the group operation can be written in terms of the group parameters as

$$g(\theta_1, a_1, b_1) \cdot g(\theta_2, a_2, b_2) = g(\theta_1 + \theta_2, \cos(\theta_1)a_2 - \sin(\theta_1)b_2 + a_1, \sin(\theta_1)a_2 + \cos(\theta_1)b_2 + b_1)$$

and the inverse mapping is given by

$$g(\theta, a, b)^{-1} = g(-\theta, -\sin(\theta_1)b_1 - a_1\cos(\theta_1), -\cos(\theta_1)b_1 + a_1\sin(\theta_1)).$$

Once we have group, we can define what is called an *action* on another set. This is a very general construction, even though we will usually assume that we have a manifold rather than a set. There are left and right actions and although the concepts are very similar there is a difference in how products of group elements are treated.

Definition 2.2.6. Given a group G, a set X and a function defined as

$$\Phi : G \times X \to X$$

$$(g,x) \mapsto \Phi(g,x),$$

that satisfies the following property:

$$\Phi(e,x) = x \quad \text{where } e \text{ is the identity element of the group.}$$

Φ is called a *left action* if

$$\Phi(g,\Phi(h,x)) = g \cdot (h \cdot x) = (gh) \cdot x = \Phi(gh,x) \quad \forall g,h \in G.$$

Φ is called a *right action* if

$$\Phi(g,\Phi(hx)) = g \star (h \star x) = (hg) \star x = \Phi(hg,x) \quad \forall g,h \in G.$$

We will denote left actions with $\cdot$ and right actions with $\star$. Even though the theory is presented below is mostly given for left actions, it can easily be extended to right actions.

An important subset of X is the one that contains the elements that are left identical by the group action.

Definition 2.2.7. An invariant of a group action is an element $x \in X$ such that $g \cdot x = x$ for any $g \in G$.

The assumptions made in Definition 2.2.6 are very general, so we want to restrict our study to some types of actions that have some "nice" properties. The first of these features is defined in terms of the *isotropy group*.

Definition 2.2.8. Given $x \in X$ we define the *isotropy group of* x as

$$G_x = \{g \in G | g \cdot x = x\}.$$

The first property we require on the group action is to be *free*.

Definition 2.2.9. A group action is said to be *free* if the isotropy group for every element in X is the trivial one, i.e.

$$G_x = \{e\} \qquad \forall x \in X.$$

For the second property we need to define a tool which is useful to study the images of the set elements under the group action.

Definition 2.2.10. The *orbit of* $x \in X$ under the action of a group G is the set

$$\mathscr{O}(z) := \{g \cdot z \,|\, g \in G\}$$

Now we can pass to define the second and last property of the group actions we are going to study.

Definition 2.2.11. A group action is said to be *regular* if the following two conditions are satisfied:

1. all orbits have the same dimensions,

2. for each $x \in X$ there is an arbitrarily small neighbourhood $\mathscr{U}(x)$ such that for all $x' \in \mathscr{U}(x)$, $\mathscr{U}(x) \cap \mathscr{O}(x')$ is connected.

Example 2.2.12. Consider the Lie group $SO(2)$ acting on column vectors in $\mathbb{R}^2$ as

$$SO(2)\times\mathbb{R}^2\to\mathbb{R}^2$$

$$(g,v)\mapsto gv.$$

This is a left action: take $g,h\in SO(2)$, $z=(x,y)^T\in\mathbb{R}^2$ then it is easy to show that

$$\begin{aligned} g\cdot(h\cdot z) &= \begin{pmatrix}\cos(\theta_1) & -\sin(\theta_1)\\ \sin(\theta_1) & \cos(\theta_1)\end{pmatrix}\cdot\left(\begin{pmatrix}\cos(\theta_2) & -\sin(\theta_2)\\ \sin(\theta_2) & \cos(\theta_2)\end{pmatrix}\cdot\begin{pmatrix}x\\ y\end{pmatrix}\right)\\ &= \left(\begin{pmatrix}\cos(\theta_1) & -\sin(\theta_1)\\ \sin(\theta_1) & \cos(\theta_1)\end{pmatrix}\begin{pmatrix}\cos(\theta_2) & -\sin(\theta_2)\\ \sin(\theta_2) & \cos(\theta_2)\end{pmatrix}\right)\cdot\begin{pmatrix}x\\ y\end{pmatrix}\\ &= (gh)\cdot z. \end{aligned}$$

The action is not free all over the domain as it can be clearly seen that $z=(0,0)$ is a fixed point, hence $G_{(0,0)}=G\neq\{e\}$.

Even if an action is not free in a particular domain, it is possible to modify the space on which the Lie group is acting and make the action free in that space. For this reason we introduce the *jet space* (or *jet bundle*). The idea that lies behind the jet space is to have a space where a fixed number of derivatives of the dependent variables are well–defined and can be treated as dependent variables themselves. Consider an open domain $X\subset\mathbb{R}^p$ of independent variables with coordinates $\mathbf{x}=(x_1,..,x_p)$ and a domain $U\subset\mathbb{R}^q$ of dependent variables with coordinates $\mathbf{u}=(u^1,..,u^q)$. Given a vector $K=(k_1,..,k_p)\in\mathbb{N}^p$, define $|K|=\sum_i k_i$. The notation we will use for the derivatives of the variables in U with respect to the variables in X is

$$u^\alpha_K=\frac{\partial^{|K|}u^\alpha}{\partial^{k_1}x_1\cdots\partial^{k_p}x_p}.$$

Definition 2.2.13. Assume the u^α, $\alpha = 1,..,q$, can be continuously differentiated n times, $n \in \mathbb{N}$, with respect to $x_1,..,x_p$. The n–th jet space, denoted as $\mathscr{J}^n(X \times U)$, is the smooth manifold whose points have coordinates

$$\mathbf{z} = (x_1,..,x_p,u^1,..,u^q,\dots u^1_K,..) \quad \text{where } |K| \leq n.$$

For more details and geometrical features of the jet set, the reader is referred to [29]. The jet set is the right space where variational problems can be defined and studied.

A Lie group action on $M = X \times U$ can be prolonged to $\mathscr{J}^n(X \times U)$ in the following way. The operator $\frac{\partial}{\partial x_i}$ extends to the *total differentiation operator* D_i acting on the algebra of the smooth functions on $\mathscr{J}^n(X \times U)$ as

$$D_i = \frac{\partial}{\partial x_i} + \sum_{\alpha=1}^{q} \sum_K u^\alpha_{Ki} \frac{\partial}{u^\alpha_K}.$$

Consider a group action acting smoothly on the independent variables as $g \cdot \mathbf{x} = \tilde{\mathbf{x}}$ and its Jacobian matrix

$$D\tilde{\mathbf{x}} = \begin{pmatrix} \frac{\partial \tilde{x}_1}{\partial x_1} & \cdots & \frac{\partial \tilde{x}_1}{\partial x_p} \\ \vdots & \ddots & \vdots \\ \frac{\partial \tilde{x}_p}{\partial x_1} & \cdots & \frac{\partial \tilde{x}_p}{\partial x_p} \end{pmatrix}.$$

It is possible to extend the total derivative operator to the transformed variables as

$$\widetilde{D}_i = \frac{D}{D\tilde{x}_i} = \sum_k (D\tilde{x})^{-1}_{ik} D_k. \tag{2.1}$$

Definition 2.2.14. The prolonged action of G on $\mathscr{J}^n(X \times U)$ is given by

$$g \cdot u^\alpha_K = \widetilde{u^\alpha_K} = \widetilde{D}^{k_1}_1 \cdots \widetilde{D}^{k_p}_p (g \cdot u^\alpha). \tag{2.2}$$

Example 2.2.15. Consider the action of $SL(2)$ on curves in the plane parametrised as $(x,u(x))$, given by

$$\tilde{x} = \frac{(ax+b)a}{1+(ax+b)c} \quad \tilde{u} = \frac{ua^2}{(acx+bc+1)^2}. \tag{2.3}$$

This action has been used in the context of the Invariant Calculus of Variations in [23]. Using (2.1)–(2.2), which in this case amounts to use the chain rule, we have

$$\widetilde{D}_x = \left(\frac{D\tilde{x}}{Dx}\right)^{-1} D_x = \frac{(acx+bc+1)^2}{a^2} D_x.$$

Hence the prolongation of action (2.3) to u_x is

$$\widetilde{u_x} = \widetilde{D}_x\tilde{u} = \left(\frac{D\tilde{x}}{Dx}\right)^{-1} D_x\tilde{u} = \frac{(1+(ax+b)c)u_x - 2uac}{1+(ax+b)c}.$$

The invariants of a prolonged action are called *differential invariants*. They form an algebra and we will see in the following how to find a set of generators. The prolongation of a Lie group action is an important tool that can be used to produce free and regular extensions starting from non–free or non–regular actions.

2.2.2 Moving frames

A *moving frame* is defined as follows.

Definition 2.2.16 ([42, p. 116])**.** Given a group G acting on a smooth manifold M, a moving frame is an equivariant map $\rho : M \to G$, namely if the group acts on the left, then the moving frame is right equivariant, i.e

$$\rho(g\cdot z) = \rho(z)g^{-1},$$

while for a right action, the equivariance of the frame is expressed as

$$\rho(g\star z) = g^{-1}\rho(z).$$

.

The reason why we want the group action to be locally free and regular is because this is a prerequisite for having the existence and uniqueness of a moving frame.

Theorem 2.2.17 ([42], p.117)**.** *If a group action is free and regular in* $\Omega \in M$*, then for every* $x \in \Omega$ *there is a neighbourhood* U *of* x *such that there exists a moving frame on* U*.*

For a discussion about the converse of Theorem 2.2.17, see [42] Section 4.2. The construction of a moving frame begins with a group action of a group G on a smooth manifold M. Then, choose an element $z \in M$ and a neighbourhood of it $\mathscr{U}(z) \subset M$. This construction produces in fact a local, and not global, map from the manifold to the group.

Definition 2.2.18. A *cross section* $\mathscr{K}$, which is a surface on M, that crosses the orbits transversally.

Remark 2.2.19. In the following, all the generic Lie groups we will consider will have dimension n, unless differently stated.

Suppose the cross section is given by the system of equations

$$f_1(z) = 0, \cdots, f_n(z) = 0.$$

The cross section is arbitrary, and there is no prescribed way to choose the functions $f_i, i = 1..n$. However, depending on the action at hand, there are some choices that can make computations easier. Finally the moving frame for the cross section $\mathscr{K}$ is the group element that solves the following set of equations, called *normalisation equations*:

$$f_1(g \cdot z) = 0, \cdots, f_n(g \cdot z) = 0.$$

We will denote this solution as $\rho(z)$. It is a result that ρ is consistent with Definition 2.2.16, meaning that it is guaranteed to be equivariant. This follows from the Implicit Function Theorem, [42].

Remark 2.2.20. A moving frame is determined by the choice of the cross section, hence from the normalisation equations

Example 2.2.21. Consider the action of $SE(2)$ on curves $z = (x, u(x))$ in the real plane, defined as

$$g \cdot z = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} x \\ u \end{pmatrix} + \begin{pmatrix} a \\ b \end{pmatrix}. \tag{2.4}$$

Action (2.4) is not free and regular in general, but we can make it so if we consider the prolongation given by

$$\tilde{u}_{\tilde{x}} = \left(\frac{D\tilde{x}}{Dx}\right)^{-1} D_x \tilde{u} = \frac{1}{\cos\theta - \sin(\theta)u_x} \left(\sin(\theta) + \cos(\theta)u_x\right).$$

Define a cross section $\mathscr{K}$ as the solution set of

$$x = 0, \qquad u = 0, \qquad u_x = 0.$$

Then a moving frame is the group element that satisfies

$$\begin{cases} \tilde{x} = \cos(\theta)x - \sin(\theta)u + a = 0 \\ \tilde{u} = \sin(\theta)x + \cos(\theta)u + b = 0 \\ \widetilde{u}_x = \frac{1}{\cos\theta - \sin(\theta)u_x} \left(\sin(\theta) + \cos(\theta)u_x\right) = 0. \end{cases}$$

In terms of the group parameters, the moving frame can be expressed as

$$\begin{pmatrix} a \\ b \end{pmatrix} = -R_\theta \begin{pmatrix} x \\ u \end{pmatrix}, \qquad \theta = -\arctan(u_x),$$

where we denoted with R_θ the rotation matrix $R_\theta = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$. Hence the moving frame in the standard representation is given as

$$\begin{pmatrix} R_\theta & -R_\theta \begin{pmatrix} x \\ u \end{pmatrix} \\ 0 & 1 \end{pmatrix}.$$

Example 2.2.22. Consider the action of $U(2)$ on pair of complex curves given by

$$g \cdot \begin{pmatrix} u(t) \\ v(t) \end{pmatrix} = \begin{pmatrix} \alpha & \beta \\ -\bar{\beta} & \bar{\alpha} \end{pmatrix} \begin{pmatrix} u(t) \\ v(t) \end{pmatrix} = \begin{pmatrix} \alpha u + \beta v \\ -\bar{\beta} u + \bar{\alpha} v \end{pmatrix},$$

where the independent variable t is left invariant, i.e. $\tilde{t} = t$. This action is free and regular in a neighbourhood of $(u, v) = (1, 0)$. Take the cross section given by

$$\begin{cases} u = 1 \\ v = 0 \end{cases}$$

Then the moving frame is the element g s.t.

$$\begin{cases} g \cdot u = 1 \\ g \cdot v = 0. \end{cases}$$

As an element of $U(2)$, the moving frame is

$$\rho = \begin{pmatrix} \bar{u} & \bar{v} \\ -v & u \end{pmatrix}$$

.

Moving frames are an important tool to study invariants and differential invariants of the Lie group action at hand as we will see in the following theorem.

Theorem 2.2.23 ([19])**.** *If $\rho(z)$ is a moving frame, then $\rho(z)\cdot z$ is invariant.*

Proof. Consider a left action and a right frame. Then

$$\rho(g\cdot z)\cdot(g\cdot z) = \rho(z)g^{-1}\cdot(g\cdot z) = (\rho(z)g^{-1}g)\cdot z = \rho(z)\cdot z.$$

The proof for a right action or left-invariant moving frame is analogous. □

Theorem 2.2.23 allows for the introduction of new coordinates in the jet bundle.

Definition 2.2.24. In the following, we will use the following notation for the *invariantised* jet bundle coordinates

$$I(x_i) = \rho(z)\cdot x_i = \tilde{x}_{i_{|g=\rho(z)}}, \qquad I(u^\alpha_K) = \rho(z)\cdot u^\alpha_K = I^\alpha_K.$$

By replacing the original variables with the invariantised ones, it is possible to match historically known invariants.

Theorem 2.2.25 (Replacement Theorem, [20])**.** *If $f(z)$ is invariant with respect to a Lie group action, then $f(z) = f(I(z))$*

We end this subsection reporting the definition of infinitesimal of a Lie group action.

Definition 2.2.26. Given a Lie group whose parameters, in a neighbourhood of the identity element, are $a_1,..,a_n$, then the *group infinitesimals* with respect to the parameters are

$$\xi_i^j = \left.\frac{\partial \widetilde{x_j}}{\partial a_i}\right|_{g=e}, \qquad \phi_i^\alpha = \left.\frac{\partial \widetilde{u^\alpha}}{\partial a_i}\right|_{g=e}, \qquad \phi_{K,i}^\alpha = \left.\frac{\partial \widetilde{u_K^\alpha}}{\partial a_i}\right|_{g=e}. \tag{2.5}$$

The matrix of infinitesimals is defined as

$$\Omega = \begin{pmatrix} \xi_1^1 & \cdots & \xi_1^p & \phi_1^1 & \cdots \\ \vdots & \ddots & \ddots & \vdots & \\ \xi_n^1 & \cdots & \xi_n^p & \phi_n^1 & \cdots \end{pmatrix},$$

where the set of infinitesimal to be included in the matrix is defined by the problem at hand. Given a moving frame for the action we are considering, the invariantised matrix of infinitesimals is defined as $\Omega(I) = \rho \cdot \Omega$.

Example 2.2.27. Consider the action of $SL(2,\mathbb{C})$ on pair of complex curves $(x(s),u(s))$ such that $\tilde{s} = s$ and

$$\tilde{x} = \frac{(ax+b)a}{acx+bc+1}, \qquad \tilde{u} = \frac{(acx+bc+1)(6ac+u(acx+bc+1))}{a^2}.$$

An important application of this and other $SL(2)$ actions can be found in [11].The matrix of infinitesimals for this action is given by

$$\Omega = \begin{array}{c} \\ a \\ b \\ c \end{array} \begin{array}{c} \begin{array}{ccccc} s & x & u & x_s & u_s \end{array} \\ \begin{pmatrix} 0 & 2x & -2u & 2x_s & -2u_s \\ 0 & 1 & 0 & 0 & 0 \\ 0 & -x^2 & 6+2ux & -2xx_s & 2(u_s x+ux_s) \end{pmatrix} \end{array}.$$

Define a frame for the action using the following normalisation equations

$$\tilde{x} = 0, \quad \tilde{x}_s = 1, \quad \tilde{u} = 0.$$

The invariantised matrix of infinitesimals is given evaluating Ω on the frame, namely

$$\Omega(I) = \rho \cdot \Omega = \begin{array}{c} \\ a \\ b \\ c \end{array} \begin{array}{c} \begin{array}{ccccc} s & x & u & x_s & u_s \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 2 & -2I_1^u \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 6 & 0 & 0 \end{pmatrix} \end{array}.$$

2.2.3 Invariant differentiation

In this subsection it is shown that the invariantisation operator does not commute with the differentiation one. This will generate what is called *syzygies*, i.e. differential relations between invariants.

We have seen in a previous section how the total differentiation operator D_i extends to the transformed variables as $\tilde{D}_i = \frac{D}{D(g \cdot x_i)}$. Since the image of a moving frame is an element of the group, it is possible to choose $g = \rho$ in the definition of $\tilde{D}_i$.

Definition 2.2.28 ([42])**.** The *invariant differential operator* is defined as

$$\mathscr{D}_i = \tilde{D}_i\big|_{g=\rho(z)} = \left.\frac{D}{D(g \cdot x_i)}\right|_{g=\rho(z)}.$$

Invariantised differential operators allow to compute derivatives of invariants in the following way. Although it is trivial to differentiate the original variables in the jet space, e.g.

$$\frac{\partial}{\partial x_i} u_K^\alpha = u_{K+\mathbf{e}_i}^\alpha,$$

it is not so straightforward to differentiate invariantised points on the manifold. It can be proved, [42], that

$$\mathscr{D}_i I_K^\alpha = I_{K+\mathbf{e}_i}^\alpha + M_{K+\mathbf{e}_i}^\alpha, \tag{2.6}$$

where M_{Ki}^α is called *correction term*. Equation (2.6) is the reason why we stated above that invariantisation and differentiation do not commute. Details on how the correction terms are computed can be found in [42]. One of the main results that is worth highlighting, is that the correction terms can be computed with the knowledge of only the matrix of infinitesimal for the action and the normalisation equations used to define the frame. In other words, it is not needed to solve explicitly for the frame in order to obtain M_{Ki}^α. In order to see this, consider the left hand side of the n normalisation equations used to define the moving frame, $f_1(z) = 0, .., f_n(z) = 0$. Denoting with $\zeta_1, .., \zeta_s$ the variables that appear in the normalisation equations, define

$$T_{ij} = I\left(\frac{D}{Dx_i}\zeta_i\right), \qquad \Omega_{ij} = \left(\frac{\partial(g\cdot\zeta_i)}{\partial a_i}\right)\Bigg|_{g=\rho}, \tag{2.7}$$

as the *invariant total derivative matrix* and the invariantised matrix of infinitesimal for the variables $\zeta_1, .., \zeta_s$ with respect to the group parameters $a_1, .., a_n$ in a neighbourhood of the identity. If $J_{ij} = \frac{\partial f_j}{\partial I(\zeta_i)}$ is the Jacobian of the left hand side of the normalisation equations, then it is possible to write the correction terms in (2.6) as

$$M_{K+\mathbf{e}_j}^\alpha = \sum_{i=1}^{n} K_{ji}\phi_{K,i}^\alpha(I), \tag{2.8}$$

where

$$K = -TJ\left(\Omega J\right)^{-1}. \tag{2.9}$$

is called the *correction matrix* and $\phi_{K,i}^\alpha(I)$ are the invariantised infinitesimals defined in (2.5). Software coded in Maple exists, [41], that computes the correction matrix and performs the invariant differentiation process.

If we consider two differential invariants I^α_J and I^α_L such that $JK = LM$, then we have, from (2.6)

$$D_K I^\alpha_J - M^\alpha_{JK} = D_M I^\alpha_L - M^\alpha_{LM} \tag{2.10}$$

as I^α_{JK} and I^α_{LM} cancel each other out. Equation (2.10) is an example of what is called a *syzygy*, i.e. a differential relation between invariants. Syzygies will be used extensively in the following as they play a pivotal role in the derivation of the invariantised Euler–Lagrange equations of a variational system. We note that, in general, syzygies take the form of nonlinear ODEs or PDEs, depending on the number of independent variables in the problem at hand.

We end this subsection giving a result about the algebra of the differential invariants. Given the set of *zeroth–invariants*, defined as

$$\mathscr{I}^0 := \left\{ I(x_1), .., I(x_p), I^1, .., I^q \right\}, \tag{2.11}$$

then it can be proved that

Theorem 2.2.29 ([30])**.** *Suppose the normalisation equations* $\{f_k = 0\}_{k=1,\dots,n}$ *yield a frame for a free and regular action on some open set of the prolonged space with coordinates* $(x_j, u^\alpha, u^\alpha_K)$*. Then the components of the correction matrix K, together with* $\mathscr{I}^0$*, form a generating set of differential invariants.*

2.2.4 The Lie algebra

Any Lie group, being a smooth manifold, has a tangent space at each point. We consider the one at the identity element, equipped with an operation called *Lie bracket*. This vector space, equipped with a Lie bracket, is known as a *Lie algebra*, a fundamental space that will often occur throughout this text. In the following, we will treat the case of matrix Lie groups, although generalisations are possible.

Definition 2.2.30. Given a Lie group G, its Lie algebra is given by $(\mathfrak{g}, [,])$, where $\mathfrak{g}$ is T_eG, i.e. the tangent space to G at the identity element, and $[,]$ is a function called *Lie bracket*, defined as

$$[,] : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$$

$$[A,B] \mapsto AB - BA,$$

and such that for every $A,B,C \in \mathfrak{g}$ and $a,b,c \in \mathbb{R}$, the Lie bracket satisfies the following three properties:

- $[aA+bB,C] = a[A,C] + b[B,C]$ and $[A,bB+cC] = b[A,B] + c[A,C]$,
- $[A,B] = -[B,A]$,
- $[A,[B,C]] + [B,[C,A]] + [C,[A,B]] = 0$. This relation is called *Jacobi identity*

Example 2.2.31. In the next chapters we will make extensive use of some notable Lie algebras. An example of these is $\mathfrak{so}(3)$, i.e. the Lie algebra associated to the Lie group $SO(3)$. It is defined as

$$\mathfrak{so}(3) = \left\{ A \in GL(3,\mathbb{R}) : A^T + A = 0 \right\}.$$

As a vector space, a basis is given by

$$v_1 = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad v_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix}, \quad v_3 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}.$$

The Lie bracket is defined as $[A,B] = AB - BA$. It is clearly bilinear and antisymmetric. The Jacobi identity can be verified as follows on the elements of a basis. It is easy to verify that $[v_1, v_2] = -v_3$, $[v_2, v_3] = -v_1$ and $[v_1, v_3] = v_2$. Once at this point, it is straightforward to see that, taken $A = a_1v_1 + a_2v_2 + a_3v_3$, $B = b_1v_1 + b_2v_2 + b_3v_3$, and $C = c_1v_1 + c_2v_2 + c_3v_3$, then the Jacobi identity holds.

2.2.5 Curvature Matrices

Given a Lie group action of G with Lie algebra $\mathfrak{g}$ consider the action of G on $\mathscr{U} \subset M$. Let $s \to z(s) \in \mathscr{U}$ be a smooth path and assume that the parameter s is invariant under the group action, so that d/ds is an invariant differential operator.

Proposition 2.2.32 ([42, p. 161])**.** *If $\rho : \mathscr{U} \subset M \to G$ is a right frame for a left action, then*

1. *The components of $\rho_s\rho^{-1}$ are invariant,*

2. $\left(\frac{d}{ds}\rho\right)\rho^{-1} : \mathscr{U} \to \mathfrak{g}$.

With this in mind we proceed to define the *curvature matrices*.

Definition 2.2.33. Under the assumptions stated above, the curvature matrices are defined as

$$\mathscr{Q}^i = (\mathscr{D}_i\rho)\rho^{-1}, \qquad i = 1,..,p. \tag{2.12}$$

It is possible to write the curvature matrices in terms of the correction matrix (2.9) and a basis of the Lie algebra in which they live.

Theorem 2.2.34 ([42, p. 163])**.** *Given a matrix representation of the Lie algebra $\mathfrak{g}$ (induced by a representation R of its associated Lie group) with basis $v_1,..,v_n$, it holds*

$$\mathscr{Q}^i = \sum_{i,j} K_{ij}v_j.$$

The following result concerns a differential relation between curvature matrices, in the presence of at least two independent variables.

Theorem 2.2.35 ([42, p. 165])**.**

$$\mathscr{D}_j(\mathscr{Q}^i) - \mathscr{D}_i(\mathscr{Q}^j) = ([\mathscr{D}_j, \mathscr{D}_i]\rho)\rho^{-1} + [\mathscr{Q}^j, \mathscr{Q}^i].$$

The differential operators are applied to the curvature matrices term by term. In the case of invariant independent variables, we have that $\mathscr{D}_i$ and $\mathscr{D}_j$ commute and the formula above simplifies to

$$\mathscr{D}_j(\mathscr{Q}^i) - \mathscr{D}_i(\mathscr{Q}^j) = [\mathscr{Q}^j, \mathscr{Q}^i]. \tag{2.13}$$

Equation (2.13) will play an important role in Chapter 4, where an application of Lie group integrators to higher dimensional invariant variational problems is presented.

Example 2.2.36. Consider the action defined in (2.3) and the frame given by the normalisation equations

$$\begin{cases} \tilde{x} = 0 \\ \tilde{u} = 1 \\ \widetilde{u}_s = 0. \end{cases}$$

A basis for the Lie algebra $\mathfrak{sl}(2)$ is

$$v_a = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad v_b = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \quad v_c = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}.$$

and the correction matrix is

$$K = \begin{array}{c} \begin{matrix} a & b & c \end{matrix} \\ \begin{pmatrix} 0 & -I_1^x & \frac{1}{2}\frac{I_{11}^u}{I_1^x} \end{pmatrix} \end{array}.$$

Hence, the curvature matrix for this action and these normalisation equations is obtained, applying Theorem 2.2.34, as

$$\mathscr{Q}^s = 0v_a - I_1^x v_b + \frac{1}{2}\frac{I_{11}^u}{I_1^x}v_c = \begin{pmatrix} 0 & -I_1^x \\ \frac{1}{2}\frac{I_{11}^u}{I_1^x} & 0 \end{pmatrix}$$

.

2.2.6 Adjoint representation

Another essential ingredient for our discussion is the Adjoint representation of an element belonging to a Lie group. This is the representation of the moving frame that we will need in order to express the conservation laws.

Consider $\mathscr{X}(M)$ as the set of all vector fields on a smooth manifold M. Then we can define an action of a Lie group on it, called the *Adjoint action*.

Definition 2.2.37. Given a Lie group G acting on the manifold M, the *Adjoint action* of G on $\mathscr{X}(M)$ is defined as

$$Ad : G \times \mathscr{X}(M) \to \mathscr{X}(M)$$

$$(g, \mathbf{v}) \mapsto Ad_g(\mathbf{v})(z) = Tg^{-1}\mathbf{v}(g \cdot z),$$

where $Tg : TM \to TM$ is the tangent map of the action $(g,z) \mapsto g \cdot z$.

It can be shown that if we fix $g \in G$, then Ad_g is a representation on $\mathscr{X}(M)$. However, we are interested in a subgroup $\mathscr{X}_G(M) \subset \mathscr{X}(M)$, which is an n–dimensional representation of the Lie algebra $\mathfrak{g}$. To construct it, take the set of all smooth paths $\gamma : [-\varepsilon, \varepsilon] \to G$, $\varepsilon > 0$, such that $\gamma(0) = e$. Define an equivalence relation on paths such that two paths are in the same equivalence class if their derivative at $t = 0$ coincides. A smooth path $\gamma(\varepsilon) \in G$ generates

a smooth path in M by considering $\gamma(\varepsilon)\cdot z$, for $z\in M$. The derivatives of these paths on M, evaluated at $t=0$, is a vector field, and it is denoted by $\mathscr{X}_G(M)$.

Lemma 2.2.38 ([42], p.108)**.** *The function $Ad_g : \mathscr{X}_G(M)\to\mathscr{X}_G(M)$ is a representation.*

Consider a basis $\{\mathbf{v}_i\}_i$ of $\mathscr{X}_G(M)$ and an element $\mathbf{w}\in\mathscr{X}_G(M)$ given by $\mathbf{w}=\sum_i \alpha_i\mathbf{v}_i$. Applying Ad_g we obtain

$$Ad_g(\mathbf{w}) = Ad_g\left(\sum_i \alpha_i\mathbf{v_i}\right) = \sum_i \alpha_i Ad_g(\mathbf{v_i}) = \sum_i \tilde{\alpha}_i\mathbf{v}_i. \tag{2.14}$$

The last equality in (2.14) hints at the fact that in practice the Adjoint representation can be computed as the induced action on the components of $\mathbf{w}$. If we write $\mathbf{w}$ as a row vector, then we have that

$$\tilde{\boldsymbol{\alpha}} = \boldsymbol{\alpha} Ad(g). \tag{2.15}$$

2.3 Invariant Calculus of Variations

So far in the chapter we have introduced the basic concepts in the theory of moving frames and prepared the ground for the main application we are interested in, which is the one to the Calculus of Variations.

2.3.1 Invariantised Euler–Lagrange equations

Given a Lagrangian

$$\mathscr{L} = \int_D L(x_1,..,x_p,u^\alpha,u^\alpha_1,u^\alpha_2,..)\,d\mathbf{x},$$

where L depends only on a finite number of derivatives, and D is a bounded and simply connected region in $\mathbb{R}^p$, consider a Lie group G acting on $M=\mathscr{J}^n(X\times U)$.

Definition 2.3.1. The Lie group G is a *variational symmetry group* of (2.3.1) if

$$\int_{g\cdot D} L(g\cdot x_1,..,g\cdot x_p,g\cdot u^\alpha,g\cdot u^\alpha_1,g\cdot u^\alpha_2,..)\,d(g\cdot\mathbf{x}) = \int_D L(x_1,..,x_p,u^\alpha,u^\alpha_1,u^\alpha_2,..)\,d\mathbf{x}. \quad (2.16)$$

It is also said that L is invariant under the action of G. Our aim is to find a solution to such variational problem, i.e. a function $\mathbf{u}$ that minimises the Lagrangian at hand. We are interested in using the theory of moving frames developed so far in order to take advantage of the invariantised variables and make the problem more tractable than using the original variables.

Lemma 2.3.2 (Fundamental Lemma of Calculus of Variations, [21, p. 9])**.** *If a function f, defined in a bounded and simply connected region $D\subset\mathbb{R}^n$, satisfies*

$$\int_D f(\mathbf{x})h(\mathbf{x})\,d\mathbf{x} = 0$$

for any compactly supported smooth function h on D, then $f\equiv 0$.

Remark 2.3.3. From here on, when specifying a domain of integration, we will denote by D a simply connected region in $\mathbb{R}^p$, for a suitable $p\in\mathbb{N}$.

The Fundamental Lemma of Calculus of Variations is one of the main tools we will need when deriving the invariantised Euler–Lagrange equations for a Lagrangian.

Denote the set of generating differential invariants with $\{I^\alpha_K\}_{\alpha,K} = \{\kappa_i\}_i$. We follow [24]–[23] and introduce a dummy, independent and invariant variable τ. Formally, we have the following equivalence for the first variation of $\mathscr{L}$:

$$\frac{d}{d\varepsilon}\mathscr{L}(\mathbf{u}+\varepsilon\mathbf{v}) = \frac{d}{d\tau}\bigg|_{\mathbf{u}_\tau=\mathbf{v}} \mathscr{L}(\mathbf{u}). \quad (2.17)$$

The introduction of a new variable generates q new invariants, denoted as $I^{\alpha}_{\tau} = g \cdot u^{\alpha}_{\tau}\big|_{g=\rho}$, $\alpha = 1,..,q$, and therefore q new syzygies, that can be written as

$$\mathscr{D}_{\tau}\begin{pmatrix} \kappa_1 \\ \vdots \\ \kappa_m \end{pmatrix} = \mathscr{H}\begin{pmatrix} I^1_{\tau} \\ \vdots \\ I^q_{\tau} \end{pmatrix}, \tag{2.18}$$

where $\mathscr{H}$ is a $m \times q$ linear differential operator containing only the generating differential invariants κ_i, and their derivatives. Supposing we have independent invariant variables $x_1,..,x_p$, and denoting with $\boldsymbol{\kappa}$ the vector containing the generating differential invariants κ_i and their derivatives, we can rewrite the Lagrangian as

$$\mathscr{L} = \int_D L(\mathbf{x}, \boldsymbol{\kappa})\, d\mathbf{x}.$$

Remark 2.3.4. Throughout the present work we will assume that the Lagrangians are regular enough to allow for the interchange of the integration and differentiation signs.

The computations that follow are taken from [42]. In order to obtain the invariantised Euler–Lagrange equations, consider

$$\begin{aligned} \frac{d}{d\tau}\mathscr{L} &= \frac{d}{d\tau}\int_D L(x_1,..,x_p,\boldsymbol{\kappa})\, d\mathbf{x} \\ &= \int_D \frac{d}{d\tau} L(x_1,..,x_p,\boldsymbol{\kappa})\, d\mathbf{x} \\ &= \int_D \left(\sum_{i,K} \frac{\partial L}{\partial \mathscr{D}_K \kappa_i} \mathscr{D}_K \mathscr{D}_{\tau} \kappa_i \right) d\mathbf{x}. \end{aligned}$$

After performing a first set of integration by parts, we obtain

$$\int_D \left(\sum_{i,K} \frac{\partial L}{\partial \mathscr{D}_K \kappa_i} \mathscr{D}_K \mathscr{D}_\tau \kappa_i \right) d\mathbf{x}$$

$$= \int_D \left(\sum_{i,K} (-1)^{|K|} \mathscr{D}_K \frac{\partial L}{\partial \mathscr{D}_K \kappa_j} \mathscr{D}_\tau \kappa_i \right) + \mathrm{Div}\,(\mathrm{B1})\, d\mathbf{x},$$

where B1 stands for the boundary terms that arise from the integration by parts. The integration by parts is a special case of the adjoint of a differential operator.

Definition 2.3.5. Given a differential operator $\mathscr{A}$, then the adjoint of $\mathscr{A}$ is the operator $\mathscr{A}^*$ such that, given two smooth functions f, g, it holds

$$\int_D f \mathscr{A}(g) d\mathbf{x} = \int_D \mathscr{A}^*(f) g + \mathrm{div}(BT) d\mathbf{x},$$

where $\mathrm{div}(BT)$ stands for some boundary terms .

In the following we will denote with

$$E^i(L) = \sum_K (-1)^{|K|} \mathscr{D}_K \frac{\partial L}{\partial \mathscr{D}_K \kappa_i}, \tag{2.19}$$

the Euler–Lagrange operator for the invariant κ_i. Resuming the computations, we express $\mathscr{D}_\tau \kappa_j$ in terms of the invariants relative to the dummy variable, via the syzygies in (2.18).

$$\int_D \left(\sum_i E^i(L) \mathscr{D}_\tau \kappa_j \right) + \mathrm{Div}\,(\mathrm{B1})\, d\mathbf{x}$$

$$= \int_D \left(\sum_{i,\alpha} E^i(L) \mathscr{H}_{i,\alpha} \mathbf{I}_\tau^\alpha \right) + \mathrm{Div}\,(\mathrm{B1})\, d\mathbf{x}$$

$$= \int_D \left(\sum_{i,\alpha} \mathscr{H}^*_{i,\alpha} E^i(L) \mathbf{I}_\tau^\alpha \right) + \mathrm{Div}\,(\mathrm{B1} + \mathrm{B2})\, d\mathbf{x},$$

where $\mathscr{H}^\star$ is the adjoint operator of $\mathscr{H}$ and B2 are the boundary terms arising from taking the adjoint of $\mathscr{H}$. Recall that we are computing the first variation as (2.17) and that the $\mathbf{I}^\alpha_\tau$ are the invariantised form of the variables u^α_τ, which are used to perform the variation. Therefore, invoking Lemma 2.3.2, we have that the invariantised Euler–Lagrange equations are

$$E^\alpha(L) = \sum_i \mathscr{H}^*_{i,\alpha} E^i(L).$$

2.3.2 Conservation laws

In the derivation of the invariantised Euler–Lagrange equations, there were two sets of boundary terms, B1 and B2, that arose from performing integration by parts and from taking the adjoint of the differential operator $\mathscr{H}$. It will be shown in this subsection that these terms can be rearranged into a divergence, i.e. some conservation laws. These can be expressed in terms of the adjoint representation of the moving frame and a vector of invariants, [42].

Theorem 2.3.6 ([24], Thm. 2.9)**.** *Consider a Lagrangian $\mathscr{L} = \int_D L(\kappa_1, \kappa_2, ..)\, d\mathbf{x}$ that is invariant under a Lie group action $G \times M \to M$, where $M = J^n(X \times U)$, that has κ_i, $i = 1, .., N$, as generating differential invariants and has $g \cdot x_i = x_i$, $i = 1, .., p$. Then, after introducing an independent, invariant dummy variable τ, the first variation of the $\mathscr{L}$ can be written as*

$$\frac{d}{d\tau}\mathscr{L} = \int \sum_{\alpha,j} E^\alpha(L) I^\alpha_\tau + \mathrm{Div}(P)\, d\mathbf{x} \tag{2.20}$$

where this defines the vector P, whose components are of the form

$$P_i = \sum_{\alpha,j} I^\alpha_{\tau J} C^\alpha_{i,J}, \quad i = 1, .., p$$

and the vectors $\mathscr{C}^\alpha_i = (C^\alpha_{i,J})$. Let $(a_1, .., a_r)$ be the coordinates of G near the identity e, and $\mathbf{v}_i$, for $i = 1, .., r$, the associated infinitesimal vector fields. Furthermore, let $Ad(g)$ be the Adjoint representation of G with respect to these vector fields. Then the r conservation laws

obtained via Noether's Theorem can be written in the form

$$\sum_i \frac{D}{Dx_i} Ad(\rho)^{-1}\mathbf{v}(I) = 0$$

where

$$\mathbf{v}(I) = \sum_\alpha \Omega^\alpha(I)\mathscr{C}_i^\alpha$$

Remark 2.3.7. In the case of one–dimensional invariant Lagrangians, the conservation laws can be written as

$$Ad(\rho)^{-1}\mathbf{v}(I) = \mathbf{c}$$

where $\mathbf{c}$ is a constant vector.

Example 2.3.8. To show how this theory works, we show an application taken from [24]. Consider the action of $SL(2)$ acting on surfaces $u = u(x,t)$ as

$$g\cdot x = x \qquad g\cdot t = t \qquad g\cdot u = \frac{au+b}{cx+d} \tag{2.21}$$

where

$$g = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \qquad ad-bc = 1$$

Prolong the action to u_x and u_{xx}, then the moving frame for the cross section given by $\tilde{u} = 0$, $\widetilde{u_x} = 1$, and $\widetilde{u_{xx}} = 0$ is

$$\rho = \begin{pmatrix} \dfrac{1}{\sqrt{u_x}} & -\dfrac{u}{\sqrt{u_x}} \\ \dfrac{u_{xx}}{2u_x^{3/2}} & \dfrac{2u_x^2 - uu_{xx}}{2u_x^{3/2}} \end{pmatrix}$$

The generating differential invariants are $I^u_{111} = \sigma$, which is the *Schwarzian derivative*, and $I^u_2 = \kappa$. Introducing a new independent, invariant variable τ, the syzygies between σ, κ and

$I^u_\tau = \rho \cdot u_\tau$ can be written as

$$\mathscr{D}_\tau \begin{pmatrix} \sigma \\ \kappa \end{pmatrix} = \begin{pmatrix} \mathscr{H}_1 \\ \mathscr{H}_2 \end{pmatrix} I^u_\tau$$

where $\mathscr{H}_1 = \mathscr{D}_x^3 + 2\sigma\mathscr{D}_x + \sigma_x$ and $\mathscr{H}_2 = \mathscr{D}_t - \kappa\mathscr{D}_x + \kappa_x$.

Given a Lagrangian

$$\mathscr{L} = \int L(\sigma, \kappa, \sigma_x, \sigma_t, \kappa_x, \kappa_t)\,\mathrm{d}x\,\mathrm{d}t$$

invariant under the action (2.21), then the invariantised Euler–Lagrange equations are

$$E^u(L) = \mathscr{H}_1^* E^\sigma(L) + \mathscr{H}_2^* E^\kappa(L)$$

where $\mathscr{H}_1^*$ and $\mathscr{H}_2^*$ are the adjoint operators of $\mathscr{H}_1$ and $\mathscr{H}_2$ respectively.

In this example, the invariantised matrix of infinitesimals is given by

$$\Omega = \begin{pmatrix} 0 & 2 & 0 & 2\sigma & 2\kappa \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -2 & 0 & 0 \end{pmatrix}$$

and this allows to write the conservation laws as

$$\mathscr{D}_x \left(Ad(\rho)^{-1} \begin{pmatrix} -2\dfrac{d}{dx}E^\sigma(L) \\ \sigma E^\sigma(L) - \kappa E^\kappa(L) + \dfrac{d^2}{dx^2}E^\sigma(L) \\ -2E^\sigma(L) \end{pmatrix} \right) + \mathscr{D}_t \left(Ad(\rho)^{-1} \begin{pmatrix} 0 \\ E^\kappa(L) \\ 0 \end{pmatrix} \right) = 0$$

We will see another application in the next chapter, where we apply the theory of Invariant Calculus of Variations to find the minimisers of a one–dimensional variational problem which is invariant under a linear action of $SU(2)$.

3. Variational problems invariant under a linear action of $SU(2)$

3.1 Introduction

As we saw in the previous chapters, Noether's first Theorem yields conservations laws for a Lagrangian with a symmetry group. In [23; 24] it has been shown how to make use of the machinery provided by the moving frames to obtain an invariantised version of the Euler–Lagrange equations and a set of conservations laws in terms of the differential invariants of the group action. On top of the theoretical interest of showing more clearly the mathematical structure underlying the conservation laws, this approach has proved to be also of practical application, as it can simplify the integration process needed to find extremal solutions. In particular, so far this method has been applied to Lagrangians left invariants by action of the Lie groups $SL(2,\mathbb{C})$, $SE(2)$ and $SE(3)$, [22; 23; 24; 42].

The aim of this chapter is to provide another application in the context of variational systems, this time considering the case where the Lagrangian is invariant under a linear action of the Lie group $SU(2)$ on complex curves. This work is motivated by the importance that $SU(2)$ has in quantum physics, for instance as a model for particles with non–integer spin. The original contributions in this chapter can be found in Section 3.2.2 and Section 3.3, where we solve the conservation laws to find a minimiser and we show how this theory

works in practice with some numerical examples. Although the results themselves are already known, we believe the way we solve the conservation laws for this specific example has not been studied before.

3.2 The one–dimensional case

Consider the group $SU(2) = \{A \in GL(2,\mathbb{C}) : AA^* = I = A^*A\,, \det(A) = 1\}$, where A^* is the conjugate transpose of A, acting linearly on pairs of complex curves $(u(t), v(t))$ as follows:

$$\begin{pmatrix} \tilde{u}(t) \\ \tilde{v}(t) \end{pmatrix} = g \cdot \begin{pmatrix} u(t) \\ v(t) \end{pmatrix} = \begin{pmatrix} \alpha & \beta \\ -\bar{\beta} & \bar{\alpha} \end{pmatrix} \begin{pmatrix} u(t) \\ v(t) \end{pmatrix} = \begin{pmatrix} \alpha u(t) + \beta v(t) \\ -\bar{\beta} u(t) + \bar{\alpha} v(t) \end{pmatrix}, \tag{3.1}$$

with $\alpha, \beta \in \mathbb{C}$, $|\alpha|^2 + |\beta|^2 = 1$, $\tilde{t} = t$.

In this case, prolonging the action is straightforward as the independent variable t is left invariant. Using the chain rule we get

$$g \cdot u_t = \tilde{u}_{\tilde{t}} = g \cdot \frac{d\tilde{u}}{dt}\frac{dt}{d\tilde{t}} = \frac{d\tilde{u}}{dt} = \tilde{u}_t.$$

The prolonged action on the n-th derivative is

$$g \cdot u_{\underbrace{t..t}_{n \text{ times}}} = \tilde{u}_{\underbrace{t..t}_{n \text{ times}}}.$$

The prolongation to the derivatives of $v(t)$ is defined in an analogous way. We want to find the minimisers of the Lagrangian

$$\mathscr{L} = \int_D L(t, u, v, u_t, v_t, u_{tt}, ..)\, dt$$

in the case where $\mathscr{L}$ is invariant under action (3.1). Note that in the following we will only consider the case where L depends on a finite number of variables.

Consider M as the jet space with coordinates $(t,u,v,u_t,v_t,u_{tt},v_{tt},..)$. In order to define a moving frame for action (3.1) we choose the following cross–section:

$$\begin{cases} \rho \cdot u = 1 \\ \rho \cdot v = 0. \end{cases} \tag{3.2}$$

This gives us a frame for $U(2)$. Since $SU(2)$ is a 3-dimensional group and system (3.2) comprises 4 equations, we will impose a condition on the solutions in order to retrieve the moving frame as an element of $SU(2)$. The solution to (3.2) is

$$\rho = \frac{1}{|u|^2+|v|^2}\begin{pmatrix} \bar{u} & \bar{v} \\ -v & u \end{pmatrix}.$$

We now impose the determinant to be equal to 1 in order to have an element of $SU(2)$, obtaining

$$|u|^2+|v|^2 = 1, \tag{3.3}$$

and therefore

$$\rho = \begin{pmatrix} \bar{u} & \bar{v} \\ -v & u \end{pmatrix}.$$

Two differential invariants of the action are given using Theorem 2.2.23, applying the moving frame to the dependent variables u_t and v_t.

$$\rho \cdot \begin{pmatrix} u_t \\ v_t \end{pmatrix} = \begin{pmatrix} \bar{u} & \bar{v} \\ -v & u \end{pmatrix}\begin{pmatrix} u_t \\ v_t \end{pmatrix} = \begin{pmatrix} \bar{u}u_t + \bar{v}v_t \\ uv_t - vu_t \end{pmatrix} = \begin{pmatrix} \kappa_u \\ \kappa_v \end{pmatrix}$$

A closer look at κ_u reveals us that it is pure imaginary. To see this, rewrite equation (3.3) as

$$\bar{u}u + \bar{v}v = 1,$$

and differentiate with respect to t:

$$\bar{u}_t u + \bar{u}u_t + \bar{v}_t v + \bar{v}v_t = 0.$$

Rearranging terms we have

$$\bar{u}_t u + \bar{v}_t v = -(\bar{u}u_t + \bar{v}v_t),$$

and hence

$$\kappa_u = -\bar{\kappa}_u,$$

which implies that κ_u is a pure imaginary number. We know that κ_u, κ_v are differential invariants, as we obtained them applying the moving frames to coordinates, but we do not know yet if they generate the whole algebra of differential invariants. However, using Theorem 2.2.29, we have that the set

$$\{\rho \cdot u_t, \rho \cdot v_t\} = \{\kappa_u, \kappa_v\}$$

is a generating set of differential invariants by computing the correction matrix. Namely, using the *Indiff* Maple package [41], we compute the correction matrix K for $U(2)$ and then we simplify the result using $\kappa_u = -\bar{\kappa}_u$, obtaining

$$K = \begin{pmatrix} 0 & I_1^{u_2} & -I^{v_1} & I_1^{v_2} \end{pmatrix},$$

where we used the notation $u = u_1 + \mathrm{i}u_2$ and $v = v_1 + \mathrm{i}v_2$.

Denote a general element of $SU(2)$ as

$$g = \begin{pmatrix} a+\mathrm{i}b & c+\mathrm{i}d \\ -c+\mathrm{i}d & a-\mathrm{i}b \end{pmatrix},$$

with $a,b,c,d \in \mathbb{R}$ and $a^2+b^2+c^2+d^2=1$. We take as basis for the infinitesimal vector field the derivatives of (3.1) with respect to the three parameters of $SU(2)$, namely b,c,d and $a=\sqrt{1-b^2-c^2-d^2}$, computed at the identity. First we note that

$$\frac{\partial a}{\partial b}\Big|_{g=e} = \frac{\partial a}{\partial c}\Big|_{g=e} = \frac{\partial a}{\partial d}\Big|_{g=e} = 0,$$

so a basis for the vector field is given by

$$V_b = \frac{\partial \tilde{u}}{\partial b}\Big|_{g=e}\partial_u + \frac{\partial \tilde{v}}{\partial b}\Big|_{g=e}\partial_v = -u_2\partial_{u_1} + \mathrm{i}u_1\partial_{u_2} + v_2\partial_{v_1} - \mathrm{i}v_1\partial_{v_2}, \tag{3.4}$$

$$V_c = \frac{\partial \tilde{u}}{\partial c}\Big|_{g=e}\partial_u + \frac{\partial \tilde{v}}{\partial c}\Big|_{g=e}\partial_v = v_1\partial_{u_1} + \mathrm{i}v_2\partial_{u_2} - u_1\partial_{v_1} - \mathrm{i}u_2\partial_{v_2}, \tag{3.5}$$

$$V_d = \frac{\partial \tilde{u}}{\partial d}\Big|_{g=e}\partial_u + \frac{\partial \tilde{v}}{\partial d}\Big|_{g=e}\partial_v = -v_2\partial_{u_1} + \mathrm{i}v_1\partial_{u_2} - u_2\partial_{v_1} + \mathrm{i}u_1\partial_{v_2}, \tag{3.6}$$

and we apply action (3.1) to (3.4)–(3.5)–(3.6) in order to derive the adjoint representation, defined by (2.15), namely

$$\begin{pmatrix} \tilde{V}_b \\ \tilde{V}_c \\ \tilde{V}_d \end{pmatrix} = Ad(g) \begin{pmatrix} V_b \\ V_c \\ V_d \end{pmatrix}.$$

In terms of the group parameters, the Adjoint representation is

$$Ad(g) = \begin{pmatrix} 2(a^2+b^2)-1 & -2(ad-bc) & 2(ac+bd) \\ 2(ad+bc) & 2(a^2+c^2)-1 & -2(ab-cd) \\ 2(bd-ac) & 2(ab+cd) & 2(a^2+d^2)-1 \end{pmatrix}. \tag{3.7}$$

Bearing in mind that $a^2+b^2+c^2+d^2=1$, matrix (3.7) is precisely the *Cayley map*, [18]. It can be easily checked that $\det(Ad(g))=1$ and $Ad(g)^T = Ad(g)^{-1}$, hence $Ad(g)$ is a rotation matrix.

We introduce a dummy independent and invariant variable τ, that we will use to perform the variation and compute both the invariantised Euler–Lagrange equations for u and v. This new variable generates a set of differential invariants given by

$$\begin{aligned} \rho \cdot \begin{pmatrix} u_\tau \\ v_\tau \end{pmatrix} &= \begin{pmatrix} \bar{u} & \bar{v} \\ -v & u \end{pmatrix} \begin{pmatrix} u_\tau \\ v_\tau \end{pmatrix} \\ &= \begin{pmatrix} \bar{u}u_\tau + \bar{v}v_\tau \\ -vu_\tau + uv_\tau \end{pmatrix} \\ &= \begin{pmatrix} \sigma_u \\ \sigma_v \end{pmatrix}. \end{aligned}$$

We note that σ_u, as it was the case for κ_u, is a pure imaginary number and this can be seen in an analogous way as we did for κ_u, using relation (3.3), this time differentiating with respect to τ.

The introduction of the dummy variable τ also comes with a set of syzygies relating the

differential invariants:

$$\frac{d}{d\tau}\begin{pmatrix}\kappa_u\\ \kappa_v\end{pmatrix} = \mathscr{J}\begin{pmatrix}\sigma_u\\ \sigma_v\end{pmatrix}. \tag{3.8}$$

The operator $\mathscr{J}$ is going to be crucial in the computations of the invariantised Euler–Lagrange equations. In order to derive $\mathscr{J}$, we are going to first compute the curvature matrices related to the two independent variables t and τ, and then use (2.13). Aiming at a more clear exposition, we split the invariants into their real and imaginary part. The notation for the differential invariants from now on will be:

$$\kappa_u = \mathrm{i}\kappa_1,$$

$$\kappa_v = \kappa_2 + \mathrm{i}\kappa_3,$$

$$\sigma_u = \mathrm{i}\sigma_1,$$

$$\sigma_v = \sigma_2 + \mathrm{i}\sigma_3,$$

where $\kappa_1, \kappa_2, \kappa_3, \sigma_1, \sigma_2, \sigma_3 \in \mathbb{R}$, so that equation (3.8) now becomes

$$\frac{d}{d\tau}\begin{pmatrix}\kappa_1\\ \kappa_2\\ \kappa_3\end{pmatrix} = \mathscr{H}\begin{pmatrix}\sigma_1\\ \sigma_2\\ \sigma_3\end{pmatrix}, \tag{3.9}$$

where $\mathscr{H}$ plays the role of $\mathscr{J}$, but now is a 3×3 matrix. The curvature matrix along the direction of t is

$$\begin{aligned} Q^t = \rho_t \rho^{-1} &= \begin{pmatrix} \bar{u}_t & \bar{v}_t \\ -v_t & u_t \end{pmatrix} \begin{pmatrix} u & -\bar{v} \\ v & \bar{u} \end{pmatrix} \\ &= \begin{pmatrix} u\bar{u}_t + v\bar{v}_t & \bar{u}\bar{v}_t - \bar{v}\bar{u}_t \\ vu_t - uv_t & \bar{u}u_t + \bar{v}v_t \end{pmatrix} \\ &= \begin{pmatrix} \bar{\kappa}_u & \bar{\kappa}_v \\ -\kappa_v & \kappa_u \end{pmatrix} \\ &= \begin{pmatrix} -\mathrm{i}\kappa_1 & \kappa_2 - \mathrm{i}\kappa_3 \\ -\kappa_2 - \mathrm{i}\kappa_3 & \mathrm{i}\kappa_1 \end{pmatrix}. \end{aligned}$$

Analogously we derive the other curvature matrix as

$$\begin{aligned} Q^\tau = \rho_\tau \rho^{-1} &= \begin{pmatrix} \bar{\sigma}_u & \bar{\sigma}_v \\ -\sigma_v & \sigma_u \end{pmatrix} \\ &= \begin{pmatrix} -\mathrm{i}\sigma_1 & \sigma_2 - \mathrm{i}\sigma_3 \\ -\sigma_2 - \mathrm{i}\sigma_3 & \mathrm{i}\sigma_1 \end{pmatrix}. \end{aligned}$$

The structure of the curvature matrices is something we were expecting: Proposition 2.2.32 says that the components of the curvature matrix can be expressed in terms of only the differential invariants and their derivatives. Also note that both $\mathscr{Q}^t$ and $\mathscr{Q}^\tau$ belong to $\mathfrak{su}(2)$. Using (2.13), since $\frac{d}{dt}$ and $\frac{d}{d\tau}$ commute, the following set of syzygies holds

$$\frac{d}{d\tau}Q^t - \frac{d}{dt}Q^\tau = [Q^\tau, Q^t], \tag{3.10}$$

where $[\cdot,\cdot]$ is the usual Lie bracket. Expanding the commutator in (3.10) leads us to

$$\frac{d}{d\tau}\begin{pmatrix} -\mathrm{i}\kappa_1 & \kappa_2-\mathrm{i}\kappa_3 \\ -\kappa_2-\mathrm{i}\kappa_3 & \mathrm{i}\kappa_1 \end{pmatrix} - \frac{d}{dt}\begin{pmatrix} -\mathrm{i}\sigma_1 & \sigma_2-\mathrm{i}\sigma_3 \\ -\sigma_2-\mathrm{i}\sigma_3 & \mathrm{i}\sigma_1 \end{pmatrix}$$

$$= 2\begin{pmatrix} \mathrm{i}(\sigma_3\kappa_2-\sigma_2\kappa_3) & \mathrm{i}\kappa_1\sigma_2+\kappa_1\sigma_3-\sigma_1\kappa_3-\mathrm{i}\sigma_1\kappa_2 \\ \mathrm{i}\kappa_1\sigma_2+\sigma_1\kappa_3-\kappa_1\sigma_3-\mathrm{i}\sigma_1\kappa_2 & \mathrm{i}(\sigma_2\kappa_3-\sigma_3\kappa_2) \end{pmatrix}. \tag{3.11}$$

Collecting terms in (3.11) we find that matrix $\mathscr{H}$ in (3.9) is

$$\mathscr{H} = \begin{pmatrix} \frac{d}{dt} & 2\kappa_3 & -2\kappa_2 \\ -2\kappa_3 & \frac{d}{dt} & 2\kappa_1 \\ 2\kappa_2 & -2\kappa_1 & \frac{d}{dt} \end{pmatrix}.$$

If we define the vector

$$\mathbf{q} = 2\begin{pmatrix} \kappa_1 \\ \kappa_2 \\ \kappa_3 \end{pmatrix},$$

then the syzygies are

$$\frac{d}{d\tau}\begin{pmatrix} \kappa_1 \\ \kappa_2 \\ \kappa_3 \end{pmatrix} = \begin{pmatrix} \frac{d}{dt} & 2\kappa_3 & -2\kappa_2 \\ -2\kappa_3 & \frac{d}{dt} & 2\kappa_1 \\ 2\kappa_2 & -2\kappa_1 & \frac{d}{dt} \end{pmatrix}\begin{pmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \end{pmatrix} \tag{3.12}$$

$$= \frac{d}{dt}\begin{pmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \end{pmatrix} - \mathbf{q}\times(\sigma_1,\sigma_2,\sigma_3)^T,$$

where $\times$ stands for the usual cross product in $\mathbb{R}^3$.

As we established that $\{\kappa_1, \kappa_2, \kappa_3\}$ is a set of generating differential invariants, now it is possible to reparametrise the Lagrangian (3.13) in terms of the differential invariants and their derivatives, obtaining

$$\mathscr{L} = \int_D L(t, \kappa_1, \kappa_2, \kappa_3, \kappa_{1,t}, \kappa_{2,t}, \ldots)\, dt = \int_D L(t, \boldsymbol{\kappa})\, dt. \tag{3.13}$$

Now we can apply the result in (2.3.1) to obtain the system of invariantised Euler–Lagrange equations, using the syzygy operator derived in (3.12), namely

$$\begin{cases} -\frac{d}{dt}\mathsf{E}^{\kappa_1}(L) + 2\kappa_3\mathsf{E}^{\kappa_2}(L) - 2\kappa_2\mathsf{E}^{\kappa_3}(L) = 0 \\ -2\kappa_3\mathsf{E}^{\kappa_1}(L) - \frac{d}{dt}\mathsf{E}^{\kappa_2}(L) + 2\kappa_1\mathsf{E}^{\kappa_3}(L) = 0 \\ 2\kappa_2\mathsf{E}^{\kappa_1}(L) - 2\kappa_1\mathsf{E}^{\kappa_2}(L) - \frac{d}{dt}\mathsf{E}^{\kappa_3}(L) = 0. \end{cases} \tag{3.14}$$

3.2.1 Conservation laws

In the previous subsection we derived the invariantised Euler–Lagrange equations in the same spirit as we would do with the original variables. We saw in Chapter 2 how the conservation laws arising from Noether's first theorem, in the one–dimensional case, can be expressed as

$$Ad(\rho)^{-1}\mathbf{w} = \mathbf{k}, \tag{3.15}$$

where **w** is a vector of invariants and **k** is constant. To find **w** we need the matrix of infinitesimal and its invariantised form. In this case we have

$$\Omega = \begin{array}{c} \\ a \\ b \\ c \end{array} \begin{array}{c} \begin{array}{cccc} u_1 & u_2 & v_1 & v_2 \end{array} \\ \begin{pmatrix} -u_2 & u_1 & v_2 & -v_1 \\ v_1 & v_2 & -u_1 & -u_2 \\ -v_2 & v_1 & -u_2 & u_1 \end{pmatrix} \end{array} \tag{3.16}$$

and recall the invariantised version of (3.16) is obtained evaluating the matrix of infinitesimals on $g = \rho$, i.e.

$$\Omega(I) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Now we can write the set of conservation laws as

$$\begin{aligned} 0 &= \frac{d}{dt}\left(\mathsf{E}^{\kappa_1}(L) Ad(\rho)^{-1} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + \mathsf{E}^{\kappa_2}(L) Ad(\rho)^{-1} \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix} + \mathsf{E}^{\kappa_3}(L) Ad(\rho)^{-1} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right) \\ &= \frac{d}{dt}\left(Ad(\rho)^{-1} \begin{pmatrix} \mathsf{E}^{\kappa_1}(L) \\ -\mathsf{E}^{\kappa_2}(L) \\ \mathsf{E}^{\kappa_3}(L) \end{pmatrix} \right). \end{aligned} \tag{3.17}$$

The vector appearing on the right hand side of (3.17) is the vector of invariants that we denoted as **w** in (3.15). If we introduce also **k**, a vector of constants, then the conservation laws can be expressed as

$$Ad(\rho)\mathbf{k} = \mathbf{w}. \tag{3.18}$$

We note that $||k|| = ||w||$, $Ad(\rho)$ being a rotation.

3.2.2 Finding the minimisers

Suppose now we have solved the invariantised Euler–Lagrange equations and therefore we know the generating differential invariants in terms of the original variables. In practical examples, it is possible we will only have numerical solutions to the Euler–Lagrange equations. Once we know the generating differential invariants, and we have chosen an appropriate vector of constants $\mathbf{k}$, then our aim is to recover the moving frame $Ad(\rho)$, i.e. to solve for the frame in the conservation laws

$$Ad(\rho)^{-1}\mathbf{w} = \mathbf{k}. \tag{3.19}$$

We are going to take advantage of the geometric setting given by the conservation laws expressed as (3.18). This means we need to find first the rotation that takes the vector $\mathbf{k}$ to the vector $\mathbf{w}$, and then we will still have a degree of freedom given by the fact that we can use $\mathbf{w}$ as a rotational axis, and perform another rotation around it. The information for this last rotation is incorporated in the curvature matrix.
The main result of this section is the following.

Theorem 3.2.1. *Given a Lagrangian invariant under action* (3.1) *and the conservation laws* (3.19)*, then the Adjoint representation of the moving frame is the product of two matrices, namely*

$$Ad(\rho) = BK, \tag{3.20}$$

where K is the rotation which takes $\mathbf{k}$ to $\mathbf{w}$ and B is another rotation matrix around $\mathbf{w}$. Representing K as a Cayley map as in (3.7)*, the parameters a,b,c, and d are given by*

$$a = 0, \qquad \begin{pmatrix} b \\ c \\ d \end{pmatrix} = \frac{\mathbf{w}+\mathbf{k}}{||\mathbf{w}+\mathbf{k}||},$$

and B, also represented as a Cayley map, is given by

$$a = \sin\left(\frac{||\mathbf{k}||}{2}\int_{t_0}^{t}\frac{\mathbf{q}\cdot(\mathbf{k}+\mathbf{w})}{\mathbf{k}\cdot(\mathbf{k}+\mathbf{w})}ds+\omega\right), \qquad \begin{pmatrix} b \\ c \\ d \end{pmatrix} = \frac{\mathbf{w}}{||\mathbf{w}||},$$

where $\mathbf{q}$ *is a vector of invariants and* ω *is a constant of integration.*

Proof. Both B and K can be thought as Cayley maps without loss of generality. To completely describe K we decide to take as axis of rotation the vector $\mathbf{w}+\mathbf{k}$ and perform a rotation of angle π around it. For this option to be feasible, we have to assume that $\mathbf{w}+\mathbf{k}\neq 0$. At the end of this section we will provide an alternative way to proceed in the case where this assumption does not hold.

The axis of rotation is the vector that is left fixed by the rotation. In algebraic terms, it is the eigenvector related to the eigenvalue 1. Since we took K in the form of (3.7), the rotational axis is given by

$$\begin{pmatrix} b \\ c \\ d \end{pmatrix} = \psi \begin{pmatrix} w_1(t)+k_1 \\ w_2(t)+k_2 \\ w_3(t)+k_3 \end{pmatrix}, \tag{3.21}$$

where $\psi \in \mathbb{R}$ is a normalisation factor, $w_i(t)$ and k_i are the i–th components of $\mathbf{w}$ and $\mathbf{k}$ respectively, and b,c,d are the group parameters as they appear in (3.7). The rotation angle θ is "controlled" entirely by the group parameter a, as the trace of the Cayley matrix gives

$$\mathrm{Tr}(K) = 4a^2 - 1 = 1 + 2\cos(\theta), \tag{3.22}$$

where the first equality is given by just summing the elements on the main diagonal of (3.7), and the second one is the trace of

$$\begin{pmatrix} \cos\theta + u_x^2(1-\cos\theta) & u_x u_y(1-\cos\theta) - u_z\sin\theta & u_x u_z(1-\cos\theta) + u_y\sin\theta \\ u_y u_x(1-\cos\theta) + u_z\sin\theta & \cos\theta + u_y^2(1-\cos\theta) & u_y u_z(1-\cos\theta) - u_x\sin\theta \\ u_z u_x(1-\cos\theta) - u_y\sin\theta & u_z u_y(1-\cos\theta) + u_x\sin\theta & \cos\theta + u_z^2(1-\cos\theta) \end{pmatrix},$$

which is the matrix representation of a rotation of angle θ around the normalised axis (u_x, u_y, u_z), see [18]. Equation (3.22) can be rewritten as

$$a = \pm\cos\left(\frac{\theta}{2}\right),$$

where the $\pm$ sign keeps track of which direction we are taking to measure angles. In our case this is irrelevant, as we rotate of π and $\cos(\frac{\pi}{2}) = 0$, hence $a = 0$. In order to keep the determinant equal to 1, we have the relation $a^2 + b^2 + c^2 + d^2 = 1$, that in this specific case simplifies to $b^2 + c^2 + d^2 = 1$. This, along with (3.21), gives us four equations for four unknowns (b, c, d, and ψ). The constant ψ normalises $\mathbf{w} + \mathbf{k}$, namely,

$$\psi(t) = \pm\frac{1}{||\mathbf{w} + \mathbf{k}||},$$

and we have two solutions as expected, but we can choose one, say the positive one, and discard the other without loss of generality. Finally we recover K as

$$K = \begin{pmatrix} 2\psi^2(k_1+w_1)^2 - 1 & 2\psi^2(k_1+w_1)(k_2+w_2) & 2\psi^2(k_1+w_1)(k_3+w_3) \\ \psi^2(k_1+w_1)(k_2+w_2) & 1 - 2\psi^2(k_2+w_2)^2 & 2\psi^2(k_3+w_3)(k_2+w_2) \\ 2\psi^2(k_1+w_1)(k_3+w_3) & 2\psi^2(k_3+w_3)(k_2+w_2) & 2\psi^2(k_3+w_3)^2 - 1 \end{pmatrix}.$$

As we have completely characterised the matrix K, we focus on matrix B in (3.20), which is a rotation that fixes $\mathbf{w}$. As above, we know 3 parameters out of 4 from

$$\begin{pmatrix} b \\ c \\ d \end{pmatrix} = \frac{1}{||w||} \begin{pmatrix} w_1(t) \\ w_2(t) \\ w_3(t) \end{pmatrix},$$

but this time we do not know the angle the moving frame rotates around this axis, i.e. we need to find the group parameter a. This information is contained in the curvature matrix $\rho_t \rho^{-1}$. In the following, our aim is to simplify the expressions contained in $\mathscr{Q}^t$ in terms of well–known geometric quantities. We begin computing the curvature matrix of the adjoint representation of the frame.

$$\mathscr{Q}^t = Ad(\rho)_t Ad(\rho)^{-1} = 2 \begin{pmatrix} 0 & \kappa_3 & -\kappa_2 \\ -\kappa_3 & 0 & \kappa_1 \\ \kappa_2 & -\kappa_1 & 0 \end{pmatrix}. \tag{3.23}$$

To obtain the above expression we simplified with respect to $I_1^{u_1} = \mathrm{Re}(\kappa_u) = 0$. On the other hand, we have, using (3.20)

$$\begin{aligned} \mathscr{Q}^t = Ad(\rho)_t Ad(\rho)^{-1} &= (B_t K + B K_t) K^{-1} B^{-1} \\ &= B_t B^{-1} + B K_t K^{-1} B^{-1}. \end{aligned} \tag{3.24}$$

The matrices appearing in (3.24) are very long and complicated functions of the quantities appearing in the conservation laws. Any attempt to find a solution for the group parameter a just via brute force is destined to fail. However, we show how it is possible to remarkably simplify these expressions in terms of very well–known and easy to handle geometric

quantities. If we define

$$W=\begin{pmatrix} 0 & w_3 & -w_2 \\ -w_3 & 0 & w_1 \\ w_2 & -w_1 & 0 \end{pmatrix}, \quad QW=\begin{pmatrix} 0 & q_3w_1-q_1w_2 & q_3w_1-q_2w_3 \\ q_1w_2-q_3w_1 & 0 & q_3w_2-q_2w_3 \\ q_2w_3-q_3w_1 & -q_3w_2+q_2w_3 & 0 \end{pmatrix},$$

then it can be proved that

$$K_tK^{-1}=\mathscr{Q}^t-\frac{\mathbf{q}\cdot(\mathbf{k}+\mathbf{w})}{||\mathbf{k}||^2+\mathbf{w}\cdot\mathbf{k}}W,$$

$$B_tB^{-1}=\frac{2}{||\mathbf{k}||^2\sqrt{1-a(t)^2}}\Bigg(||\mathbf{k}||^2\sqrt{1-a(t)^2}(1-a(t)^2)\mathscr{Q}^t$$

$$+\left(||\mathbf{k}||a'(t)-\mathbf{q}\cdot\mathbf{w}\sqrt{1-a(t)^2}(1-a(t)^2)\right)W-||\mathbf{k}||a(t)(1-a(t)^2)QW\Bigg).$$

This is done by direct computation. More details are given in the Appendix A. To express the term $BK_tK^{-1}B^{-1}$ in a simpler way, we proceed as follows. In general, if R is a rotation matrix and S is skew–symmetric, denote

$$S=\begin{pmatrix} 0 & S_3 & -S_2 \\ -S_3 & 0 & S_1 \\ S_2 & -S_1 & 0 \end{pmatrix}, \qquad \mathbf{s}=\begin{pmatrix} S_1 \\ S_2 \\ S_3 \end{pmatrix},$$

then it is well known that $RSR^T=RSR^{-1}$ can be expressed as follows:

$$RSR^{-1} = \begin{pmatrix} R_1^T \\ R_2^T \\ R_3^T \end{pmatrix} S \begin{pmatrix} R_1 & R_2 & R_3 \end{pmatrix}$$

$$= - \begin{pmatrix} R_1^T \\ R_2^T \\ R_3^T \end{pmatrix} \begin{pmatrix} \mathbf{s} \times R_1 & \mathbf{s} \times R_2 & \mathbf{s} \times R_3 \end{pmatrix}$$

$$= \begin{pmatrix} 0 & \mathbf{s} \cdot R_3 & -\mathbf{s} \cdot R_2 \\ -\mathbf{s} \cdot R_3 & 0 & \mathbf{s} \cdot R_1 \\ \mathbf{s} \cdot R_2 & -\mathbf{s} \cdot R_1 & 0 \end{pmatrix}.$$

Applying this computational trick to $BK_tK^{-1}B^{-1}$, and recalling that $B\mathbf{w} = \mathbf{w}$, we obtain

$$BK_tK^{-1}B^{-1} = QB - \frac{\mathbf{q} \cdot (\mathbf{k} + \mathbf{w})}{||\mathbf{k}||^2 + \mathbf{w} \cdot \mathbf{k}} W, \tag{3.25}$$

where we defined QB as

$$QB = \begin{pmatrix} 0 & \mathbf{q} \cdot B_3 & -\mathbf{q} \cdot B_2 \\ -\mathbf{q} \cdot B_3 & 0 & \mathbf{q} \cdot B_1 \\ \mathbf{q} \cdot B_2 & -\mathbf{q} \cdot B_1 & 0 \end{pmatrix},$$

and B_i is the $i-th$ column of B. If we substitute (3.23)–(3.25) into (3.24) we obtain

$$0 = B_t B^{-1} + B K_t K^{-1} B^{-1} - \mathscr{Q}^t \tag{3.26}$$

$$= \left(1-2a(t)^2\right)\mathscr{Q}^t + \left(\frac{||\mathbf{k}||a'(t) - \mathbf{q}\cdot\mathbf{w}\sqrt{1-a(t)^2}(1-a(t)^2)}{\frac{1}{2}||\mathbf{k}||^2\sqrt{1-a(t)^2}} - \frac{\mathbf{q}\cdot(\mathbf{k}+\mathbf{w})}{||\mathbf{k}||^2+\mathbf{w}\cdot\mathbf{k}}\right)W$$

$$- \frac{||\mathbf{k}||a(t)(1-a(t)^2)}{\frac{1}{2}||\mathbf{k}||^2\sqrt{1-a(t)^2}}QW + QB.$$

Now we note that the matrices $\mathscr{Q}^t$, W and QW only have 3 degrees of freedom. Assuming that $\mathscr{Q}^t$ and W are linearly independent, then we have that $(\mathscr{Q}^t, W, QW)$ is a basis for $\mathbb{R}^3$ as a subspace of $\mathbb{R}^9$. We can identify the triple $(\mathscr{Q}^t, W, QW)$ with $(\mathbf{q}, \mathbf{w}, \mathbf{q}\times\mathbf{w})$, and this defines an isomorphism. In $\mathbb{R}^3$ equation (3.26) becomes

$$\left(1-2a(t)^2\right)\mathbf{q} + \left(\frac{||\mathbf{k}||a'(t) - \mathbf{q}\cdot\mathbf{w}\sqrt{1-a(t)^2}(1-a(t)^2)}{\frac{1}{2}||\mathbf{k}||^2\sqrt{1-a(t)^2}} - \frac{\mathbf{q}\cdot(\mathbf{k}+\mathbf{w})}{||\mathbf{k}||^2+\mathbf{w}\cdot\mathbf{k}}\right)\mathbf{w} -$$

$$- \frac{||\mathbf{k}||a(t)(1-a(t)^2)}{\frac{1}{2}||\mathbf{k}||^2\sqrt{1-a(t)^2}}\mathbf{q}\times\mathbf{w} + B^T\mathbf{q} = 0. \tag{3.27}$$

Multiplying (3.27) everything on the left by $\mathbf{w}^T$, and recalling that $\mathbf{w}^T B^T = \mathbf{w}^T$ we obtain a scalar ODE for $a(t)$, the only parameter missing to reconstruct the moving frame $Ad(\rho)$:

$$0 = \left(1-2a(t)^2\right)\mathbf{w}^T\mathbf{q} + \left(\frac{||\mathbf{k}||a'(t) - \mathbf{q}\cdot\mathbf{w}\sqrt{1-a(t)^2}(1-a(t)^2)}{\frac{1}{2}||\mathbf{k}||^2\sqrt{1-a(t)^2}} - \frac{\mathbf{q}\cdot(\mathbf{k}+\mathbf{w})}{||\mathbf{k}||^2+\mathbf{w}\cdot\mathbf{k}}\right)||\mathbf{w}||^2 + \mathbf{w}^T\mathbf{q}$$

$$= 2(1-a(t)^2)\mathbf{q}\cdot\mathbf{w} + \left(\frac{||\mathbf{k}||a'(t) - \mathbf{q}\cdot\mathbf{w}\sqrt{1-a(t)^2}(1-a(t)^2)}{\frac{1}{2}||\mathbf{k}||^2\sqrt{1-a(t)^2}} - \frac{\mathbf{q}\cdot(\mathbf{k}+\mathbf{w})}{||\mathbf{k}||^2+\mathbf{w}\cdot\mathbf{k}}\right)||\mathbf{w}||^2.$$

Finally we get an ODE for $a(t)$ as

$$\begin{aligned}\frac{a'(t)}{\sqrt{1-a(t)^2}} &= \frac{||\mathbf{k}||}{2}\frac{\mathbf{q}\cdot(\mathbf{k}+\mathbf{w})}{||\mathbf{k}||^2+\mathbf{k}\cdot\mathbf{w}} \\ &= \frac{||\mathbf{k}||}{2}\frac{\mathbf{q}\cdot(\mathbf{k}+\mathbf{w})}{\mathbf{k}\cdot(\mathbf{k}+\mathbf{w})}.\end{aligned} \tag{3.28}$$

Equation (3.28) can be solved by quadratures, as

$$\int \frac{a'(t)}{\sqrt{1-a(t)^2}}\,dt = \arcsin\left(a(t)\right).$$

So the solution for $a(t)$ is

$$a(t) = \sin\left(\frac{||\mathbf{k}||}{2}\int_{t_0}^{t}\frac{\mathbf{q}\cdot(\mathbf{k}+\mathbf{w})}{\mathbf{k}\cdot(\mathbf{k}+\mathbf{w})}\,ds+\omega\right), \tag{3.29}$$

where ω is a constant of integration and t_0 the initial time. This ends the proof. □

Once we know $Ad(\rho)$ it is possible to derive the minimising curves $(u(t),v(t))$ using the normalisation equations (3.2). Recall we defined the frame as the solution of

$$\begin{cases}\rho\cdot u = 1\\ \rho\cdot v = 0.\end{cases} \tag{3.30}$$

Inverting the moving frame in (3.30), the minimisers of (3.13) are

$$\begin{pmatrix}u(t)\\ v(t)\end{pmatrix} = \rho^{-1}(t)\begin{pmatrix}u(t)\\ v(t)\end{pmatrix} = \begin{pmatrix}a-\mathrm{i}b\\ c-\mathrm{i}d\end{pmatrix},$$

where a, b, c, and d are the moving frame's parameters, namely

$$\rho = \begin{pmatrix} a & b \\ c & d \end{pmatrix}.$$

Remark 3.2.2. Equation (3.29) holds when $\mathbf{k} \neq -\mathbf{w}$. If this is not the case, we can take any vector orthogonal to $\mathbf{k}$ as axis of rotation for K. In this way, the matrix K is independent of t, hence:

$$Ad(\rho)_t Ad(\rho)^{-1} = B_t B^{-1} + B K_t K^{-1} B^{-1}$$

$$= B_t B^{-1}.$$

as K_t is the null matrix. This gives a singularity–free equation for $a(t)$ as

$$a(t) = \sin\left(\frac{||\mathbf{w}||}{2}\mathbf{q}\cdot\mathbf{w}\right).$$

3.3 Numerical examples

To see how the theory presented in this chapter works in practice we are going to give some explicit examples of how to solve a variational problem in this setting. In the following we will study four simple Lagrangians that are invariant under action (3.1). The Maple code used to generate the examples in this section can be found in Appendix A.

3.3.1 Example 1

The first example we will consider is

$$\mathscr{L} = \int \kappa_1^2(t)\, dt. \tag{3.31}$$

To find κ_1 in terms of the original variable t, we need to solve the Euler–Lagrange equations (3.14), that in this case greatly simplify to

$$\kappa_{1,t} = 0.$$

The solution is simply κ_1 being constant and κ_2, κ_3 are free. We set them to zero. The vector of invariants is

$$\mathbf{w} = \begin{pmatrix} \mathsf{E}^{\kappa_1}(L) \\ -\mathsf{E}^{\kappa_2}(L) \\ \mathsf{E}^{\kappa_3}(L) \end{pmatrix} = \begin{pmatrix} 2\kappa_1 \\ 0 \\ 0 \end{pmatrix},$$

while as a vector of constant $\mathbf{k}$ we can choose any vector whose norm is $2|\kappa_1|$. We choose

$$\mathbf{k} = \begin{pmatrix} \kappa_1 \\ 0 \\ \sqrt{3}\kappa_1 \end{pmatrix},$$

so we are also guaranteed that $\mathbf{w} + \mathbf{k} \neq 0$. The last vector we need is $\mathbf{q}$, which contains the components of the curvature matrix $\mathscr{Q}^t$. Recall it takes the form

$$\mathbf{q} = 2 \begin{pmatrix} \kappa_1 \\ \kappa_2 \\ \kappa_3 \end{pmatrix}.$$

Using (3.18) to find $Ad(\rho)$, the first step is to find the matrix K, that rotates $\mathbf{k}$ onto $\mathbf{w}$. In this case we obtain

$$K = \begin{pmatrix} \frac{1}{2} & 0 & \frac{\sqrt{3}}{2} \\ 0 & -1 & 0 \\ \frac{\sqrt{3}}{2} & 0 & -\frac{1}{2} \end{pmatrix}.$$

Then we solve the ODE for the rotation angle of B, which is the matrix that fixes $\mathbf{w}$. As we saw in (3.28), this is in general depending on $\mathbf{k}, \mathbf{w}$ and $\mathbf{q}$. In this case the ODE reduces to

$$\frac{a(t)'}{\sqrt{1-a(t)^2}} + \kappa_1 = 0.$$

If we take the initial condition $a(0) = \frac{1}{2}$, the solution is

$$a(t) = -\sin(t\kappa_1).$$

Now we have everything to express B as

$$B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2\sin(t\kappa_1)^2 - 1 & -\sqrt{\sin(t\kappa_1)\frac{4\left(1-\sin(t\kappa_1)^2\right)}{\kappa_1^2}}\kappa_1 \\ 0 & \sin(t\kappa_1)\sqrt{\frac{4\left(1-\sin(t\kappa_1)^2\right)}{\kappa_1^2}}\kappa_1 & 2\sin(t\kappa_1)^2 - 1 \end{pmatrix}.$$

The adjoint representation of the moving frame is then given by $Ad(\rho) = BK$.

In order to find the couple of solution curves $u(t), v(t)$, we need the frame in the standard representation of $SU(2)$

$$\begin{pmatrix} a+\mathrm{i}b & c+\mathrm{i}d \\ -c+\mathrm{i}d & a-\mathrm{i}b \end{pmatrix}.$$

Given a Cayley map A, with $a \neq 0$, it is possible to pass to $SU(2)$ via

$$a = \pm\sqrt{\frac{(Tr(A)+1)}{4}},$$

$$b = -\frac{1}{4a}(A_{23} - A_{32}),$$

$$c = -\frac{1}{4a}(A_{31} - A_{13}),$$

$$d = -\frac{1}{4a}(A_{12} - A_{21}).$$

If $a = 0$, the parameters (b,c,d) are given from the axis of rotation, as we saw above. Due to the choice of a sign in a, we have that two elements ρ and $-\rho$ are mapped to the same image by the Cayley map. After having found the corresponding element of $SU(2)$, recall that the solution curves are given as

$$\begin{pmatrix} u \\ v \end{pmatrix} = \rho^{-1} \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

where ρ^{-1} is the inverse of ρ in $SU(2)$. Choosing $\kappa_1 = 2$, a pair of minimisers of (3.31) is

$$u = a - ib = \frac{-\frac{3\mathrm{i}}{2}(\sqrt{-(\sin(t)-1)(\sin(t)+1)}\sin(t) - \sin(t)^2\mathrm{i} + \mathrm{i})}{\sqrt{-3(\sin(t)-1)(\sin(t)+1)}},$$

$$v = c - id = \frac{-\frac{\mathrm{i}}{2}\sqrt{3}(\sqrt{-(\sin(t)-1)(\sin(t)+1)}\sin(t) - \sin(t)^2\mathrm{i} + \mathrm{i})}{\sqrt{-3(\sin(t)-1)(\sin(t)+1)}}.$$

Both $u(t)$ and $v(t)$ are periodic functions, so the plots below do not change if we take a bigger range for t. In this specific case, the range was $t \in [-6,6]$. The plot of $u(t)$ is given in Figure 3.1. A plot of $v(t)$ is given in Figure 3.2.

Remark 3.3.1. The two different colours in Figure 3.1 and Figure 3.2 correspond to the choice of ρ and $-\rho$ to compute the solution. This applies for all the examples in this section.

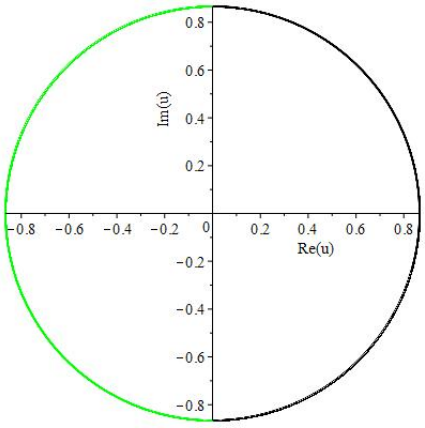


Fig. 3.1 A plot of $u(t)$, with $t \in [-6,6]$

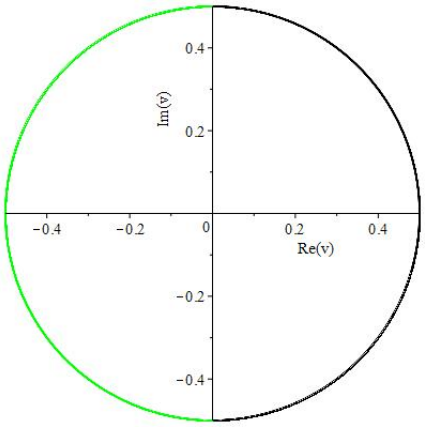


Fig. 3.2 A plot of $v(t)$, with $t \in [-6,6]$

3.3.2 Example 2

In this example we take the Lagrangian

$$\mathscr{L} = \int \kappa_{1,t}^2 \, dt.$$

The Euler–Lagrange equations also simplify in this case and we are left with the single scalar ODE

$$\kappa_{1,ttt} = 0.$$

the general integral of which is $\kappa_1(t) = ft^2 + gt + h$, where f, g, h are constants of integration. For this example, the values of (f, g, h) have been set to $(-1, 3, 2)$. The vector of invariants in this case is

$$\mathbf{w} = \begin{pmatrix} -2\kappa_{1,tt} \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 4 \\ 0 \\ 0 \end{pmatrix}.$$

The vector of constants can be chosen within the set $\left\{\mathbf{t} \in \mathbb{R}^3 \middle| \, ||t||^2 = 16\right\}$. We take

$$\mathbf{k} = \begin{pmatrix} -3 \\ 2 \\ -\sqrt{3} \end{pmatrix}.$$

so we will always have $\mathbf{w} + \mathbf{k} \neq 0$ for all $t \in I \subset \mathbb{R}$. The ODE for $a(t)$ is

$$\frac{a'(t)}{\sqrt{1 - a(t)^2}} = -t^2 + 3t + 2.$$

Taking the initial condition $a(0) = \frac{1}{2}$, we get the solution

$$a(t) = -\sin\left(-\frac{1}{3}t^3 - \frac{3}{2}t^2 - 2t\right).$$

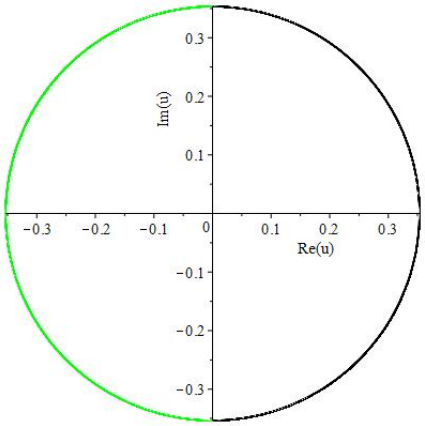


Fig. 3.3 A plot of $u(t)$, with $t \in [0.001, 3.15]$.

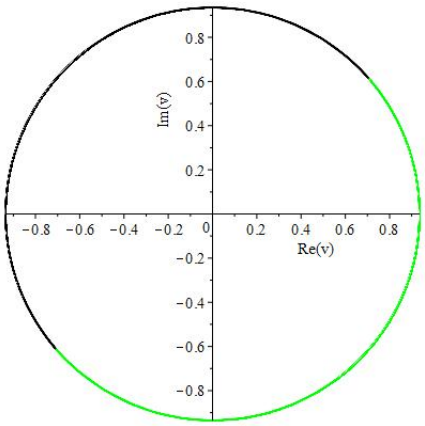


Fig. 3.4 A plot of $v(t)$, with $t \in [0.001, 3.15]$.

and we do not report B as it does not fit the page. The minimisers $u(t)$ and $v(t)$ are rational functions of $\cos(t)$ and $\sin(t)$. In this examples, choosing the positive a, we obtain the plot for $u(t)$ displayed in Figure 3.3. while a plot for $v(t)$ can be seen in Figure 3.4. The range of the independent variable t used to draw the plots was $t \in [-2, 2]$.

3.3.3 Example 3

Consider now the Lagrangian

$$\mathscr{L} = \int \left(\kappa_1^2(t) + \kappa_2^2(t)\right) dt.$$

We find the generating differential invariants solving the Euler–Lagrange equations, that now are a system of coupled of ODEs

$$\begin{cases} \frac{d}{dt}\kappa_1(t) = 2\kappa_3(t)\kappa_2(t) \\ \frac{d}{dt}\kappa_2(t) = -2\kappa_3(t)\kappa_1(t). \end{cases} \tag{3.32}$$

If we choose as initial conditions

$$\begin{cases} \kappa_1(0) = 1 \\ \kappa_2(0) = 0 \\ \kappa_3(0) = 0, \end{cases}$$

equations (3.32) are solved by

$$\kappa_1(t) = \cos\left(2\int_0^t \kappa_3(t_1)\, dt_1\right),$$

$$\kappa_2(t) = -\sin\left(2\int_0^t \kappa_3(t_1)\, dt_1\right),$$

where κ_3 is free. The vector of invariants now is

$$\mathbf{w} = \begin{pmatrix} 2\kappa_1 \\ -2\kappa_2 \\ 0 \end{pmatrix},$$

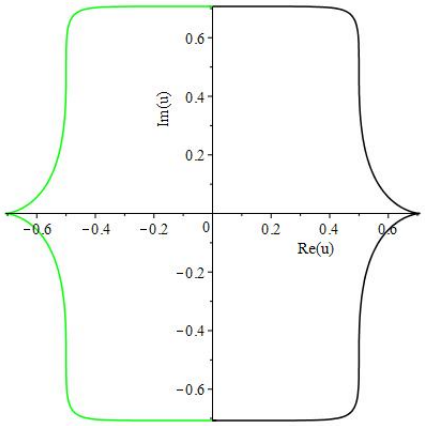


Fig. 3.5 A plot of $u(t)$, with $t \in [0.001, 3.15]$.

which has norm equal to 2 fo every t. So the vector of constants can be any vector with the above norm, and we choose

$$\mathbf{k} = \begin{pmatrix} 0 \\ 0 \\ 2 \end{pmatrix}.$$

In this setting, both the matrix K and B are quite complicated and it would be hard to fit them in a readable way on this page. Assume we computed them and recall they both depend on t, while in the previous examples this was the case only for B. It should be remarked that the solution of this problem is influenced by the choice of the third generating differential invariant, κ_3. In the following plots, we chose the simple case where it is constant and we set its value to $\kappa_3 = 2$. A plot for $u(t)$ is given in Figure 3.5. A plot of $v(t)$ is shown in Figure 3.6. The range for the independent variable t used to draw the two plots was $t \in [0.001, 3.15]$. It was chosen to avoid singularities in the solution, which correspond the the zeroes of $a(t)$. This can be seen from the plot in Appendix to Chapter 3, Example 3.

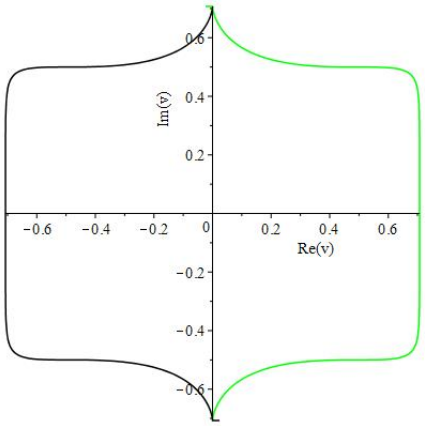


Fig. 3.6 A plot of $v(t)$, with $t \in [0.001, 3.15]$.

3.3.4 Example 4

In this example we consider the Lagrangian

$$\mathscr{L} = \int \left(\kappa_1(t)^2 + \kappa_2(t)^2 + \kappa_3(t)^2\right) dt,$$

where all three generating differential invariants appear at the same time. The Euler–Lagrange equations for this case simplify to

$$\begin{cases} \frac{d}{dt}\kappa_1 = 0 \\ \frac{d}{dt}\kappa_2 = 0 \\ \frac{d}{dt}\kappa_3 = 0. \end{cases}$$

The vector of invariants is

$$\mathbf{w} = 2\begin{pmatrix} \kappa_1 \\ -\kappa_2 \\ \kappa_3 \end{pmatrix}.$$

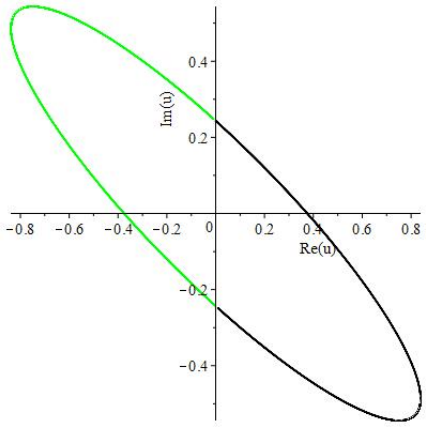


Fig. 3.7 A plot of $u(t)$, with $t \in [-13, 13]$.

Taking $\kappa_1 = -1$, $\kappa_2 = 2$ and $\kappa_3 = 1$, we choose a vector of constant given by

$$\mathbf{k} = 2 \begin{pmatrix} 0 \\ 0 \\ \sqrt{6} \end{pmatrix}.$$

Also in this case the matrices K and B do not fit easily this page, so we will just show the plots of the solutions. The minimisers $u(t)$ and $v(t)$ are periodic functions, and the range used to draw the plots below was $t \in [-13, 13]$. The plot of $u(t)$ is given in Figure 3.7 and a plot of $v(t)$ is shown in Figure 3.8.

3.4 The two–dimensional case

We focus now on the case where the variational problem is defined by a 2D Lagrangian. Namely, we will be looking for surfaces $(u(s,t), v(s,t))$, rather than curves, that extremise the Lagrangian.

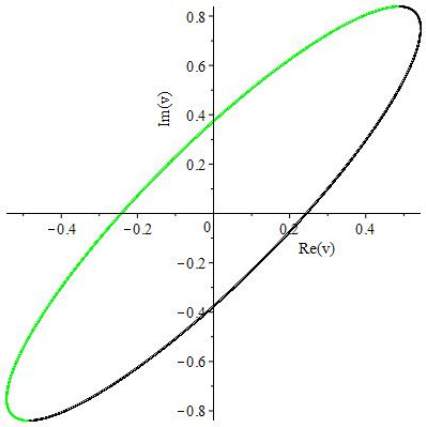


Fig. 3.8 A plot of $v(t)$, with $t \in [-13, 13]$.

Consider the Lagrangian

$$\mathscr{L} = \int_D L(s,t,u(s,t),v(s,t),v_s(s,t),\ldots)\,ds\,dt \tag{3.33}$$

where L depends on a finite number of partial derivatives of u and v and D is a simply connected open subset of $\mathbb{R}^2$. Consider the action of $SU(2)$ on pairs of complex surfaces defined by

$$\begin{pmatrix} \alpha & \beta \\ -\bar{\beta} & \bar{\alpha} \end{pmatrix} \cdot \begin{pmatrix} u_1(s,t) + iu_2(s,t) \\ v_1(s,t) + iv_2(s,t) \end{pmatrix} = \begin{pmatrix} \alpha(u_1 + iu_2) + \beta(v_1 + iv_2) \\ -\bar{\beta}(u_1 + iu_2) + \bar{\alpha}(v_1 + iv_2) \end{pmatrix}. \tag{3.34}$$

We will assume that the independent variables s,t are invariant. A frame for the action (3.34) can be defined taking the normalisation equations

$$\begin{cases} \rho \cdot u = 1 \\ \rho \cdot v = 0, \end{cases} \tag{3.35}$$

and applying the condition $u\bar{u}+v\bar{v}=1$ we obtain

$$\rho = \begin{pmatrix} \bar{u} & \bar{v} \\ -v & u \end{pmatrix},$$

where we emphasise that now $u=u(s,t)$ and $v=v(s,t)$ are complex surfaces, as opposed to (3.2), where u and v were complex curves. The set of generating differential invariants is given by the generating differential invariants from the one–dimensional case plus the ones arising from the introduction of the new variable t. Recall that in the one–dimensional case we had that the generating differential invariants were κ_1, κ_2, and κ_3. Introducing the variable t, we have that also $\kappa_4=\rho\cdot\frac{d}{dt}u_2$, $\kappa_5=\rho\cdot\frac{d}{dt}v_1$, and $\kappa_6=\rho\cdot\frac{d}{dt}v_2$ are generating differential invariants. Given the normalisation equations (3.35) and the action (3.34), the curvature matrices with respect to s and t are

$$\mathscr{Q}^s = \begin{pmatrix} -i\kappa_1 & \kappa_2-i\kappa_3 \\ -\kappa_2-i\kappa_3 & i\kappa_1 \end{pmatrix} \qquad \mathscr{Q}^t = \begin{pmatrix} -i\kappa_4 & \kappa_5-i\kappa_6 \\ -\kappa_5-i\kappa_6 & i\kappa_4 \end{pmatrix}.$$

The curvature matrices above satisfy

$$\frac{d}{ds}\mathscr{Q}^t - \frac{d}{dt}\mathscr{Q}^s = [\mathscr{Q}^s, \mathscr{Q}^t]. \tag{3.36}$$

Equation (3.36) allows us to find the syzygies equations between the invariants $\mathbf{I}_s = (\kappa_1, \kappa_2, \kappa_3)^T$ and $\mathbf{I}_t = (\kappa_4, \kappa_5, \kappa_6)^T$. Namely, we have

$$\frac{d}{dt}\mathbf{I}_s = \begin{pmatrix} \frac{d}{ds} & 2\kappa_3 & -2\kappa_2 \\ -2\kappa_3 & \frac{d}{ds} & 2\kappa_1 \\ 2\kappa_2 & 2\kappa_1 & \frac{d}{ds} \end{pmatrix} \mathbf{I}_t$$

$$= \mathscr{H}\mathbf{I}_t. \tag{3.37}$$

Equation (3.37) is a system of 3 PDEs that has to be solved in conjunction with the invariantised Euler–Lagrange equations (that we still have to derive), in order to compute the six generating differential invariants as functions of s and t. To obtain the invariantised Euler–Lagrange equations, we let u and v depend also on a dummy independent and invariant variable τ. With the introduction of the new variable, there are 3 new differential invariants that we will denote as $\mathbf{I}_\tau = (\sigma_1, \sigma_2, \sigma_3)$. There are also new syzygies appearing, relating $\mathbf{I}_s$ and $\mathbf{I}_t$ to $\mathbf{I}_\tau$. They are obtained using (3.36) replacing first s and then t with τ. The syzygies are

$$\frac{\partial}{\partial\tau}\mathbf{I}_s = \begin{pmatrix} \frac{\partial}{\partial s} & 2\kappa_3 & -2\kappa_2 \\ -2\kappa_3 & \frac{\partial}{\partial s} & 2\kappa_1 \\ 2\kappa_2 & 2\kappa_1 & \frac{\partial}{\partial s} \end{pmatrix} \mathbf{I}_\tau$$

$$= \mathscr{H}\mathbf{I}_\tau,$$

and

$$\frac{d}{d\tau}\mathbf{I}_t = \begin{pmatrix} \frac{d}{ds} & 2\kappa_6 & -2\kappa_5 \\ -2\kappa_6 & \frac{d}{ds} & 2\kappa_4 \\ 2\kappa_5 & 2\kappa_4 & \frac{d}{ds} \end{pmatrix} \mathbf{I}_\tau$$

$$= \mathscr{J}\,\mathbf{I}_\tau.$$

After rewriting the Lagrangian (3.33) in terms of the generating differential invariants and their derivatives, denote with $\boldsymbol{\kappa} = (\kappa_1, \cdots, \kappa_6)$ and consider a Lagrangian

$$\mathscr{L} = \int_D L(s,t,\boldsymbol{\kappa}, \frac{\partial}{\partial s}\boldsymbol{\kappa}, \frac{\partial}{\partial t}\boldsymbol{\kappa}, \ldots)\, ds\, dt, \tag{3.38}$$

where L depends on a finite number of derivatives of the generating differential invariants. Performing the variation with respect to the dummy variable τ and using (2.3.1), we have

$$\begin{aligned} \frac{d}{d\tau}\mathscr{L} &= \int \frac{d}{d\tau} L(s,t,\boldsymbol{\kappa}, \frac{d}{ds}\boldsymbol{\kappa}, \ldots)\, ds\, dt \\ &= \int \left(\sum_i \mathscr{H}_i^* E^{\kappa_i}(L) + \mathscr{J}_i^* E^{\kappa_{i+3}}(L) \right) \mathbf{I}_\tau + \mathrm{BT}\, ds\, dt, \end{aligned}$$

where BT stands for the boundary terms that arise from the integration by parts and from taking the adjoint of $\mathscr{H}$ and $\mathscr{J}$. The Euler–Lagrange equations relative to the Lagrangian (3.38) can be written in full as

$$\begin{cases} -\frac{d}{ds}E^{\kappa_1}(L) - 2\kappa_3 E^{\kappa_2}(L) + 2\kappa_2 E^{\kappa_3}(L) - \frac{d}{dt}E^{\kappa_4}(L) - 2\kappa_6 E^{\kappa_5}(L) + 2\kappa_5 E^{\kappa_6}(L) = 0 \\ 2\kappa_3 E^{\kappa_1}(L) - \frac{d}{ds}E^{\kappa_2}(L) - 2\kappa_1 E^{\kappa_3}(L) + 2\kappa_6 E^{\kappa_4}(L) - \frac{d}{dt}E^{\kappa_5}(L) - 2\kappa_4 E^{\kappa_6}(L) = 0 \\ -2\kappa_2 E^{\kappa_1}(L) + 2\kappa_1 E^{\kappa_2}(L) - \frac{d}{ds}E^{\kappa_3}(L) - 2\kappa_5 E^{\kappa_4}(L) + 2\kappa_4 E^{\kappa_5}(L) - \frac{d}{dt}E^{\kappa_6}(L) = 0. \end{cases} \tag{3.39}$$

A set of generating differential invariants is a solution to the system given by (3.37), (3.39) and suitable boundary conditions. We will see in the next chapter how can we use such sets to numerically compute the frame at every point in the solution's domain. Once the frame has been computed, a solution to the variational problem defined by the Lagrangian (3.38) can be obtained inverting the normalisation equations (3.35) as done in the one–dimensional case, namely

$$\begin{pmatrix} u(s,t) \\ v(s,t) \end{pmatrix} = \rho^{-1} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \bar{\alpha} \\ \bar{\beta} \end{pmatrix},$$

where ρ^{-1} is the inverse of ρ in $SU(2)$.

In the previous sections of this chapter, we managed to compute solutions of invariant variational problems using the fact that the conservation laws can be written as

$$\frac{d}{dt}\left(Ad(\rho)^{-1}\mathbf{v}\right) = 0, \tag{3.40}$$

where $\mathbf{v}$ is a vector of invariants and $Ad(\rho)$ is the adjoint representation of the moving frame. Expression (3.40) allowed us to recover the moving frame ρ, after some simplification dictated by the geometrical setting, using only quadratures. In the two–dimensional case, this is not possible anymore, as the conservation laws become

$$\frac{d}{ds}\left(Ad(\rho)^{-1}\mathbf{z}\right) + \frac{d}{dt}\left(Ad(\rho)^{-1}\mathbf{w}\right) = 0, \tag{3.41}$$

where $\mathbf{w}$ and $\mathbf{z}$ are vector of invariants. Equation (3.41) does not lend itself to be integrated anymore as we did in the one–dimensional case. Hence, we need to proceed in a different way in order to find a solution to the variational problem. In the next chapter, we will show how to use the invariantised Euler–Lagrange equations to solve invariant variational problems of dimension two and higher.

4. Solutions to higher dimensional invariant variational problems

4.1 Introduction

In Chapter 3 we have shown how to use the moving frame theory to solve one–dimensional variational problems that are invariant under a linear action of $SU(2)$. Lagrangians invariant under actions of $SL(2), SE(2)$, and $SE(3)$ have been solved exactly using analogous methods in [22; 23; 24; 42]. For a one–dimensional problem, Noether's laws yield algebraic equations for the frame and these can be used to ease the integration problem for the minimising solution. However, for higher dimensional problems, the laws do not in general lend themselves to finding exact solutions.

In this chapter we reduce the problem of finding the minimiser, to that of solving the Euler–Lagrange equations for the invariants and then solving the compatible system of differential equations,

$$\begin{cases} \dfrac{\partial}{\partial x_i}\rho = \mathscr{Q}^i\rho, \qquad i = 1,\ldots,p \\ \rho(\mathbf{x}_0) = \rho_0 \end{cases} \tag{4.1}$$

for ρ, where G is the Lie group, $\rho : M \to G$ is the moving frame, $\mathfrak{g}$ is the Lie algebra of G, and $\mathscr{Q}^i : M \to \mathfrak{g}$ are the curvature matrices. Once the frame has been found in the original

variables, the minimiser is given by

$$u_K^\alpha = \rho(\mathbf{x})^{-1} \cdot I(u_K^\alpha), \tag{4.2}$$

where $\rho(\mathbf{x})^{-1}$ is the group inverse of $\rho(\mathbf{x})$ and the action is that appropriate for the jet bundle coordinate, u_K^α. The content of this chapter has been taken from [56]. The original contributions can be found in Section 4.2.3, Section 4.2.4 and Section 4.3.

We begin providing a summary of the main results concerning the Magnus expansion, on which an important class of Lie group integrators is based, for a matrix ODE system evolving on a Lie group (see [2; 9; 10; 35] for surveys on the topic, [17] for numerical software).

We then present the main result: that the Magnus expansion solution may be used to solve the compatible differential system (4.1) in the case $p = 2$, at least to order 5, in the neighbourhood of a point where the components of the curvature matrices are regular. We do this by showing that applying the expansion sequentially, yields a result which is independent of the order in which the two differential equations are solved, to order 5. This then implies directly, that for a set of p pairwise compatible equations of the form (4.1) the Magnus expansion can be applied sequentially, with respect to each independent variable, yielding a well defined result, at least to order 5. We chose to prove this result to order 5 for two reasons: first of all, the computations' complexity grows quickly with the order; second, most of the numerical methods implemented in Diffman, [17] are of order less than 5. However, it seems reasonable to think that the methods presented in this Chapter can be used to prove the result to a higher order. We then demonstrate in a range of examples, that an efficient implementation, [17], mirrors this result, to the relevant order of approximation.

We conclude with a conjecture, that compatibility of the system (4.1) implies that the Magnus expansion may be used sequentially to obtain a well-defined result, to all orders, in the neighbourhood of a point where the components of the curvature matrices equal their Taylor series.

4.2 Lie group integrators

We have seen in Chapter 2 how the curvature matrices are defined in terms of the moving frames. Multiplying on the right both sides of (2.12) by ρ we obtain a system of first–order PDEs for the frame, namely

$$\begin{cases} \dfrac{\partial}{\partial x_i}\rho = \mathscr{Q}^i\rho, & i = 1,\dots,p \\ \rho(\mathbf{x}_0) = \rho_0 \end{cases}$$

where ρ_0 is an initial condition that can be expressed in terms of the initial condition on the minimiser, given in the definition of the variational problem at hand.

Standing Assumption. We assume throughout the chapter that the independent variables are invariant under the Lie group action,

$$g\cdot(\mathbf{x},\mathbf{u}) = (\mathbf{x},\widetilde{\mathbf{u}}) = (x_1,..,x_p,\widetilde{u^1},..,\widetilde{u^q})$$

We have then that for all $K = (k_1,\dots,k_p)$,

$$g\cdot u^\alpha_K = \frac{\partial^{|K|}}{\partial x^K}\widetilde{u^\alpha} = \frac{\partial^{|K|}}{\partial^{k_1}x_1\cdots\partial^{k_p}x_p}\widetilde{u^\alpha}.$$

We discuss how to relax this assumption slightly, at the end of this section.

Definition 4.2.1. A system of PDEs is compatible if all the equations share a common solution.

Given these assumptions, the invariant differential operators are standard, commutative differential operators, hence compatibility for system (4.1) is guaranteed by (2.13), namely

$$\frac{\partial}{\partial x_i}\mathscr{Q}^j - \frac{\partial}{\partial x_j}\mathscr{Q}^i = [\mathscr{Q}^i,\mathscr{Q}^j], \quad (i,j)\in\{1,2,..,p\}^2, \quad i\neq j \tag{4.3}$$

Remark 4.2.2. Our results extend readily to the case where the action on the independent variables is translation, so that

$$g\cdot(\mathbf{x},\mathbf{u})=(\mathbf{x}+\varepsilon,\widetilde{\mathbf{u}})=(x_1+\varepsilon_1,..,x_p+\varepsilon_p,\widetilde{u^1},..,\widetilde{u^q})$$

in which case the operators $\dfrac{\partial}{\partial x_i}$ are still invariant, and the equations (4.3) still hold. The only real difference is that the Lagrangian can not depend on the independent variables. The moving frame for the group parameter ε is taken to be $\varepsilon=\mathbf{c}-\mathbf{x}$ where $\mathbf{c}$ is constant, so that $I(x_i)=c_i$. We note that the choice of the c_i can lead to more or less complicated expressions, and are often taken to be either zero or unity. More details on this can be found in [22].

4.2.1 Matrix ODEs

We have seen how the moving frame ρ is the solution of the compatible system, rewritten here for the two dimensional case,

$$\begin{cases}\dfrac{\partial}{\partial x}\rho=\mathscr{Q}^x\rho & (4.4)\\[2ex] \dfrac{\partial}{\partial y}\rho=\mathscr{Q}^y\rho & (4.5)\\[2ex] \rho(x_0,y_0)=\rho_0\end{cases}$$

where the compatibility condition $\frac{\partial}{\partial x}\mathscr{Q}^y-\frac{\partial}{\partial y}\mathscr{Q}^x-[\mathscr{Q}^x,\mathscr{Q}^y]=0$ is guaranteed to hold.

Equations (4.4)–(4.5) are linear coupled PDEs which evolve on a Lie group. However, each equation contains only a derivative in a single direction. Hence, it is possible to solve each of them numerically using numerical schemes developed to solve ODEs on Lie groups: the so–called "Lie group integrators". In the following subsection we review the main facts concerning the theory of Lie group integrators. In–depth surveys on this can be found in [2; 9; 35].

As our focus is on matrix Lie group actions, we will assume we are dealing with matrix Lie groups. Moreover, the matrices $\mathscr{Q}^x$ and $\mathscr{Q}^y$ depend only on the generating differential invariants of the action and not on the moving frame itself. This means that equations (4.4)–(4.5) form a system of linear PDEs.

Suppose we have a matrix Lie group G with Lie algebra $\mathfrak{g}$. We consider the initial value problem on G,

$$\begin{cases} Y'(t) = A(t)Y(t), & t \geq 0 \\ Y(0) = Y_0 \end{cases} \tag{4.6}$$

where $Y \in G$ and $A : \mathbb{R} \to \mathfrak{g}$. The convergence of the Magnus expansion has been studied also in [3]. To solve the initial value problem (4.6) it is necessary to extend the exponential function to Lie algebras.

Definition 4.2.3 ([28])**.** If $\mathfrak{g}$ is the Lie algebra of a Lie group G, then the exponential map is defined as

$$\exp : \mathfrak{g} \to G, \qquad A \mapsto \sum_{k=0}^{\infty} \frac{A^k}{k!}$$

It can be shown that the series $\exp(A)$ indeed maps into G. The exponential is one–to–one only in a neighbourhood of $0 \in \mathfrak{g}$. However, it is not globally one–to–one, nor surjective. When it does exist, the inverse function is known as the logarithm and denoted as log.

Definition 4.2.4 ([35])**.** Suppose $A : \mathbb{R} \to \mathfrak{g}$ is differentiable. Then the differential of $\exp(A(t))$, denoted by dexp, is given by

$$\frac{\mathrm{d}}{\mathrm{d}t} \exp(A(t)) = \mathrm{dexp}_{A(t)}(A'(t)) \exp(A(t))$$

Definition 4.2.5 ([35])**.** Given $A \in \mathfrak{g}$, the *adjoint map* ad_A is defined as

$$\mathrm{ad}_A : \mathfrak{g} \to \mathfrak{g}, \qquad Y \mapsto [A, Y].$$

It can be proved [55], that dexp_A is an analytic function of ad_A, namely

$$\mathrm{dexp}_A(B) = \sum_{i=0}^{\infty} \frac{\mathrm{ad}_A^i B}{(i+1)!} = \left.\frac{\exp(z)-1}{z}\right|_{z=\mathrm{ad}_A} (B) \tag{4.7}$$

and we used the notation

$$\mathrm{ad}_A^i B = \underbrace{[A,[A,[A,..,[A}_{i-1\,\text{times}},[A,B]]]] \quad \text{for}\, i \in \mathbb{N}$$

We follow [35] and read the ratio in the second equality of (4.7) in the sense of the power series

$$\frac{\exp(z)-1}{z} = \sum_{i=0}^{\infty} \frac{z^j}{(j+1)!}$$

where z is replaced by ad_A. As dexp is an analytic function, we can invert it and write

$$\mathrm{dexp}_A^{-1} = \left.\frac{z}{\exp(z)-1}\right|_{z=\mathrm{ad}_A}$$

This last equation should also be read as a power series, recalling that

$$\frac{z}{\exp(z)-1} = \sum_{i=0}^{\infty} \frac{B_i}{i!} z^i$$

where B_i is the ith–Bernoulli number [15, Eq. 24.2.1]. A sufficient condition for the radius of convergence of (4.8) to be strictly positive is given in Theorem 4.2.6. Hence

$$\mathrm{dexp}_A^{-1}(B) = \sum_{i=0}^{\infty} \frac{B_i}{i!} \mathrm{ad}_A^i(B) \tag{4.8}$$

We now state the fundamental result that lies behind the theory of the Lie group integrators.

Theorem 4.2.6 ([35], [45]). *Consider the initial value problem on G given in* (4.6) *and define*

$$T_{max} = \sup_{T} \left\{ \int_0^T ||A(\xi)||_2 \, d\xi < \pi \right\}$$

Then, for every $t_0 \in (0, T_{max})$*, the solution of* (4.6) *in* $[0, t_0]$ *is given by*

$$Y(t) = \exp(\Theta(t)) Y_0$$

and $\Theta(t) \in \mathfrak{g}$ *is the solution of*

$$\begin{cases} \Theta(t)' = \operatorname{dexp}^{-1}_{\Theta(t)}(A(t)) \\ \Theta(0) = 0 \end{cases} \tag{4.9}$$

4.2.2 The Magnus expansion

We are interested in using a class of numerical methods that goes under the name of "Magnus expansion methods" [36]. This is a particular case of the Runge–Kutta–Munthe–Kaas methods developed in [46; 47; 48; 49]. In order to solve (4.9), the method of Picard iteration is used, which relies on the concept of *uniformly Lipschitz continuous function* [27, p. 2]. We recall the following definitions.

Definition 4.2.7 ([27]). A function f: $\mathbb{R}^m \to \mathbb{R}^n$ is said to be *uniformly Lipschitz continuous* if there exists a constant $L \geq 0$, such that, for every $x, y \in \mathbb{R}^m$

$$||f(x) - f(y)||_{\mathbb{R}^n} \leq L||x - y||_{\mathbb{R}^m}$$

holds.

Definition 4.2.8. For the initial value problem

$$\begin{cases} y'(t) = f(t, y(t)) \\ y(t_0) = y_0 \end{cases} \tag{4.10}$$

the *Picard iteration* is defined as the sequence

$$\begin{cases} u^{[0]} = y_0 \\ u^{[m+1]} = y_0 + \displaystyle\int_{t_0}^{t} f(s, u^{[m]})\, ds \qquad m \geq 0 \end{cases} \tag{4.11}$$

The two definitions above play a central role in the Picard–Lindelöf theorem:

Theorem 4.2.9 (Picard–Lindelöf, [27])**.** *Consider the initial value problem given by* (4.10)*. If $f(t, y(t))$ is uniformly Lipschitz continuous in y and continuous in t, then there exists $\varepsilon > 0$ such that there exists a unique solution to* (4.10) *on the interval $[t_0 - \varepsilon, t_0 + \varepsilon]$. Further, this solution is the limit of the Picard iterations, i.e.*

$$y(t) = \lim_{m \to \infty} u^{[m]}(t), \tag{4.12}$$

where $u^{[m]}(t)$ is defined in (4.11)*.*

As seen in (4.8), the inverse of dexp can be written as a series involving powers of the ad operator. Applying the Picard iterations to (4.9) yields

$$\begin{cases} \Theta^{[0]} = 0 \\ \Theta^{[m+1]} = \displaystyle\int_0^t \operatorname{dexp}^{-1}_{\Theta^{[m]}(\xi)} A(\xi)\, d\xi = \sum_{i=0}^{\infty} \frac{B_i}{i!} \int_0^t \operatorname{ad}^i_{\Theta^{[m]}(\xi)} A(\xi)\, d\xi \end{cases}$$

for $m = 0, 1, 2, \ldots$

In our case, the matrix function $A(t)$ has no dependence on Y. Since it is assumed to be smooth and hence continuous, in t, Picard's theorem can be applied, to yield a unique local solution to (4.9), namely $\Theta(t) = \lim_{m\to\infty} \Theta^{[m]}(t)$. It can be seen [35], that it is possible to rearrange the terms in Θ as

$$\Theta(t) = \sum_{i=0}^{\infty} H_i(t) \tag{4.13}$$

where $H_i(t)$ comprises those terms involving precisely i commutators and $i+1$ integrals. The expression defined in (4.13) is called the *Magnus expansion*.

4.2.3 Magnus expansion and coupled systems of PDEs

We now restrict to the two dimensional case, for simplicity. We are interested in applying the theory of Lie group integrators based on the Magnus expansion to solve two–dimensional variational problems. Let us recall we want to solve system (4.4)–(4.5) in order to find the moving frame ρ. Equations (4.4)–(4.5) form a system of two linear matrix differential equations to be solved in a suitable domain of $\mathbb{R}^2$ and we want the solution to belong to the Lie group G at every point where it is defined. We also recall the compatibility condition (2.13) for (4.4)–(4.5) to have a solution. We denote this condition by R, that is,

$$R = \frac{\partial}{\partial x}\mathscr{Q}^y - \frac{\partial}{\partial y}\mathscr{Q}^x - [\mathscr{Q}^x, \mathscr{Q}^y] \tag{4.14}$$

which must be identically zero for the system to be compatible. We apply Lemma (4.2.6) to equations (4.4)–(4.5), obtaining the coupled system of differential equations,

$$\begin{cases} \dfrac{\partial}{\partial x}\Theta(x,y) = \operatorname{dexp}^{-1}_{\Theta(x,y)} \mathscr{Q}^x(x,y) & (4.15) \\ \dfrac{\partial}{\partial y}\Theta(x,y) = \operatorname{dexp}^{-1}_{\Theta(x,y)} \mathscr{Q}^y(x,y) & (4.16) \end{cases}$$

The method of Picard iterations is applied to each of the differential equations (4.15)–(4.16) to yield,

$$\begin{cases} \Theta^x_{[0]} = 0 \\ \Theta^y_{[0]} = 0 \\ \Theta^x_{[n+1]} = \sum_{i=0}^{\infty} \dfrac{B_i}{i!} \displaystyle\int_0^x \mathrm{ad}^i_{\Theta^x_{[n]}(\xi,y)} \mathscr{Q}^x(\xi,y)\, d\xi \\ \Theta^y_{[n+1]} = \sum_{i=0}^{\infty} \dfrac{B_i}{i!} \displaystyle\int_0^y \mathrm{ad}^i_{\Theta^y_{[n]}(x,\xi)} \mathscr{Q}^y(x,\xi)\, d\xi \end{cases}$$

for $n = 0, 1, 2, \ldots$, where the iterations of Θ^x and Θ^y solve the equation for (4.15) and (4.16) respectively. We use the superscripts x and y to denote the integrations in the x and y direction respectively. As in (4.13), we rearrange terms such that

$$\Theta^x(y) = \sum_{i=0}^{\infty} M^x_i(y)$$

$$\Theta^y(x) = \sum_{i=0}^{\infty} M^y_i(x)$$

where M^x_i, M^y_i comprise those terms containing exactly i commutators and $i+1$ integrals.

4.2.4 Magnus expansions commute up to order 5

We now show that the Magnus expansion, considered as an exact, albeit infinite series solution, yields a well defined integration method for a system of the form (4.4)–(4.5), in the neighbourhood of a point (x_0, y_0) for which the curvature matrices both have a Taylor series expansion. We consider the result obtained by sequential integration in the two different directions. We show, in fact, that the difference in the results obtained by changing the order of integration, can be expressed in terms of a differential operator acting on the compatibility condition, R, (4.14). The two different solutions are the same, then, provided $R = 0$. While we show the result only to order 5, it is clear that the calculations may be continued to any order, albeit they become increasingly complex.

Definition 4.2.10. If $Q \in \mathfrak{g}$ is a matrix Lie algebra element, then Q is of order n in h if

$$n = \inf\left\{ j \in \mathbb{Z} : \lim_{h\to+\infty} \frac{Q}{h^{j+1}} = 0 \right\}$$

In our calculations, we will make strong use of the Baker–Campbell–Hausdorff (BCH) formula which shows how two matrix exponentials may be multiplied to obtain a single matrix exponential. Although we will use a truncated BCH expansion up to order 5, a recursive formula to determine every term has been proved by Dynkin, [16].

Theorem 4.2.11. *[BCH formula, [16; 54]] If* $||X||_2 + ||Y||_2 < \log 2$*, then*

$$\log(\exp(X)\exp(Y)) = \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n} \sum_{\substack{r_1+s_1>0 \\ \vdots \\ r_n+s_n>0}} \frac{[X^{r_1}Y^{s_1}X^{r_2}Y^{s_2}\cdots X^{r_n}Y^{s_n}]}{\sum_{i=1}^{n}(r_i+s_i)\prod_{i=1}^{n} r_i!s_i!}$$

where

$$[X^{r_1}Y^{s_1}X^{r_2}Y^{s_2}\cdots X^{r_n}Y^{s_n}]$$

$$= [\underbrace{X,[X,\cdots[X}_{r_1},\underbrace{[Y,[Y,\cdots[Y}_{s_1},\cdots\underbrace{[X,[X,\cdots[X}_{r_n},\underbrace{[Y,[Y,\cdots Y]}_{s_n}]\cdots]$$

Theorem 4.2.12. *Let* (x_0, y_0) *be a point in the domain of the moving frame, for which the curvature matrices have a (local) Taylor series expansion. Then in a neighbourhood of this point, the Magnus expansion may be used sequentially, to yield a well–defined solution for the compatible system* (4.4)–(4.5)*, to order at least 5.*

Proof. Consider a rectangular neighbourhood of (x_0, y_0), given by $[x_0, x_0+h] \times [y_0, y_0+k]$, where $h, k \in \mathbb{R}$ are sufficiently small, that is, $[x_0, x_0+h] \times [y_0, y_0+k]$ lies within the domain of validity of the Taylor series of the curvature matrices. In order to have a well–defined solution, we need to prove that if we start from the initial datum $\rho_0 = \rho(x_0, y_0)$, then we obtain a unique expression for $\rho(x_0+h, y_0+k)$, regardless of the order of integration, that is, regardless of whether we integrate first with respect to x or with respect to y.

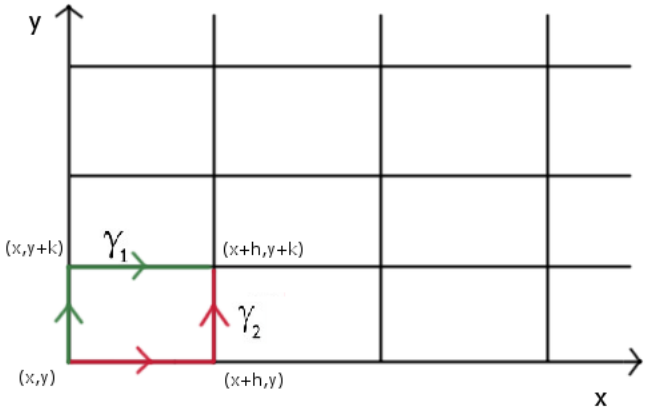


Fig. 4.1 The two different paths γ_1,γ_2.

Let us consider two paths, say γ_1 and γ_2, such that they both start at (x_0,y_0) and end at $(x_0+h,y_0+k)=(x_1,y_1)$, but γ_1 first goes to (x_0,y_1) and then to (x_1,y_1), while γ_2 travels first to (x_1,y_0) before going to (x_1,y_1) (see Figure 4.1). We compute the solution $\rho(x_1,y_1)$ along the two paths, and compare the two results. We call $\rho^{\gamma_1}(x_1,y_1)$ and $\rho^{\gamma_2}(x_1,y_1)$ the solution $\rho(x_1,y_1)$ obtained along γ_1 and γ_2 respectively. To make the calculations tractable, we will approximate the solutions ρ^{γ_1} and ρ^{γ_2} to order five.

Using Lemma (4.2.6) we compute $\rho^{\gamma_1}(x_1,y_1)$ and $\rho^{\gamma_2}(x_1,y_1)$ in two steps. First we obtain the solution of

$$\begin{cases} \dfrac{\partial}{\partial y}\rho^{\gamma_1} = \mathscr{Q}^y\rho^{\gamma_1} \\ \rho^{\gamma_1}(x_0,y_0)=\rho_0 \\ (x,y)\in\{x_0\}\times[y_0,y_1] \end{cases} \qquad \begin{cases} \dfrac{\partial}{\partial x}\rho^{\gamma_2} = \mathscr{Q}^x\rho^{\gamma_2} \\ \rho^{\gamma_2}(x_0,y_0)=\rho_0 \\ (x,y)\in[x_0,x_1]\times\{y_0\} \end{cases}$$

as

$$\rho^{\gamma_1}(x_0,y_1) = \exp(\Theta^y(x_0))\rho_0$$

$$\rho^{\gamma_2}(x_1,y_0) = \exp(\Theta^x(y_0))\rho_0$$

Then the following step is to solve the systems

$$\begin{cases} \dfrac{\partial}{\partial x}\rho^{\gamma_1} = \mathscr{D}^x\rho^{\gamma_1} \\ \rho^{\gamma_1}(x_0,y_1) = \exp(\Theta^y(x_0))\rho_0 \\ (x,y) \in [x_0,x_1]\times\{y_1\} \end{cases} \qquad \begin{cases} \dfrac{\partial}{\partial y}\rho^{\gamma_2} = \mathscr{D}^y\rho^{\gamma_2} \\ \rho^{\gamma_2}(x_1,y_0) = \exp(\Theta^x(y_0))\rho_0 \\ (x,y) \in \{x_1\}\times[y_0,y_1] \end{cases}$$

and we obtain the two solutions that we want to compare, namely

$$\rho^{\gamma_1}(x_1,y_1) = \exp(\Theta^x(y_1))\exp(\Theta^y(x_0))\rho_0$$

$$\rho^{\gamma_2}(x_1,y_1) = \exp(\Theta^y(x_1))\exp(\Theta^x(y_0))\rho_0$$

Therefore, we consider

$$\begin{aligned} \log(\rho^{\gamma_1}(x_1,y_1)\rho_0^{-1}) - \log(\rho^{\gamma_2}(x_1,y_1)\rho_0^{-1}) = \log(\exp(\Theta^x(y_1))\exp(\Theta^y(x_0))) \\ - \log(\exp(\Theta^y(x_1))\exp(\Theta^x(y_0))) \end{aligned} \tag{4.17}$$

We will show that the right hand side of (4.17) is zero up to order 5 in h, k. We will present the computations only for $\log(\rho^{\gamma_1}(x_1,y_1)\rho_0^{-1})$ as those for $\log(\rho^{\gamma_2}(x_1,y_1)\rho_0^{-1})$ can be obtained by interchanging x and y.

We begin applying the BCH formula to the RHS of (4.17). As we truncate the expansion at order 5, the terms that are relevant for our result are

$$
\begin{aligned}
\log(\rho^{\gamma_1}(x_1,y_1)\rho_0^{-1}) &= \log\left(\exp\left(\Theta^x(y_1)\right)\exp\left(\Theta^y(x_0)\right)\right) \\
&= \Theta^x(y_1)+\Theta^y(x_0)+\frac{1}{2}[\Theta^x(y_1),\Theta^y(x_0)] \\
&+\frac{1}{2}\left([\Theta^x(y_1),[\Theta^x(y_1),\Theta^y(x_0)]]+[\Theta^y(x_0),[\Theta^y(x_0),\Theta^x(y_1)]]\right) \\
&-\frac{1}{24}[\Theta^y(x_0),[\Theta^y(x_0),[\Theta^x(y_1),\Theta^y(x_0)]]] \\
&-\frac{1}{720}[\Theta^y(x_0),[\Theta^y(x_0),[\Theta^y(x_0),[\Theta^y(x_0),\Theta^x(y_1)]]]] \\
&-\frac{1}{720}[\Theta^x(y_1),[\Theta^x(y_1),[\Theta^x(y_1),[\Theta^x(y_1),\Theta^y(x_0)]]]] \\
&+\frac{1}{360}[\Theta^x(y_1),[\Theta^y(x_0),[\Theta^y(x_0),[\Theta^y(x_0),\Theta^x(y_1)]]]] \\
&+\frac{1}{360}[\Theta^y(x_0),[\Theta^x(y_1),[\Theta^x(y_1),[\Theta^x(y_1),\Theta^y(x_0)]]]] \\
&+\frac{1}{120}[\Theta^y(x_0),[\Theta^y(x_0),[\Theta^x(y_1),[\Theta^y(x_0),\Theta^x(y_1)]]]] \\
&+\frac{1}{120}[\Theta^x(y_1),[\Theta^x(y_1),[\Theta^y(x_0),[\Theta^x(y_1),\Theta^y(x_0)]]]]+\text{h.o.t.}
\end{aligned}
$$

where ‘h.o.t’ stands for higher order terms. The expansion for $\log(\rho^{\gamma_2}(x_1,y_1)\rho_0^{-1})$ is analogous.

The second step is to express $\Theta^x(y_1)$ and $\Theta^y(x_1)$ as Taylor polynomials around y_0 and x_0 respectively.

The terms we need for the Magnus expansion of $\Theta^x(y_0)$ are,

$$\Theta^x(y_0) = \int_{x_0}^{x_1} \mathscr{Q}^x(\xi, y_0)\, d\xi - \frac{1}{2}\int_{x_0}^{x_1} \left[\int_{x_0}^{\xi_1} \mathscr{Q}^x(\xi_2, y_0)\, d\xi_2, \mathscr{Q}^x(\xi_1, y_0)\right] d\xi_1$$

$$+ \frac{1}{12}\int_{x_0}^{x_1} \left[\int_{x_0}^{\xi_1} \mathscr{Q}^x(\xi_2, y_0)\, d\xi_2, \left[\int_{x_0}^{\xi_1} \mathscr{Q}^x(\xi_2, y_0), \mathscr{Q}^x(\xi_1, y_0)\right]\right] d\xi_1$$

$$+ \frac{1}{4}\int_{x_0}^{x_1} \left[\int_{x_0}^{\xi_1} \left[\mathscr{Q}^x(\xi_3, y_0)\, \xi_3, \mathscr{Q}^x(\xi_2, y_0)\right] d\xi_2, \mathscr{Q}^x(\xi_1, y_0)\right] d\xi_1$$

$$- \frac{1}{24}\int_{x_0}^{x_1} \left[\int_{x_0}^{\xi_1} \left[\int_{x_0}^{\xi_2} \mathscr{Q}^x(\xi_3, y_0)\, d\xi_3, \left[\int_{x_0}^{\xi_2} \mathscr{Q}^x(\xi_3, y_0)\, d\xi_3, \mathscr{Q}^x(\xi_2, y_0)\right]\right] d\xi_2,$$

$$\mathscr{Q}^x(\xi_1, y_0)\right] d\xi_1$$

$$- \frac{1}{24}\int_{x_0}^{x_1} \left[\int_{x_0}^{\xi_1} \left[\int_{x_0}^{\xi_2} \mathscr{Q}^x(\xi_3, y_0)\, d\xi_3, \mathscr{Q}^x(\xi_2, y_0)\right] d\xi_2, \left[\int_{x_0}^{\xi_1} \mathscr{Q}^x(\xi_2, y_0)\, d\xi_2,\right.\right.$$

$$\left.\left.\mathscr{Q}^x(\xi_1, y_0)\right]\right] d\xi_1$$

$$- \frac{1}{24}\int_{x_0}^{x_1} \left[\int_{x_0}^{\xi_1} \mathscr{Q}^x(\xi_2, y_0), \left[\int_{x_0}^{\xi_1} \left[\int_{x_0}^{\xi_2} \mathscr{Q}^x(\xi_3, y_0)\, d\xi_3, \mathscr{Q}^x(\xi_2, y_0)\right] d\xi_2,\right.\right.$$

$$\left.\left.\mathscr{Q}^x(\xi_1, y_0)\right]\right] d\xi_1$$

$$- \frac{1}{8}\int_{x_0}^{x_1} \left[\int_{x_0}^{\xi_1} \mathscr{Q}^x(\xi_2, y_0), \left[\int_{x_0}^{\xi_1} \mathscr{Q}^x(\xi_2, y_0)\, d\xi_2, \left[\int_{x_0}^{\xi_1} \mathscr{Q}^x(\xi_2, y_0), \mathscr{Q}^x(\xi_1, y_0)\right]\right]\right] d\xi_1$$

The expression for $\Theta^y(x_0)$ is analogous.

The third step is to expand the integrand functions inside $\Theta^x(y_0)$ and $\Theta^y(x_0)$, that is, $\mathscr{Q}^x(\xi, y_0)$ and $\mathscr{Q}^y(x_0, \xi)$, around x_0 and y_0 respectively as Taylor polynomials up to order 5. The coefficients of this Taylor expansion are functions of the curvature matrices $\mathscr{Q}^x$ and

$\mathscr{Q}^y$ and their partial derivatives evaluated at (x_0, y_0). After this step, it becomes trivial to compute the integrals as they are polynomial in the dummy variables of integration, ξ, ξ_1 and ξ_2, and to collect terms of each order.

In this way, the right hand side of (4.17) may be written down in terms of $\mathscr{Q}^x$ and $\mathscr{Q}^y$ and their partial derivatives, all evaluated at (x_0, y_0).

The final step is to write this resulting expression in terms of the compatibility expression R defined in (4.14) and its partial derivatives, all evaluated at the arbitrary initial point (x_0, y_0). We summarise the result in the table below, noting that the coefficient of $h^n k^m$ can be obtained from that of $h^m k^n$ by interchanging x and y. It can be seen that every coefficient is a differential expression in R which is identically zero when R is zero, and hence the right hand side of (4.17) is zero. This ends the proof.

Order	Monomial	Coefficient
2	hk	R
3	h^2k	$\frac{1}{2}\partial_x R$
4	h^3k	$\frac{1}{6}\partial_x^2 R - \frac{1}{12}\operatorname{ad}_{\mathscr{Q}^x}(\partial_x R) + \frac{1}{12}\operatorname{ad}_{\partial_x \mathscr{Q}^x}(R)$
	h^2k^2	$\frac{1}{4}\partial_x\partial_y R - \frac{1}{6}\operatorname{ad}_{\operatorname{ad}_{\mathscr{Q}^x}(\mathscr{Q}^y)}(R) - \frac{1}{12}\operatorname{ad}_{\mathscr{Q}^y}(\operatorname{ad}_{\mathscr{Q}^x}(R)) + \frac{1}{4}[\partial_y \mathscr{Q}^y, R]$
5	h^4k	$\frac{1}{24}\partial_x^3 R - \frac{1}{24}\operatorname{ad}_{\mathscr{Q}^x}(\partial_x^2 R) + \frac{1}{24}\operatorname{ad}_{\partial_x^2 \mathscr{Q}^x}(R)$
	h^3k^2	$\frac{1}{12}\partial_x^2\partial_y R - \frac{1}{24}\operatorname{ad}_{\mathscr{Q}^x}(\partial_x\partial_y R) + \frac{1}{24}\operatorname{ad}_{\partial_x \mathscr{Q}^x}(\partial_y R)$
		$-\frac{1}{24}\operatorname{ad}_{\mathscr{Q}^x}(\operatorname{ad}_{\mathscr{Q}^y}(\partial_x R)) - \frac{1}{12}\operatorname{ad}_R(\partial_x R) + \frac{1}{12}\operatorname{ad}_{\partial_x \mathscr{Q}^y}(\partial_x R)$
		$+\frac{1}{6}\operatorname{ad}_{\partial_x^2 \mathscr{Q}^y}(R) - \frac{1}{24}\operatorname{ad}_{\mathscr{Q}^y}(\operatorname{ad}_{\partial_x \mathscr{Q}^x}(R)) - \frac{1}{24}\operatorname{ad}_{\partial_x \mathscr{Q}^y}(\operatorname{ad}_{\mathscr{Q}^x}(R))$
		$+\frac{1}{8}\operatorname{ad}_{[\partial_x \mathscr{Q}^y, \mathscr{Q}^x]}(R) + \frac{1}{8}\operatorname{ad}_{[\mathscr{Q}^y, \partial_x \mathscr{Q}^x]}(R)$

□

It can be seen that the calculations become increasingly complex as the order increases. While obtaining a recursive expression for these expressions seems out of reach, nevertheless, it seems reasonable to conjecture that the result holds to every order. Of interest is the emergence of an operator acting on R at every order, which combines differential and ad operators, both of which are derivations acting on the free Lie algebra generated not only by

the curvature matrices but also their derivatives. Understanding the structure of the sequence of operators acting on R, as exhibited in the Table, is an open problem.

4.3 Numerical examples

We showed in the previous section that the Magnus expansions commute at least up to order 5. These hint that the Lie group integrators based on the Magnus expansion may also commute to some related order, and we investigate some simple examples.

We consider four variational problems and, in order to solve the system of coupled matrix PDEs for the frame, we use a sixth–order Magnus series method which is included in the Matlab package *DiffMan* ([17], Algorithm A.2.5). This numerical scheme is cost efficient [4; 7; 37], which means that not all the terms in the Magnus expansion are used in the calculations. This makes the multivariate integrals in the formulae, which in general are computationally expensive to approximate, [12], computable in an efficient way. Moreover, the algorithm numerically approximates integrals using a fifth–order Gauss–Legendre scheme. Further research needs to be done in order to understand fully how the compatibility condition can be used to prove, to some order, a result like Theorem (4.2.12) for the solvers implemented in *Diffman*. However, as we will see in the numerical examples in this section, neither the omission of some terms in the name of efficiency, nor the replacement of quadrature for exact integration, appear to affect unduly the numerical compatibility.

In the following we will numerically solve some variational problems. We first find a simple exact solution to the Euler–Lagrange equations which may readily be used as components of the curvature matrices $\mathscr{Q}^i$ in the software[1]. We then solve for the frame using two different methods:

[1]Using a numerical solution seems to require the data representations to be aligned in some sense, for example, that the meshes match. This is a development for the future.

1 integrating first **with respect to** y along the line $x = x_0$, and then, for $j = 0,..,n$, use the points $\rho(x_j, y_0)$ as initial condition for the solution found integrating with respect to x along the line $y = y_j$.

2 integrating first **with respect to** x along the line $y = y_0$, and then, for $j = 0,..,n$, use the points $\rho(x_0, y_j)$ as initial condition for the solution found integrating with respect to y along the line $x = x_j$.

and we will compare the solutions obtained. Finally, we use (4.2) to plot the minimiser, given the frame.

4.3.1 An example using a linear action of $SU(2)$

Recall that in Chapter 3, we studied a linear action of $SU(2)$ on pair of complex surfaces $u(x,t)$ and $v(x,t)$. A set of generating differential invariants, using the notation $u = u_1 + \mathrm{i}u_2$, $v = v_1 + \mathrm{i}v_2$, is given by

$$\kappa_1 = \rho \cdot \frac{\partial u_2}{\partial x}, \quad \kappa_2 = \rho \cdot \frac{\partial v_1}{\partial x}, \quad \kappa_3 = \rho \cdot \frac{\partial v_2}{\partial x}$$

and

$$\kappa_4 = \rho \cdot \frac{\partial u_2}{\partial t}, \quad \kappa_5 = \rho \cdot \frac{\partial v_1}{\partial t}, \quad \kappa_6 = \rho \cdot \frac{\partial v_2}{\partial t}$$

In order to find κ_i, $i = 1,..,6$, in terms of the original variables, the following Euler–Lagrange equations must be solved:

$$\begin{cases} -\frac{d}{ds}E^{\kappa_1}(L) - 2\kappa_3 E^{\kappa_2}(L) + 2\kappa_2 E^{\kappa_3}(L) - \frac{d}{dt}E^{\kappa_4}(L) - 2\kappa_6 E^{\kappa_5}(L) + 2\kappa_5 E^{\kappa_6}(L) = 0 \\ 2\kappa_3 E^{\kappa_1}(L) - \frac{d}{ds}E^{\kappa_2}(L) - 2\kappa_1 E^{\kappa_3}(L) + 2\kappa_6 E^{\kappa_4}(L) - \frac{d}{dt}E^{\kappa_5}(L) - 2\kappa_4 E^{\kappa_6}(L) = 0 \\ -2\kappa_2 E^{\kappa_1}(L) + 2\kappa_1 E^{\kappa_2}(L) - \frac{d}{ds}E^{\kappa_3}(L) - 2\kappa_5 E^{\kappa_4}(L) + 2\kappa_4 E^{\kappa_5}(L) - \frac{d}{dt}E^{\kappa_6}(L) = 0 \end{cases} \tag{4.18}$$

along with the syzygies

$$\frac{d}{dt}\begin{pmatrix}\kappa_1\\ \kappa_2\\ \kappa_3\end{pmatrix}=\begin{pmatrix}\frac{d}{ds} & 2\kappa_3 & -2\kappa_2\\ -2\kappa_3 & \frac{d}{ds} & 2\kappa_1\\ 2\kappa_2 & 2\kappa_1 & \frac{d}{ds}\end{pmatrix}\begin{pmatrix}\kappa_4\\ \kappa_5\\ \kappa_6\end{pmatrix} \tag{4.19}$$

Consider the invariant Lagrangian

$$\frac{1}{2}\int_D \kappa_1^2 \quad dx\,dt \tag{4.20}$$

where D is the square $[0,1]\times[0,1]$, and note that a simple exact solution to the system (4.18),(4.19), with boundary conditions

$$\begin{cases}\kappa_1(0,t)=t^3\\ \kappa_4(0,t)=-t^2\\ \kappa_5(0,t)=-\frac{t^4}{3}\\ \kappa_6(0,t)=t+5\end{cases}$$

is given by

$$\begin{cases}\kappa_1=t^3\\ \kappa_2=\kappa_3=0\\ \kappa_4=3xt^2-t^2\\ \kappa_5=-(t+5)\sin(2t^3x)-\frac{1}{3}t^4\cos(2t^3x)\\ \kappa_6=-\frac{1}{3}t^4\sin(2t^3x)+(t+5)\cos(2t^3x)\end{cases}$$

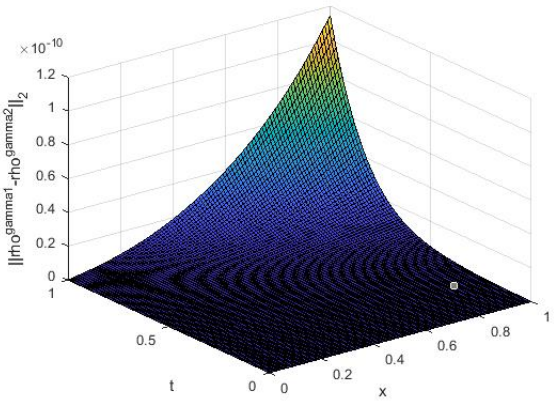


Fig. 4.2 2-Norm of the difference between the two moving frames

Since we do not impose initial data for (u,v) , we may take the (randomly chosen) initial condition for the moving frame to be

$$\rho_0 = \begin{pmatrix} -\frac{1}{3}+\frac{1}{4}\mathrm{i} & \frac{1}{2}-\frac{\sqrt{83}}{12}\mathrm{i} \\ -\frac{1}{2}-\frac{\sqrt{83}}{12}\mathrm{i} & -\frac{1}{3}-\frac{1}{4}\mathrm{i} \end{pmatrix}$$

We use *Diffman* to solve the system for the moving frame in two different ways; first by solving the equation $\rho_x = \mathscr{Q}^x\rho$ for $\rho(x,0)$, and then by solving the equation $\rho_t = \mathscr{Q}^t\rho$ equation with $\rho(x,0)$ as the initial data, and second, by reversing the order of integrations. In order to keep the number of plots low (there are 4 surfaces corresponding to real and imaginary parts of u and v, and 4 plots related to the difference between each surface computed along the two paths), we show in Figure (4.2) the 2-norm of the difference of the two moving frames computed. The step sizes in the x and y directions were chosen to be $h = k = 0.01$. It can be seen that the two possible solutions to the equations for the frame, coincide at least up to order 5.

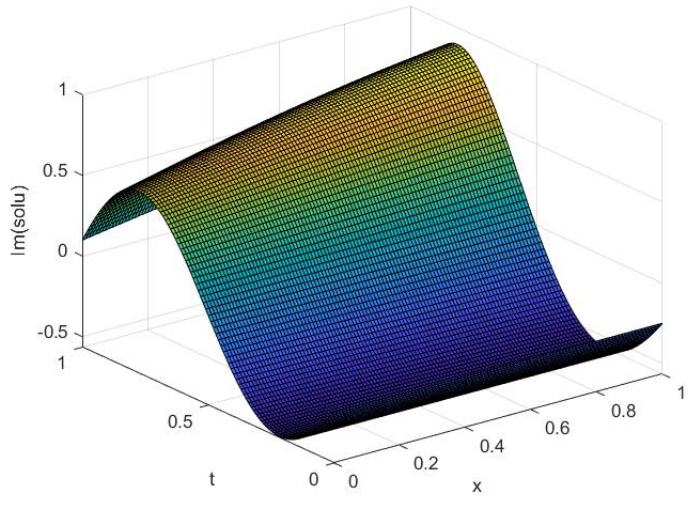


Fig. 4.3 The imaginary component of u

Once the frame has been computed on some domain, we may use the normalisation to recover the minimisers of (4.20), namely

$$\begin{pmatrix} u \\ v \end{pmatrix} = \rho(x,t)^{-1} \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

where $\rho(x,t)^{-1}$ is the group inverse to $\rho(x,t)$. In Figure 4.3 we plot the imaginary component of u; the three other possible plots are similar.

4.3.2 Examples using the projective action of $SL(2)$

The next two examples are related to the Lie group $SL(2)$, given by

$$SL(2) = \left\{ g = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \mid ad - bc = 1 \right\}$$

and we let it act projectively on surfaces as

$$g\cdot x = x, \qquad g\cdot y = y, \qquad g\cdot u = \frac{au+b}{cu+d}$$

This action and its use in the Calculus of Variations is studied in complete detail in [22; 42]. For convenience, we record here the information needed to complete the calculations.

Given the frame ρ defined by the normalisation equations $g\cdot u = 0$, $g\cdot u_x = 1$ and $g\cdot u_{xx} = 0$, the generating differential invariants are,

$$\kappa(x,y) = \rho\cdot u_y = \frac{u_y}{u_x}$$

$$\sigma(x,y) = \rho\cdot u_{xxx} = \frac{u_{xxx}}{u_x} - \frac{3u_{xx}^2}{2u_x^2}.$$

The two curvature matrices are

$$\mathscr{Q}^x = \begin{pmatrix} 0 & -1 \\ \frac{1}{2}\sigma & 0 \end{pmatrix} \quad \mathscr{Q}^y = \begin{pmatrix} -\frac{1}{2}\kappa_x & -\kappa \\ \frac{1}{2}(\kappa_{xx}+\sigma\kappa) & \frac{1}{2}\kappa_x \end{pmatrix}$$

and the syzygy is

$$\frac{\partial}{\partial y}\sigma = \left(\frac{\partial^3}{\partial x^3} + 2\sigma\frac{\partial}{\partial x} + \sigma_x\right)\kappa. \tag{4.21}$$

Introducing a dummy variable τ to effect the variation yields the new invariant $\omega = u_\tau/u_x$ and the syzygies,

$$\frac{\partial}{\partial\tau}\begin{pmatrix}\kappa\\ \sigma\end{pmatrix} = \mathscr{H}\omega = \begin{pmatrix} \dfrac{\partial}{\partial y} - \kappa\dfrac{\partial}{\partial x} + \kappa_x \\ \dfrac{\partial^3}{\partial x^3} + 2\sigma\dfrac{\partial}{\partial x} + \sigma_x \end{pmatrix}\omega.$$

The invariantised Euler–Lagrange equation is, [23],

$$-\left(\frac{\partial^3}{\partial x^3} + 2\sigma\frac{\partial}{\partial x} + \sigma_x\right)E^\sigma(L) + \left(-\frac{\partial}{\partial y} + \kappa\frac{\partial}{\partial x} + 2\kappa_x\right)E^\kappa(L) = 0$$

Finally, the equations for the moving frame ρ, are

$$\begin{cases} \frac{\partial}{\partial x}\rho = \mathscr{Q}^x\rho \\ \frac{\partial}{\partial y}\rho = \mathscr{Q}^y\rho \\ \rho(x_0,y_0) = \rho_0 \\ (x,y) \in [x_0,x_1]\times[y_0,y_1] \end{cases} \tag{4.22}$$

We now consider two different Lagrangians. Our aim here is to investigate the numerical compatibility of the Lie group integrator in some simple examples. Therefore, the region D for this example and the ones that follow have been chosen such that it is possible to compute the solution in a reasonable time and the solution itself is well defined all over the domain. Further, the boundary and initial conditions in the following examples have been chosen in order to have the existence of a solution guaranteed and to make computations tractable.

Example 1

Consider the Lagrangian given by

$$\mathscr{L} = \int_D \kappa^2(x,y) \quad dx\,dy \tag{4.23}$$

where D is the square $[3,4]\times[3,4]$ and we choose a step size equal in both directions $h = k = 0.01$. The Euler–Lagrange equation is

$$\kappa_y = 3\kappa\kappa_x \tag{4.24}$$

and if we add a boundary condition as $\kappa(x,1) = x$, then a simple exact solution is

$$\kappa(x,y) = -\frac{x}{3y-4} \tag{4.25}$$

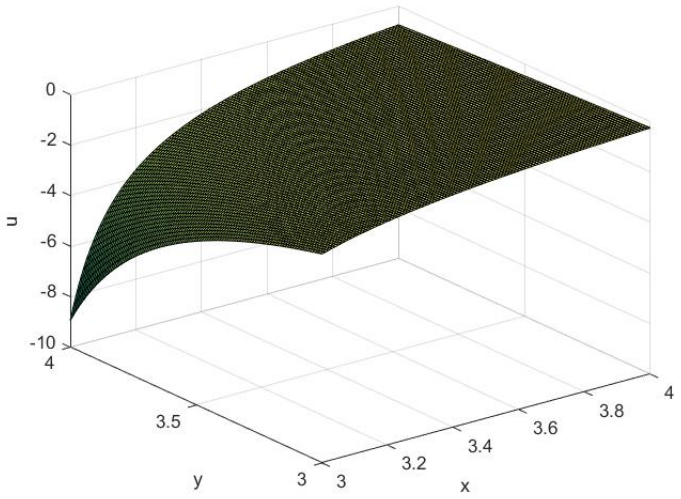


Fig. 4.4 Plots of solutions to the variational problem defined by (4.23), computed integrating the two different ways; the plots look identical to the naked eye.

Setting κ into the syzygy equation (4.21), we obtain an equation for σ,

$$\sigma_y = -2\frac{\sigma}{(3y-4)} - \frac{x\sigma_x}{(3y-4)}$$

and if we impose that $\sigma(1,y) = y$, we obtain the solution

$$\sigma(x,y) = \frac{4x^3+3y-4}{3x^5} \tag{4.26}$$

Inserting (4.25) and (4.26) into (4.22), adding an initial condition

$$\rho_0 = \begin{pmatrix} \frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{\sqrt{3}}{2} & \frac{1}{2} \end{pmatrix}$$

and integrating as we described using the two methods above, we obtain two surfaces, identical to the naked eye, shown in Figure 4.4. A plot of the absolute difference between the

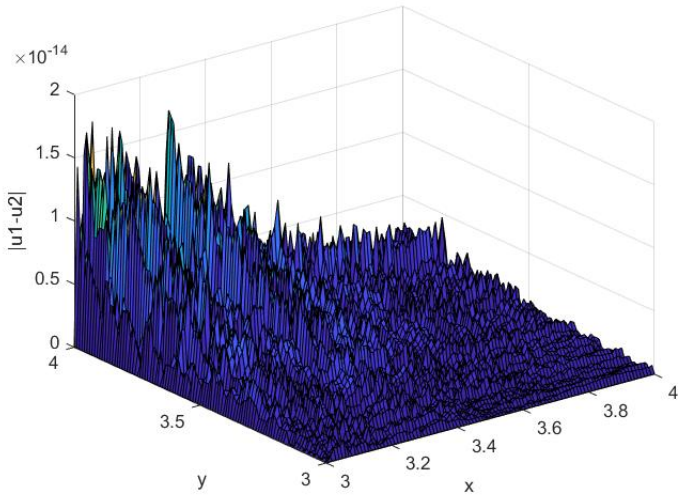


Fig. 4.5 Absolute value of the difference between the two surfaces in Figure 4.4.

two surfaces is shown in Figure 4.5. We can see in this case, that the point–wise difference of the two surfaces plotted in Figure 4.4 is of order at least 7 in h, k.

Remark 4.3.1. The Euler–Lagrange equation (4.24) is the inviscid Burgers equation, well–known for its shock wave solutions. Such solutions lead to the curvature matrices not being continuous, and hence not satisfying the hypotheses for the Picard iteration solution method to be valid. The use of moving frames to study such extremal solutions is an open problem.

Example 2

Consider next the Lagrangian given by

$$\mathscr{L} = \int_D \sigma_x^2(x,y) \quad dxdy \tag{4.27}$$

where D is the square $[1,2]\times[1,2]$ and we choose a step size equal in both directions $h = k = 0.01$. The Euler–Lagrange equation becomes

$$\sigma_{xxxxx} + 2\sigma\sigma_{xxx} + \sigma_x\sigma_{xx} = 0$$

and we notice that all summands in the differential equation above contain one factor with at least a second order derivative in x. So a simple exact solution is

$$\sigma(x,y) = x - y \tag{4.28}$$

Now we can substitute the expression for σ into the syzygy equation (4.21), obtaining an equation for κ

$$\kappa_{xxx} + (2x - 2y)\kappa_x + \kappa + 1 = 0 \tag{4.29}$$

and if we impose that

$$\begin{cases} \kappa(0,y) = y \\ \kappa_x(0,y) = 0 \\ \kappa_{xx}(0,y) = \frac{1}{y} \end{cases} \tag{4.30}$$

we obtain a solution in terms of the Airy functions of first and second kind (and their first derivative). Inserting (4.28) and the solution to (4.29)–(4.30) into (4.22), adding an initial condition

$$\rho_0 = \begin{pmatrix} \frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{\sqrt{3}}{2} & \frac{1}{2} \end{pmatrix}$$

and integrating as we described in 1–2 above, we obtain the two surfaces shown in Figure 4.6. A plot of the absolute difference between the two surfaces is given in Figure 4.7. In this example we obtain that the difference between the two surfaces is of order greater than 5.

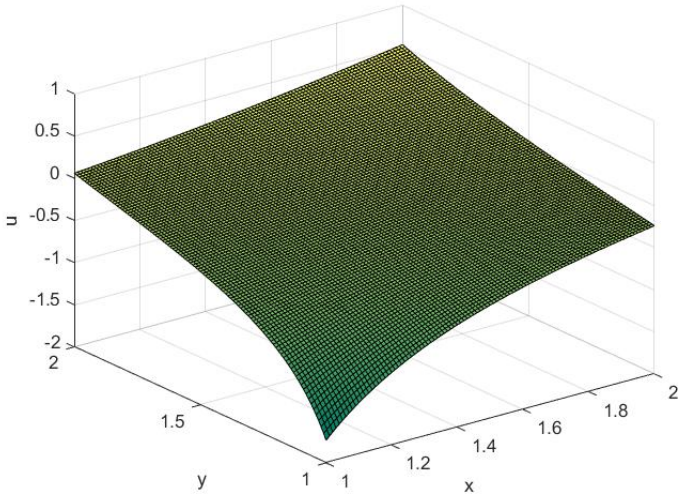


Fig. 4.6 Plots of solutions to the variational problem defined by (4.27), computed integrating the two different ways; the plots look identical to the naked eye.

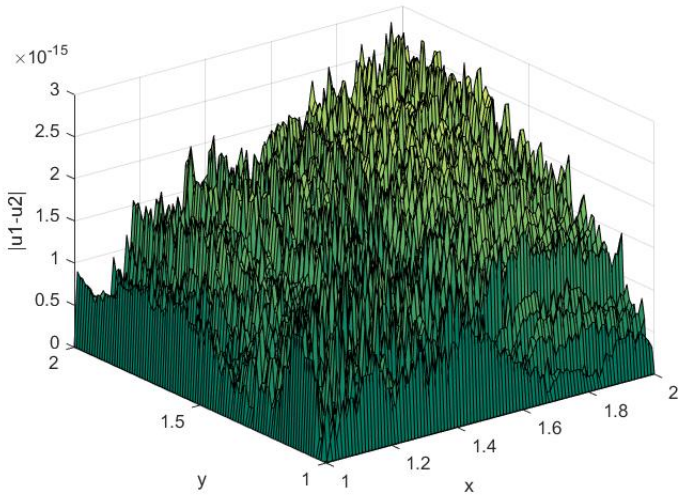


Fig. 4.7 Absolute value of the difference between the two surfaces in Figure 4.6.

4.3.3 An example using the standard action of $SE(2)$

We end this section with a numerical example involving an action of $SE(2)$ on parametrised surfaces $(s,t) \to (x(s,t),u(s,t))$. In many applications, we consider $(x(s,t),u(s,t))$ as an evolving curve, $(x(s),u(s))$, in the (x,u) plane. In this case, it is common to take s to be arc length. Here, we achieve this, while maintaining both $\dfrac{\partial}{\partial s}$ and $\dfrac{\partial}{\partial t}$ to be standard, commuting operators, by taking $x_s^2+u_s^2=1$ as a constraint in the Lagrangian.

Remark 4.3.2. If we define $u=u(x,t)$ and take the standard arc length derivative, $\dfrac{\partial}{\partial s} = (1+u_x^2)^{-1/2}\dfrac{\partial}{\partial x}$, then $\dfrac{\partial}{\partial s}$ and $\dfrac{\partial}{\partial t}$ do not commute, since $u_t \neq 0$. In this case, the compatibility condition will not take the form (4.14).

The action is given by

$$g\cdot\begin{pmatrix} x \\ u \\ 1 \end{pmatrix} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & a \\ \sin(\theta) & \cos(\theta) & b \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ u \\ 1 \end{pmatrix} \tag{4.31}$$

where $(\theta,a,b)\in\mathbb{R}^3$.

Moving frames for this and related actions and their use in the Calculus of Variations are well studied, see [24; 42]. For convenience, we record here the information we need. Given the normalisation equations

$$g\cdot x=0, \qquad g\cdot u=0, \qquad g\cdot u_s=0,$$

the frame is

$$\rho = \frac{1}{(x_s^2+u_s^2)^{1/2}}\begin{pmatrix} x_s & u_s & -(xx_s+uu_s) \\ -u_s & x_s & u_sx-x_su \\ 0 & 0 & 1 \end{pmatrix}.$$

The normalisation equations give $\rho \cdot x = I(x) = 0$ and similarly $I(u) = 0$ and $I(u_s) = 0$, while $\rho \cdot x_s = I(x_s) = (x_s^2 + u_s^2)^{1/2}$.

Calculating the curvature matrices and applying the Replacement Rule, Theorem 2.2.25, yields

$$\mathscr{Q}^s = \begin{pmatrix} 0 & \dfrac{\kappa_1}{\kappa_2} & -\kappa_2 \\ -\dfrac{\kappa_1}{\kappa_2} & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{4.32}$$

$$\mathscr{Q}^t = \begin{pmatrix} 0 & \dfrac{I(u_{st})}{\kappa_2} & -\kappa_4 \\ -\dfrac{I(u_{st})}{\kappa_2} & 0 & -\kappa_3 \\ 0 & 0 & 0 \end{pmatrix} \tag{4.33}$$

where

$$\rho \cdot u_{ss} = \kappa_1, \qquad \rho \cdot x_s = \kappa_2, \qquad \rho \cdot u_t = \kappa_3, \qquad \rho \cdot x_t = \kappa_4.$$

Calculating the syzygies from the compatibility condition yields

$$I(u_{st}) = \kappa_{3,s} + \kappa_1 \kappa_4 / \kappa_2$$

and therefore, the generating invariants are κ_i, $i = 1, \ldots, 4$, together with the invariant independent variables.

The famous invariant of this action, the Euclidean curvature, can be expressed as $\kappa_1 \kappa_2^{-3}$. It is usual to set $\kappa_2 = (x_s^2 + u_s^2)^{1/2} = 1$ to fix the parametrisation and ease the calculations.

Setting $\kappa_2 = 1$, the syzygies for our invariants are $\kappa_{4,s} = \kappa_1 \kappa_3$ together with

$$\frac{\partial}{\partial t}\kappa_1 = \frac{\partial}{\partial s}\left(\frac{\partial}{\partial s}\kappa_3 + \kappa_1 \kappa_4\right).$$

In order to effect the variation, we introduce a dummy invariant independent variable, τ. We obtain two new invariants, $\sigma_1 = \rho \cdot u_\tau$ and $\sigma_2 = \rho \cdot x_\tau$, and then the syzygy operator $\mathscr{H}$ needed to calculate the Euler–Lagrange equations is,

$$\frac{\partial}{\partial \tau}\begin{pmatrix}\kappa_1\\ \kappa_2\\ \kappa_3\\ \kappa_4\end{pmatrix} = \mathscr{H}\begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix} = \begin{pmatrix} \frac{\partial^2}{\partial s^2} & \kappa_{1,s} + \kappa_1 \frac{\partial}{\partial s} \\ -\kappa_1 & \frac{\partial}{\partial s} \\ \frac{\partial}{\partial t} - \kappa_4 \frac{\partial}{\partial s} & \kappa_{3,s} \\ \kappa_3 \frac{\partial}{\partial s} - \kappa_{3,s} - \kappa_1 \kappa_4 & \frac{\partial}{\partial t} + \kappa_1 \kappa_3 \end{pmatrix}\begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}$$

where we have set $\kappa_2 = 1$ in $\mathscr{H}$.

Consider the Lagrangian

$$\int_D \frac{1}{2}\left(\frac{\partial}{\partial t}\kappa_1\right)^2 - \lambda\left(\kappa_2 - 1\right) \mathrm{d}s\mathrm{d}t \tag{4.34}$$

where $D = [1,2] \times [1,2]$ and λ is a Lagrange multiplier for the constraint, $\kappa_2 = 1$. Given (4.34), the system to be solved is made of the two Euler–Lagrange equations for the invariants and their syzygies, which in this case is

$$\begin{cases} \kappa_1 \lambda - \kappa_{1,sstt} = 0 \\ \lambda_s - \kappa_1 \kappa_{1,stt} = 0 \\ \kappa_{4,s} - \kappa_1 \kappa_3 = 0 \\ \kappa_{3,ss} + \kappa_{1,s}\kappa_4 + \kappa_1^2 \kappa_3 - \kappa_{1,t} = 0 \end{cases} \tag{4.35}$$

A simple exact solution to (4.35) is

$$\kappa_1 = -4(s+t)^{-1} \quad \lambda = 24(s+t)^{-4} \quad \kappa_3 = s+t+\sin\left(4\ln(s+t)\right)+\cos\left(4\ln(s+t)\right) \tag{4.36}$$

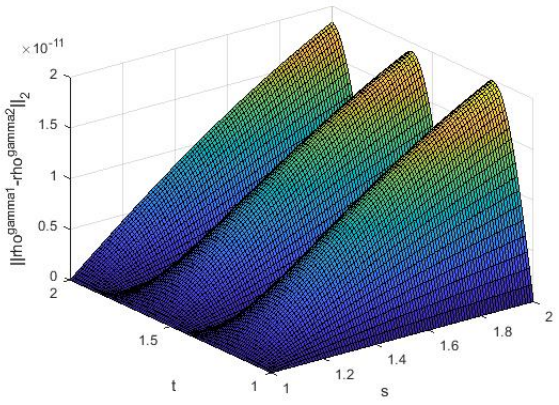


Fig. 4.8 2–norm of the difference between the two moving frames

and

$$\kappa_4 = \cos(4\ln(s+t)) - \sin(4\ln(s+t)) + 1 - 4(s+t) \tag{4.37}$$

Substituting (4.36)–(4.37) into (4.32)–(4.33), we solve system (4.22) using the procedure described above, with a constant step size in both direction equal to $h = k = 0.01$. A plot of the 2–norm of the difference of the two moving frames obtained in this way can be found in Figure 4.8. From the plot it can be seen that in this case our theoretical result is mirrored in the numerical result. Once the frame has been computed, recall the minimisers are given by

$$\begin{pmatrix} x(s,t) \\ u(s,t) \\ 1 \end{pmatrix} = \rho(s,t)^{-1} \begin{pmatrix} I(x) \\ I(u) \\ 1 \end{pmatrix} = \rho(s,t)^{-1} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

where the right–hand side is determined by the first two of the normalisation equations, $I(x) = \rho \cdot x = 0$ and $I(u) = \rho \cdot u = 0$, which define the frame. A plot of the minimisers is provided in Figure 4.9.

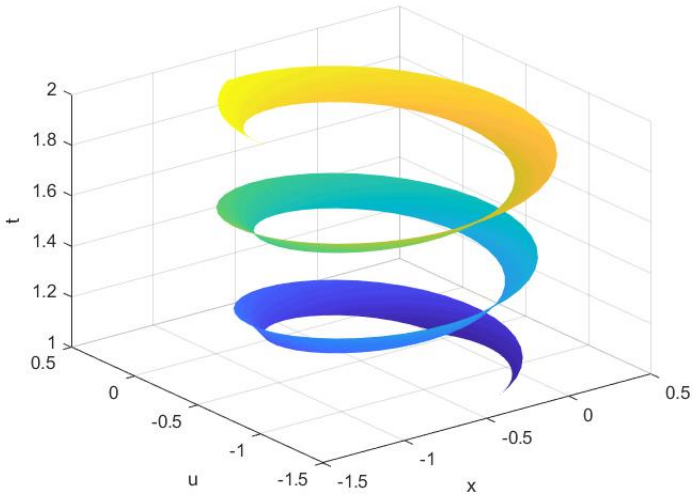


Fig. 4.9 A plot of the minimiser as an evolving curve, $(t, x(s,t), u(s,t))$

4.3.4 Considering bigger domains

The examples above show that the theory we proved in this chapter is confirmed numerically. However, as we wanted to use a reasonably small step size, we had to reduce the area we were integrating. An interesting question that this approach leaves open is what happen when the domain of integration gets bigger? Here we will consider a variational problems seen already in this section and reduce the step size to $h = k = 0.1$, but taking as a domain of integration a much bigger area.

Consider the variational problem studied in Section 4.3.3, with Lagrangian

$$\int \frac{1}{2}\left(\frac{\partial}{\partial t}\kappa_1\right)^2 - \lambda\left(\kappa_2 - 1\right) \mathrm{d}s\mathrm{d}t$$

We look for a solution in $[1,15] \times [1,15]$, rather than in $[1,2] \times [1,2]$ as before. Recall we lower the step size to $h = k = 0.1$, so a fifth–order error now will have magnitude of 10^{-5}. A plot of the solution can be found in Figure 4.10, while the 2–norm difference of the moving frames can be found in Figure 4.11.

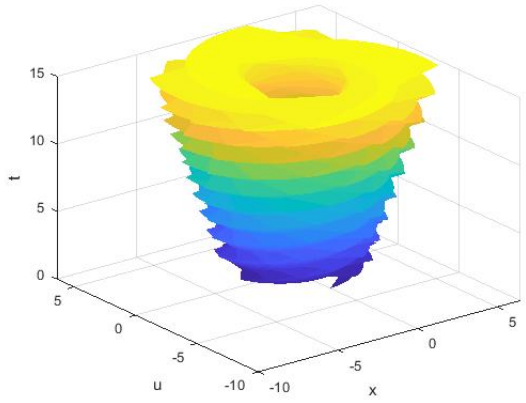


Fig. 4.10 Solution to the variational problem (4.34) in $[1,15] \times [1,15]$

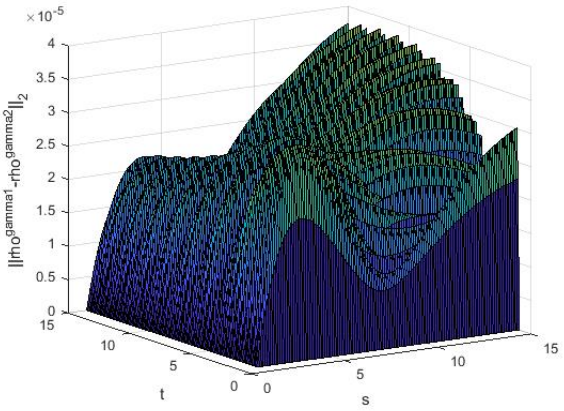


Fig. 4.11 2–norm of the difference between the two moving frames in $[1,15] \times [1,15]$

Remark 4.3.3. Although these are preliminary results, they suggest that the method presented in this section can be applied also to larger domains. More research will have to be done to formally understand which are the limits of this method in terms of domain of integration. We think that one of the biggest challenges is to understand the qualitative behaviour of the approximated solution in order to predict if any singularities will appear and where.

4.4 Final remarks and a conjecture

In this chapter, we have shown that Magnus expansions may be used to solve the system of equations for a moving frame, (4.1), which evolves on a Lie group, in the case where the base space has two dimensions. Our result extends immediately to the system of equations for a moving frame on an p-dimensional base space, $p \geq 2$, as these equations are pairwise compatible.

Our method can, in principle, be applied to any variational problem with a Lie group symmetry, where

1. the Lie group action leaves the independent variables invariant or acts by translation on them, so that the invariant differential operators are the standard, commuting operators,
2. which can be described and analysed using a Lie group based moving frame,
3. and for which the solutions of the Euler–Lagrange equations lead to continuous curvature matrices,

with the caveat that our proof requires the curvature matrices to be smooth.

We have applied our result to find, numerically, simple extremal solutions for variational problems which are invariant either under a linear action of $SU(2)$, the projective action of $SL(2)$ or the affine action of $SE(2)$. Cost efficient Lie group integrators [4; 7; 37] reduce the number of commutators involved in the numerical computation, and the implementation we

have used, [17], takes advantage of these ideas. The precise interplay between compatibility and efficiency is a topic for further study. Further, the use of Lie group integrators for the computation of the frame for numerical solutions of the Euler–Lagrange solutions will depend on whether or not they may take as input, numerical coefficients in the matrices $\mathscr{Q}^i$ appearing in the equations for the frame.

While we have shown that the Magnus expansions are compatible to order 5, it is clear that our proof of the compatibility (4.17) could have continued to higher orders. However, the calculations become less and less tractable, and there is no clear, discernible, recursive pattern. The object our calculations uncover, to wit, an infinite set of operators acting on the compatibility condition R, involving not only the curvature matrices $\mathscr{Q}^i$ but also their derivatives, seems to be new. We conclude by stating the general result as a conjecture.

Conjecture 4.4.1. The Magnus expansions for compatible systems will commute to all orders, that is, the right–hand side of (4.17) is identically zero to all orders of h, k.

We may state the conjecture more precisely, that the right–hand side of (4.17) is a differential operator acting on the compatibility condition R given in (4.14), and which therefore must be identically zero for the Magnus expansions to commute.

5. Variational problems in multispaces

In the previous chapters we have studied different ways to solve smooth invariant variational problems. However, in many applications, it is necessary to pass from a smooth setting, to a discrete one. Since we are most interested in variational problems invariant under a Lie group action, how is it possible to be sure that the discretised problem still retains the geometric features we want to study? This problem has been addressed in [44] with the creation of lattice–based multispaces. These are spaces that contain both the jet bundle, as a submanifold, and the space of lattices, as an open subset. When we restrict the quantities we are studying to the jet bundle, then we are working in a smooth setting, while the lattice provides a framework for the discretisation. Under some conditions, the authors showed that the multispace is a smooth manifold and it is possible to have well–defined interpolating Lagrange polynomials, whose coefficients can be used as coordinates.

This chapter starts with the basic definitions and notation related to the multispace construction and how to introduce variational problems in this setting. In the presence of a Lie group invariance, we will show how the formula for the induced infinitesimal action on the first derivative can be generalised in the case of the multispace.

Since the discretisation process is so delicate and important, we move to study how to build discrete approximants to the generating differential invariants of a Lie group action. We show how it is possible to obtain discrete versions of the generating differential invariants and their derivatives, up to any order of accuracy in the discretised variables.

In the final part of the chapter we will study discrete variational problems that are invariant under the standard action of $SE(2)$. Motivated by the computational cost and difficulties that arise in this case, we introduce a constraint in the Lagrangian and study its effect in the numerical output.

The original contributions in this Chapter can be found in Section 5.1.3, Section 5.2.2, Section 5.2.3, Section 5.3.3, Section 5.3.4 and Section 5.4.

5.1 Lattice–based multispaces

In this section the basic ideas in the theory of lattice–based multispaces will be presented, along with the notation we will use. Most of the notation derives from the context of Lagrangian interpolation, hence divided differences will play an important role. For more details on divided difference and interpolation theory, see [53].

5.1.1 From first order Lagrange interpolation of functions to first order multispace

Let U be an open domain in $\mathbb{R}^p$ which has coordinates $\mathbf{x} = (x_1, x_2, \ldots, x_p)$. If $\mathbf{x}^i = (x_1^i, x_2^i, \ldots, x_p^i) \in U$ for $i = 0, \ldots p$, and $\Gamma = \{\mathbf{x}^i \,|\, i = 0, \ldots, p\}$, then we denote as $\Delta(\mathbf{x}^0, \ldots, \mathbf{x}^p)$ the $(p+1) \times (p+1)$ determinant

$$\Delta(\mathbf{x}^0, \ldots, \mathbf{x}^p) = \begin{vmatrix} 1 & x_1^0 & x_2^0 & \cdots & x_p^0 \\ 1 & x_1^1 & x_2^1 & \cdots & x_p^1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^p & x_2^p & \cdots & x_p^p \end{vmatrix}.$$

Further for a function $u : U \to \mathbb{R}$, we define

$$\Delta_k(\mathbf{x}^1,\ldots,\mathbf{x}^p;u) = \begin{vmatrix} 1 & x_1^0 & \cdots & x_{k-1}^0 & u(\mathbf{x}^0) & x_{k+1}^0 & \cdots & x_p^0 \\ 1 & x_1^1 & \cdots & x_{k-1}^1 & u(\mathbf{x}^1) & x_{k+1}^1 & \cdots & x_p^1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^p & \cdots & x_{k-1}^p & u(\mathbf{x}^p) & x_{k+1}^p & \cdots & x_p^p \end{vmatrix}.$$

Note that in Δ_k, it is not $\mathbf{x}^k$ which has been replaced, but rather, the column of the kth components of the $\mathbf{x}^i$. Continuing, we define $\Delta_{k,j}(\mathbf{x}^1,\ldots,\mathbf{x}^p;u,v)$ to be the value of the determinant Δ where the column of the kth co-ordinates of the $\mathbf{x}^i$ has been replaced by $(u(\mathbf{x}^0),u(\mathbf{x}^1),\ldots,u(\mathbf{x}^p))^T$ and the column of the jth coordinates has been replaced by $(v(\mathbf{x}^0),v(\mathbf{x}^1),\ldots,v(\mathbf{x}^p))^T$.

Definition 5.1.1. We say that $\Gamma = \{\mathbf{x}^i \in U \mid i = 0,\ldots p\}$ is a *multispace lattice* with basepoint $\mathbf{x}^0$ if $\Delta(\mathbf{x}^0,\mathbf{x}^1,\cdots,\mathbf{x}^p) \neq 0$.

Multispaces are based on the Lagrange interpolation; using the notation defined above, the Lagrange interpolation theorem can be expressed as follows.

Proposition 5.1.2. *Given $u : U \to \mathbb{R}$, the first order Lagrange approximation $p(u)$ of the function u based on interpolation at the $p+1$ points Γ is given by*

$$\begin{aligned} p(u)(\mathbf{x}) \quad &= u(\mathbf{x}^0) + \frac{\Delta_1(\mathbf{x}^0,\mathbf{x}^1,\ldots,\mathbf{x}^p;u)}{\Delta(\mathbf{x}^0,\mathbf{x}^1,\ldots,\mathbf{x}^p)}(x_1 - x_1^0) + \frac{\Delta_2(\mathbf{x}^0,\mathbf{x}^1,\ldots,\mathbf{x}^p;u)}{\Delta(\mathbf{x}^0,\mathbf{x}^1,\ldots,\mathbf{x}^p)}(x_2 - x_2^0) \\ &\quad + \cdots + \frac{\Delta_p(\mathbf{x}^0,\mathbf{x}^1,\ldots,\mathbf{x}^p;u)}{\Delta(\mathbf{x}^0,\mathbf{x}^1,\ldots,\mathbf{x}^p)}(x_p - x_p^0). \end{aligned} \tag{5.1}$$

Further, if $\mathbf{x}_i = h\mathbf{e}_i$ then as $h \to 0$, we have that the coefficient of $(x_i - x_i^0)$ converges to $\partial u/\partial x_i\big|_{\mathbf{x}=\mathbf{x}_0}$.

The coefficient of $(x_j - x_j^0)$ in the expression of $p(u)(\mathbf{x})$ in Equation (5.1) is denoted as $\mathscr{M}(u,x_j)$, the *multispace approximation to $\partial u/\partial x_j$*. Points in the multispace have the

coefficients of the Lagrange interpolation as coordinates, namely

$$z = (\mathbf{x}^0, \mathbf{x}^1, \ldots, \mathbf{x}^p, u(\mathbf{x}^0), \mathscr{M}(u, x_1), \ldots, \mathscr{M}(u, x_p)).$$

5.1.2 Lie group actions on multispaces

If g is an element of a Lie group G, an action on the multispace variables can be defined as usual on the first $p+1$ coordinates. The induced action on the multispace coordinates relative to the approximations of the derivative is

$$g \cdot \mathscr{M}(u^\alpha, x_k) = \frac{\Delta_k(g \cdot \mathbf{x}^0, g \cdot \mathbf{x}^1, \ldots, g \cdot \mathbf{x}^p; g \cdot u^\alpha)}{\Delta(g \cdot \mathbf{x}^0, g \cdot \mathbf{x}^1, \ldots, g \cdot \mathbf{x}^p)}.$$

Example 5.1.3. Consider the case where $p = 1$, $\mathbf{x}^0 = (x_1^0)$, $\mathbf{x}^1 = (x_1^1)$, $\Delta(\mathbf{x}^0, \mathbf{x}^1) = \begin{vmatrix} 1 & x_1^0 \\ 1 & x_1^1 \end{vmatrix}$ and $\Delta_1(\mathbf{x}^0, \mathbf{x}^1; u) = \begin{vmatrix} 1 & u(x_1^0) \\ 1 & u(x_1^1) \end{vmatrix}$. We look at the group of scaling and translation, acting on $\mathbf{x}^i$ and u as

$$g \cdot \mathbf{x} = e^\mu x, \qquad g \cdot u = u + \varepsilon,$$

where $\mu, \varepsilon \in \mathbb{R}$. The induced action is calculated as follows. The action is induced on the divided difference elements as

$$\Delta(g \cdot \mathbf{x}^0, g \cdot \mathbf{x}^1) = \begin{vmatrix} 1 & \mathrm{e}^\mu x_1^0 \\ 1 & \mathrm{e}^\mu x_1^1 \end{vmatrix} = \mathrm{e}^\mu \Delta(\mathbf{x}^0, \mathbf{x}^1).$$

From $g\cdot u = u+\varepsilon$ we obtain $g\cdot u(\mathbf{x}^0) = g\cdot u|_{\mathbf{x}=\mathbf{x}^0,u=u(\mathbf{x}^0)} = u(\mathbf{x}^0)+\varepsilon$ and denoting $\mathscr{M}(u,x_1)$ as $\mathscr{M}(u,x)$,

$$g\cdot\mathscr{M}(u,x) = \frac{\Delta_1(g\cdot\mathbf{x}^0, g\cdot\mathbf{x}^1; g\cdot u)}{\Delta(g\cdot\mathbf{x}^0, g\cdot\mathbf{x}^1)} = \frac{\begin{vmatrix} 1 & u(\mathbf{x}^0)+\varepsilon \\ 1 & u(\mathbf{x}^1)+\varepsilon \end{vmatrix}}{\mathrm{e}^{\mu}\Delta(\mathbf{x}^0,\mathbf{x}^1)} = \mathrm{e}^{-\mu}\mathscr{M}(u,x).$$

5.1.3 Infinitesimal actions on multispace

Proposition 5.1.4 ([51, p. 110])**.** *Given a Lie group action $G\times M\times\mathbb{R}^q\to M\times\mathbb{R}^q$ and its induced action on the jet bundle $J^k(M,\mathbb{R}^q)$, we let $\gamma\colon[-1,1]\to G$ be a smooth path in G with $\gamma(0)=e$, the identity of G, and define the infinitesimal actions as*

$$\frac{\mathrm{d}}{\mathrm{d}\varepsilon}\Big|_{\varepsilon=0}\gamma(\varepsilon)\cdot x_i = \xi_i(\mathbf{x},\mathbf{u}), \qquad \frac{\mathrm{d}}{\mathrm{d}\varepsilon}\Big|_{\varepsilon=0}\gamma(\varepsilon)\cdot u^\alpha = \phi^\alpha(\mathbf{x},\mathbf{u}), \qquad \frac{\mathrm{d}}{\mathrm{d}\varepsilon}\Big|_{\varepsilon=0}\gamma(\varepsilon)\cdot u^\alpha_K = \phi^\alpha_K(\mathbf{x},\mathbf{u}).$$

Then, it is possible to prolong the infinitesimal action to the derivatives of u^α in the following way

$$\phi^\alpha_K = \frac{\mathrm{D}^{|K|}}{\mathrm{D}x_1^{K_1}\mathrm{D}x_2^{K_2}\cdots\mathrm{D}x_p^{K_p}}\left(\phi^\alpha - \sum_i u_i^\alpha\xi_i\right) + \sum_i u^\alpha_{K+\mathbf{e}_i}\xi_i,$$

where $u^\alpha_i = \partial u^\alpha/\partial x_i$ and $\mathrm{D}/\mathrm{D}x_i$ is the total derivative with respect to x_i. In particular,

$$\phi^\alpha_{\mathbf{e}_j} = \frac{\mathrm{D}\phi^\alpha}{\mathrm{D}x_j} - \sum_k u_k^\alpha\frac{\mathrm{D}\xi_k}{\mathrm{D}x_j}.$$

We now obtain the infinitesimal action on the multispace coordinates.

Proposition 5.1.5. *Given a Lie group G acting on a multispace Γ, let $\gamma\colon[-1,1]\to G$ be a smooth path in G with $\gamma(0)=e$, define the induced action on Γ as the action on each point, i.e.*

$$\frac{\mathrm{d}}{\mathrm{d}\varepsilon}\Big|_{\varepsilon=0}\gamma(\varepsilon)\cdot\mathbf{x}^i = (\xi_1,\ldots\xi_p)\Big|_{(\mathbf{x}^i,\mathbf{u}(\mathbf{x}^i))}$$

and further

$$\frac{\mathrm{d}}{\mathrm{d}\varepsilon}\Big|_{\varepsilon=0}\gamma(\varepsilon)\cdot u^{\alpha}(\mathbf{x}^0)=\phi^{\alpha}(\mathbf{x}^0,\mathbf{u}(\mathbf{x}^0)).$$

Then, it is possible to prolong the infinitesimal action to the multispace approximation of the derivative of u^{α} in the following way

$$\frac{\mathrm{d}}{\mathrm{d}\varepsilon}\Big|_{\varepsilon=0}\gamma(\varepsilon)\cdot\mathscr{M}(u^{\alpha},x_j)=\mathscr{M}(\phi^{\alpha},x_j)-\sum_k\mathscr{M}(u^{\alpha},x_k)\mathscr{M}(\xi_k,x_j).$$

Proof: To obtain the infinitesimal action on the $\mathscr{M}(u^{\alpha},x_j)$, note that the infinitesimal action on the ith column of $\Delta(\mathbf{x}^0,\ldots,\mathbf{x}^p)$ is $(\xi_i(\mathbf{x}^0),\xi_i(\mathbf{x}^1),\ldots,\xi_i(\mathbf{x}^p))^T$ and so we have, setting $\mathbf{x}=(\mathbf{x}^0,\mathbf{x}^1,\ldots,\mathbf{x}^p)$ to ease the appearance of the formulae,

$$\begin{aligned}
&\tfrac{\mathrm{d}}{\mathrm{d}\varepsilon}\Big|_{\varepsilon=0}\gamma(\varepsilon)\cdot\mathscr{M}(u^{\alpha},x_j)\\
&\quad=\frac{\Delta_j(\mathbf{x};\phi^{\alpha})}{\Delta(\mathbf{x})}+\sum_{i\neq j}\frac{\Delta_{j,i}(\mathbf{x};u^{\alpha},\xi_i)}{\Delta(\mathbf{x})}-\sum_k\frac{\Delta_j(\mathbf{x};u^{\alpha})}{\Delta(\mathbf{x})^2}\Delta_k(\mathbf{x};\xi_k)\\
&\quad=\mathscr{M}(\phi^{\alpha},x_j)-\mathscr{M}(u^{\alpha},x_j)\mathscr{M}(\xi_j,x_j)+\textstyle\sum_{k\neq j}\left[\dfrac{\Delta_{j,k}(\mathbf{x};u^{\alpha},\xi_k)}{\Delta(\mathbf{x})}-\dfrac{\Delta_j(\mathbf{x};u^{\alpha})}{\Delta(\mathbf{x})^2}\Delta_k(\mathbf{x};\xi_k)\right],\\
&\quad=\mathscr{M}(\phi^{\alpha},x_j)-\mathscr{M}(u^{\alpha},x_j)\mathscr{M}(\xi_j,x_j)+\textstyle\sum_{k\neq j}\left[-\dfrac{\Delta_k(\mathbf{x};u^{\alpha})}{\Delta(\mathbf{x})}\dfrac{\Delta_j(\mathbf{x};\xi_k)}{\Delta(\mathbf{x})}\right]\\
&\quad=\mathscr{M}(\phi^{\alpha},x_j)-\textstyle\sum_k\mathscr{M}(u^{\alpha},x_k)\mathscr{M}(\xi_j,x_k)
\end{aligned}$$

using the identity

$$\Delta_{j,k}(\mathbf{x};f,g)\Delta(\mathbf{x})=\Delta_j(\mathbf{x};f)\Delta_k(\mathbf{x};g)-\Delta_k(\mathbf{x};f)\Delta_j(\mathbf{x};g).$$

□

5.1.4 Lagrangians in multispaces

We focus our attention on how to rewrite smooth dimensional Lagrangians in the multispace formalism. We will restrict our attention to one–dimensional, first order Lagrangians, namely

$$\mathscr{L} = \int_a^b L(x, u(x), u_x)\,\mathrm{d}x. \tag{5.2}$$

In this case, the notation simplifies to $\mathbf{x}^0 = (x_1^0)$, $x^1 = (x_1^1)$. Since there is only one independent variable and there is no possibility for ambiguity, we will denote $\mathbf{x}^0$ and $\mathbf{x}^1$ as x_0 and x_1 respectively. Then we have

$$\Delta(\mathbf{x}_0, \mathbf{x}_1) = \begin{vmatrix} 1 & x_0 \\ 1 & x_1 \end{vmatrix} = x_1 - x_0, \tag{5.3}$$

and we take (5.3) as the multispace analogue of $\mathrm{d}x$. Next, we cover the interval $[a,b] = \cup_n [x_n, x_{n+1}]$ using a partition, $a \leq x_0 < x_1 < x_2 < \cdots < x_N = b$. By taking a sequence of lattices $\Gamma_n = \{x_n, x_{n+1}\}$ we can write a Lagrangian on multispace which is analogous to (5.2), specifically,

$$\mathscr{L}_n = \sum_{\Gamma_n} L(x_n, u(x_n), \mathscr{M}(u,x)|_{\Gamma_n})\,\Delta(x_n, x_{n+1}).$$

We now adopt standard practice which is to drop the index n and to write our multispace Lagrangian as

$$\mathscr{L}_n = \sum_{\Gamma} L(x_0, u(x_0), \mathscr{M}(u,x))\,\Delta(x_0, x_1), \tag{5.4}$$

where in this sum, x_0 is the basepoint of Γ and with an abuse of notation we denoted $\mathscr{M}(u,x)|_{\Gamma_n}$ with $\mathscr{M}(u,x)$.

5.1.5 Euler–Lagrange equations and Noether's first theorem

Suppose the multispace Lagrangian (5.4) is invariant, i.e. for all $g \in G$ it holds

$$\sum_{\Gamma} L(g\cdot x_0, g\cdot u(x_0), g\cdot \mathscr{M}(u,x)|_{\Gamma})\; g\cdot\Delta(x_0,x_1) = \sum_{\Gamma} L(x_0,u(x_0),\mathscr{M}(u,x)|_{\Gamma})\Delta(x_0,x_1).$$

Equivalently, taking a path $\gamma(-1,1)\to G$ with $\gamma(0)=e$, the identity element of G, we have, setting $\xi_i = \xi(x_i,u(x_i))$ for $i=0,1$, $\phi_0=\phi(x_0,u(x_0))$

$$\begin{aligned} 0 &= \tfrac{\mathrm{d}}{\mathrm{d}\varepsilon}\big|_{\varepsilon=0}\sum_{\Gamma} L(\gamma(\varepsilon)\cdot x_0,\gamma(\varepsilon)\cdot u(x_0),\gamma(\varepsilon)\cdot\mathscr{M}(u,x))\gamma(\varepsilon)\cdot\Delta(x_0,x_1) \\ &= \sum_{\Gamma}\Bigg\{\left[\frac{\partial L}{\partial x_0}\xi_0+\frac{\partial L}{\partial u(x_0)}\phi_0+\frac{\partial L}{\partial\mathscr{M}(u,x)}\left(\mathscr{M}(\phi,x)-\mathscr{M}(u,x)\mathscr{M}(\xi,x)\right)\right](x_1-x_0) \\ &\qquad +L(\xi_1-\xi_0)\Bigg\}, \end{aligned}$$

where we have used Proposition 5.1.5. In this one dimensional case, we have

$$\mathscr{M}(\xi,x)=\frac{\xi_1-\xi_0}{x_1-x_0},\qquad \mathscr{M}(\phi,x)=\frac{\phi_1-\phi_0}{x_1-x_0},$$

and so the above simplifies to

$$\begin{aligned} 0=\sum_{\Gamma}\Bigg[&\frac{\partial L}{\partial x_0}(x_1-x_0)\xi_0+\frac{\partial L}{\partial u(x_0)}(x_1-x_0)\phi_0+\frac{\partial L}{\partial\mathscr{M}(u,x)}(\phi_1-\phi_0) \\ &+\left(L-\frac{\partial L}{\partial\mathscr{M}(u,x)}\mathscr{M}(u,x)\right)(\xi_1-\xi_0)\Bigg]. \end{aligned} \tag{5.5}$$

As the partition of the interval $[a,b]$ introduces an ordering on the set Γ, we may thus define a shift operator and its inverse as

$$S(f(x_n)) = S(f_n) = f(x_{n+1}) = f_{n+1}, \tag{5.6}$$

$$S^{-1}(f(x_n)) = S^{-1}(f_n) = f(x_{n-1}) = f_{n-1}, \tag{5.7}$$

for any function f that is defined for both x_n and x_{n+1}, where the (5.6)–(5.7) define f_n and f_{n-1}. In the following, the composition of more shifts is denoted as

$$S^j f_n = \underbrace{S(S(\cdots S(f_n)))}_{j \text{ times}}, \quad S^{-j} f_n = \underbrace{S^{-1}(S^{-1}(\cdots S^{-1}(f_n)))}_{j \text{ times}}, \quad j \in \mathbb{N}.$$

In the derivation of the Euler–Lagrange equations of a smooth variational problem, we made extensive use of integration by parts, see Section 2.3. As we are now in a discrete setting, the shift operator allows for an analogous operation called summation by parts. Given two functions f, g that are defined at the points x_n, x_{n-1} and x_{n+1}, it holds

$$\sum_n f_n(g_{n+1} - g_n) = \sum_n (f_{n-1} - f_n) g_n + (S - \mathrm{id})(f_{n-1} g_n),$$

where id is the identity operator $\mathrm{id}(f_n) = f_n$. Applying summation by parts to (5.5) we obtain

$$\begin{aligned} 0 \;=\; \sum_\Gamma \Bigg\{ & \left[\frac{\partial L}{\partial x_0}(x_1 - x_0) + (S^{-1} - \mathrm{id}) \left(L - \frac{\partial L}{\partial \mathscr{M}(u,x)} \mathscr{M}(u,x) \right) \right] \xi_0 \\ & + \left[\frac{\partial L}{\partial u(x_0)}(x_1 - x_0) + (S^{-1} - \mathrm{id}) \frac{\partial L}{\partial \mathscr{M}(u,x)} \right] \phi_0 \\ & + (S - \mathrm{id}) \left[S^{-1} \left(L - \frac{\partial L}{\partial \mathscr{M}(u,x)} \mathscr{M}(u,x) \right) \xi_0 + S^{-1} \left(\frac{\partial L}{\partial \mathscr{M}(u,x)} \right) \phi_0 \right] \Bigg\}. \end{aligned} \tag{5.8}$$

Comparing (5.8) to the general smooth analogue,

$$0 = \int_a^b \left(\sum_\alpha Q^\alpha E^\alpha(L) + \frac{\mathrm{d}}{\mathrm{d}x} A \right),$$

we set

$$\begin{aligned} E^x(L) &= \frac{\partial L}{\partial x_0}(x_1 - x_0) + (S^{-1} - \mathrm{id}) \left(L - \frac{\partial L}{\partial \mathscr{M}(u,x)} \mathscr{M}(u,x) \right), \\ E^u(L) &= \frac{\partial L}{\partial u(x_0)}(x_1 - x_0) + (S^{-1} - \mathrm{id}) \frac{\partial L}{\partial \mathscr{M}(u,x)}, \\ A &= S^{-1} \left(L - \frac{\partial L}{\partial \mathscr{M}(u,x)} \mathscr{M}(u,x) \right) \xi_0 + S^{-1} \left(\frac{\partial L}{\partial \mathscr{M}(u,x)} \right) \phi_0. \end{aligned}$$

Remark 5.1.6. The appearance of an equation $E^x(L) = 0$ is justified by the fact that one can formulate (5.2) as

$$\mathscr{L}[x,u] = \int_{t_0}^{t_1} L \left(x(t), u(t), \frac{u_t}{x_t} \right) x_t \, \mathrm{d}t. \tag{5.9}$$

This sets x and u as dependent variables, and the standard result is

$$\begin{aligned} E^x(L) &= x_t \frac{\partial L}{\partial x} - \frac{\mathrm{d}}{\mathrm{d}t} \left[L - \frac{u_t}{x_t} D_3(L) \right], \\ E^u(L) &= x_t \frac{\partial L}{\partial u} - \frac{\mathrm{d}}{\mathrm{d}t} D_3(L), \\ A &= \left(L - u_x \frac{\partial L}{\partial u_x} \right) \xi + \frac{\partial L}{\partial u_x} \phi. \end{aligned}$$

where $D_3(L)$ is the standard way in the literature to express the partial derivative of L with respect to its third argument, $\dfrac{u_t}{x_t}$. It can be also noted that the continuum limit of the above quantities can be obtained in the multispace evaluating the same objects on the jet bundle.

Remark 5.1.7. No new information is introduced by writing (5.2) as (5.9). The smooth Euler Lagrange equations of (5.9) satisfy

$$u_t E^u(L) + x_t E^x(L) = 0,$$

by virtue of

$$\frac{\mathrm{d}L}{\mathrm{d}t} = x_t \frac{\partial L}{\partial x} + u_t \frac{\partial L}{\partial u} + D_3(L) \frac{\mathrm{d}}{\mathrm{d}t} \left(\frac{u_t}{x_t} \right).$$

On the other hand, the compatibility condition of the discrete Euler Lagrange equations is

$$\begin{aligned} \left(\mathrm{id} - S^{-1}\right) L \quad &= (x_1 - x_0) \left[\frac{\partial L}{\partial x_0} + \mathscr{M}(u,x) \frac{\partial L}{\partial u_0} \right] \\ &\quad + S^{-1} \frac{\partial L}{\partial \mathscr{M}(u,x)} \left(\mathrm{id} - S^{-1}\right) (\mathscr{M}(u,x)) \end{aligned},$$

showing that on solutions of the discrete Euler Lagrange equations, a relation having its continuum limit to that of the total derivative operator acting on L holds.

5.2 Discretisation of invariant Lagrangians

In this section we will show how moving frames can be used in multispaces similarly to the case where we the Lie group is acting on a jet bundle. We will use as a running example a variational problem that is invariant under the affine action of $SE(2)$. Unless otherwise stated, the notation for the multispace approximation to the derivative of u, computed at $x = x_0$, will be denoted as $\mathscr{M}(u_0)$, rather than $\mathscr{M}(u,x)|_{x=x_0}$. This allows for a more readable exposition as the shifts can be expressed simply through the index of the variable u_0. For instance, with this notation we have $S(\mathscr{M}(u,x)|_{x=x_0}) = \mathscr{M}(u_1)$.

Consider the multispace Lagrangian

$$\mathscr{L}_n = \sum L(x_0, u_0, \mathscr{M}(u_0))\, \Delta(x_0, x_1). \tag{5.10}$$

Recall that, if $g \in SE(2)$, the standard representation is

$$g = \begin{pmatrix} R_{\theta_0} & v_0 \\ 0 & 1 \end{pmatrix},$$

where $R_{\theta_0} = \begin{pmatrix} \cos(\theta_0) & -\sin(\theta_0) \\ \sin(\theta_0) & \cos(\theta_0) \end{pmatrix}$ is the rotational part of g, while $v_0^T = (a_0, b_0)$ is the translation vector. We consider the action of SE(2) on the multispace coordinates defined as

$$g \cdot \begin{pmatrix} x_0 \\ u_0 \end{pmatrix} = R_{\theta_0} \cdot \begin{pmatrix} x_0 \\ u_0 \end{pmatrix} + v_0. \tag{5.11}$$

We prolong the action to $\mathscr{M}(u_0)$ as follows:

$$\begin{aligned} g \cdot \mathscr{M}(u_0) &= \frac{g \cdot (u_1 - u_0)}{g \cdot (x_1 - x_0)} \\ &= \frac{\sin(\theta_0)(x_1 - x_0) + \cos(\theta_0)(u_1 - u_0)}{\cos(\theta_0)(x_1 - x_0) - \sin(\theta_0)(u_1 - u_0)} \\ &= \frac{\tan(\theta_0) + \mathscr{M}(u_0)}{1 - \tan(\theta_0)\mathscr{M}(u_0)}. \end{aligned}$$

Practically, moving frames can be introduced in multispaces using a similar procedure to the one described for the jet bundles. More details on the moving frames' construction for multispaces can be found in [44], along with a discussion on the invariants and the difference–differential relations between them. We will use later on these syzygies to derive discrete approximants to smooth generating invariants.

As this is a free and regular action in a suitable neighbourhood of $(x_0 = 0, u_0 = 0, \mathscr{M}(u_0) = 0)$, we can introduce a moving frame using the following normalisation equations:

$$g \cdot x_0 = 0, \qquad g \cdot u_0 = 0, \qquad g \cdot \mathscr{M}(u_0) = 0. \tag{5.12}$$

From (5.11) we obtain

$$\begin{cases} g \cdot x_0 = \cos(\theta_0)x_0 - \sin(\theta_0)u_0 + a_0 = 0 \\ g \cdot u_0 = \sin(\theta_0)x_0 + \cos(\theta_0)u_0 + b_0 = 0 \\ g \cdot \mathscr{M}(u_0) = \dfrac{\tan(\theta_0) + \mathscr{M}(u_0)}{1 - \tan(\theta_0)\mathscr{M}(u_0)} = 0. \end{cases} \tag{5.13}$$

We can solve equations (5.13) for the frame's parameters, obtaining

$$\begin{cases} \theta_0 = -\arctan(\mathscr{M}(u_0)) \\ a_0 = -\dfrac{\mathscr{M}(u_0)u_0 + x_0}{\sqrt{1 + \mathscr{M}(u_0)^2}} \\ b_0 = \dfrac{\mathscr{M}(u_0)x_0 - u_0}{\sqrt{1 + \mathscr{M}(u_0)^2}}. \end{cases} \tag{5.14}$$

Hence, the standard representation of the frame in multispace coordinates will be

$$\rho_0 = \begin{pmatrix} \dfrac{1}{\sqrt{1+\mathscr{M}(u_0)^2}} & \dfrac{\mathscr{M}(u_0)}{\sqrt{1+\mathscr{M}(u_0)^2}} & -\dfrac{\mathscr{M}(u_0)u_0 + x_0}{\sqrt{1+\mathscr{M}(u_0)^2}} \\ -\dfrac{\mathscr{M}(u_0)}{\sqrt{1+\mathscr{M}(u_0)^2}} & \dfrac{1}{\sqrt{1+\mathscr{M}(u_0)^2}} & \dfrac{\mathscr{M}(u_0)x_0 - u_0}{\sqrt{1+\mathscr{M}(u_0)^2}} \\ 0 & 0 & 1 \end{pmatrix}. \tag{5.15}$$

Comparing the frame obtained in the multispace setting to the one related to the smooth equivalent to action (5.11), which using the following normalisation equations

$$g \cdot x = 0, \qquad g \cdot u = 0, \qquad g \cdot u_x = 0$$

gives

$$\begin{cases} \theta = -\arctan(u_x) \\ a = -\dfrac{x+uu_x}{\sqrt{1+u_x^2}} \\ b = \dfrac{xu_x-u}{\sqrt{1+u_x^2}} \end{cases}, \tag{5.16}$$

we can see that the quantities (5.14) converge in limit to the ones in (5.16) as $x_1 - x_0 \to 0$ and $x_0 \to x$. As we have seen in previous chapters, the smooth equivalent action of (5.11), once the parametrisation through arc length has been been fixed, has a single generating differential invariant, κ. In order to find the multispace equivalent of κ, we let ρ_0 act on the point $(x_1, u_1, \mathscr{M}(u_1))$, which is the shift of the point that we considered in the normalisation equations (5.12). Namely, for the first two coordinates, we have

$$\begin{aligned} \rho_0 \cdot \begin{pmatrix} x_1 \\ u_1 \\ 1 \end{pmatrix} &= \begin{pmatrix} \frac{(u_1-u_0)\mathscr{M}(u_0)-x_0+x_1)}{\sqrt{(1+\mathscr{M}(u_0)^2)}} \\ \frac{(x_0-x_1)\mathscr{M}(u_0)-u_0+u_1)}{\sqrt{(1+\mathscr{M}(u_0)^2)}} \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} \ell_0 \\ 0 \\ 1 \end{pmatrix}, \end{aligned} \tag{5.17}$$

where in (5.17) we used (5.12) and we denoted with $\ell_0 = \sqrt{(x_1-x_0)^2+(u_1-u_0)^2}$, the discrete analogous of the arclength infinitesimal ds. Regarding the third coordinate, $\mathscr{M}(u_1)$,

we have

$$\begin{aligned}
\rho_0 \cdot \mathscr{M}(u_1) &= \frac{(x_2-x_1)\sin(\theta_0)+(u_2-u_1)\cos(\theta_0)}{(x_2-x_1)\cos(\theta_0)-(u_2-u_1)\sin(\theta_0)} \\
&= \frac{\tan(\theta_0)-\mathscr{M}(u_1)}{1-\tan(\theta_0)\mathscr{M}(u_1)} \\
&= -\frac{(S-\mathrm{id})\mathscr{M}(u_0)}{1+\mathscr{M}(u_0)\mathscr{M}(u_1)} \\
&= -\tan(\theta_1-\theta_0) \\
&= -\tan(\Delta\theta_0) \\
&= \kappa_0, \qquad (5.18)
\end{aligned}$$

where the last two equalities define $\Delta\theta_0$ and κ_0. Later in this chapter, we will use κ_0 and ℓ_0 to approximate the smooth generating differential invariant $\kappa = \frac{u_{xx}}{(1+u_x^2)^{3/2}}$. Now that we have found the invariants ℓ_0 and κ_0, we introduce a dummy variable, say t, and we want to derive expressions for $\frac{\partial \ell_0}{\partial t}$ and $\frac{\partial \kappa_0}{\partial t}$, as these will be necessary when performing the first variation of the Lagrangian. In fact now we can rewrite the Lagrangian (5.10) in terms of only κ_0 and ℓ_0 (and their shifts) and if we perform the total differentiation with respect to the variable t, we have

$$\frac{\partial L}{\partial t} = \frac{\partial L}{\partial \kappa_0}\frac{\partial \kappa_0}{\partial t} + \frac{\partial L}{\partial \ell_0}\frac{\partial \ell_0}{\partial t}. \qquad (5.19)$$

5.2.1 Discrete approximations to smooth differential invariants

Recall now that in the smooth case, given a moving frame $\rho(s)$, it is possible to define a curvature matrix as $\rho_s\rho^{-1}$. This is an element of the Lie algebra and its entries can be written in terms of only the generating differential invariants and their derivatives. Then, given Δs

"small enough", we can express the moving frame $\rho(s)$ as a Taylor series, namely

$$\rho(s+\Delta s) = \rho(s) + \Delta s \rho_s(s) + \mathscr{O}\left((\Delta s)^2\right).$$

Evaluating the smooth moving frame on points where the discrete one is defined, allows us to compare the two, i.e.

$$\begin{aligned}\rho_1 = \rho(s_0 + (s_1 - s_0)) &= \rho(s_0) + (s_1 - s_0)\rho_s(s_0) + \mathscr{O}\left((s_1 - s_0)^2\right) \\ &= \rho_0 + (s_1 - s_0)\rho_{0,s} + \mathscr{O}\left((s_1 - s_0)^2\right), \end{aligned} \tag{5.20}$$

where we used the notation $\rho_{0,s} = \rho_s(s_0)$. Even though the following is a general procedure and could be adapted to any other Lie group action, we restrict our attention to the case where $G = SE(2)$. In this case, as it is common practice to parametrise the (smooth) moving frame via the arclength, we identify $s_1 - s_0$ as the discrete version of ds, which we called above ℓ_0. Then we can rewrite (5.20) as

$$\rho_1 = \rho_0 + \ell_0 \rho_{0,s} + \mathscr{O}\left(\ell_0^2\right).$$

If we denote by $K_0 = \rho_1 \rho_0^{-1}$ and by I the identity matrix of the correct dimension, we have

$$K_0 = \rho_1 \rho_0^{-1} = I + \ell_0 \rho_{0,s} \rho_0^{-1} + \mathscr{O}\left(\ell_0^2\right). \tag{5.21}$$

Equation (5.21) can be read as $(K_0 - I)\ell_0^{-1}$ gives a first–order approximation to the smooth curvature matrix. This is going to be very important as it will give us a way to match the

smooth and discrete invariants. Explicitly, in this setting, the matrix K_0 is given by

$$K_0 = \begin{pmatrix} \dfrac{1+\mathscr{M}(u_0)\mathscr{M}(u_1)}{\sqrt{1+\mathscr{M}(u_0)^2}\sqrt{1+\mathscr{M}(u_1)^2}} & \dfrac{\mathscr{M}(u_1)-\mathscr{M}(u_0)}{\sqrt{1+\mathscr{M}(u_0)^2}\sqrt{1+\mathscr{M}(u_1)^2}} & -R_{\theta_1}\begin{pmatrix} x_1-x_0 \\ u_1-u_0 \end{pmatrix} \\ -\dfrac{\mathscr{M}(u_1)-\mathscr{M}(u_0)}{\sqrt{1+\mathscr{M}(u_0)^2}\sqrt{1+\mathscr{M}(u_1)^2}} & \dfrac{1+\mathscr{M}(u_0)\mathscr{M}(u_1)}{\sqrt{1+\mathscr{M}(u_0)^2}\sqrt{1+\mathscr{M}(u_1)^2}} & \\ 0 & 0 & 1 \end{pmatrix}.$$

Using the fact that the $(3,3)-$minor of K_0 must be equal to 1, we have the following simplification in terms of the invariant κ_0

$$\sqrt{1+\mathscr{M}(u_0)^2}\sqrt{1+\mathscr{M}(u_1)^2} = (1+\mathscr{M}(u_0)\mathscr{M}(u_1))\left(\sqrt{1+\kappa_0^2}\right). \tag{5.22}$$

Another simplification can be performed on the matrix K:

$$\begin{aligned} R_{\theta_1}\begin{pmatrix} x_1-x_0 \\ u_1-u_0 \end{pmatrix} &= R_{\Delta\theta_0}R_{\theta_0}\begin{pmatrix} x_1-x_0 \\ u_1-u_0 \end{pmatrix} \\ &= R_{\Delta\theta_0}\begin{pmatrix} \ell_0 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} \dfrac{\ell_0}{\sqrt{1+\kappa_0^2}} \\ -\dfrac{\ell_0\kappa_0}{\sqrt{1+\kappa_0^2}} \end{pmatrix} = \begin{pmatrix} \cos(\Delta\theta_0)\ell_0 \\ \sin(\Delta\theta_0)\ell_0 \end{pmatrix}. \end{aligned} \tag{5.23}$$

Therefore we can express all the entries of the matrix K in terms of only κ_0 and ℓ_0, as expected from the comparison with the smooth curvature matrix. Substituting both (5.22)–(5.23) into

the matrix K_0, we have

$$K_0 = \begin{pmatrix} \frac{1}{\sqrt{1+\kappa_0^2}} & \frac{\kappa_0}{\sqrt{1+\kappa_0^2}} & -\frac{\ell_0}{\sqrt{1+\kappa_0^2}} \\ -\frac{\kappa_0}{\sqrt{1+\kappa_0^2}} & \frac{1}{\sqrt{1+\kappa_0^2}} & \frac{\ell_0\kappa_0}{\sqrt{1+\kappa_0^2}} \\ 0 & 0 & 1 \end{pmatrix}.$$

Another useful way to express the components of the matrix K is in terms of $\Delta\theta_0$ and ℓ_0.

$$K_0 = \begin{pmatrix} \cos(\Delta\theta_0) & -\sin(\Delta\theta_0) & -\ell_0\cos(\Delta\theta_0) \\ \sin(\Delta\theta_0) & \cos(\Delta\theta_0) & -\ell_0\sin(\Delta\theta_0) \\ 0 & 0 & 1 \end{pmatrix} \tag{5.24}$$

$$= \begin{pmatrix} R_{\Delta\theta_0} & -R_{\Delta\theta_0}\begin{pmatrix} \ell_0 \\ 0 \end{pmatrix} \\ 0 \quad\quad 1 \end{pmatrix}.$$

A discrete approximation of the smooth generating differential invariant κ can be recovered directly by the comparison between K_0 and the curvature matrix $\rho_s\rho^{-1}$. In this setting we have

$$\rho_s\rho^{-1} = \begin{pmatrix} 0 & \kappa & -1 \\ -\kappa & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{5.25}$$

Substituting (5.25) and (5.24) into (5.21) we obtain a first–order discrete approximation of κ as

$$\kappa = -\frac{\sin(\Delta\theta_0)}{\ell_0} + \mathscr{O}(\ell_0). \tag{5.26}$$

5.2.2 Higher order approximations to smooth differential invariants

We can derive higher order approximations of κ if we take more terms in the expansion of K_0 in (5.21). First we consider one more term to expansion (5.21)

$$K = \rho_1\rho_0^{-1} = I + \ell_0\rho_{0,s}\rho_0^{-1} + \frac{\ell_0^2}{2}\rho_{0,ss}\rho^{-1} + \mathscr{O}(\ell_0^3). \tag{5.27}$$

Then we expand $\rho_{-1}\rho_0^{-1}$ around $s = s_0$. Recall that with the notation that has been used so far, $\ell_{-1} = S^{-1}\ell_0$ and $\rho_{-1} = S^{-1}\rho_0$ and we assume that these quantities are well–defined.

$$\rho_{-1}\rho_0^{-1} = I - \ell_{-1}\rho_{0,s}\rho_0^{-1} + \frac{\ell_{-1}^2}{2}\rho_{0,ss}\rho_0^{-1} + \mathscr{O}(\ell_{-1}^3). \tag{5.28}$$

After suppressing the quadratic terms in ℓ_0 and ℓ_{-1} in (5.27)–(5.28), we are left with

$$\ell_{-1}^2\rho_1\rho_0^{-1} - \ell_0^2\rho_{-1}\rho_0^{-1} = (\ell_0^2 - \ell_{-1}^2)I + \ell_0\ell_{-1}(\ell_0 + \ell_{-1})\rho_{0,s}\rho_0^{-1} + \mathscr{O}(\ell_0^3\ell_{-1}^2, \ell_{-1}^3\ell_0^2).$$

Isolating $\rho_{0,s}\rho_0^{-1}$ we obtain

$$\rho_{0,s}\rho_0^{-1} = \left(\frac{\ell_{-1}^2\rho_1\rho_0^{-1} - \ell_0^2\rho_{-1}\rho_0^{-1} - (\ell_0^2 - \ell_{-1}^2)I}{\ell_0\ell_{-1}(\ell_0 + \ell_{-1})}\right) + \frac{1}{\ell_0 + \ell_{-1}}\mathscr{O}(\ell_0^2\ell_{-1}, \ell_{-1}^2\ell_0). \tag{5.29}$$

This is a second–order approximation in ℓ_0 to the curvature matrix (5.25), from which we can obtain a second–order approximation for the invariant κ. Computing the $(1,2)$ entry in the first term of the right-hand side of (5.29) gives

$$\left(\frac{\ell_{-1}^2\rho_1\rho_0^{-1} - \ell_0^2\rho_{-1}\rho_0^{-1} - (\ell_0^2 - \ell_{-1}^2)I}{\ell_0\ell_{-1}(\ell_0 + \ell_{-1})}\right)_{(1,2)} = -\frac{(\ell_0^2 + \ell_{-1}^2)\left(\sin(\Delta\theta_0) + \sin(\Delta\theta_{-1})\right)}{\ell_0\ell_{-1}(\ell_0 + \ell_{-1})} \tag{5.30}$$

$$= \kappa + \frac{1}{\ell_0 + \ell_{-1}}\mathscr{O}(\ell_0^2\ell_{-1}, \ell_{-1}^2\ell_0).$$

Equation (5.30) depends on $\ell_0, \ell_{-1}, \Delta\theta_0, \Delta\theta_{-1}$, rather than just on ℓ_0 and $\Delta\theta_0$. Hence, the approximation in (5.30) can be used to produce a more accurate numerical scheme for Lagrangians depending on κ, at the cost of having to solve for a higher–order system of difference equations. In this way, invariants of any order of approximation can be built, just taking in consideration more variables and adding the relevant terms in the Taylor expansion. It can be seen, that for a nth–order approximant to κ, n points have to be considered and for each of them $n+1$ terms are needed in the relative Taylor expansion.

5.2.3 Discrete approximation of derivatives of invariants

We focus now on a slightly different task. So far we have seen how to compute approximations of any order to the generating differential invariants. This construction is useful when we have a Lagrangian that is first–order (in the sense of the order of a Lagrangian). However, if we want to consider variational problems defined by higher–order Lagrangians, we need to also approximate the derivatives of the generating differential invariants. This will be enough, as we have seen already that a Lagrangian that is invariant under some Lie group action, can be rewritten in terms of only the generating differential invariants and their derivatives.

In the case of a multispace variational problem invariant under the affine action of $SE(2)$, recall the system

$$\begin{cases} \rho_1\rho_0^{-1} = I + \ell_0\rho_{0,s}\rho_0^{-1} + \frac{\ell_0^2}{2}\rho_{0,ss}\rho^{-1} + \mathscr{O}(\ell_0^3) \\ \rho_{-1}\rho_0^{-1} = I - \ell_{-1}\rho_{0,s}\rho_0^{-1} + \frac{\ell_{-1}^2}{2}\rho_{0,ss}\rho^{-1} + \mathscr{O}(\ell_{-1}^3) \end{cases}$$

If we instead of suppressing the second–order term in ℓ_0, ℓ_{-1}, we remove the first–order one, we obtain

$$\frac{2}{\ell_0\ell_{-1}(\ell_0+\ell_{-1})}\left(\ell_{-1}\rho_1\rho_0^{-1} + \ell_0\rho_{-1}\rho_0^{-1} - (\ell_0+\ell_{-1})I\right) = \rho_{0,ss}\rho_0^{-1} + \frac{1}{\ell_0+\ell_{-1}}\mathscr{O}(\ell_0^2, \ell_{-1}^2) \tag{5.31}$$

which is a first–order approximation to $\rho_{0,ss}\rho_0^{-1}$. The matrix $\rho_{0,ss}\rho^{-1}$ is not a curvature matrix anymore, as it was the case of $\rho_{0,s}\rho_0^{-1}$ in the first–order expansion of K_0 in (5.21). In fact, $\rho_{0,ss}\rho_0^{-1}$ is not even a map to the Lie Algebra. A priori, it is not guaranteed that all the element of $\rho_{0,ss}\rho_0^{-1}$ could be written in terms of only κ and its derivatives. However we show that this is the case, namely

$$\begin{aligned}\frac{d}{ds}\left(\rho_{0,s}\rho_0^{-1}\right) &= \rho_{0,ss}\rho_0^{-1} - \rho_{0,s}\rho_0^{-1}\rho_{0,s}\rho_0^{-1} \\ &= \rho_{0,ss}\rho_0^{-1} - \left(\rho_{0,s}\rho_0^{-1}\right)^2,\end{aligned}$$

hence

$$\rho_{0,ss}\rho_0^{-1} = \frac{d}{ds}\left(\rho_{0,s}\rho_0^{-1}\right) + \left(\rho_{0,s}\rho_0^{-1}\right)^2, \tag{5.32}$$

where the right–hand side of (5.32) can be written in terms only of the invariant κ and its derivatives as it is a function of only the curvature matrix. This result can be generalised as follows.

Lemma 5.2.1. *Suppose $\rho(s)$ is a smooth moving frame and denote $\left(\frac{d^n}{ds^n}\rho\right) = \rho^{(n)}$, for $n \geq 1$. Then $\rho^{(n)}\rho^{-1}$ can be written in terms of only the invariants of the action and its derivatives.*

Proof. This can be proved by induction. For the case where $n = 1$ a proof can be found in [42, Lemma 5.2.1].

If the statement holds for $n = j$, then we have

$$\frac{d}{ds}\left(\rho^{(j)}\rho^{-1}\right) = \rho^{(j+1)}\rho^{-1} - \rho^{(j)}\rho^{-1}\rho^{(1)}\rho^{-1},$$

hence

$$\rho^{(j+1)}\rho^{-1} = \frac{d}{ds}\left(\rho^{(j)}\rho^{-1}\right) + \rho^{(j)}\rho^{-1}\rho^{(1)}\rho^{-1}, \tag{5.33}$$

where the right–hand side can be written in terms of only the invariants of the action and its derivatives because of the inductive hypothesis and the case $n = 1$. □

Example 5.2.2. For practical purposes, now that we know that $\rho^{(n)}\rho_0^{-1}$ contains the invariant κ and its derivatives, we need to explore if these expressions can lead us to approximants for the derivatives of κ. Let us compute the first few elements of $\rho^{(n)}\rho^{-1}$, say for $n = 2,3$. We have for $n = 2$

$$\rho^{(2)}\rho^{-1} = \begin{pmatrix} -\kappa^2 & \kappa_s & 0 \\ -\kappa_s & -\kappa^2 & \kappa \\ 0 & 0 & 0 \end{pmatrix}, \tag{5.34}$$

and its discrete counterpart from (5.31), which for layout reasons we will write as

$$2\begin{pmatrix} A & B & C \\ -B & A & D \\ 0 & 0 & 0 \end{pmatrix},$$

where

$$\begin{cases} A = \dfrac{\ell_{-1}\cos(\Delta\theta_0) + \ell_0\cos(\Delta\theta_{-1}) - \ell_{-1} - \ell_0)}{(\ell_{-1}^2\ell_0 + \ell_{-1}\ell_0^2)} \\ B = \dfrac{-\ell_{-1}\sin(\Delta\theta_0) + \ell_0\sin(\Delta\theta_{-1})}{\ell_{-1}^2\ell_0 + \ell_{-1}\ell_0^2} \\ C = \dfrac{-\cos(\Delta\theta_0) + 1}{\ell_0 + \ell_{-1}} \\ D = -\dfrac{\sin(\Delta\theta_0)}{\ell_0 + \ell_{-1}}. \end{cases} \tag{5.35, 5.36}$$

Comparing the $(2,3)$ entry in (5.34) and (5.36), we could construct an approximation for κ, but we have already seen how to do it efficiently and for higher order than this. However, comparing the $(1,2)$ entry in (5.34) and (5.35), we obtain a first–order approximation of κ_s,

that could be used to study first–order (in the sense of the order of a Lagrangian) Lagrangians. The approximation is

$$\kappa_s = 2\frac{-\ell_{-1}\sin(\Delta\theta_0)+\ell_0\sin(\Delta\theta_{-1})}{\ell_{-1}^2\ell_0+\ell_{-1}\ell_0^2}+\frac{1}{\ell_0+\ell-1}\mathscr{O}(\ell_{-1}^2,\ell_0^2).$$

For $n=3$, we have

$$\rho^{(3)}\rho^{-1}=\begin{pmatrix}-3\kappa\kappa_s & \kappa_{ss}-\kappa^3 & \kappa^2\\ -\kappa_{ss}+\kappa^3 & -3\kappa\kappa_s & 2\kappa_s\\ 0 & 0 & 0\end{pmatrix}. \tag{5.37}$$

This time we are interested in approximating κ_{ss}. Consider the $(1,2)$ entry in (5.37). It can be noticed that we do not have expressions in only the invariant we want to approximate, κ_{ss}, rather also κ is appearing. Hence, we need to substitute a suitable approximation of κ so that we can isolate κ_{ss}. The order of the approximation of κ that is needed depends on the required order of accuracy of the discrete analogue of κ_{ss}. In the case of (5.37), we are looking for a first–order approximant to κ_{ss}, so we need at least a first–order expression for κ. Any further degree of accuracy in κ would be lost and hence is not worth taking into account a more sophisticated formula. As we want to eliminate the first– and second–order terms in the Taylor expansion of $\rho(s_0+\ell_i)\rho_0^{-1}$, for $i=-1,0,1$, we will now consider a system of three Taylor expansions around s_0. We could also consider a system of four expansions obtaining a central difference approximant. We have

$$\begin{cases}\rho_1\rho_0^{-1}=I+\ell_0\rho_{0,s}\rho_0^{-1}+\dfrac{\ell_0^2}{2}\rho_{0,ss}\rho_0^{-1}+\dfrac{\ell_0^3}{6}\rho_{0,sss}\rho_0^{-1}+\mathscr{O}(\ell_0^4)\\ \rho_{-1}\rho_0^{-1}=I-\ell_{-1}\rho_{0,s}\rho_0^{-1}+\dfrac{\ell_{-1}^2}{2}\rho_{0,ss}\rho_0^{-1}-\dfrac{\ell_{-1}^3}{6}\rho_{0,sss}\rho_0^{-1}+\mathscr{O}(\ell_{-1}^4)\\ \rho_2\rho_0^{-1}=I+(\ell_0+\ell_1)\rho_{0,s}\rho_0^{-1}+\dfrac{(\ell_0+\ell_1)^2}{2}\rho_{0,ss}\rho_0^{-1}+\dfrac{(\ell_0+\ell_1)^3}{6}\rho_{0,sss}\rho_0^{-1}+\mathscr{O}((\ell_0+\ell_1)^4).\end{cases} \tag{5.38}$$

We can eliminate the terms in $\rho_{0,s}\rho_0^{-1}$ and $\rho_{0,ss}\rho_0^{-1}$ and we are left with

$$\rho_0^{(3)}\rho_0^{-1} = \frac{1}{(\ell_0+\ell_{-1}+\ell_1)(\ell_0+\ell_{-1})\ell_1}\Big(-6(\ell_0+\ell_1+\ell_{-1})\ell_0^{-1}\rho_1\rho_0^{-1}+\ell_1\ell_{-1}^{-1}\rho_{-1}\rho_0^{-1}$$

$$-(\ell_0+\ell_{-1})(\ell_0+\ell_1)^{-1}-(\ell_0+\ell_{-1})(\ell_0+\ell_{-1}+\ell_1)\ell_0^{-1}\ell_{-1}^{-1}(\ell_0+\ell_1)^{-1}I\Big). \quad (5.39)$$

Entry $(1,2)$ in (5.39) will give us a first–order approximation of $\kappa_{ss}-\kappa^3$. Considering again system (5.38), we could also eliminate the first– and third–order terms, obtaining a second–order approximation for $\rho_{0,ss}\rho_0^{-1}$, which we have seen containing κ_s. This strategy can be carried forward, meaning that we can construct discrete approximants to κ_s with any order of accuracy. The same holds for κ_{ss}, and in general, discrete approximants of any order of accuracy can be built for any derivative of κ.

Remark 5.2.3. Although we used the Lie group $SE(2)$ as a running example, this method to approximate differential invariants and their derivatives using Taylor approximations of $\rho^{(n)}\rho^{-1}$ is valid for any matrix Lie group.

5.3 Lagrangians invariant under $SE(2)$

5.3.1 Difference–Differential Syzygies

As we have shown how it is possible to approximate the smooth generating differential invariants, and their derivatives, we now go back where we left the computations for the Euler–Lagrange equations in the $SE(2)$ case. We ended the presentation with equation (5.19), where a dummy variable t was introduced to perform the Lagrangian variation. The

introduction of t generates two new invariants, namely

$$\begin{cases} \rho_0 \cdot \dfrac{\partial x_0}{\partial t} = \sigma_0^x \\ \rho_0 \cdot \dfrac{\partial u_0}{\partial t} = \sigma_0^u. \end{cases}$$

Consider now the matrix

$$N_0 = \rho_{0,t}\rho_0^{-1}$$

and observe that it's related to $K_0 = \rho_1\rho_0^{-1}$ in the following way

$$\begin{aligned} \frac{\partial K_0}{\partial t} &= \rho_{1,t}\rho_0^{-1} - \rho_1\rho_0^{-1}\rho_{0,t}\rho_0^{-1} \\ &= \rho_{1,t}\rho_1^{-1}\rho_1\rho_0^{-1} - \rho_1\rho_0^{-1}\rho_{0,t}\rho_0^{-1} \\ &= N_1 \cdot K_0 - K_0 \cdot N_0. \end{aligned} \tag{5.40}$$

This means that if we compute N_0, and this is something that can be done directly by hand or using *Indiff*, [43], we obtain expressions for the derivatives of the invariants κ_0 and ℓ_0 with respect to t. These derivatives play a crucial role in the derivation of the invariantised Euler–Lagrange equations, as we have already mentioned above. In this context the matrix N_0 is

$$N_0 = \begin{pmatrix} 0 & \dfrac{\rho_0 \cdot u_{1,t} - \sigma_0^u}{\ell_0} & 0 \\ -\dfrac{\rho_0 \cdot u_{1,t} - \sigma_0^u}{\ell_0} & 0 & -\sigma_0^u \\ 0 & 0 & 0 \end{pmatrix}.$$

The next step is to express $\rho_0 \cdot u_{1,t}$ in terms of σ_0^x, σ_0^u and their shifts. Let us rewrite it as

$$\begin{aligned}
\rho_0 \cdot \begin{pmatrix} x_{1,t} \\ u_{1,t} \\ 1 \end{pmatrix} &= \left(\rho_0 \rho_1^{-1}\right) \rho_1 \begin{pmatrix} x_{1,t} \\ u_{1,t} \\ 1 \end{pmatrix} \\
&= K_0^{-1} S \left(\rho_0 \cdot \begin{pmatrix} x_{0,t} \\ u_{0,t} \\ 1 \end{pmatrix} \right) \\
&= K_0^{-1} \begin{pmatrix} \sigma_1^x \\ \sigma_1^u \\ 1 \end{pmatrix},
\end{aligned}$$

then we have

$$\rho_0 \cdot \begin{pmatrix} x_{1,t} \\ u_{1,t} \\ 1 \end{pmatrix} = K_0^{-1} \begin{pmatrix} \sigma_1^x \\ \sigma_1^u \\ 1 \end{pmatrix} \tag{5.41}$$

$$\begin{aligned}
&= \begin{pmatrix} \dfrac{1}{\sqrt{1+\kappa_0^2}} & -\dfrac{\kappa_0}{\sqrt{1+\kappa_0^2}} & \ell_0 \\ \dfrac{\kappa_0}{\sqrt{1+\kappa_0^2}} & \dfrac{1}{\sqrt{1+\kappa_0^2}} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \sigma_1^x \\ \sigma_1^u \\ 1 \end{pmatrix} \\
&= \frac{1}{\sqrt{1+\kappa_0^2}} \begin{pmatrix} \sigma_1^x - \kappa_0 \sigma_1^u + \ell_0 \\ \kappa_0 \sigma_1^x + \sigma_1^u \\ \sqrt{1+\kappa_0^2} \end{pmatrix}.
\end{aligned} \tag{5.42}$$

Comparing the left hand side of (5.41) to the right hand side of (5.42), we find that

$$\rho_0 \cdot u_{1,t} = \frac{(\kappa_0 \sigma_1^x + \sigma_1^u)}{\sqrt{1+\kappa_0^2}}. \tag{5.43}$$

Equation (5.43) is an expressions dependent only on κ_0, ℓ_0, σ_0^x, σ_0^u and their shifts, as desired. If we insert (5.43) into N_0, and use N_0 to compute equation (5.40), we obtain the following formulae for $\frac{\partial \kappa_0}{\partial t}$ and $\frac{\partial \ell_0}{\partial t}$:

$$\begin{cases} \frac{\partial \kappa_0}{\partial t} = A\sigma_0^u + B\sigma_1^u + C\sigma_2^u + D\sigma_1^x + E\sigma_2^x \\ \frac{\partial \ell_0}{\partial t} = -\sigma_0^x + \frac{\sigma_1^x}{\sqrt{1+\kappa_0^2}} - \frac{\kappa_0}{\sqrt{1+\kappa_0^2}}\sigma_1^u = -\sigma_0^x + \cos(\Delta\theta_0)\sigma_1^x + \sin(\Delta\theta_0)\sigma_1^u, \end{cases} \tag{5.44}$$

where

$$\begin{cases} A &= \frac{1+\kappa_0^2}{\ell_0} \\ B &= -\frac{\ell_0\kappa_0^2 + \ell_1\sqrt{1+\kappa_0^2} + \ell_0}{\ell_0\ell_1} \\ C &= \frac{1+\kappa_0^2}{\ell_1\sqrt{1+\kappa_1^2}} \\ D &= -\frac{\kappa_0\sqrt{1+\kappa_0^2}}{\ell_0} \\ E &= \frac{\kappa_1(1+\kappa_0^2)}{\ell_1\sqrt{1+\kappa_1^2}}. \end{cases}$$

As the expression for $\frac{\partial \kappa_0}{\partial t}$ is quite complicated and can easily lead to some errors in the numerical implementation stage, we simplify it. Recall from (5.18) that $\kappa_0 = -\tan(\Delta\theta_0)$. If

we substitute for $\kappa_0 = -\tan(\Delta\theta_0)$ in the coefficients A,B,C,D,E above, we obtain

$$\begin{aligned}\frac{\partial \tan(\Delta\theta_0)}{\partial t} &= \frac{-1}{\cos(\Delta\theta_0)^2}\frac{\partial \Delta\theta_0}{\partial t} \\ &= \frac{1}{\cos^2(\Delta\theta_0)}\left(-\frac{\sigma_0^u}{\ell_0} + \frac{(\ell_0 - \ell_1\cos(\Delta\theta_0))\sigma_1^u}{\ell_1\ell_0} + \frac{\cos(\Delta\theta_1)\sigma_2^u}{\ell_1} - \frac{\sin(\Delta\theta_0)\sigma_1^x}{\ell_0} + \frac{\sin(\Delta\theta_1)\sigma_2^x}{\ell_1}\right) \\ &= \frac{1}{\cos(\Delta\theta_0)^2}(S-\mathrm{id})\left[\frac{1}{\ell_0}\left(\sigma_0^u - \cos(\Delta\theta_0)\sigma_1^u + \sin(\Delta\theta_0)\sigma_1^x\right)\right], \qquad (5.45)\end{aligned}$$

hence

$$\frac{\partial \Delta\theta_0}{\partial t} = (S-\mathrm{id})\left[\frac{1}{\ell_0}\left(\sigma_0^u - \cos(\Delta\theta_0)\sigma_1^u + \sin(\Delta\theta_0)\sigma_1^x\right)\right].$$

5.3.2 Euler–Lagrange equations and Conservation laws

Now that we have a more compact expression for the derivatives of the invariants with respect to t, we derive the invariantised Euler-Lagrange equations for the Lagrangian

$$\mathscr{L}_n = \sum L(\Delta\theta_0, \ell_0)\ell_0 \qquad (5.46)$$

where we rewrote the Lagrangian in terms of the invariants needed to approximate the generating differential invariant κ. Computations are as follows:

$$\begin{aligned}\sum \frac{\partial L}{\partial t} &= \sum \frac{\partial L}{\partial \Delta\theta_0}\frac{\partial \Delta\theta_0}{\partial t} + \frac{\partial L}{\partial \ell_0}\frac{\partial \ell_0}{\partial t} \\ &= \sum \frac{\partial L}{\partial \Delta\theta_0}\left((S-\mathrm{id})\left[\frac{1}{\ell_0}\left(\sigma_0^u - \cos(\Delta\theta_0)\sigma_1^u + \sin(\Delta\theta_0)\sigma_1^x\right)\right]\right) + \qquad (5.47) \\ &\quad + \frac{\partial L}{\partial \ell_0}\left(-\sigma_0^x + \cos(\Delta\theta_0)\sigma_1^x + \sin(\Delta\theta_0)\sigma_1^u\right). \qquad (5.48)\end{aligned}$$

Now we perform a first summation by parts in order to remove the $(S-\mathrm{id})$ operator. Recall that given two functions f,g, it holds

$$fSg = gS^{-1}f + (S-\mathrm{id})(gS^{-1}f), \tag{5.49}$$

$$fS^2g = gS^{-2}f + (S-\mathrm{id})(SgS^{-1}f + gS^{-2}f). \tag{5.50}$$

Equation (5.48) becomes

$$\begin{aligned}
&\sum \frac{\partial L}{\partial \Delta\theta_0}\left((S-\mathrm{id})\left[\frac{1}{\ell_0}(\sigma_0^u - \cos(\Delta\theta_0)\sigma_1^u + \sin(\Delta\theta_0)\sigma_1^x)\right]\right)\\
&+\frac{\partial L}{\partial \ell_0}(-\sigma_0^x + \cos(\Delta\theta_0)\sigma_1^x + \sin(\Delta\theta_0)\sigma_1^u)\\
&=\sum\left(\frac{S^{-1}-\mathrm{id}}{\ell_0}\right)\left(\frac{\partial L}{\partial \Delta\theta_0}\right)[(\sigma_0^u - \cos(\Delta\theta_0)\sigma_1^u + \sin(\Delta\theta_0)\sigma_1^x)]\\
&+\frac{\partial L}{\partial \ell_0}(-\sigma_0^x + \cos(\Delta\theta_0)\sigma_1^x + \sin(\Delta\theta_0)\sigma_1^u)\\
&+(S-\mathrm{id})\left(\frac{1}{\ell_0}(\sigma^u - \cos(\Delta\theta_0)\sigma_1^u + \sin(\Delta\theta_0)\sigma_1^x)S^{-1}\left(\frac{\partial L}{\partial \Delta\theta_0}\right).\right)
\end{aligned} \tag{5.51}$$

The next step is to perform another summation by parts in order to obtain something in the following form

$$\sum \frac{\partial L}{\partial t} = \sum E^x(L)\sigma_0^x + E^u(L)\sigma_0^u + (S-\mathrm{id})(BT1), \tag{5.52}$$

where we called 'BT1' the boundary terms arising from performing the two summation by parts and (5.52) defines $E^x(L)$, $E^u(L)$, that stand for the invariantised Euler-Lagrange

equations. Performing another summation by parts in (5.51) we obtain

$$\sum \left(\frac{S^{-1}-\mathrm{id}}{\ell_0}\right)\left(\frac{\partial L}{\partial \Delta\theta_0}\right)(\sigma^u-\cos(\Delta\theta_0)S\sigma^u+\sin(\Delta\theta_0)S\sigma^x)$$

$$+\frac{\partial L}{\partial \ell_0}(-\sigma_0^x+\cos(\Delta\theta_0)\sigma_1^x+\sin(\Delta\theta_0)\sigma_1^u)$$

$$+(S-\mathrm{id})\left(\frac{1}{\ell_0}(\sigma^u-\cos(\Delta\theta_0)\sigma_1^u+\sin(\Delta\theta_0)\sigma_1^x)S^{-1}\left(\frac{\partial L}{\partial \Delta\theta_0}\right)\right)$$

$$=\sum \sigma_0^u\left(\frac{(S^{-1}-\mathrm{id})}{\ell_0}\left(\frac{\partial L}{\partial \Delta\theta_0}\right)+S^{-1}\left(-\frac{\cos\Delta\theta_0}{\ell_0}(S^{-1}-\mathrm{id})\left(\frac{\partial L}{\partial \Delta\theta_0}\right)+\sin\Delta\theta_0\left(\frac{\partial L}{\partial \ell_0}\right)\right)\right)$$

$$+\sigma_0^x\left(-\frac{\partial L}{\partial \ell_0}+S^{-1}\left(\sin\Delta\theta_0\frac{(S^{-1}-\mathrm{id})}{\ell_0}\left(\frac{\partial L}{\partial \Delta\theta_0}\right)+\cos\Delta\theta_0\left(\frac{\partial L}{\partial \ell_0}\right)\right)\right)$$

$$+(S-\mathrm{id})\left(\frac{1}{\ell_0}(\sigma_0^u-\cos(\Delta\theta_0)\sigma_1^u+\sin(\Delta\theta_0)\sigma_0^x)S^{-1}\left(\frac{\partial L}{\partial \Delta\theta_0}\right)\right. \tag{5.53}$$

$$+\sigma_0^u S^{-1}\left(-\frac{\cos\Delta\theta_0}{\ell_0}(S^{-1}-\mathrm{id})\left(\frac{\partial L}{\partial \Delta\theta_0}\right)+\sin(\Delta\theta_0)\frac{\partial L}{\partial \ell_0}\right) \tag{5.54}$$

$$\left.+\sigma_0^x S^{-1}\left(\sin\Delta\theta_0\frac{(S^{-1}-\mathrm{id})}{\ell_0}\left(\frac{\partial L}{\partial \Delta\theta_0}\right)+\cos(\Delta\theta_0)\frac{\partial L}{\partial \ell_0}\right)\right). \tag{5.55}$$

Since we performed the variation with respect to σ_0^x and σ_0^u, then the Euler–Lagrange equations are

$$E^u(L)=\frac{(S^{-1}-\mathrm{id})}{\ell_0}\left(\frac{\partial L}{\partial \Delta\theta_0}\right)+S^{-1}\left(-\frac{\cos\Delta\theta_0}{\ell_0}(S^{-1}-\mathrm{id})\left(\frac{\partial L}{\partial \Delta\theta_0}\right)+\sin\Delta\theta_0\left(\frac{\partial L}{\partial \ell_0}\right)\right)=0, \tag{5.56}$$

$$E^x(L)=-\frac{\partial L}{\partial \ell_0}+S^{-1}\left(\sin\Delta\theta_0\frac{(S^{-1}-\mathrm{id})}{\ell_0}\left(\frac{\partial L}{\partial \Delta\theta_0}\right)+\cos\Delta\theta_0\left(\frac{\partial L}{\partial \ell_0}\right)\right)=0. \tag{5.57}$$

Aiming at a more clear and insightful presentation of the the Euler–Lagrange equations, we define a vector Z as

$$Z_0 = \begin{pmatrix} \left(\frac{S^{-1}-\mathrm{id}}{\ell_0}\right)\left(\frac{\partial L}{\partial \Delta\theta_0}\right) \\ \frac{\partial L}{\partial \ell_0} \end{pmatrix}. \tag{5.58}$$

If we consider (5.57) and the opposite of (5.56), then the Euler–Lagrange equations can be written as

$$\begin{pmatrix} E^u(L) \\ E^x(L) \end{pmatrix} = S^{-1}\left(R_{\Delta\theta_0} Z_0\right) - Z_0 = 0, \tag{5.59}$$

which is equivalent, applying the shift operator to both sides, to

$$R_{\Delta\theta_0} Z_0 = Z_1.$$

Solutions of the Euler–Lagrange equations give us the invariants $\Delta\theta_0$ and ℓ_0 that we can use to recover the curve that extremises the Lagrangian (5.46). Recall that

$$\Delta\theta_0 = \theta_1 - \theta_0 \qquad \Longrightarrow \qquad \theta_1 = \Delta\theta_0 + \theta_0,$$

hence, given a point (x_0, u_0, θ_0) as initial datum, we can find the solution curve recursively as

$$\begin{cases} x_1 = x_0 + \ell_0 \cos(\theta_0) \\ u_1 = u_0 + \ell_0 \sin(\theta_0). \end{cases}$$

If we take a closer look at the boundary terms in (5.53)–(5.54)–(5.55), we can express them in terms of the invariantised matrix of infinitesimal of the Lie group action and the adjoint representation of the moving frame we defined for that action. If we suppose that the Euler-Lagrange equations are satisfied, then the boundary terms will give us the expression of the

conservation laws arising from the Lie group symmetry, namely

$$\frac{1}{\ell_0}\left(\sigma_0^u-\cos(\Delta\theta_0)\sigma_1^u+\sin(\Delta\theta_0)\sigma_1^x\right)S^{-1}\left(\frac{\partial L}{\partial\Delta\theta_0}\right)+S^{-1}\left(R_{\Delta\theta_0}Z_0\right)^T\begin{pmatrix}\sigma_0^u\\ \sigma_0^x\end{pmatrix}$$

$$=\sigma_0^x\left(S^{-1}\left(\sin(\Delta\theta_0)\frac{(S^{-1}-\mathrm{id})}{\ell_0}\left(\frac{\partial L}{\partial\Delta\theta_0}\right)\right)+\cos(\Delta\theta_0)\frac{\partial L}{\partial\ell_0}\right)$$

$$+\sigma_0^u\left(S^{-1}\left(-\cos(\Delta\theta_0)\frac{(S^{-1}-\mathrm{id})}{\ell_0}\left(\frac{\partial L}{\partial\Delta\theta_0}\right)\right)+\sin(\Delta\theta_0)\frac{\partial L}{\partial\ell_0}+\frac{1}{\ell_1}\frac{\partial L}{\partial\Delta\theta_0}\right)+$$

$$-S^{-1}\left(\frac{\partial L}{\partial\Delta\theta_0}\right)\frac{\cos(\Delta\theta_0)}{\ell_0}\sigma_1^u+S^{-1}\left(\frac{\partial L}{\partial\Delta\theta_0}\right)\frac{\sin(\Delta\theta_0)}{\ell_0}\sigma_1^x.$$

In the discrete case, the conservation laws can be written as, [43],

$$\mathbf{v}(\mathbf{I})\mathrm{Ad}(\rho)=\mathbf{c}, \tag{5.60}$$

where $\mathbf{v}(\mathbf{I})$ is a vector depending only on invariants of action (5.11), while $\mathrm{Ad}(\rho)$ is the Adjoint representation of the frame (5.15) and $\mathbf{c}$ is a constant vector. In this setting, the Adjoint representation of the frame is

$$Ad(\rho)=\begin{pmatrix}\cos(\theta) & -\sin(\theta) & b\\ -\sin(\theta) & \cos(\theta) & -a\\ 0 & 0 & 1\end{pmatrix}.$$

Consider the matrix of infinitesimals for the action (5.11), given by

$$\Phi_0(x_0,u_0)=\begin{array}{c}\\ x_0\\ u_0\end{array}\begin{array}{c}\begin{array}{ccc}a & b & \theta\end{array}\\ \begin{pmatrix}1 & 0 & -u_0\\ 0 & 1 & x_0\end{pmatrix}\end{array},$$

and its invariantised form

$$\Phi_0(\mathbf{I}) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

The vector $\mathbf{v}(\mathbf{I})$ in (5.60) is computed, [43], as

$$\mathbf{v}(\mathbf{I}) = \sum C_\alpha^j (S^j \Phi_0^\alpha(I)) Ad(\rho_j \rho_0^{-1}), \qquad \alpha = x, u \tag{5.61}$$

where C_α^j is the coefficient of σ_j^α in BT1. With this notation, the vector of invariants $\mathbf{v}(I) = (v_1, v_2, v_3)$ is given by

$$v_1 = S^{-1}\left(\sin\Delta\theta_0 \frac{(S^{-1} - \mathrm{id})}{\ell_0} \left(\frac{\partial L}{\partial \Delta\theta_0} \right) + \cos\Delta\theta_0 \left(\frac{\partial L}{\partial \ell_0} \right) \right),$$

$$v_2 = S^{-1}\left(-\frac{\cos\Delta\theta_0}{\ell_0} (S^{-1} - \mathrm{id}) \left(\frac{\partial L}{\partial \Delta\theta_0} \right) + \sin\Delta\theta_0 \left(\frac{\partial L}{\partial \ell_0} \right) \right),$$

$$v_3 = -S^{-1}\left(\frac{\partial L}{\partial \Delta\theta_0} \right).$$

5.3.3 Higher–order Lagrangians

In the previous subsection, we have shown how to derive the Euler–Lagrange equations and conservation laws for Lagrangians that depend only on $\Delta\theta_0$ and ℓ_0. We have seen though, that we can construct either higher–order approximants to κ, or even discrete analogues to derivatives of κ, that depend on shifts of $\Delta\theta_0$ and ℓ_0. In this paragraph we will discuss how to adapt computations for the Euler–Lagrange equations and conservation laws seen in the previous subsection, in order to take into consideration also the shift of the discrete invariants. The key mathematical fact here is that the $\frac{d}{dt}$ and S operators do commute. In the case of the affine action of $SE(2)$ the two invariants we need to consider in order to approximate κ are

$\Delta\theta_0$ and ℓ_0. For these two quantities, it holds that

$$\frac{d}{dt}(\Delta\theta_{-1}) = S^{-1}\frac{d}{dt}(\Delta\theta_0), \tag{5.62}$$

$$\frac{d}{dt}(\ell_{-1}) = S^{-1}\frac{d}{dt}(\ell_0). \tag{5.63}$$

Equation (5.62)–(5.63) also is true in the presence of multiple shifts. We proceed now to compute the Euler–Lagrange equations and conservation laws for Lagragians that involve $\Delta\theta_0, \Delta\theta_{-1}, \ell_0, \ell_{-1}$. An example of such Lagrangian is any Lagrangian that is a function of a second–order approximation to κ. Euler–Lagrange equations and conservation laws for Lagrangians containing composition of shifts of the invariants can be computed analogously. Consider a Lagrangian of the form

$$\mathscr{L}_n = \sum L(\Delta\theta_0, \Delta\theta_{-1}, \ell_0, \ell_{-1}).$$

Differentiating with respect to t both sides yields

$$\begin{aligned}
\frac{d}{dt}\mathscr{L}_n &= \frac{d}{dt}\sum L(\Delta\theta_0, \Delta\theta_{-1}, \ell_0, \ell_{-1}) \\
&= \sum \frac{d}{dt} L(\Delta\theta_0, \Delta\theta_{-1}, \ell_0, \ell_{-1}) \\
&= \sum \frac{\partial L}{\partial \Delta\theta_0}\frac{\partial \Delta\theta_0}{\partial t} + \frac{\partial L}{\partial \Delta\theta_{-1}}\frac{\partial \Delta\theta_{-1}}{\partial t} + \frac{\partial L}{\partial \ell_0}\frac{\partial \ell_0}{\partial t} + \frac{\partial L}{\partial \ell_{-1}}\frac{\partial \ell_{-1}}{\partial t}.
\end{aligned} \tag{5.64}$$

Equations (5.62)–(5.63) can now be used to substitute in (5.64) for the syzygies relative to $\Delta\theta_{-1}$ and ℓ_{-1}, obtained just shifting the ones for $\Delta\theta_0$ and ℓ_0.

$$
\begin{aligned}
&\sum \frac{\partial L}{\partial \Delta\theta_0}\frac{\partial \Delta\theta_0}{\partial t} + \frac{\partial L}{\partial \Delta\theta_{-1}}\frac{\partial \Delta\theta_{-1}}{\partial t} + \frac{\partial L}{\partial \ell_0}\frac{\partial \ell_0}{\partial t} + \frac{\partial L}{\partial \ell_{-1}}\frac{\partial \ell_{-1}}{\partial t} \\
&= \frac{\partial L}{\partial \Delta\theta_0}\left((S-\mathrm{id})\left[\frac{1}{\ell_0}\left(\sigma_0^u - \cos(\Delta\theta_0)\sigma_1^u + \sin(\Delta\theta_0)\sigma_1^x\right)\right]\right) \\
&+ \frac{\partial L}{\partial \Delta\theta_{-1}}\left((\mathrm{id}-S^{-1})\left[\frac{1}{\ell_0}\left(\sigma_0^u - \cos(\Delta\theta_0)\sigma_1^u + \sin(\Delta\theta_0)\sigma_1^x\right)\right]\right) \\
&+ \frac{\partial L}{\partial \ell_0}\left(-\sigma_0^x + \cos(\Delta\theta_0)\sigma_1^x + \sin(\Delta\theta_0)\sigma_1^u\right) \\
&+ \frac{\partial L}{\partial \ell_{-1}}S^{-1}\left(-\sigma_0^x + \cos(\Delta\theta_0)\sigma_1^x + \sin(\Delta\theta_0)\sigma_1^u\right).
\end{aligned}
$$

Recall that summation by parts can also be written as

$$fS^{-1}g = gSf + (S^{-1} - \mathrm{id})(gSf), \tag{5.65}$$

$$fS^{-2}g = gS^2f + (S^{-1} - \mathrm{id})(S^{-1}gSf + gS^2f). \tag{5.66}$$

Using (5.49)–(5.50)–(5.65)–(5.66) we perform the first summation by parts, obtaining

$$\begin{aligned}
\frac{d}{dt}\mathscr{L}_n = \sum \frac{(S^{-1}-\mathrm{id})}{\ell_0}\left(\frac{\partial L}{\partial \Delta\theta_0}\right)(\sigma_0^u - \cos(\Delta\theta_0)\sigma_1^u + \sin(\Delta\theta_0)\sigma_1^x) \\
+ \frac{(\mathrm{id}-S)}{\ell_0}\left(\frac{\partial L}{\partial \Delta\theta_{-1}}\right)(\sigma_0^u - \cos(\Delta\theta_0)\sigma_1^u + \sin(\Delta\theta_0)\sigma_1^x) \\
+ \frac{\partial L}{\partial \ell_0}(-\sigma_0^x + \cos(\Delta\theta_0)\sigma_1^x + \sin(\Delta\theta_0)\sigma_1^u) \\
+ S\left(\frac{\partial L}{\partial \ell_{-1}}\right)(-\sigma_0^x + \cos(\Delta\theta_0)\sigma_1^x + \sin(\Delta\theta_0)\sigma_1^u) \\
+ (S-\mathrm{id})\left(\frac{1}{\ell_0}(\sigma_0^u - \cos(\Delta\theta_0)\sigma_1^u + \sin(\Delta\theta_0)\sigma_1^x)S^{-1}\left(\frac{\partial L}{\partial \Delta\theta_0}\right)\right) \\
+ (S^{-1}-\mathrm{id})\left(\frac{1}{\ell_0}(\sigma_0^u - \cos(\Delta\theta_0)\sigma_1^u + \sin(\Delta\theta_0)\sigma_1^x)S\left(\frac{\partial L}{\partial \Delta\theta_{-1}}\right)\right. \\
\left. + (-\sigma_0^x + \cos(\Delta\theta_0)\sigma_1^x + \sin(\Delta\theta_0)\sigma_1^u)S\left(\frac{\partial L}{\partial \ell_{-1}}\right)\right).
\end{aligned}$$

Performing another series of summation by parts, we regroup the expressions in terms of the invariants σ_0^u, σ_0^x, namely

$$\begin{aligned}
\frac{d}{dt}\mathscr{L}_n = \sum \sigma_0^u \Bigg(& \frac{(S^{-1}-\mathrm{id})}{\ell_0}\left(\frac{\partial L}{\partial \Delta\theta_0}\right) + \frac{(\mathrm{id}-S)}{\ell_0}\left(\frac{\partial L}{\partial \Delta\theta_{-1}}\right) \\
& + S^{-1}\Bigg(-\cos(\Delta\theta_0)\Bigg(\frac{(S^{-1}-\mathrm{id})}{\ell_0}\left(\frac{\partial L}{\partial \Delta\theta_0}\right) + \frac{(\mathrm{id}-S)}{\ell_0}\left(\frac{\partial L}{\partial \Delta\theta_{-1}}\right)\Bigg) \\
& + \sin(\Delta\theta_0)\left(\frac{\partial L}{\partial \ell_0} + S\frac{\partial L}{\partial \ell_{-1}}\right)\Bigg) \\
& + \sigma_0^x\Bigg(-\left(\frac{\partial L}{\partial \ell_0} + S\frac{\partial L}{\partial \ell_{-1}}\right) + S^{-1}\Bigg(\sin(\Delta\theta_0)\Bigg(\frac{(S^{-1}-\mathrm{id})}{\ell_0}\left(\frac{\partial L}{\partial \Delta\theta_0}\right) \\
& + \frac{(\mathrm{id}-S)}{\ell_0}\left(\frac{\partial L}{\partial \Delta\theta_{-1}}\right)\Bigg) + \cos(\Delta\theta_0)\left(\frac{\partial L}{\partial \ell_0} + S\frac{\partial L}{\partial \ell_{-1}}\right)\Bigg)\Bigg) \\
& + (S-\mathrm{id})\Bigg(\frac{1}{\ell_0}\left(\sigma_0^u - \cos(\Delta\theta_0)\sigma_1^u + \sin(\Delta\theta_0)\sigma_1^x\right) S^{-1}\left(\frac{\partial L}{\partial \Delta\theta_0}\right) \\
& + \sigma_0^u S^{-1}\Bigg(-\cos(\Delta\theta_0)\Bigg(\frac{(S^{-1}-\mathrm{id})}{\ell_0}\left(\frac{\partial L}{\partial \Delta\theta_0}\right) + \frac{(\mathrm{id}-S)}{\ell_0}\left(\frac{\partial L}{\partial \Delta\theta_{-1}}\right)\Bigg) \\
& + \sin(\Delta\theta_0)\left(\frac{\partial L}{\partial \ell_0} + S\frac{\partial L}{\partial \ell_{-1}}\right)\Bigg) + \sigma_0^x S^{-1}\Bigg(\sin(\Delta\theta_0)\Bigg(\frac{(S^{-1}-\mathrm{id})}{\ell_0}\left(\frac{\partial L}{\partial \Delta\theta_0}\right) \\
& + \frac{(\mathrm{id}-S)}{\ell_0}\left(\frac{\partial L}{\partial \Delta\theta_{-1}}\right)\Bigg) + \cos(\Delta\theta_0)\left(\frac{\partial L}{\partial \ell_0} + S\frac{\partial L}{\partial \ell_{-1}}\right)\Bigg)\Bigg) + (S^{-1}-\mathrm{id})\Bigg(\frac{1}{\ell_0}\left(\sigma_0^u\sigma_1^u\right. \\
& \left. - \cos(\Delta\theta_0) + \sin(\Delta\theta_0)\sigma_1^x S\left(\frac{\partial L}{\partial \Delta\theta_{-1}}\right) + \left(-\sigma_0^x + \cos(\Delta\theta_0)\sigma_1^x + \sin(\Delta\theta_0)\sigma_1^u\right) S\left(\frac{\partial L}{\partial \ell_{-1}}\right)\Bigg)\right).
\end{aligned}$$

Notice that, given a sequence of points f_0, then it holds

$$(S^{-1}-\mathrm{id})f_0 = -(S-\mathrm{id})f_{-1}.$$

Hence we can rewrite the boundary terms in the expression above as

$$(S-\mathrm{id})\Bigg(\frac{1}{\ell_0}\left(\sigma_0^u-\cos(\Delta\theta_0)\sigma_1^u+\sin(\Delta\theta_0)\sigma_1^x\right)S^{-1}\left(\frac{\partial L}{\partial\Delta\theta_0}\right)$$

$$+\sigma_0^u S^{-1}\Bigg(-\cos(\Delta\theta_0)\Bigg(\frac{(S^{-1}-\mathrm{id})}{\ell_0}\left(\frac{\partial L}{\partial\Delta\theta_0}\right)+\frac{(\mathrm{id}-S)}{\ell_0}\left(\frac{\partial L}{\partial\Delta\theta_{-1}}\right)\Bigg)+\sin(\Delta\theta_0)\left(\frac{\partial L}{\partial\ell_0}+S\frac{\partial L}{\partial\ell_{-1}}\right)\Bigg)$$

$$+\sigma_0^x S^{-1}\Bigg(\sin(\Delta\theta_0)\Bigg(\frac{(S^{-1}-\mathrm{id})}{\ell_0}\left(\frac{\partial L}{\partial\Delta\theta_0}\right)+\frac{(\mathrm{id}-S)}{\ell_0}\left(\frac{\partial L}{\partial\Delta\theta_{-1}}\right)\Bigg)+\cos(\Delta\theta_0)\left(\frac{\partial L}{\partial\ell_0}+S\frac{\partial L}{\partial\ell_{-1}}\right)\Bigg)$$

$$-S^{-1}\Bigg(\frac{1}{\ell_0}\left(\sigma_0^u-\cos(\Delta\theta_0)\sigma_1^u+\sin(\Delta\theta_0)\sigma_1^x\right)S\left(\frac{\partial L}{\partial\Delta\theta_{-1}}\right)$$

$$+\left(-\sigma_0^x+\cos(\Delta\theta_0)\sigma_1^x+\sin(\Delta\theta_0)\sigma_1^u\right)S\left(\frac{\partial L}{\partial\ell_{-1}}\right)\Bigg)\Bigg).$$

Let us define the vectors

$$U_0=\begin{pmatrix}\dfrac{(S^{-1}-\mathrm{id})}{\ell_0}\left(\dfrac{\partial L}{\partial\Delta\theta_0}\right)\\[2ex] \dfrac{\partial L}{\partial\ell_0}\end{pmatrix},\qquad V_0=\begin{pmatrix}\dfrac{(\mathrm{id}-S)}{\ell_0}\left(\dfrac{\partial L}{\partial\Delta\theta_{-1}}\right)\\[2ex] S\dfrac{\partial L}{\partial\ell_{-1}}\end{pmatrix}.$$

The Euler–Lagrange equations can be written as

$$\begin{pmatrix}E^u(L)\\ E^x(L)\end{pmatrix}=S^{-1}\left(R_{\Delta\theta_0}\left(U_0+V_0\right)\right)-(U_0+V_0).$$

Using (5.61) it is possible to write down the vector of invariant $\mathbf{w}(\mathbf{I})$ in the conservation laws.

For each solution of the Euler–Lagrange equations we have

$$\mathbf{w}(\mathbf{I})Ad(\rho)=\mathbf{c},$$

where **c** is a vector of constants and

$$\mathbf{w} = C_x^0 \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}^T + C_u^x \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}^T + C_x^1 \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}^T Ad(K_0) + C_u^1 \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}^T Ad(K_0)$$
$$+ C_x^{-1} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}^T Ad(K_{-1})^{-1} + C_u^{-1} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}^T Ad(K_{-1})^{-1},$$

where the C_α^j are as in (5.61) and stand for the coefficients of σ_α^j in the boundary terms above.

5.3.4 Constrained Lagrangians

Another way to approach the problem defined by (5.46), is to introduce a constraint on the invariant ℓ_0. We will show how this results in a computationally far simpler problem, compared to the unconstrained one. Suppose we want to find solutions where ℓ_0 is a constant, say $h > 0$. This translates into considering a new Lagrangian given by

$$\mathscr{L}_n = \sum \left(L(\Delta\theta_0, \ell_0) + \lambda_0 \left(\ell_0 - h \right) \right) \ell_0. \tag{5.67}$$

From a geometric point of view, we are looking at discrete curves such that the distance between two consecutive points is equal to h. In this case, in order to find the point (x_1, u_1) we need only the previous point (x_0, u_0) and the angle between the two. Therefore, it is expected that to find the minimising solution to (5.67), only one equation has to be solved.

Computations for the Euler–Lagrange equations for (5.67) are as follows:

$$\begin{aligned}
\frac{d}{dt}\mathscr{L}_n &= \frac{d}{dt}\sum \left(L(\Delta\theta_0,\ell_0)+\lambda_0\left(\ell_0-h\right)\right)\ell_0 \\
&= \sum \frac{d}{dt}\left(\left(L(\Delta\theta_0,\ell_0)+\lambda_0\left(\ell_0-h\right)\right)\ell_0\right) \\
&= \sum \left(\frac{dL}{d\Delta\theta_0}\frac{d\Delta\theta_0}{dt}+\frac{dL}{d\ell_0}\frac{d\ell_0}{dt}\right)\ell_0 + L\frac{d\ell_0}{dt}+\lambda_0\ell_0\frac{d\ell_0}{dt}+\lambda_0(\ell_0-h)\frac{d\ell_0}{dt} \\
&= \sum \frac{dL}{d\Delta\theta_0}\frac{d\Delta\theta_0}{dt}\ell_0 + \left(\frac{dL}{d\ell_0}\ell_0+L+\lambda_0+\lambda_0(\ell_0-h)\right)\frac{d\ell_0}{dt}.
\end{aligned} \tag{5.68}$$

We substitute expressions (5.44)–(5.45) for $\dfrac{d\Delta\theta_0}{dt}$ and $\dfrac{d\ell_0}{dt}$ into (5.68), obtaining

$$\begin{aligned}
\frac{d}{dt}\mathscr{L}_n &= \sum \frac{dL}{d\Delta\theta_0}\frac{d\Delta\theta_0}{dt}\ell_0 + \left(\frac{dL}{d\ell_0}\ell_0+L\right)\frac{d\ell_0}{dt} \\
&= \sum \frac{dL}{d\Delta\theta_0}\,(\text{S-id})\left(\frac{\sigma_0^u-\cos(\Delta\theta_0)\sigma_1^u+\sin(\Delta\theta_0)\sigma_1^x}{\ell_0}\right)\ell_0 \\
&+ \left(\frac{dL}{d\ell_0}\ell_0+L+\lambda_0+\lambda_0(\ell_0-h)\right)\left(-\sigma_0^x+\cos(\Delta\theta_0)\sigma_1^x+\sin(\Delta\theta_0)\sigma_1^u\right).
\end{aligned} \tag{5.69}$$

As in the non–constrained case, we use (5.49)–(5.50) with the aim to express (5.69) in terms of σ_0^x and σ_0^u. In doing so, we will need to add some boundary terms, that will give rise to

conservation laws. The first summation by parts gives

$$
\begin{aligned}
&= \sum \frac{dL}{d\Delta\theta_0} (\mathrm{S} - \mathrm{id}) \left(\frac{\sigma_0^u - \cos(\Delta\theta_0)\sigma_1^u + \sin(\Delta\theta_0)\sigma_1^x}{\ell_0} \right) \ell_0 \\
&+ \left(\frac{dL}{d\ell_0}\ell_0 + L + \lambda_0 + \lambda_0(\ell_0 - h) \right) (-\sigma_0^x + \cos(\Delta\theta_0)\sigma_1^x + \sin(\Delta\theta_0)\sigma_1^u) \\
&= \sum \left(\frac{\mathrm{S}^{-1} - \mathrm{id}}{\ell_0} \right) \left(\frac{dL}{d\Delta\theta_0}\ell_0 \right) (\sigma_0^u - \cos(\Delta\theta_0)\sigma_1^u + \sin(\Delta\theta_0)\sigma_1^x) \\
&+ \left(\frac{dL}{d\ell_0}\ell_0 + L + \lambda_0 + \lambda_0(\ell_0 - h) \right) (-\sigma_0^x + \cos(\Delta\theta_0)\sigma_1^x + \sin(\Delta\theta_0)\sigma_1^u) \\
&+ (\mathrm{S} - \mathrm{id}) \left((\sigma_0^u - \cos(\Delta\theta_0)\sigma_1^u + \sin(\Delta\theta_0)\sigma_1^x \ell_0)\, \mathrm{S}^{-1} \left(\frac{dL}{d\Delta\theta_0}\ell_0 \right) \right).
\end{aligned}
$$

If we perform another summation by parts, we can express σ_1^x and σ_1^u in terms of σ_0^x and σ_0^u plus some other boundary terms. Computations are as follows:

$$
\begin{aligned}
&= \sum \left(\frac{\mathrm{S}^{-1}-\mathrm{id}}{\ell_0}\right)\left(\frac{dL}{d\Delta\theta_0}\ell_0\right)(\sigma_0^u - \cos(\Delta\theta_0)\sigma_1^u + \sin(\Delta\theta_0)\sigma_1^x) \\
&+ \left(\frac{dL}{d\ell_0}\ell_0 + L + \lambda_0 + \lambda_0(\ell_0 - h)\right)(-\sigma_0^x + \cos(\Delta\theta_0)\sigma_1^x + \sin(\Delta\theta_0)\sigma_1^u) \\
&+ (\mathrm{S}-\mathrm{id})\left(\left(\frac{\sigma_0^u - \cos(\Delta\theta_0)\sigma_1^u + \sin(\Delta\theta_0)\sigma_1^x}{\ell_0}\right)\mathrm{S}^{-1}\left(\frac{dL}{d\Delta\theta_0}\ell_0\right)\right) \\
&= \sum \sigma_0^u\left(-\left(\frac{\mathrm{S}^{-1}-\mathrm{id}}{\ell_0}\right)\left(\frac{dL}{d\Delta\theta_0}\ell_0\right) + \mathrm{S}^{-1}\left(-\cos(\Delta\theta_0)\left(\frac{\mathrm{S}^{-1}-\mathrm{id}}{\ell_0}\right)\left(\frac{dL}{d\Delta\theta_0}\ell_0\right)\right)\right. \\
&\left. + \sin(\Delta\theta_0)\left(\frac{dL}{d\ell_0}\ell_0 + L + \lambda_0 + \lambda_0(\ell_0 - h)\right)\right) + \sigma_0^x\left(-\left(\frac{dL}{d\ell_0}\ell_0 + L + \lambda_0 + \lambda_0(\ell_0 - h)\right)\right. \\
&\left. + \mathrm{S}^{-1}\left(\sin(\Delta\theta_0)\left(\frac{\mathrm{S}^{-1}-\mathrm{id}}{\ell_0}\right)\left(\frac{dL}{d\Delta\theta_0}\ell_0\right) + \cos(\Delta\theta_0)\left(\frac{dL}{d\ell_0}\ell_0 + L + \lambda_0 + \lambda_0(\ell_0 - h)\right)\right)\right) \\
&+ (\mathrm{S}-\mathrm{id})\left(\left(\frac{\sigma_0^u - \cos(\Delta\theta_0)\sigma_1^u + \sin(\Delta\theta_0)\sigma_1^x}{\ell_0}\right)\mathrm{S}^{-1}\left(\frac{dL}{d\Delta\theta_0}\ell_0\right)\right. \\
&+ \sigma_0^x\mathrm{S}^{-1}\left(\sin(\Delta\theta_0)\left(\frac{\mathrm{S}^{-1}-\mathrm{id}}{\ell_0}\right)\left(\frac{dL}{d\Delta\theta_0}\ell_0\right) + \cos(\Delta\theta_0)\left(\frac{dL}{d\ell_0}\ell_0 + L + \lambda_0 + \lambda_0(\ell_0 - h)\right)\right) \\
&\left. + \sigma_0^u\mathrm{S}^{-1}\left(-\cos(\Delta\theta_0)\left(\frac{\mathrm{S}^{-1}-\mathrm{id}}{\ell_0}\right)\left(\frac{dL}{d\Delta\theta_0}\ell_0\right) + \sin(\Delta\theta_0)\left(\frac{dL}{d\ell_0}\ell_0 + L + \lambda_0 + \lambda_0(\ell_0 - h)\right)\right)\right).
\end{aligned}
$$

The expression above contains the Euler–Lagrange equations and the conservation laws of the constrained problem. Since we introduced the constrain on ℓ_0, we can now evaluate all

the quantities at $\ell_0 = h$ in order to obtain a first simplification as

$$\begin{aligned}
\frac{d}{dt}\mathscr{L}_n = \sum \sigma_0^u \Bigg(& \left(\mathrm{S}^{-1} - \mathrm{id}\right)\left(\frac{dL}{d\Delta\theta_0}\right) + \mathrm{S}^{-1}\left(-\cos(\Delta\theta_0)\left(\mathrm{S}^{-1} - \mathrm{id}\right)\left(\frac{dL}{d\Delta\theta_0}\right)\right) \\
& + \sin(\Delta\theta_0)\left(\frac{dL}{d\ell_0}h + L + \lambda_0\right)\Bigg) + \sigma_0^x\Bigg(- \left(\frac{dL}{d\ell_0}h + L + \lambda_0\right) \\
& + \mathrm{S}^{-1}\left(\sin(\Delta\theta_0)\left(\mathrm{S}^{-1} - \mathrm{id}\right)\left(\frac{dL}{d\Delta\theta_0}\right) + \cos(\Delta\theta_0)\left(\frac{dL}{d\ell_0}h + L + \lambda_0\right)\right)\Bigg) \\
& + (\mathrm{S} - \mathrm{id})\Bigg((\sigma_0^u - \cos(\Delta\theta_0)\sigma_1^u + \sin(\Delta\theta_0)\sigma_1^x)\,\mathrm{S}^{-1}\left(\frac{dL}{d\Delta\theta_0}\right) \\
& + \sigma_0^x \mathrm{S}^{-1}\left(\sin(\Delta\theta_0)\left(\mathrm{S}^{-1} - \mathrm{id}\right)\left(\frac{dL}{d\Delta\theta_0}\right) + \cos(\Delta\theta_0)\left(\frac{dL}{d\ell_0}h + L + \lambda_0\right)\right) \\
& + \sigma_0^u \mathrm{S}^{-1}\left(-\cos(\Delta\theta_0)\left(\mathrm{S}^{-1} - \mathrm{id}\right)\left(\frac{dL}{d\Delta\theta_0}\right) + \sin(\Delta\theta_0)\left(\frac{dL}{d\ell_0}h + L + \lambda_0\right)\right)\Bigg).
\end{aligned}$$

We stress again the fact that all the quantities above are evaluated at $\ell_0 = h$, even if we do not specify it with the usual notation to make it easier to read. We reduce it to a more compact expression defining the vector

$$W = \begin{pmatrix} \left(\mathrm{S}^{-1} - \mathrm{id}\right)\left(\dfrac{dL}{d\Delta\theta_0}\right) \\ \dfrac{dL}{d\ell_0}h + L + \lambda_0 \end{pmatrix}.$$

Hence, the Euler–Lagrange equations become

$$R_{\Delta\theta_0} W_0 = W_1, \tag{5.70}$$

where $R_{\Delta\theta_0}$ is the rotation matrix with angle $\Delta\theta_0$.

Also the conservation laws can be written in a more compact form, as we did in the unconstrained case. The invariantised matrix of infinitesimal depends only on the action and the normalisation equations, so it stays the same as in the unconstrained case. However, the vector of invariants includes the information contained in the constraint. Performing analogous computations as the ones for the unconstrained case, we have that the conservation laws can be expressed as

$$\mathbf{w}(I)\mathrm{Ad}(\rho) = \mathbf{c},$$

where in this case, the vector of invariants $\mathbf{w}(I) = (w_1, w_2, w_3)$ is

$$w_1 = \mathrm{S}^{-1}\left(\sin(\Delta\theta_0)\left(\mathrm{S}^{-1} - \mathrm{id}\right)\left(\frac{dL}{d\Delta\theta_0}\right) + \cos(\Delta\theta_0)\left(\frac{dL}{d\ell_0}h + L + \lambda_0\right)\right),$$

$$w_2 = \mathrm{S}^{-1}\left(-\cos(\Delta\theta_0)\left(\mathrm{S}^{-1} - \mathrm{id}\right)\left(\frac{dL}{d\Delta\theta_0}\right) + \sin(\Delta\theta_0)\left(\frac{dL}{d\ell_0}h + L + \lambda_0\right)\right),$$

$$w_3 = -S^{-1}\left(\frac{\partial L}{\partial\Delta\theta_0}\right).$$

5.4 Numerical Examples

In this section we will present two simple numerical examples to show how the theory works in practice. Of particular interest is to see if there are some computional benefits in introducing a constraint as we did in a previous section. Therefore, we will consider the same Lagrangian, first unconstrained, and then with ℓ_0 fixed.

5.4.1 An unconstrained Lagrangian

Let us now look at the numerical aspects of the Euler-Lagrange equations for a $SE(2)$-invariant Lagrangian. As a first example, let us consider the Lagrangian given by

$$\mathscr{L}_n = \sum L(\Delta\theta_0, \ell_0) = \sum \frac{\sin^2(\Delta\theta_0)}{\ell_0^2}\ell_0. \tag{5.71}$$

We have seen that we can match the generating invariants of the smooth action with the ones of the discrete action, up to some order of accuracy in ℓ_0 and its shifts. Recall from (5.26) that

$$\kappa = -\frac{\sin(\Delta\theta_0)}{\ell_0} + \mathscr{O}(\ell_0),$$

then the Lagrangian (5.71) is a first–order approximation of the smooth

$$\mathscr{L} = \int \kappa^2\, ds.$$

In order to compute the invariantised Euler-Lagrange equations for the Lagrangian (5.71) we first compute the components of the vector Z in (5.58). We have

$$\frac{\partial L}{\partial \Delta\theta_0} = \frac{\sin(2\Delta\theta_0)}{\ell_0}, \qquad \frac{\partial L}{\partial \ell_0} = -\frac{\sin^2(\Delta\theta_0)}{\ell_0^2}.$$

Hence,

$$Z_0 = \begin{pmatrix} \frac{(S^{-1}-\mathrm{id})}{\ell_0}\left(\frac{\sin^2(\Delta\theta_0)}{\ell_0}\right) \\ -\frac{\sin^2(\Delta\theta_0)}{\ell_0^2} \end{pmatrix}. \tag{5.72}$$

From (5.59) the Euler–Lagrange equations are

$$R_{\Delta\theta_0} Z_0 = Z_1. \tag{5.73}$$

Equation (5.73) can be solved as an initial value problem. Given $\Delta\theta_0, \Delta\theta_{-1}, \ell_0, \ell_1$, we are able to compute $R_{\Delta\theta_0} Z_0$. Then $R_{\Delta\theta_0} Z_0 = Z_1$ can be solved for the unknowns $\ell_1, \Delta\theta_1$ and thus the procedure can be continued. This gives us an explicit numerical scheme for solving (5.73). Consider now the vector $Z_1 = SZ_0$, where Z_0 is as in (5.72). While there is no restriction on the sign of the first component, the second component is always negative. Therefore the rotation will always happen in the lower half of the circle C centered at the origin and with radius $||Z_0||$, where $||\cdot||$ stands for the Euclidean norm. This induces a necessary condition on the initial data as

$$-\frac{\sin(\Delta\theta_1)^2}{\ell_1^2} = \sin(\Delta\theta_0)\frac{\left(S^{-1}-\mathrm{id}\right)}{\ell_0}\left(\frac{\sin^2(\Delta\theta_0)}{\ell_0}\right) - \cos(\Delta\theta_0)\frac{\sin^2(\Delta\theta_0)}{\ell_0^2} < 0. \tag{5.74}$$

If we assume that $\sin(\Delta\theta_0) < 0$, as it is the case in the lower–half of the circle C, then (5.74) implies that

$$\frac{\left(S^{-1}-\mathrm{id}\right)}{\ell_0}\left(\frac{\sin^2(\Delta\theta_0)}{\ell_0}\right) - \frac{\sin(2\Delta\theta_0)}{\ell_0^2} > 0.$$

We use MATLAB to code a routine that solves the Euler–Lagrange equations and compute the curve that minimises (5.71). A plot of the first 250 points in $[0,1]$ of the solution and the correspondent conservation laws are given in Figure 5.1 and Figure 5.2 respectively.

5.4.2 A constrained Lagrangian

Consider the Lagrangian given by

$$\mathscr{L}_n = \sum\left(\frac{\sin^2(\Delta\theta_0)}{\ell_0^2} + \lambda_0\left(\ell_0 - h\right)\right)\ell_0. \tag{5.75}$$

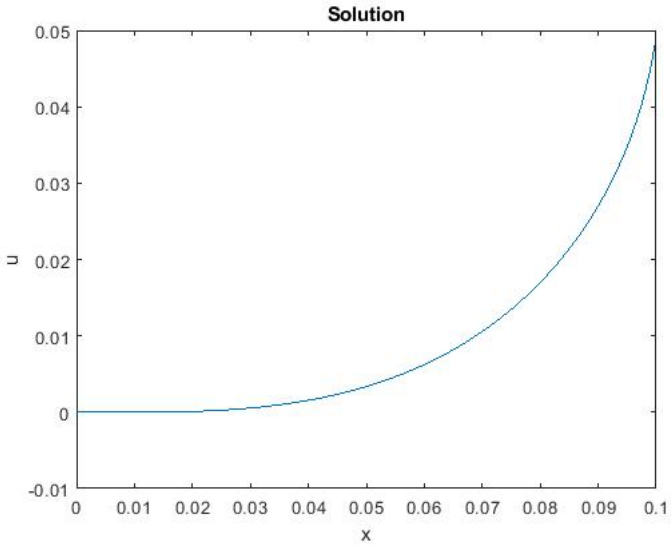


Fig. 5.1 Plot of the solution $(x(t), u(t))$, with $t \in [0,1]$

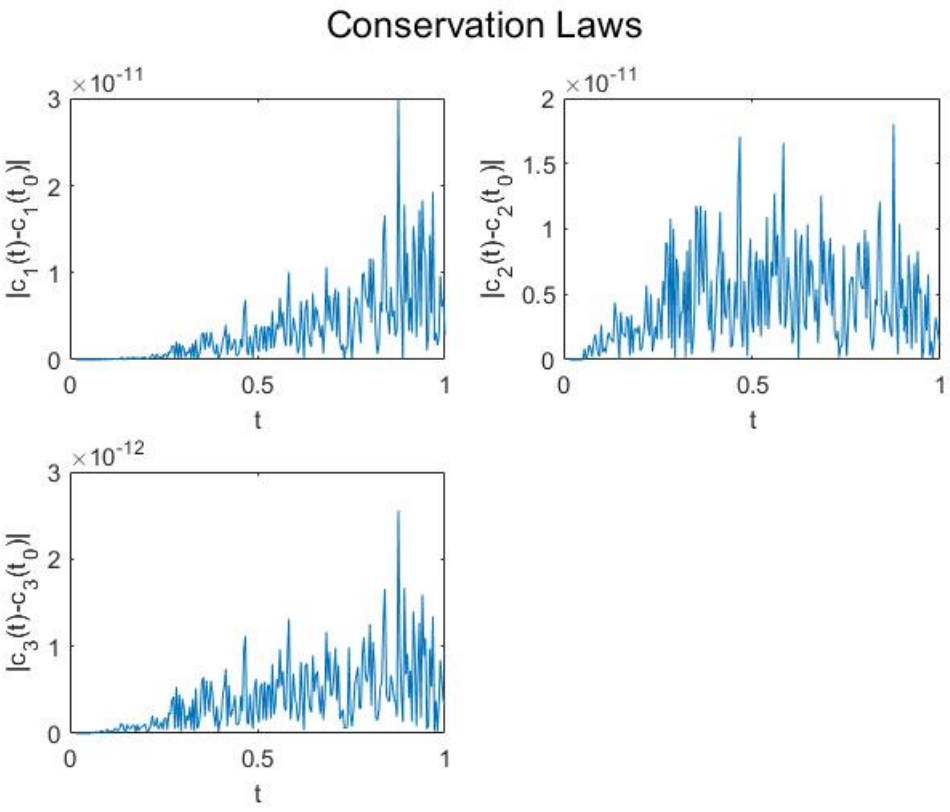


Fig. 5.2 Plot of the conservation laws

In this case, the vector W is

$$W_0 = \begin{pmatrix} \left(\mathrm{S}^{-1} - \mathrm{id}\right)\left(\dfrac{\sin(2\Delta\theta_0)}{h^2}\right) \\ -\dfrac{\sin^2(\Delta\theta_0)}{h^2} + \lambda_0 \end{pmatrix}. \tag{5.76}$$

Recall that the solution to the variational problem defined by (5.75) can be constructed with the knowledge of an initial point (x_0, u_0) and the expressions for $\Delta\theta_0$ and ℓ_0. Since in this case we have fixed ℓ_0 we just need to solve for $\Delta\theta_0$.

Suppose we are given initial data for the difference equation (5.70), where W is specified in (5.76). This means that we are given $\Delta\theta_0$, $\Delta\theta_{-1}$ and λ_0. We can express W_1 also as

$$W_1 = \begin{pmatrix} (\mathrm{id} - \mathrm{S})\left(\frac{\sin(2\Delta\theta_0)}{h^2}\right) \\ -\frac{\sin^2(\Delta\theta_1)}{h^2} + \lambda_1 \end{pmatrix}. \tag{5.77}$$

To find $\Delta\theta_1$ we proceed as follows:

- Compute $R_{\Delta\theta_0} W_0$ using the initial data
- Solve for $\Delta\theta_1$ in $W_1 = R_{\Delta\theta_0} W_0$, where W_1 is as in (5.77)

If we denote by $W_1^{(1)}$ the first component of $R_\Delta \theta_0 W_0$, then we have from (5.77) and (5.70)

$$W_1^{(1)} = (\mathrm{id} - \mathrm{S})\left(\frac{\sin(2\Delta\theta_0)}{h^2}\right),$$

which implies

$$\Delta\theta_1 = \frac{\arcsin\left(-W_1^{(1)} h^2 + \sin(2\Delta\theta_0)\right)}{2}.$$

We have an explicit formula to advance $\Delta\theta_0$ and λ_1 can be easily computed once $\Delta\theta_1$ is known. Plots for the first 250 points of the solution in $[0,1]$ (with $h = 0.0015$) and its conservation laws can be found in Figure 5.3 and Figure 5.4 respectively.

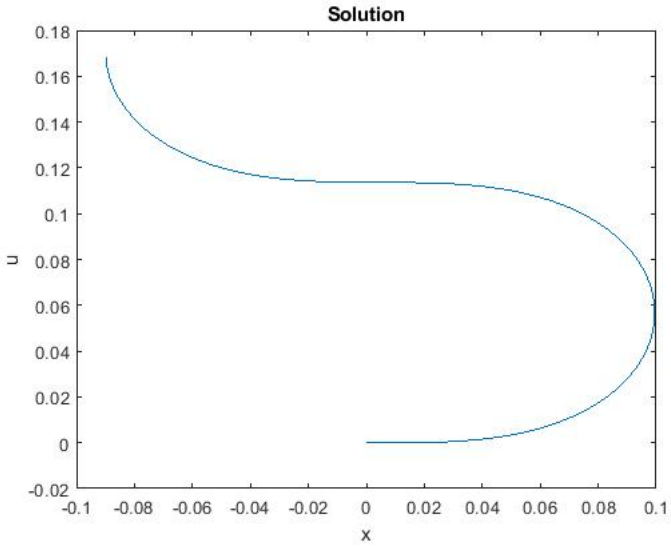


Fig. 5.3 Plot of the solution $(x(t), u(t))$, with $t \in [0,1]$

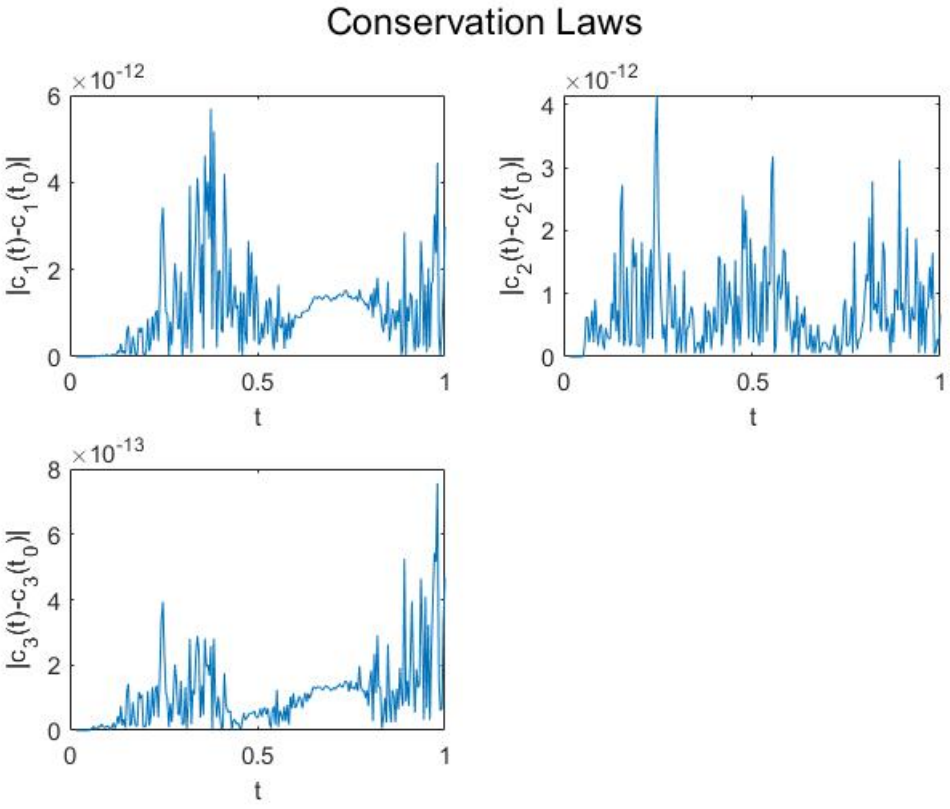


Fig. 5.4 Plot of the conservation laws

5.4.3 Comments on the numerical examples

The use of a constraint in a smooth Lagrangian that is invariant under the standard action of $SE(2)$ has already been studied in [24]. In the previous sections we have introduced an analogous constraint, this time in a multispace setting. This makes for a computationally simpler problem, compared to the unconstrained case. The Euler–Lagrange equations become more tractable and the numerical execution is faster. The plots of the conservation laws also hint at the fact that no preserved quantities have been lost with the introduction of the constraint on ℓ_0. However, there are differences between the two cases, as the behaviour of ℓ_0 can differ. It is not obvious how to compare the two solutions. This is because in the unconstrained case, ℓ_0 is left free to assume very small values, compared to the fixed value $\ell_0 = h$ in the constrained case. Hence, in the same number of points, the solution curve spans less space. How to measure the error introduced with the constraint is still an open problem.

6. Conclusion and Future Work

This work is devoted to the study of some theoretical and numerical topics arising in the Invariant Calculus of Variations.

In Chapter 2 we introduce the concept of manifold, Lie group and Lie group actions on smooth manifolds. These concepts are the building blocks that allow for the construction of a moving frame, i.e. an equivariant map from a suitable open set of a manifold to the Lie group. We show how a moving frame can be used to find and study the invariants of the action prolonged to jet bundle, the so–called differential invariants. Then we move to give the definition of a curvature matrix, a map from the manifold to the Lie algebra. Curvature matrices play an important role throughout this thesis as they both provide a set of generating differential invariants and define some differential relations, called syzygies, between these generators. After a brief review of the Adjoint representation of a Lie group, we enunciate the main results needed to apply the moving frame theory invariant problems in the Calculus of Variations. It is presented how the invariantised Euler–Lagrange equations can be derived and how the conservation laws coming from Noether's First Theorem can be written in terms of the adjoint representation of the group and a vector of invariants.

One–dimensional variational problems that are invariant under a linear action of $SU(2)$ are the main subject of Chapter 3. We apply the theory presented in Chapter 2 and derive a set of generating differential invariants, the invariantised Euler–Lagrange equations and the conservation laws. The second part of the chapter is devoted to the solution of the conservation laws in terms of the adjoint representation of the moving frame. In order to do

so, we show how we can use the geometrical context in order to make computations tractable and derive a solution for the minimisers. Numerical examples with some simple Lagrangians show how this theory works in practice. We end the chapter deriving the invariantised Euler–Lagrange equations for the two–dimensional case and introducing the issue of integrating the conservation laws in higher dimensional problems.

Chapter 4 contains a discussion on how to find solutions to invariant variational problems in higher dimension. In particular, the theory of Lie group integrators is presented, with a specific focus on the Magnus expansion. The latter is a series such that its exponential provides an exact solution of an ODE evolving on a manifold, in our case a Lie group. We use the Magnus expansion to show that integrating sequentially the PDEs for the curvature matrices provides a well–defined solution for the frame up to order 5 in the discretisation variables. The chapter continues with some numerical examples where we solve for the minimisers of two–dimensional variational problems invariant under actions of the Lie groups $SU(2)$, $SE(2)$ and $SL(2)$. Finally, we conjecture that our result does hold up to every order.

Our last contribution is exposed in Chapter 5, where the case of invariant variational problems on lattice–based multispaces is considered. After introducing the basic definitions and results about the multispace, we show how the formula for the prolongation of the infinitesimal action to the first derivative can be extended in this setting. We then study first–order invariant variational problems in multispace coordinates, with particular emphasis given to the case of the affine action of $SE(2)$ on curves. We show how to discretise generating differential invariants, and their derivatives, up to any order, using the curvature matrices. The chapter continues considering the case of higher order Lagrangians and the one where a constraint is introduced to ease the computation for the affine action of $SE(2)$. Finally, the chapter ends with two numerical examples and some considerations on the introduction of the above constraint.

Given more time, some ideas on how to continue the research presented in this work are the following:

- Apply the theory developed in Chapter 3 to variational systems invariant under some other $SU(2)$ actions, directly coming from applications in physics.
- For two or higher dimensional systems, use Stokes' Theorem to develop a numerical scheme for solving the conservation laws on a discretised grid.
- Lie group integrators, for efficiency reasons, are implemented in such a way that some terms in the Magnus expansion are omitted. The very terms that are used to compute numerical solutions in Diffman throughout this thesis can be found in [17, Algorithm A.2.5]. How the compatibility condition can be used to prove an analogue of Theorem 4.2.12 in this setting is an open problem.
- Related to Chapter 4, study the case where the independent variables are not left invariant by the Lie group action.
- Prove Conjecture 4.4.1.
- From the numerical point of view, it would be interesting and useful to extend the capabilities of the *Diffman* package. The package itself comprises already the implementation of many Lie groups, Lie algebras and Lie group integrators. However, it is not immediate to code numerical inputs for the Lie algebra elements and to handle systems of higher dimensions. There are some other features that could be included, an example being the Lie brackets for all the implemented Lie algebras.
- Consider the Euler–Lagrange equations as boundary value problems, especially in the multispace setting. The difficulties in this case lie in the nonlinearity of the equations, often leading to issues in the convergence of the numerical scheme. It could be interesting to utilise "shooting arguments" to explore the existence of some solutions.

- A generalisation to higher–order and higher–dimensional Lagrangian then will expand greatly the range of problems to which multispaces can be applied. It is worth studying how to rewrite higher–order systems as it is done in the smooth case.

- From the numerical point of view, the introduction of a constraint in invariant problems under $SE(2)$ gives promising results, but there are still areas to look at as to provide an estimate for the error and how to rigorously relate the unconstrained and constrained solution.

References

[1] Résolution del l'equation matricielle u̇ = pu par produit infini d'exponentielles matricielles. *Bull. Classe des Sci. Acad. Royal Belg.*, 44:818 – 829, 1958.

[2] B. Blanes, F. Casas, J. A. Oteo, and J. Ros. The Magnus expansion and some of its applications. *Physics Reports*, 470 (2009):151–238, 2009.

[3] S. Blanes, F. Casas, J. Oteo, and J. Ros. Magnus and fer expansions for matrix differential equations: The convergence problem. *Journal of Physics A: Mathematical and General*, 31:259, 1999.

[4] S. Blanes, F. Casas, and J. Ros. Improved high order integrators based on the Magnus expansion. *BIT Numerical Mathematics*, 40:434–450, 2000. https://doi.org/10.1023/A:1022311628317.

[5] J. C. Butcher. Coefficients for the study of runge-kutta integration processes. *Journal of the Australian Mathematical Society*, 3(2):185–201, 1963.

[6] M. P. Calvo, A. Iserles, and A. Zanna. Numerical solution of isospectral flows. *Math. Comput.*, 66:1461–1486, 1997.

[7] F. Casas and B. Owren. Cost efficient Lie group integrators in the RKMK class. *BIT Numerical Mathematics*, 43 (2003):723–742, 2003.

[8] E. Celledoni, A. Marthinsen, and B. Owren. Commutator-free lie group methods. *Future Gener. Comput. Syst.*, 19(3):341–352, 2003.

[9] E. Celledoni, H. Marthinsen, and B. Owren. An introduction to Lie group integrators – basics, new developments and applications. *Journal of Computational Physics*, 257:1040–1061, 2014.

[10] S. H. Christiansen, H. Z. Munthe-Kaas, and B. Owren. Topics in structure-preserving discretization. *Acta Numerica*, 20:1—119, 2011.

[11] P. Clarkson and P. Olver. Symmetry and the chazy equation. *Journal of differential equations*, 124(1):225–246, 1996.

[12] R. Cools. Constructing cubature formulae: the science behind the art. *Acta Numerica*, 6:1–54, 1997.

[13] G. Cooper. Stability of Runge–Kutta methods for trajectory problems. *IMA J. Num. Anal.*, 7:1–13, 1987.

[14] P. E. Crouch and R. Grossman. Numerical integration of ordinary differential equations on manifolds. *Journal of Nonlinear Science*, 3(1):1–33, 1993.

[15] *NIST Digital Library of Mathematical Functions*. http://dlmf.nist.gov/, Release 1.0.21 of 2018-12-15. http://dlmf.nist.gov/.

[16] E. B. Dynkin. Calculation of the coefficients in the Campbell-Hausdorff formula. *Doklady Akademii Nauk SSSR (N.S.)*, 57:323–326, 1947. English translation available at: http://people.math.umass.edu/~gunnells/S14/lie/dynkin-BCHfmla.pdf.

[17] K. Engø, A. Martinsen, and H. Z. Munthe-Kaas. Diffman: An object-oriented MATLAB toolbox for solving differential equations on manifolds. *Applied Numerical Mathematics*, 39 (2012):323–347, 2012.

[18] A. Fässler and E. Stiefel. *Group Theoretical Methods and Their Applications*. Birkhäuser, Boston, 1992.

[19] M. Fels and P. J. Olver. Moving coframes: I. A practical algorithm. *Acta Applicandae Mathematicae*, 51:161–213, 1998. https://doi.org/10.1023/A:1005878210297.

[20] M. Fels and P. J. Olver. Moving coframes: II. regularization and theoretical foundations. *Acta Applicandae Mathematicae*, 55:127–208, 1999. https://doi.org/10.1023/A:1006195823000.

[21] I. M. Gelfand and S. V. Fomin. *Calculus of Variations*. Prentice–Hall, 1963.

[22] T. M. Gonçalves and E. L. Mansfield. Moving frames and Noether's conservation laws - the general case. *Forum Math. Sigma*, 4, 2016.

[23] T. M. N. Gonçalves and E. L. Mansfield. On moving frames and Noether's conservation laws. *Studies in Applied Mathematics*, 128:1–29, 2012.

[24] T. M. N. Gonçalves and E. L. Mansfield. Moving frames and conservation laws for Euclidean invariant Lagrangians. *Studies in Applied Mathematics*, 130:134–166, 2013.

[25] E. Hairer, C. Lubich, and G. Wanner. *Geometric Numerical Integration*. Springer, 2006.

[26] E. Hairer, S. P. Norsett, and G. Wanner. *Solving Ordinary Differential Equations I. Nonstiff Problems*. Springer, Berlin, 2nd rev. ed. 1993. corr. 3rd printing edition, 1993.

[27] J. K. Hale. *Ordinary Differential Equations*. Krieger Publishing Company, 1980.

[28] B. Hall. *Lie Groups, Lie Algebras, and Representations: an elementary introduction*. Springer, 2015.

[29] M. W. Hirsch. *Differential Topology*. Springer, 1976.

[30] E. Hubert. Generation properties of Maurer-Cartan invariants. 2007. https://hal.inria.fr/inria-00194528/en.

[31] P. E. Hydon and E. L. Mansfield. A variational complex for difference equations. *Foundations of Computational Mathematics*, 4(2):187–217, 2004. https://doi.org/10.1007/s10208-002-0071-9.

[32] A. Iserles. Solving linear ordinary differential equations by exponentials of iterated commutators. *Numerische Mathematik*, 45(2):183–199, 1984.

[33] A. Iserles. Multistep methods on manifolds. *Math. Comput.*, 66:1461–1486, 1997.

[34] A. Iserles. Multistep methods on manifolds. *IMA Journal of Numerical Analysis*, 21(1):407–419, 2001.

[35] A. Iserles, H. Z. Munthe-Kaas, S. P. Nørsett, and A. Zanna. Lie–group methods. *Acta Numerica*, 9:215–365, 2000.

[36] A. Iserles and S. P. Nørsett. On the solution of linear differential equations in Lie groups. *Philosophical transactions of the Royal Society: mathematical, physical and engineering sciences*, 357:983–1020, 1999.

[37] A. Iserles, S. P. Nørsett, and A. F. Rasmussen. Time symmetry and high–order Magnus methods. *Applied Numerical Mathematics*, 39:379–401, 2001. http://www.sciencedirect.com/science/article/pii/S0168927401000885.

[38] Y. Kosmann-Schwarzbach. *The Noether Theorems*. Springer-Verlag, New York, 2011.

[39] D. Lewis and J. C. Simo. Conserving algorithms for the dynamics of hamiltonian systems on lie groups. *Journal of Nonlinear Science*, 4(1):253, 1994.

[40] W. Magnus. On the exponential solution of differential equations for a linear operator. *Communications on Pure and Applied Mathematics*, 7(4):649–673, 1954.

[41] E. L. Mansfield. Algorithms for symmetric differential systems. *E. Found. Comput. Math.*, 1:335–383, 2001.

[42] E. L. Mansfield. *A practical guide to invariant calculus*. Cambridge University Press, Cambridge, 2010.

[43] E. L. Mansfield, P. E. Hydon, L. Peng, and A. Rojo-Echeburua. Moving frames and noether's finite difference conservation laws I. *Transactions of Mathematics and Its Applications*, 3, 2019.

[44] G. Marí Beffa and E. L. Mansfield. Discrete moving frames on lattice varieties and lattice–based multispaces. *J. Found Comput Math*, 18(1):181–247, 2018.

[45] P. C. Moan and J. Niesen. Convergence of the Magnus series. *J. Found Comput Math*, 9 (3):291–301, 2009. https://doi.org/10.1007/s10208-007-9010-0.

[46] H. Z. Munthe-Kaas. Lie-Butcher theory for Runge-Kutta methods. *BIT*, 35, 1995. https://doi.org/10.1007/BF01739828.

[47] H. Z. Munthe-Kaas. Runge-Kutta methods on Lie groups. *BIT*, 38, 1998. https://doi.org/10.1007/BF02510919.

[48] H. Z. Munthe-Kaas. Higher order Runge-Kutta methods on manifolds. *Appl. Numer. Math.*, 29:115–127, 1999.

[49] H. Z. Munthe-Kaas and A. Zanna. Numerical integration of ordinary differential equations on homogenous manifolds. *Foundations of computational mathematics*, 29, 1997. https://doi.org/10.1007/978-3-642-60539-024.

[50] P. J. Olver. A survey of moving frames. *Lecture Notes in Computer Science*, 3519:105–138, 2004. https://doi.org/10.1007/1149925111.

[51] P. J. Olver. *Applications of Lie groups to differential equations*. Springer, 2013.

[52] B. Owren and A. Marthinsen. Runge-kutta methods adapted to manifolds and based on rigid frames. *BIT Numerical Mathematics*, 39(1):116–142, 1999.

[53] L. Schumaker. *Spline Functions: Basic Theory*. Cambridge Mathematical Library. Cambridge University Press, 3rd edition, 2007.

[54] M. Suzuki. On the convergence of exponential operators—the Zassenhaus formula, BCH formula and systematic approximants. *Commun.Math. Phys.*, 57(3):193–200, 1977. https://doi.org/10.1007/BF01614161.

[55] G. M. Tuynman. The derivation of the exponential map of matrices. *Amer. Math. Monthly*, 102:818–820, 1995.

[56] M. Zadra and E. L. Mansfield. Using lie group integrators to solve two and higher dimensional variational problems with symmetry. *Journal of Computational Dynamics*, 6(2):485–511, 2019.

[57] A. Zanna. The method of iterated commutators for ordinary differential equations on Lie groups. Technical report, DAMTP, University of Cambridge, 1996.

[58] A. Zanna. *On the Numerical Solution of Isospectral Flows*. PhD thesis, University of Cambridge, 1998.

[59] A. Zanna and H. Munthe-Kaas. Iterated commutators, lie's reduction method and ordinary differential equations on matrix lie groups. In F. Cucker and M. Shub, editors, *Foundations of Computational Mathematics*, pages 434–441, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

A. Appendix to Chapter 3

Computations

The following Maple code details all the computations performed in the proof of Theorem 3.2.1. The code is commented thoroughly in order to make it more readable and understandable.The Maple file can be found on Zenodo at the following link: https://zenodo.org/record/3661107/files/computations_second_chapter_final.mw?download=1.

```
restart : with(LinearAlgebra) :

#Definition of the Cayley map, which is also the adjoint representation of the moving frame
adj1 := (a, b, c, d) → Matrix([[2*a^2+2*b^2-1, -2*(a*d-b*c), 2*(a*c+b*d)], [2*(a*d+b*c), 2*a^2+2*c^2-1, -2*(a*b-c*d)], [2*(b*d-a*c), 2*(a*b+c*d), 2*a^2+2*d^2-1]]) :
#KK is the rotation that takes k to w. This is the matrix denoted with K in section 3.2.2
KK := adj1(0, 1/(sqrt(2·k1^2+2·k2^2+2·k3^2+2·(k1·w1(z)+k2·w2(z)+k3·w3(z)))) · (k1+w1(z)), 1/(sqrt(2·k1^2+2·k2^2+2·k3^2+2·(k1·w1(z)+k2·w2(z)+k3·w3(z)))) · (k2+w2(z)), 1/(sqrt(2·k1^2+2·k2^2+2·k3^2+2·(k1·w1(z)+k2·w2(z)+k3·w3(z)))) · (k3+w3(z))) :

#eq1 is the equation ||\mathbf{w}||=||\mathbf{k}||, i.e. expressing the fact that the norms of \mathbf{v} and \mathbf{k} are equal (as adj1 is a rotation)
eq1 := w1(z)^2+w2(z)^2+w3(z)^2-k1^2-k2^2-k3^2 :

#definition of the vector \mathbf{k}
kMT := Matrix([[k1, k2, k3]]) :

#check that KK is the actual rotation matrix that takes \mathbf{k} to \mathbf{v}
KK • Transpose(kMT) : map(simplify, %, {eq1});
```

$$\begin{bmatrix} w1(z) \\ w2(z) \\ w3(z) \end{bmatrix} \tag{1}$$

```
#normalisation equation for the rotation axis of the operator BB, which is the rotation denoted as B in Section 3.2.2
eq2 := a(z)^2+μ^2·(k1^2+k2^2+k3^2)-1 :

#definiton of BB
BB := adj1(a(z), sqrt(1-a(z)^2)/sqrt(k1^2+k2^2+k3^2) · w1(z), sqrt(1-a(z)^2)/sqrt(k1^2+k2^2+k3^2) · w2(z), sqrt(1-a(z)^2)/sqrt(k1^2+k2^2+k3^2) · w3(z)) :

#definition of the vector \mathbf{w}
wM := Matrix([[w1(z)], [w2(z)], [w3(z)]]) :

#check that BB is the rotation that fixes wM
BB • wM : map(simplify, %, {eq1});
```

$$\begin{bmatrix} w1(z) \\ w2(z) \\ w3(z) \end{bmatrix} \tag{2}$$

#a set of differential relations between invariants
$subs1 := diff(w1(z), z) = Q3 \cdot w2(z) - Q2 \cdot w3(z), diff(w2(z), z) = -Q3 \cdot w1(z) + Q1 \cdot w3(z), diff(w3(z), z) = Q2 \cdot w1(z) - Q1 \cdot w2(z)$:

#Differentiating BB and KK. The differentiated variables, Bz and Kz, correspond to B_t and K_t in Section 3.2.2
$Bz := map(factor, simplify(subs(subs1, map(t \to diff(t, z), BB)), \{eq1\}))$:
$simplify(subs(subs1, map(t \to diff(t, z), KK)), \{eq1\})$: $subs(subs1, \%)$: $Kz := map(simplify, \%)$:

#QQ is the curvature matrix, \mathcal{Q}^t in Section 3.2.2
$QQ := Matrix([[0, Q3, -Q2], [-Q3, 0, Q1], [Q2, -Q1, 0]])$:

#eq4 and eq5 are again the fact that kMt and wM have the same norm, which we call M
$eq4 := k1^2 + k2^2 + k3^2 - M^2$:
$eq5 := w1(z)^2 + w2(z)^2 + w3(z)^2 - M^2$:

#definition of the dot product wMT \cdot wM, which we call WK
$eq6 := w1(z) * k1 + w2(z) * k2 + w3(z) * k3 - WK$:

#definition of the dot product \mathbf{q} \cdot wM, which we call QW. \mathbf{q} can be found in Section 3.2.2
$eq9 := Q1 * w1(z) + Q2 * w2(z) + Q3 * w3(z) - QW$:

#definition of the dot product \mathbf{q} \cdot kMT, which we call QK. \mathbf{q} can be found in Section 3.2.2
$eq10 := Q1 * k1 + Q2 * k2 + Q3 * k3 - QK$:

#eq11, eq12, eq13 are the three components of the cross product \mathbf{q} \times wMT
$eq11 := Q3 * w2(z) - Q2 * w3(z) - QCW1$:
$eq12 := -Q3 * w1(z) + Q1 * w3(z) - QCW2$:
$eq13 := Q2 * w1(z) - Q1 * w2(z) - QCW3$:

#The following 6 lines are the computations to prove Lemma 1
$(Bz \bullet Transpose(BB))(1, 2)$: $simplify(\%, \{eq4, eq5, eq9, eq11, eq12, eq13\}, \{w1(z), Q1, Q2, k1, w2(z), w3(z)\})$: $simplify(\%, symbolic)$;

$$-\frac{1}{\sqrt{1 - a(z)^2}\, M^2}\left(2\left((a(z) - 1)\,(a(z) + 1)\,(M^2\, Q3 - QW\, w3(z))\sqrt{1 - a(z)^2}\right.\right. \tag{3}$$
$$\left.\left. - M\left(QCW3\, a(z)^3 - a(z)\, QCW3 + w3(z)\left(\frac{d}{dz}\, a(z)\right)\right)\right)$$

$(Bz \bullet Transpose(BB))(1, 3)$: $simplify(\%, \{eq4, eq5, eq9, eq11, eq12\}, \{w1(z), w2(z), k1, Q1, Q3\})$: $simplify(\%, symbolic)$;

$$\frac{1}{\sqrt{1 - a(z)^2}\, M^2}\left(2\left((a(z) - 1)\,(a(z) + 1)\,(M^2\, Q2 - QW\, w2(z))\sqrt{1 - a(z)^2}\right.\right. \tag{4}$$
$$\left.\left. - M\left(QCW2\, a(z)^3 - QCW2\, a(z) + w2(z)\left(\frac{d}{dz}\, a(z)\right)\right)\right)$$

$(Bz \bullet Transpose(BB))(2,3) : simplify(\%, \{eq4, eq5, eq9, eq11, eq12\}, \{k1, w2(z), Q2, Q3, w3(z)\}) :$
$simplify(\%, symbolic);$

$$-\frac{1}{\sqrt{1-a(z)^2}\,M^2}\left(2\left((a(z)-1)\,(a(z)+1)\,(M^2\,Q1-QW\,w1(z))\sqrt{1-a(z)^2}\right.\right. \tag{5}$$
$$\left.\left.-M\left(QCW1\,a(z)^3-QCW1\,a(z)+w1(z)\left(\frac{d}{dz}\,a(z)\right)\right)\right)\right)$$

$kzkt12 := simplify((Kz \bullet Transpose(KK))(1,2), \{eq4, eq5, eq6, eq9, eq10\}, \{k1, k2, w1(z), w3(z), Q1, Q2\});$

$$kzkt12 := \frac{(-QK-QW)\,w3(z)+Q3\,(M^2+WK)}{M^2+WK} \tag{6}$$

$kzkt13 := simplify((Kz \bullet Transpose(KK))(1,3), \{eq4, eq5, eq6, eq9, eq10\}, \{k1, k2, w1(z), w3(z), Q1, Q2\});$

$$kzkt13 := \frac{(QK+QW)\,w2(z)-Q2\,(M^2+WK)}{M^2+WK} \tag{7}$$

$kzkt23 := simplify((Kz \bullet Transpose(KK))(2,3), \{eq4, eq5, eq6, eq9, eq10\}, \{k1, k2, w2(z), w3(z), Q1, Q2\});$

$$kzkt23 := \frac{(-QK-QW)\,w1(z)+Q1\,(M^2+WK)}{M^2+WK} \tag{8}$$

#Simplifying BB \cdot \mathbf{q}
$DotProduct(BB \bullet Matrix([[Q1],[Q2],[Q3]]), wM, conjugate=false) : simplify(\%, \{eq4, eq5, eq9\}, \{w1(z), Q1, Q2, k1, w2(z), w3(z)\});$

$$QW \tag{9}$$

#definition of \mathbf{q}
$q := \langle Q1(z), Q2(z), Q3(z)\rangle :$

#This is the equation to solve for a(t) at the end of Section 3.2.2
$finaleq := diff(a(z),z)/sqrt(1-a(z)^2) - M/(2*DotProduct(Transpose(kMT), Transpose(kMT)+wM, conjugate=false))*DotProduct(q, wM+Transpose(kMT), conjugate=false);$

$$finaleq := \frac{\frac{d}{dz}\,a(z)}{\sqrt{1-a(z)^2}} - \frac{M\,(Q1(z)\,(k1+w1(z))+Q2(z)\,(k2+w2(z))+Q3(z)\,(k3+w3(z)))}{2\,(k1\,(k1+w1(z))+k2\,(k2+w2(z))+k3\,(k3+w3(z)))} \tag{10}$$

#Solution to the differential equation above
$dsolve(finaleq, a(z)) : simplify(\%, \{k1^2+k2^2+k3^2-M^2\}) : simplify(\%, symbolic);$

$$a(z) = \sin\left(\frac{1}{2}\left(M\left(k3\left(\int\frac{Q3(z)}{k1\,w1(z)+k2\,w2(z)+k3\,w3(z)+M^2}\,dz\right)+k2\left(\right.\right.\right.\right. \tag{11}$$
$$\left.\left.\int\frac{Q2(z)}{k1\,w1(z)+k2\,w2(z)+k3\,w3(z)+M^2}\,dz\right)+k1\left(\right.$$
$$\left.\left.\int\frac{Q1(z)}{k1\,w1(z)+k2\,w2(z)+k3\,w3(z)+M^2}\,dz\right)+\int\frac{Q3(z)\,w3(z)}{k1\,w1(z)+k2\,w2(z)+k3\,w3(z)+M^2}$$

$$dz + \int \frac{Q2(z)\ w2(z)}{k1\ w1(z) + k2\ w2(z) + k3\ w3(z) + M^2}\ dz + \int \frac{Q1(z)\ w1(z)}{k1\ w1(z) + k2\ w2(z) + k3\ w3(z) + M^2}\ dz + \_C1 \Bigg) \Bigg) \Bigg)$$

Example 1

The following Maple code relates to Section 3.3.1and can be found at the following link: https://zenodo.org/record/3661107/files/app_second_first_example.mw?download=1.

```
> restart:
> assume(h>0);
```

#Lagrangian definition

```
> L=lambda^2;
```

$$L = \lambda^2 \tag{1}$$

#Solution to the Euler-Lagrange equations. In this case it's trivial to find it

```
> lambda:=-h:
> mu1:=0:
> mu2:=0
```

#Definition of the vectors \mathbf{w} and \mathbf{k} in Section 3.2.2

```
> w:=<-2*lambda,0,0>;
```

$$w := \begin{bmatrix} 2\,h\sim \\ 0 \\ 0 \end{bmatrix} \tag{2}$$

```
> k:=<h,0,sqrt(3)*h>;
```

$$k := \begin{bmatrix} h\sim \\ 0 \\ \sqrt{3}\,h\sim \end{bmatrix} \tag{3}$$

#The Cayley map, which in this case is the Adjoint representation of the moving frame

```
> adj1 := (a, b, c, d) -> Matrix([[2*a^2 + 2*b^2 - 1, -2*a*d + 2*b*
  c, 2*a*c + 2*b*d], [2*a*d + 2*b*c, 2*a^2 + 2*c^2 - 1, -2*b*a + 2*
  c*d], [2*b*d - 2*a*c, 2*b*a + 2*c*d, 2*a^2 + 2*d^2 - 1]]);
```

$$adj1 := (a, b, c, d) \mapsto Matrix([[2\,a^2 + 2\,b^2 - 1,\ -2\,a\,d + 2\,b\,c,\ 2\,a\,c + 2\,b\,d],\ [2\,a\,d + 2\,b\,c,\ 2\,a^2 + 2\,c^2 - 1,\ -2\,b\,a + 2\,c\,d],\ [-2\,a\,c + 2\,b\,d,\ 2\,b\,a + 2\,c\,d,\ 2\,a^2 + 2\,d^2 - 1]]) \tag{4}$$

#Matrix K in Section 3.2.2

```
> KK := subs({k1=k[1],k2=k[2],k3=k[3],w1(z)=w[1],w2(z)=w[2],w3(z)=w
  [3]},adj1(0, (k1+w1(z))/sqrt(2*k1^2+2*k2^2+2*k3^2+2*(k1*w1(z)+k2*
  w2(z)+k3*w3(z))), (k2+w2(z))/sqrt(2*k1^2+2*k2^2+2*k3^2+2*(k1*w1
  (z)+k2*w2(z)+k3*w3(z))),(k3+w3(z))/sqrt(2*k1^2+2*k2^2+2*k3^2+2*
  (k1*w1(z)+k2*w2(z)+k3*w3(z))));
```

$$KK := \begin{bmatrix} \frac{1}{2} & 0 & \frac{\sqrt{3}}{2} \\ 0 & -1 & 0 \\ \frac{\sqrt{3}}{2} & 0 & -\frac{1}{2} \end{bmatrix} \tag{5}$$

#Check that KK takes k to w

```
> KK . k
```

$$\tag{6}$$

$$\begin{bmatrix} 2\,h\sim \\ 0 \\ 0 \end{bmatrix} \tag{6}$$

#Definition of the vector \mathbf{q} in Section 3.2.2

```
> q:=2*<lambda,mu1,mu2>:
```

#Equation for a(t)

```
> eqaz:=subs({Q1(z)=q[1],Q2(z)=q[2],Q3(z)=q[3],M=norm(k,2),k1=k[1],
  k2=k[2],k3=k[3],w1(z)=w[1],w2(z)=w[2],w3(z)=w[3]},(diff(a(z), z))
  /sqrt(1-a(z)^2)-(1/2)*M*(Q1(z)*(k1+w1(z))+Q2(z)*(k2+w2(z))+Q3(z)*
  (k3+w3(z)))/(k1*(k1+w1(z))+k2*(k2+w2(z))+k3*(k3+w3(z))))
```

$$eqaz := \frac{\frac{\mathrm{d}}{\mathrm{d}z}\,a(z)}{\sqrt{1-a(z)^2}} + h\sim \tag{7}$$

#Solution for a(t)

```
> sola:=rhs(dsolve({%,a(0)=0},a(z)))
```

$$sola := -\sin(z\,h\sim) \tag{8}$$

#Matrix B in Section 3.2.2

```
> BB := subs({Q1(z)=q[1],Q2(z)=q[2],Q3(z)=q[3],k1=k[1],k2=k[2],k3=k
  [3],w1(z)=w[1],w2(z)=w[2],w3(z)=w[3],a(z)=sola},adj1(a(z), sqrt(1
  -a(z)^2)*w1(z)/sqrt(k1^2+k2^2+k3^2), sqrt(1-a(z)^2)*w2(z)/sqrt
  (k1^2+k2^2+k3^2),sqrt(1-a(z)^2)*w3(z)/sqrt(k1^2+k2^2+k3^2)));
```

$$BB := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2\sin(z\,h\sim)^2 - 1 & \frac{\sqrt{1-\sin(z\,h\sim)^2}\;h\sim\sqrt{4}\;\sin(z\,h\sim)}{\sqrt{h\sim^2}} \\ 0 & -\frac{\sqrt{1-\sin(z\,h\sim)^2}\;h\sim\sqrt{4}\;\sin(z\,h\sim)}{\sqrt{h\sim^2}} & 2\sin(z\,h\sim)^2 - 1 \end{bmatrix} \tag{9}$$

#Definition of the adjoint representation of the moving frame, as the product of the two rotations BB and KK

```
> adrho:=BB.KK;
```

$$adrho := \left[\left[\frac{1}{2},\ 0,\ \frac{\sqrt{3}}{2}\right],\ \left[\frac{\sqrt{1-\sin(z\,h\sim)^2}\;h\sim\sqrt{4}\;\sin(z\,h\sim)\sqrt{3}}{2\sqrt{h\sim^2}},\ -2\sin(z\,h\sim)^2+1,\ -\frac{\sqrt{1-\sin(z\,h\sim)^2}\;h\sim\sqrt{4}\;\sin(z\,h\sim)}{2\sqrt{h\sim^2}}\right],\ \left[\frac{(2\sin(z\,h\sim)^2-1)\sqrt{3}}{2},\ \frac{\sqrt{1-\sin(z\,h\sim)^2}\;h\sim\sqrt{4}\;\sin(z\,h\sim)}{\sqrt{h\sim^2}},\ -\sin(z\,h\sim)^2+\frac{1}{2}\right]\right] \tag{10}$$

#Check that BB fixes w

> $BB \cdot w$

$$\begin{bmatrix} 2\,h\sim \\ 0 \\ 0 \end{bmatrix} \tag{11}$$

#Check that the moving frame takes k to w

> $adrho \cdot k : map(simplify, \%);$

$$\begin{bmatrix} 2\,h\sim \\ 0 \\ 0 \end{bmatrix} \tag{12}$$

#Inversion of the Cayley map, in order to find the parameters a,b,c and d in Section 3.2.2

```
> a1:=subs({h=1},1/2*sqrt(1+adrho[1,1]+adrho[2,2]+adrho[3,3]));
```

$$a1 := \frac{\sqrt{3 - 3\sin(z)^2}}{2} \tag{13}$$

> $b1 := -\frac{1}{4 \cdot a1} \cdot (adrho[2,3] - adrho[3,2]);$

$$b1 := \frac{3\sqrt{1 - \sin(z\,h\sim)^2}\; h\sim \sqrt{4}\; \sin(z\,h\sim)}{4\sqrt{3 - 3\sin(z)^2}\;\sqrt{h\sim^2}} \tag{14}$$

> $c1 := -\frac{1}{4 \cdot a1} \cdot (adrho[3,1] - adrho[1,3]);$

$$c1 := -\frac{\frac{\left(2\sin(z\,h\sim)^2 - 1\right)\sqrt{3}}{2} - \frac{\sqrt{3}}{2}}{2\sqrt{3 - 3\sin(z)^2}} \tag{15}$$

> $d1 := -\frac{1}{4 \cdot a1} \cdot (adrho[1,2] - adrho[2,1]) : simplify(\%);$

$$\frac{\mathrm{csgn}(\cos(z))\;\mathrm{csgn}(\cos(z\,h\sim))\;\cos(z\,h\sim)\;\sin(z\,h\sim)}{2\cos(z)} \tag{16}$$

#Check that the squared parameters sum to 1 (it holds for h=1)

> $a1^2 + b1^2 + c1^2 + d1^2 : simplify(\%);$

$$\frac{-3\cos(z\,h\sim)^4 + 3\cos(z)^4 + 4\cos(z\,h\sim)^2}{4\cos(z)^2} \tag{17}$$

#compute the minimisers u and v as done at the end of Section 3.2.2

```
> u:=subs(h=1,a1-I*b1):factor(%);
```

$$-\frac{3\left(\mathrm{I}\sqrt{-(\sin(z)-1)(\sin(z)+1)}\,\sin(z) + \sin(z)^2 - 1\right)}{2\sqrt{-3(\sin(z)-1)(\sin(z)+1)}} \tag{18}$$

```
> v:=subs(h=1,c1-I*d1):factor(%);
```

$$-\frac{\sqrt{3}\left(\mathrm{I}\sqrt{-(\sin(z)-1)(\sin(z)+1)}\,\sin(z) + \sin(z)^2 - 1\right)}{2\sqrt{-3(\sin(z)-1)(\sin(z)+1)}} \tag{19}$$

#Plot the minimisers

```
> plot([[evalc(Re(u)),evalc(Im(u)),z=-6..6],[-evalc(Re(u)),-evalc
```

```
   (Im(u)),z=-6..6]],labels=["Re(u)","Im(u)"],labeldirections=
   [horizontal,vertical],color=["black","green"]):
> plot([[evalc(Re(v)),evalc(Im(v)),z=-6..6],[-evalc(Re(v)),-evalc
   (Im(v)),z=-6..6]],labels=["Re(v)","Im(v)"],labeldirections=
   [horizontal,vertical],color=["black","green"]):
```

Example 2

The following Maple code relates to Section 3.3.2and can be found at the following link: https://zenodo.org/record/3661107/files/app_second_second_example.mw?download=1.

```
> restart:
```

#Lambda, mu1 and mu2 can be easily derived from the Euler-Lagrange equations for this example

```
> f := -1 : g := 3 : h := 2 :
> lambda:=f*z^2+g*z+h;
```

$$\lambda := -z^2 + 3z + 2 \tag{1}$$

```
> mu1:=0:
> mu2:=0:
```

#Definition of the vectors \mathbf{w} and \mathbf{k} in Section 3.2.2

```
> w:=<-2*diff(lambda,z,z),0,0>;
```

$$w := \begin{bmatrix} 4 \\ 0 \\ 0 \end{bmatrix} \tag{2}$$

```
> k:=<-3,2,-sqrt(3)>;
```

$$k := \begin{bmatrix} -3 \\ 2 \\ -\sqrt{3} \end{bmatrix} \tag{3}$$

#Definition of the Cayley map, which is the adjoint representation of the moving frame in this case

```
> adj1 := (a, b, c, d) -> Matrix([[2*a^2 + 2*b^2 - 1, -2*a*d + 2*b*
  c, 2*a*c + 2*b*d], [2*a*d + 2*b*c, 2*a^2 + 2*c^2 - 1, -2*b*a + 2*
  c*d], [2*b*d - 2*a*c, 2*b*a + 2*c*d, 2*a^2 + 2*d^2 - 1]]);
```

$$adj1 := (a, b, c, d) \mapsto Matrix([[2a^2 + 2b^2 - 1, -2ad + 2bc, 2ac + 2bd], [2ad + 2bc, 2a^2 + 2c^2 - 1, -2ba + 2cd], [-2ac + 2bd, 2ba + 2cd, 2a^2 + 2d^2 - 1]]) \tag{4}$$

#Definition of matrix K in section 3.2.2

```
> KK := subs({k1=k[1],k2=k[2],k3=k[3],w1(z)=w[1],w2(z)=w[2],w3(z)=w
  [3]},adj1(0, (k1+w1(z))/sqrt(2*k1^2+2*k2^2+2*k3^2+2*(k1*w1(z)+k2*
  w2(z)+k3*w3(z))), (k2+w2(z))/sqrt(2*k1^2+2*k2^2+2*k3^2+2*(k1*w1
  (z)+k2*w2(z)+k3*w3(z))),(k3+w3(z))/sqrt(2*k1^2+2*k2^2+2*k3^2+2*
  (k1*w1(z)+k2*w2(z)+k3*w3(z)))));
```

$$KK := \begin{bmatrix} -\frac{3}{4} & \frac{1}{2} & -\frac{\sqrt{3}}{4} \\ \frac{1}{2} & 0 & -\frac{\sqrt{3}}{2} \\ -\frac{\sqrt{3}}{4} & -\frac{\sqrt{3}}{2} & -\frac{1}{4} \end{bmatrix} \tag{5}$$

#Check that KK takes k to w

```
> KK . k
```

$$\begin{bmatrix} 4 \\ 0 \\ 0 \end{bmatrix} \tag{6}$$

#Definition of vector \mathbf{q} in Section 3.2.2

```
> q:=2*<lambda,mu1,mu2>:
```

#Equation for a(t)

```
> eqaz:=subs({Q1(z)=q[1],Q2(z)=q[2],Q3(z)=q[3],M=norm(k,2),k1=k[1],
  k2=k[2],k3=k[3],w1(z)=w[1],w2(z)=w[2],w3(z)=w[3]},(diff(a(z), z))
  /sqrt(1-a(z)^2)-(1/2)*M*(Q1(z)*(k1+w1(z))+Q2(z)*(k2+w2(z))+Q3(z)*
  (k3+w3(z)))/(k1*(k1+w1(z))+k2*(k2+w2(z))+k3*(k3+w3(z))))
```

$$eqaz := \frac{\frac{d}{dz}a(z)}{\sqrt{1-a(z)^2}} + z^2 - 3z - 2 \tag{7}$$

```
> sola:=rhs(dsolve({%,a(0)=1/2},a(z)))
```

$$sola := \cos\left(\frac{1}{3}z^3 - \frac{3}{2}z^2 + \frac{1}{3}\pi - 2z\right) \tag{8}$$

#Definition of the matrix B in Section 3.2.2

```
> BB := subs({Q1(z)=q[1],Q2(z)=q[2],Q3(z)=q[3],k1=k[1],k2=k[2],k3=k
  [3],w1(z)=w[1],w2(z)=w[2],w3(z)=w[3],a(z)=sola},adj1(a(z), sqrt(1
  -a(z)^2)*w1(z)/sqrt(k1^2+k2^2+k3^2), sqrt(1-a(z)^2)*w2(z)/sqrt
  (k1^2+k2^2+k3^2),sqrt(1-a(z)^2)*w3(z)/sqrt(k1^2+k2^2+k3^2)));
```

$$BB := \left[\left[1, 0, 0\right], \left[0, 2\cos\left(\frac{1}{3}z^3 - \frac{3}{2}z^2 + \frac{1}{3}\pi - 2z\right)^2 - 1, -\frac{\sqrt{1-\cos\left(\frac{1}{3}z^3 - \frac{3}{2}z^2 + \frac{1}{3}\pi - 2z\right)^2}\sqrt{16}\cos\left(\frac{1}{3}z^3 - \frac{3}{2}z^2 + \frac{1}{3}\pi - 2z\right)}{2}\right], \left[0, \frac{\sqrt{1-\cos\left(\frac{1}{3}z^3 - \frac{3}{2}z^2 + \frac{1}{3}\pi - 2z\right)^2}\sqrt{16}\cos\left(\frac{1}{3}z^3 - \frac{3}{2}z^2 + \frac{1}{3}\pi - 2z\right)}{2}, 2\cos\left(\frac{1}{3}z^3 - \frac{3}{2}z^2 + \frac{1}{3}\pi - 2z\right)^2 - 1\right]\right] \tag{9}$$

#Check that BB fixes w

```
> BB . w
```

$$\begin{bmatrix} 4 \\ 0 \\ 0 \end{bmatrix} \tag{10}$$

#Definition of the moving frame

```
> adrho:=BB.KK;
```

$$adrho := \left[\left[-\frac{3}{4}, \frac{1}{2}, -\frac{\sqrt{3}}{4}\right], \right. \tag{11}$$

$$\left[-\frac{1}{2} + \cos\left(\frac{1}{3} z^3 - \frac{3}{2} z^2 + \frac{1}{3} \pi - 2 z \right)^2 \right.$$

$$+ \frac{1}{8} \left(\sqrt{1 - \cos\left(\frac{1}{3} z^3 - \frac{3}{2} z^2 + \frac{1}{3} \pi - 2 z \right)^2} \sqrt{16} \cos\left(\frac{1}{3} z^3 - \frac{3}{2} z^2 + \frac{1}{3} \pi \right.\right.$$

$$\left.\left. - 2 z \right) \sqrt{3} \right),$$

$$\frac{\sqrt{1 - \cos\left(\frac{1}{3} z^3 - \frac{3}{2} z^2 + \frac{1}{3} \pi - 2 z \right)^2} \sqrt{16} \cos\left(\frac{1}{3} z^3 - \frac{3}{2} z^2 + \frac{1}{3} \pi - 2 z \right) \sqrt{3}}{4},$$

$$- \frac{\left(2 \cos\left(\frac{1}{3} z^3 - \frac{3}{2} z^2 + \frac{1}{3} \pi - 2 z \right)^2 - 1 \right) \sqrt{3}}{2}$$

$$\left. + \frac{\sqrt{1 - \cos\left(\frac{1}{3} z^3 - \frac{3}{2} z^2 + \frac{1}{3} \pi - 2 z \right)^2} \sqrt{16} \cos\left(\frac{1}{3} z^3 - \frac{3}{2} z^2 + \frac{1}{3} \pi - 2 z \right)}{8} \right],$$

$$\left[\frac{\sqrt{1 - \cos\left(\frac{1}{3} z^3 - \frac{3}{2} z^2 + \frac{1}{3} \pi - 2 z \right)^2} \sqrt{16} \cos\left(\frac{1}{3} z^3 - \frac{3}{2} z^2 + \frac{1}{3} \pi - 2 z \right)}{4} \right.$$

$$- \frac{\left(2 \cos\left(\frac{1}{3} z^3 - \frac{3}{2} z^2 + \frac{1}{3} \pi - 2 z \right)^2 - 1 \right) \sqrt{3}}{4},$$

$$- \frac{\left(2 \cos\left(\frac{1}{3} z^3 - \frac{3}{2} z^2 + \frac{1}{3} \pi - 2 z \right)^2 - 1 \right) \sqrt{3}}{2}, \frac{1}{4}$$

$$- \frac{\sqrt{1 - \cos\left(\frac{1}{3} z^3 - \frac{3}{2} z^2 + \frac{1}{3} \pi - 2 z \right)^2} \sqrt{16} \cos\left(\frac{1}{3} z^3 - \frac{3}{2} z^2 + \frac{1}{3} \pi - 2 z \right) \sqrt{3}}{4}$$

$$\left. \left. - \frac{\cos\left(\frac{1}{3} z^3 - \frac{3}{2} z^2 + \frac{1}{3} \pi - 2 z \right)^2}{2} \right] \right]$$

#Check that the moving frame takes k to w

> $adrho \cdot k : map(simplify, \%);$

$$\begin{bmatrix} 4 \\ 0 \\ 0 \end{bmatrix}$$

(12)

#Inverting the Cayley map to obtain the frame in terms of the parameters

```
> a1:=1/2*sqrt(1+adrho[1,1]+adrho[2,2]+adrho[3,3]);
```

$$a1 := \frac{\sqrt{2-2\cos\left(\frac{1}{3}z^3-\frac{3}{2}z^2+\frac{1}{3}\pi-2z\right)^2}}{4} \tag{13}$$

```
> b1 := -1/(4*a1)*(adrho[2,3]-adrho[3,2]);
```

$$b1 := -\frac{\sqrt{1-\cos\left(\frac{1}{3}z^3-\frac{3}{2}z^2+\frac{1}{3}\pi-2z\right)^2}\,\sqrt{16}\,\cos\left(\frac{1}{3}z^3-\frac{3}{2}z^2+\frac{1}{3}\pi-2z\right)}{8\sqrt{2-2\cos\left(\frac{1}{3}z^3-\frac{3}{2}z^2+\frac{1}{3}\pi-2z\right)^2}} \tag{14}$$

```
> c1 := -1/(4*a1)*(adrho[3,1]-adrho[1,3]);
```

$$c1 := \tag{15}$$

$$-\frac{1}{\sqrt{2-2\cos\left(\frac{1}{3}z^3-\frac{3}{2}z^2+\frac{1}{3}\pi-2z\right)^2}}\left(\frac{1}{4}\left(\sqrt{1-\cos\left(\frac{1}{3}z^3-\frac{3}{2}z^2+\frac{1}{3}\pi-2z\right)^2}\,\sqrt{16}\,\cos\left(\frac{1}{3}z^3-\frac{3}{2}z^2+\frac{1}{3}\pi-2z\right)\right)\right.$$

$$\left.-\frac{\left(2\cos\left(\frac{1}{3}z^3-\frac{3}{2}z^2+\frac{1}{3}\pi-2z\right)^2-1\right)\sqrt{3}}{4}+\frac{\sqrt{3}}{4}\right)$$

```
> d1 := -1/(4*a1)*(adrho[1,2]-adrho[2,1]) : simplify(%);
```

$$\frac{1}{4\sin\left(\frac{1}{3}z^3-\frac{3}{2}z^2+\frac{1}{3}\pi-2z\right)}\left(\sqrt{2}\left(\operatorname{csgn}\left(\sin\left(\frac{1}{3}z^3-\frac{3}{2}z^2+\frac{1}{3}\pi-2z\right)\right)\sin\left(\frac{1}{3}z^3\right.\right.\right. \tag{16}$$

$$\left.-\frac{3}{2}z^2+\frac{1}{3}\pi-2z\right)\cos\left(\frac{1}{3}z^3-\frac{3}{2}z^2+\frac{1}{3}\pi-2z\right)\sqrt{3}+2\cos\left(\frac{1}{3}z^3-\frac{3}{2}z^2\right.$$

$$\left.\left.\left.+\frac{1}{3}\pi-2z\right)^2-2\right)\operatorname{csgn}\left(\sin\left(\frac{1}{3}z^3-\frac{3}{2}z^2+\frac{1}{3}\pi-2z\right)\right)\right)$$

#Computing the minimisers as done at the end of Section 3.2.2

```
> u:=a1-I*b1;
```

$$u := \frac{\sqrt{2-2\cos\left(\frac{1}{3}z^3-\frac{3}{2}z^2+\frac{1}{3}\pi-2z\right)^2}}{4} \tag{17}$$

$$+\frac{I\sqrt{1-\cos\left(\frac{1}{3}z^3-\frac{3}{2}z^2+\frac{1}{3}\pi-2z\right)^2}\sqrt{16}\cos\left(\frac{1}{3}z^3-\frac{3}{2}z^2+\frac{1}{3}\pi-2z\right)}{8\sqrt{2-2\cos\left(\frac{1}{3}z^3-\frac{3}{2}z^2+\frac{1}{3}\pi-2z\right)^2}}$$

```
> v:=c1-I*d1;
```

$v :=$ **(18)**

$$-\frac{1}{\sqrt{2-2\cos\left(\frac{1}{3}z^3-\frac{3}{2}z^2+\frac{1}{3}\pi-2z\right)^2}}\left(\frac{1}{4}\left(\sqrt{1-\cos\left(\frac{1}{3}z^3-\frac{3}{2}z^2+\frac{1}{3}\pi-2z\right)^2}\sqrt{16}\cos\left(\frac{1}{3}z^3-\frac{3}{2}z^2+\frac{1}{3}\pi-2z\right)\right)-\frac{\left(2\cos\left(\frac{1}{3}z^3-\frac{3}{2}z^2+\frac{1}{3}\pi-2z\right)^2-1\right)\sqrt{3}}{4}+\frac{\sqrt{3}}{4}\right)$$

$$+\frac{1}{\sqrt{2-2\cos\left(\frac{1}{3}z^3-\frac{3}{2}z^2+\frac{1}{3}\pi-2z\right)^2}}\left(I\left(1-\cos\left(\frac{1}{3}z^3-\frac{3}{2}z^2+\frac{1}{3}\pi-2z\right)^2-\frac{1}{8}\left(\sqrt{1-\cos\left(\frac{1}{3}z^3-\frac{3}{2}z^2+\frac{1}{3}\pi-2z\right)^2}\sqrt{16}\cos\left(\frac{1}{3}z^3-\frac{3}{2}z^2+\frac{1}{3}\pi-2z\right)\sqrt{3}\right)\right)\right)$$

#Plot the minimisers

```
> plot([[evalc(Re(u)), evalc(Im(u)), z=-2..2], [-evalc(Re(u)), -evalc(Im(u)), z=-2..2]],
      labels=["Re(u)", "Im(u)"], labeldirections=[horizontal, vertical], color=["black",
      "green"]) :
```

```
> plot([[evalc(Re(v)), evalc(Im(v)), z=-2..2], [-evalc(Re(v)), -evalc(Im(v)), z=-2..2]],
      labels=["Re(v)", "Im(v)"], labeldirections=[horizontal, vertical], color=["black",
      "green"]) :
```

#Check that the parameters sum to 1

```
> a1^2+b1^2+c1^2+d1^2 : simplify(%);
```

$$1$$ **(19)**

Example 3

The following Maple code relates to Section 3.3.3and can be found at the following link: https://zenodo.org/record/3661107/files/app_second_third_example_1.mw?download=1.

```
> restart:
```
#Euler-Lagrange equations for this example
```
> el1:=-diff(lambda(z),z)+2*mu2(z)*mu1(z):
> el2:=diff(mu1(z),z)+2*mu2(z)*lambda(z):
```
#Solution to the Euler-Lagrange equations
```
> elsol:=dsolve({el1,el2,lambda(0)=1,mu1(0)=0},{lambda(z),mu1(z)});
```

$$elsol := \left\{ \lambda(z) = \frac{e^{\int_0^z 2\,I\mu2(\_z1)\,d\_z1}}{2} + \frac{e^{-\left(\int_0^z 2\,I\mu2(\_z1)\,d\_z1\right)}}{2},\ \mu1(z) = I\left(\frac{e^{\int_0^z 2\,I\mu2(\_z1)\,d\_z1}}{2} - \frac{e^{-\left(\int_0^z 2\,I\mu2(\_z1)\,d\_z1\right)}}{2} \right) \right\} \tag{1}$$

```
> lambda:=evalc(rhs(elsol[1]));
```

$$\lambda := \cos\left(\int_0^z 2\,\mu2(\_z1)\,d\_z1\right) \tag{2}$$

```
> mu1:=evalc(rhs(elsol[2]));
```

$$\mu1 := -\sin\left(\int_0^z 2\,\mu2(\_z1)\,d\_z1\right) \tag{3}$$

```
> mu2(_z1):=1:
```
#Definition of the vectors w and k in Section 3.2.2
```
> w:=<2*lambda,-2*mu1,0>;
```

$$w := \begin{bmatrix} 2\cos\left(\int_0^z 2\,d\_z1\right) \\ 2\sin\left(\int_0^z 2\,d\_z1\right) \\ 0 \end{bmatrix} \tag{4}$$

```
> k:=<0,0,2>;
```

$$k := \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix} \tag{5}$$

#Definition of the Cayley map
```
> adj1 := (a, b, c, d) -> Matrix([[2*a^2 + 2*b^2 - 1, -2*a*d + 2*b*
  c, 2*a*c + 2*b*d], [2*a*d + 2*b*c, 2*a^2 + 2*c^2 - 1, -2*b*a + 2*
  c*d], [2*b*d - 2*a*c, 2*b*a + 2*c*d, 2*a^2 + 2*d^2 - 1]]);
```

$$adj1 := (a,b,c,d) \mapsto Matrix\left(\left[\left[2a^2+2b^2-1,\ -2ad+2bc,\ 2ac+2bd\right],\left[2ad+2bc,\ 2a^2+2c^2-1,\ -2ab+2cd\right],\left[-2ac+2bd,\ 2ab+2cd,\ 2a^2+2d^2-1\right]\right]\right) \tag{6}$$

#Definition of the matrix K from Section 3.2.2
```
> KK := subs({k1=k[1],k2=k[2],k3=k[3],w1(z)=w[1],w2(z)=w[2],w3(z)=w
  [3]},adj1(0, (k1+w1(z))/sqrt(2*k1^2+2*k2^2+2*k3^2+2*(k1*w1(z)+k2*
```

```
w2(z)+k3*w3(z))), (k2+w2(z))/sqrt(2*k1^2+2*k2^2+2*k3^2+2*(k1*w1
(z)+k2*w2(z)+k3*w3(z))),(k3+w3(z))/sqrt(2*k1^2+2*k2^2+2*k3^2+2*
(k1*w1(z)+k2*w2(z)+k3*w3(z))));
```

$$KK := \begin{bmatrix} -1+\cos\left(\int_0^z 2\,d\_z1\right)^2 & \cos\left(\int_0^z 2\,d\_z1\right)\sin\left(\int_0^z 2\,d\_z1\right) & \cos\left(\int_0^z 2\,d\_z1\right) \\ \cos\left(\int_0^z 2\,d\_z1\right)\sin\left(\int_0^z 2\,d\_z1\right) & -1+\sin\left(\int_0^z 2\,d\_z1\right)^2 & \sin\left(\int_0^z 2\,d\_z1\right) \\ \cos\left(\int_0^z 2\,d\_z1\right) & \sin\left(\int_0^z 2\,d\_z1\right) & 0 \end{bmatrix} \quad (7)$$

#Check that K takes k to w

```
> KK.k: map(simplify,%);
```

$$\begin{bmatrix} 2\cos\left(2\left(\int_0^z 1\,d\_z1\right)\right) \\ 2\sin\left(2\left(\int_0^z 1\,d\_z1\right)\right) \\ 0 \end{bmatrix} \quad (8)$$

#Definition of the vector q from Section 3.2.2

```
> q:=2*<lambda,mu1,mu2(_z1)>:
```

#Equation for a(t) from Section 3.2.2

```
> eqaz:=subs({Q1(z)=q[1],Q2(z)=q[2],Q3(z)=q[3],M=norm(k,2),k1=k[1],
  k2=k[2],k3=k[3],w1(z)=w[1],w2(z)=w[2],w3(z)=w[3]},(diff(a(z), z))
  /sqrt(1-a(z)^2)-(1/2)*M*(Q1(z)*(k1+w1(z))+Q2(z)*(k2+w2(z))+Q3(z)*
  (k3+w3(z)))/(k1*(k1+w1(z))+k2*(k2+w2(z))+k3*(k3+w3(z))));
```

$$eqaz := \frac{\frac{d}{dz}a(z)}{\sqrt{1-a(z)^2}} - \cos\left(\int_0^z 2\,d\_z1\right)^2 + \sin\left(\int_0^z 2\,d\_z1\right)^2 - 1 \quad (9)$$

#Solving for a(t)

```
> sola:=rhs(dsolve({%,a(0)=1},a(z)));
```

$$sola := \cos\left(8\left(\int_0^z \cos\left(\int_0^{\_z2} 1\,d\_z1\right)^4 d\_z2\right) - 8\left(\int_0^z \cos\left(\int_0^{\_z2} 1\,d\_z1\right)^2 d\_z2\right) + 2\left(\int_0^z 1\,d\_z2\right)\right) \quad (10)$$

#Definition of Matrix B from Section 3.2.2

```
> BB := subs({Q1(z)=q[1],Q2(z)=q[2],Q3(z)=q[3],k1=k[1],k2=k[2],k3=k
  [3],w1(z)=w[1],w2(z)=w[2],w3(z)=w[3],a(z)=sola},adj1(a(z), sqrt(1
  -a(z)^2)*w1(z)/sqrt(k1^2+k2^2+k3^2),sqrt(1-a(z)^2)*w2(z)/sqrt
  (k1^2+k2^2+k3^2),sqrt(1-a(z)^2)*w3(z)/sqrt(k1^2+k2^2+k3^2))):
```

#Definition of the moving frame

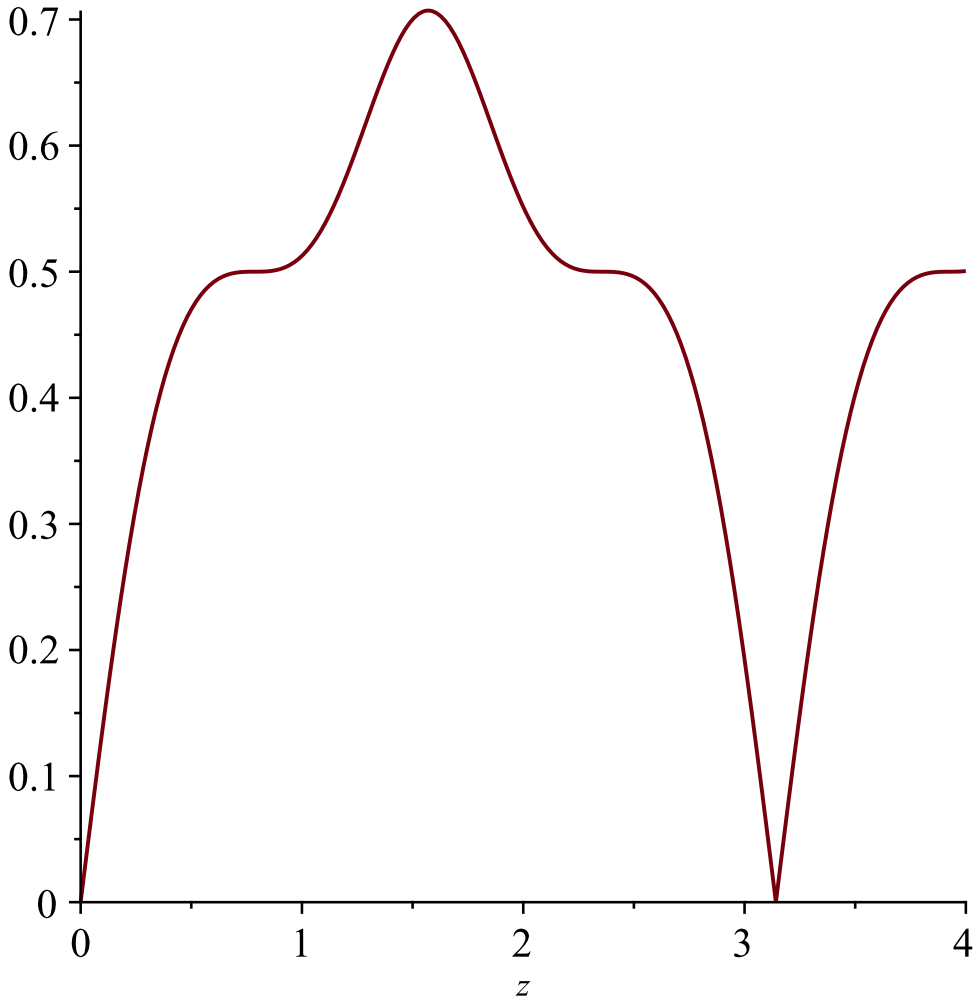
```
> adrho:=BB.KK:
```

#Inverting the Cayley map to get the frame in terms of the parameters

```
> a1:=1/2*sqrt(1+adrho[1,1]+adrho[2,2]+adrho[3,3]):
```

#The Cayley map is only locally invertible in general. That's why we need to pay attention to the zeroes of a(t), as those are the points where it's not invertible

> $plot(a1, z=0..4)$



> $b1 := -\frac{1}{4 \cdot a1} \cdot (adrho[2,3] - adrho[3,2])$:

> $c1 := -\frac{1}{4 \cdot a1} \cdot (adrho[3,1] - adrho[1,3])$:

> $d1 := -\frac{1}{4 \cdot a1} \cdot (adrho[1,2] - adrho[2,1])$:

#Computing the minimisers as done at the end of Section 3.2.2

```
> u:=a1-I*b1:
> v:=c1-I*d1:
```

#Plotting the minimisers

```
> with(plots):
> plot([[a1,-b1,z=0.001..3.15],[-a1,b1,z=0.001..3.15]],labels=["Re
  (u)","Im(u)"],labeldirections=[horizontal,vertical],color=
  ["black","green"]):
> plot([[c1,-d1,z=0.001..3.15],[-c1,d1,z=0.001..3.15]],labels=["Re
  (v)","Im(v)"],labeldirections=[horizontal,vertical],color=
  ["black","green"]):
```

Example 4

The following Maple code relates to Section 3.3.4and can be found at the following link: https://zenodo.org/record/3661107/files/app_second_fourth_example.mw?download=1.

```
> restart:
```

#Defining the generating differential invariants for this example (trivial to obtain them from the Euler-Lagrange equations)

```
> lambda:=-1:
> mu1:=2:
> mu2:=1:
```

#Definition of the vectors w and k from Section 3.2.2

```
> w:=<-2*lambda,-2*mu1,2*mu2>;
```

$$w := \begin{bmatrix} 2 \\ -4 \\ 2 \end{bmatrix} \tag{1}$$

```
> k:=<0,0,sqrt(24)>;
```

$$k := \begin{bmatrix} 0 \\ 0 \\ 2\sqrt{6} \end{bmatrix} \tag{2}$$

#Definition of the Cayley map

```
> adj1 := (a, b, c, d) -> Matrix([[2*a^2 + 2*b^2 - 1, -2*a*d + 2*b*
  c, 2*a*c + 2*b*d], [2*a*d + 2*b*c, 2*a^2 + 2*c^2 - 1, -2*b*a + 2*
  c*d], [2*b*d - 2*a*c, 2*b*a + 2*c*d, 2*a^2 + 2*d^2 - 1]]);
```

$$adj1 := (a, b, c, d) \mapsto Matrix([[2a^2 + 2b^2 - 1, -2ad + 2bc, 2ac + 2bd], [2ad + 2bc, 2a^2 + 2c^2 - 1, -2ba + 2cd], [-2ac + 2bd, 2ba + 2cd, 2a^2 + 2d^2 - 1]]) \tag{3}$$

#Definition of the matrix K from Section 3.2.2

```
> KK := subs({k1=k[1],k2=k[2],k3=k[3],w1(z)=w[1],w2(z)=w[2],w3(z)=w
  [3]},adj1(0, (k1+w1(z))/sqrt(2*k1^2+2*k2^2+2*k3^2+2*(k1*w1(z)+k2*
  w2(z)+k3*w3(z))), (k2+w2(z))/sqrt(2*k1^2+2*k2^2+2*k3^2+2*(k1*w1
  (z)+k2*w2(z)+k3*w3(z))),(k3+w3(z))/sqrt(2*k1^2+2*k2^2+2*k3^2+2*
  (k1*w1(z)+k2*w2(z)+k3*w3(z))));
```

$$KK := \begin{bmatrix} -1 + \frac{8}{48 + 8\sqrt{6}} & -\frac{16}{48 + 8\sqrt{6}} & \frac{4(2\sqrt{6} + 2)}{48 + 8\sqrt{6}} \\ -\frac{16}{48 + 8\sqrt{6}} & -1 + \frac{32}{48 + 8\sqrt{6}} & -\frac{8(2\sqrt{6} + 2)}{48 + 8\sqrt{6}} \\ \frac{4(2\sqrt{6} + 2)}{48 + 8\sqrt{6}} & -\frac{8(2\sqrt{6} + 2)}{48 + 8\sqrt{6}} & -1 + \frac{2(2\sqrt{6} + 2)^2}{48 + 8\sqrt{6}} \end{bmatrix} \tag{4}$$

#Check that the matrix K takes k to w

```
> KK . k : simplify(%);
```

$$\begin{bmatrix} 2 \\ -4 \\ 2 \end{bmatrix} \tag{5}$$

#Definition of the vector q from Section 3.2.2

```
> q:=2*<lambda,mu1,mu2>:
```

#Equation for a(t)

```
> eqaz:=subs({Q1(z)=q[1],Q2(z)=q[2],Q3(z)=q[3],M=norm(k,2),k1=k[1],
  k2=k[2],k3=k[3],w1(z)=w[1],w2(z)=w[2],w3(z)=w[3]},(diff(a(z), z))
  /sqrt(1-a(z)^2)-(1/2)*M*(Q1(z)*(k1+w1(z))+Q2(z)*(k2+w2(z))+Q3(z)*
  (k3+w3(z)))/(k1*(k1+w1(z))+k2*(k2+w2(z))+k3*(k3+w3(z))));
```

$$eqaz := \frac{\frac{d}{dz}a(z)}{\sqrt{1-a(z)^2}} - \frac{-16+4\sqrt{6}}{2\left(2\sqrt{6}+2\right)} \tag{6}$$

```
> sola:=rhs(dsolve({%,a(0)=1/2},a(z)));
```

$$sola := \sin\left(\frac{\pi\sqrt{6}+6z\sqrt{6}+\pi-24z}{6\left(\sqrt{6}+1\right)}\right) \tag{7}$$

#Definition of matrix B from Section 3.2.2

```
> BB := subs({Q1(z)=q[1],Q2(z)=q[2],Q3(z)=q[3],k1=k[1],k2=k[2],k3=k
  [3],w1(z)=w[1],w2(z)=w[2],w3(z)=w[3],a(z)=sola},adj1(a(z), sqrt(1
  -a(z)^2)*w1(z)/sqrt(k1^2+k2^2+k3^2),sqrt(1-a(z)^2)*w2(z)/sqrt
  (k1^2+k2^2+k3^2),sqrt(1-a(z)^2)*w3(z)/sqrt(k1^2+k2^2+k3^2))):
```

#check that BB fixes w

```
> BB . w
```

$$\begin{bmatrix} 2 \\ -4 \\ 2 \end{bmatrix} \tag{8}$$

#Definition of the moving frame for this example

```
> adrho:=BB.KK:
```

#Inverting the Cayley map to obtain the frame in terms of the parameters

```
> a1:=1/2*sqrt(1+adrho[1,1]+adrho[2,2]+adrho[3,3]):
```

$$> \quad b1 := -\frac{1}{4 \cdot a1} \cdot (adrho[2,3] - adrho[3,2]) :$$

$$> \quad c1 := -\frac{1}{4 \cdot a1} \cdot (adrho[3,1] - adrho[1,3]) :$$

$$> \quad d1 := -\frac{1}{4 \cdot a1} \cdot (adrho[1,2] - adrho[2,1]) :$$

#Computing the minimisers u and v as done at the end of Section 3.2.2

```
> u:=a1-I*b1:
> v:=c1-I*d1:
```

#Plotting the minimisers

```
> with(plots):
> plot([[a1,-b1,z=-13..13],[-a1,b1,z=-13..13]],labels=["Re(u)","Im
  (u)"],labeldirections=[horizontal,vertical],color=["black",
  "green"]):
> plot([[c1,-d1,z=-14..15],[-c1,d1,z=-14..15]],labels=["Re(v)","Im
  (v)"],labeldirections=[horizontal,vertical],color=["black",
  "green"]):
```

#Check the frame parameters sum to 1

```
> a1^2+b1^2+c1^2+d1^2 : simplify(%);
```

$$1 \tag{9}$$

B. Appendix to Chapter 4

Computations

The following Maple code relates to Section 4.2.4. The Maple file can be found at: https://zenodo.org/record/3661107/files/app_third_1.mw?download=1 .

Below we include pseudocode for the most important routines in the algorithm presented at the end of this section.

```
Function myHx(h,k,ordera,orderb)
%Creates all the terms of \Theta^x from Section 3.2.2 and expand
%the integrands in a Taylor polynomial of order=[orderh]+[orderk]
%around ([x],[y])
 if (orderh<=0 )
    return 0;
 else  order=orderh+orderk
 end
 Expand Q^x(xi,y_0) around x_0 in a Taylor polynomial of degree order
 Create the commutators of the Magnus expansion up to order 3
 if (order == 4)
 Create commutators of the Magnus expansion of order 4
 end
 Return the sum of the commutators with their coefficients
```

Fig. B.1 Pseudocode for the routine myHx

The function myHy works exactly as myHx, but replacing x with y, ξ with x_0 and y with ξ.

```
Function myTerms(term, a, b, ordera, orderb)
%myTerms isolates the expressions of order [ordera] in [a] and
%[orderb] in [b] in an expression [term]
x1=x0+h;
y1=y0+k;
substitute (x1,y1) in [term]
coefficient1=extract the coefficient of order ordera in a from term
coefficient2=extract the coefficient of order orderb in b from term
return b^orderb*a^ordera*coefficient1*coefficient2

```

Fig. B.2 Pseudocode for the routine myTerms

```
Function Comm2(A, B, a, b, ordera, orderb)
%Comm2 picks the expressions of order [ordera] in [a] and [orderb] in [b]
%from the commutator [A,B]
for i from 0 to [ordera] do
    for j from 0 to [orderb] do
    A1 := myTerms(A, a, b, i, j);
    B1 := myTerms(B, a, b, ordera - i, orderb - j);
    comm := comm + Commutator(A1, B1);  %Commutator(A1,B1) creates the
    %commutator [A1,B1]=A1*B1-B1*A1
    comm1 := myTerms(comm, a, b, ordera, orderb);
    collect the terms in the powers of [a] and [b] from comm1
    end;
end;
return The terms of of order=[ordera] in [a] and [orderb] in [b]
       from the commutator [A,B]

```

Fig. B.3 Pseudocode for the routine Comm2

All the other functions with the name starting as “Comm” work in a similar way. Their aim is to isolate the terms of order [ordera] in [a] and [orderb] in [b] from a specific commutator appearing in the Magnus expansion. The relevant commutator is specified in the code with a comment before each routine.

```
1   Function LOOP(orderh,orderk)
2   %This routine computes all the terms up to order=[orderh] in h and [orderk]
3   %in k in the equation (4.17)
4   LOOP1=Construct log(\rho^{\gamma_1}(x_1,y_1) \rho_0^{-1}) up to order 5
5   LOOP2=Construct log(\rho^{\gamma_2}(x_1,y_1) \rho_0^{-1}) up to order 5
6   return (LOOP1-LOOP2)/(h^[orderh]*k^[orderk])
7
```

Fig. B.4 Pseudocode for the routine LOOP

We think the rest of the code is easier to understand looking at the comments in it. We define the differential relation myR and use it, and its derivatives, to simplify the different instances of LOOP(orderh,orderk).

```
> restart :
> with(Physics) : Setup(mathematicalnotation = true) : Setup(noncommutativeprefix = {R, Qx,
      Qy}) :
```

myHx creates the terms up to order [orderh] in h and [orderk] in k for the matrix Q^x evaluated at the point ([x],[y]). It works up to [orderh]+[orderk]=4

```
> myHx := proc(x, y, orderh, orderk)
  local Sum, Y, texp, m1, m2, m3, m4, m5, m6, m7, texpint, order :
  global x0, y0, Qx, h, k, xi1, xi2, xi3, h2, h3, x1, y1;
  if ( orderh ≤ 0 ) then
  0;
  else
  order := orderh + orderk;
  texp := subs( Y = y, mtaylor( Qx(xi1, Y), [xi1 = x0, Y = y0], order) ) :
  texpint := int(texp, xi1 = x .. xi2) :
  Sum[1] := myTerms(subs(xi2 = x + h, texpint), h, k, orderh, orderk) :
  m1 := Comm2(subs(xi2 = x0 + h2, texpint), subs( {xi1 = x0 + h2}, texp), h2, k, orderh − 1,
      orderk) :
  Sum[2] := - 1/2 · int(m1, h2 = 0 ..h) :
  if order ≥ 4 then
  m2 := Comm3(subs(xi2 = x0 + h2, texpint), subs(xi2 = x0 + h2, texpint), subs(xi1 = x0 + h2,
      texp), h2, k, orderh − 1, orderk) :
  m3 := Comm3intx(subs(xi2 = x0 + h2, texpint), subs(xi1 = x0 + h2, texp), subs(xi1 = x0 + h3,
      texp), orderh − 1, orderk) :
  Sum[3] := 1/12 ·int(m2, h2 = 0 ..h) + 1/4 ·int(m3, h3 = 0 .. h);
  m4 := Comm41x(subs( {xi2 = x0 + h2}, texpint), subs(xi2 = x0 + h2, texpint), subs(xi1 = x0
      + h2, texp), subs(xi1 = x0 + h3, texp), orderh − 1, orderk);
  m5 := Comm42x(subs( {xi2 = x0 + h2}, texpint), subs(xi1 = x0 + h2, texp), subs(xi2 = x0 + h3,
      texpint), subs(xi1 = x0 + h3, texp), orderh − 1, orderk);
  m6 := Comm43x(subs( {xi2 = x0 + h3}, texpint), subs(xi2 = x0 + h2, texpint), subs(xi1 = x0
      + h2, texp), subs(xi1 = x0 + h3, texp), orderh − 1, orderk);
  m7 := Comm44x(subs( {xi2 = x0 + h3}, texpint), subs(xi2 = x0 + h3, texpint), subs(xi2 = x0
      + h3, texpint), subs(xi1 = x0 + h3, texp), orderh − 1, orderk);
  Sum[4] := - 1/24 ·int(m4 + m5 + m7, h3 = 0 ..h) - 1/8 ·int(m7, h3 = 0 ..h);
  Sum[1] + Sum[2] + Sum[3] + Sum[4];
    else
  Sum[1] + Sum[2];
  end if;
  end if;
   end:
```

myHy creates the terms up to order [orderh] in h and [orderk] in k for the matrix Q^v evaluated at the point ([x],[y]). It works up to [orderh]+[orderk]=4

```
> myHy := proc(x, y, orderh, orderk)
  local Sum, X, texp, m1, m2, m3, m4, m5, m6, m7, texpint, order :
  global x0, y0, Qy, h, k, xi1, xi2, xi3, k2, k3;
  if orderk ≤ 0 then
  0;
  else
  order := orderh + orderk;
```

```
texp := subs(X=x, mtaylor(Qy(X, xi1), [xi1=y0, X=x0], order)) :
texpint := int(texp, xi1=y .. xi2) :
Sum[1] := myTerms(subs(xi2=y+k, texpint), h, k, orderh, orderk) :
m1 := Comm2(subs(xi2=y0+k2, texpint), subs({xi1=y0+k2}, texp), h, k2, orderh, orderk
      -1) :
Sum[2] := -1/2 · int(m1, k2=0..k) :
if order ≥ 4 then
m2 := Comm3(subs(xi2=y0+k2, texpint), subs(xi2=y0+k2, texpint), subs(xi1=y0+k2,
      texp), h, k2, orderh, orderk-1) :
m3 := Comm3inty(subs(xi2=y0+k2, texpint), subs(xi1=y0+k2, texp), subs(xi1=y0+k3,
      texp), orderh, orderk-1) :
Sum[3] := 1/12 ·int(m2, k2=0..k) + 1/4 ·int(m3, k3=0 .. k) :

m4 := Comm41y(subs(xi2=y0+k2, texpint), subs(xi2=y0+k2, texpint), subs(xi1=y0+k2,
      texp), subs(xi1=y0+k3, texp), orderh, orderk-1);
m5 := Comm42y(subs(xi2=y0+k2, texpint), subs(xi1=y0+k2, texp), subs(xi2=y0+k3,
      texpint), subs(xi1=y0+k3, texp), orderh, orderk-1);
m6 := Comm43y(subs(xi2=y0+k3, texpint), subs(xi2=y0+k2, texpint), subs(xi1=y0+k2,
      texp), subs(xi1=y0+k3, texp), orderh, orderk-1);
m7 := Comm44y(subs(xi2=y0+k3, texpint), subs(xi2=y0+k3, texpint), subs(xi2=y0+k3,
      texpint), subs(xi1=y0+k3, texp), orderh, orderk-1);
Sum[4] := -1/24 ·int(m4+m5+m6, k3=0..k) - 1/8 ·int(m7, k3=0..k);
Sum[1]+Sum[2]+Sum[3]+Sum[4];
 else
Sum[1] + Sum[2];
 end if;
 end if;
  end:
```

myTerms isolates the expressions of order [ordera] in [a] and [orderb] in [b] in an expression [term]

```
> myTerms :=proc(term, a, b, ordera, orderb)
  local term1;
  term1 := subs({x1=x0+h, y1=y0+k}, term) :
  coeff(term1, b, orderb) : b^orderb·a^ordera·coeff(%, a, ordera);
  end:
```

All the following routines up to the routine Comm2Hx (excluded) serve the function of creating the commutators appearing in the magnus expansions of Theta^h and Theta^v up to order 5 in h,k. They are all called by the functions myHx and myHy

Comm2 picks the expressions of order [ordera] in [a] and [orderb] in [b] from the commutator [A,B]

```
> Comm2 := proc(A, B, a, b, ordera, orderb)
  local i, j, A1, B1, comm, comm1;
  comm := 0;
  for i from 0 to ordera do
  for j from 0 to orderb do
  A1 := simplify(myTerms(A, a, b, i, j)) :
  B1 := simplify(myTerms(B, a, b, ordera-i, orderb-j)) :
  comm := comm + Commutator(A1, B1) :
  comm1 := myTerms(comm, a, b, ordera, orderb);
```

```
    collect(comm1, {a, b}, distributed);
    end do: end do;
    end:
```
Comm3 picks the expressions of order [ordera] in [a] and [orderb] in [b] from the commutator [A,[B,C]]
```
> Comm3 :=proc(A, B, C, a, b, ordera, orderb)
    local r, s, mysum, commprov;
     mysum := 0;
    for r from 0 to ordera do
    for s from 0 to orderb do
    commprov := Comm2(B, C, a, b, r, s);
    mysum := mysum + Comm2(A, commprov, a, b, ordera, orderb);
    end do:
    end do:
    end proc:
```
Comm3intx picks the expressions of order [ordera] in [a] and [orderb] in [b] from the commutator [\int [A,B],C] where \int is horizontal integration
```
> Comm3intx :=proc(A, B, C, ordera, orderb)
    local r, s, mysum, commprov;
    mysum := 0;
    for r from 0 to ordera  do
    for s from 0 to orderb   do
    commprov := Comm2(A, B, h2, k, r, s);
    mysum := mysum + Comm2(int(commprov, h2 = 0 ..h3), C, h3, k, ordera, orderb)
    end do:
    end do:
    end proc:
```
Comm3inty picks the expressions of order [ordera] in [a] and [orderb] in [b] from the commutator [\int [A,B],C] where \int is vertical integration
```
> Comm3inty :=proc(A, B, C, ordera, orderb)
    local r, s, mysum, commprov;
    mysum := 0;
    for r from 0 to ordera  do
    for s from 0 to orderb   do
    commprov := Comm2(A, B, h, k2, r, s);
   mysum := mysum + Comm2(int(commprov, k2 = 0 ..k3), C, h, k3, ordera, orderb)
    end do:
    end do:
    end proc:
```
Comm41x picks the expressions of order [ordera] in [a] and [orderb] in [b] from the commutator [[A,[B,C]],DD]
```
> Comm41x :=proc(A, B, C, DD, ordera, orderb)
    local r, s, mysum, sumprov;
    mysum := 0;
    for r from  0 to ordera do
    for s from 0 to orderb do
    sumprov := int(Comm3(A, B, C, h2, k, r, s), h2 = 0 ..h3);
    mysum := mysum + Comm2(sumprov, DD, h3, k, ordera, orderb);
    end do;
    end do;
```

```
    end proc:
```
Comm42x picks the expressions of order [ordera] in [a] and [orderb] in [b] from the commutator [[A, B],[C,DD]]
```
> Comm42x := proc(A, B, C, DD, ordera, orderb)
    local r, s, mysum, sumprov1, sumprov2;
    mysum := 0;
    for r from 0 to ordera do
    for s from 0 to orderb do
    sumprov1 := int(Comm2(A, B, h2, k, r, s), h2 = 0 ..h3);
    sumprov2 := Comm2(C, DD, h3, k, ordera − r, orderb − s);
    mysum := mysum + Commutator(sumprov1, sumprov2);
    end do;
    end do;
    end proc:
```
Comm43x picks the expressions of order [ordera] in [a] and [orderb] in [b] from the commutator [A, [\int[B,C],DD]], where \int is horizontal integration
```
> Comm43x := proc(A, B, C, DD, ordera, orderb)
    local r, s, mysum, sumprov;
    mysum := 0;
    for r from 0 to ordera do
    for s from 0 to orderb do
    sumprov := Comm3intx(B, C, DD, r, s);
    mysum := mysum + Comm2(A, sumprov, h3, k, ordera, orderb);
    end do;
    end do;
    end proc:
```
Comm44x picks the expressions of order [ordera] in [a] and [orderb] in [b] from the commutator [A,[B, [C,DD]]]
```
> Comm44x := proc(A, B, C, DD, ordera, orderb)
    local r, s, mysum, sumprov;
    mysum := 0;
    for r from 0 to ordera do
    for s from 0 to orderb do
    sumprov := Comm3(B, C, DD, h3, k, r, s);
    mysum := mysum + Comm2(A, sumprov, h3, k, ordera, orderb);
    end do;
    end do;
    end proc:
```
Comm41y picks the expressions of order [ordera] in [a] and [orderb] in [b] from the commutator [[A, [B,C]],DD]
```
> Comm41y := proc(A, B, C, DD, ordera, orderb)
    local r, s, mysum, sumprov;
    mysum := 0;
    for r from  0 to ordera do
    for s from 0 to orderb do
    sumprov := int(Comm3(A, B, C, h, k2, r, s), k2 = 0 ..k3);
    mysum := mysum + Comm2(sumprov, DD, h, k3, ordera, orderb);
    end do;
    end do;
    end proc:
```
Comm42y picks the expressions of order [ordera] in [a] and [orderb] in [b] from the commutator [\int

[A,B],[C,DD]], where \int is vertical integration

```
> Comm42y := proc(A, B, C, DD, ordera, orderb)
  local r, s, mysum, sumprov1, sumprov2;
  mysum := 0;
  for r from 0 to ordera do
  for s from 0 to orderb do
  sumprov1 := int(Comm2(A, B, h, k2, r, s), k2 = 0 ..k3);
  sumprov2 := Comm2(C, DD, h, k3, ordera - r, orderb - s);
  mysum := mysum + Commutator(sumprov1, sumprov2);
  end do;
  end do;
  end proc:
```

Comm43y picks the expressions of order [ordera] in [a] and [orderb] in [b] from the commutator [A, [\int[B,C],DD]]

```
> Comm43y := proc(A, B, C, DD, ordera, orderb)
  local r, s, mysum, sumprov;
  mysum := 0;
  for r from 0 to ordera do
  for s from 0 to orderb do
  sumprov := Comm3inty(B, C, DD, r, s);
  mysum := mysum + Comm2(A, sumprov, h, k3, ordera, orderb);
  end do;
  end do;
  end proc:
```

Comm44y picks the expressions of order [ordera] in [a] and [orderb] in [b] from the commutator [A,[B, [C,DD]]]

```
> Comm44y := proc(A, B, C, DD, ordera, orderb)
  local r, s, mysum, sumprov;
  mysum := 0;
  for r from 0 to ordera do
  for s from 0 to orderb do
  sumprov := Comm3(B, C, DD, h, k3, r, s);
  mysum := mysum + Comm2(A, sumprov, h, k3, ordera, orderb);
  end do;
  end do;
  end proc:
```

The following set of routines up to Loop (excluded) create the terms we need in the BCH truncated expansion of rho^gamma1(x1,y1) and rho^gamma2(x1,y1)

Comm2Hx picks the terms of order [orderh] in h and [orderk] in k of [Theta^h(xx,xy),Theta^v(yx,yy)]

```
> Comm2Hx := proc(xx, xy, yx, yy, orderh, orderk)
  local i, j, Hx, Hy;
  global commhx;
  commhx := 0;
   for i from 1 to orderh  do
  for j from 1 to orderk  do
  Hx := myHx(xx, xy, i, j - 1);
  Hy := myHy(yx, yy, orderh - i, orderk - j + 1);
  commhx := commhx + Commutator(Hx, Hy);
   end do;
  end do;
  collect(commhx, {h, k}, distributed);
```

```
    end proc:
```
Comm2Hy picks the terms of order [orderh] in h and [orderk] in k of [Theta^v(yx,yy),Theta^h(xx,xy)]
```
> Comm2Hy := proc(yx, yy, xx, xy, orderh, orderk)
    local i, j, Hx, Hy;
    global commhy;
    commhy := 0;
     for i from 1 to orderh  do
    for j from 1 to orderk  do
    Hx := myHx(xx, xy, i, j - 1);
    Hy := myHy(yx, yy, orderh - i, orderk - j + 1);
    commhy := commhy + Commutator(Hy, Hx);
     end do;
    end do;
   collect(commhy, {h, k}, distributed);
    end proc:
```
Comm3Hx picks the terms of order [orderh] in h and [orderk] in k of [Theta^h(xx,xy),[Theta^h(xx,xy), Theta^v(yx,yy)]]
```
> Comm3Hx := proc(xx, xy, yx, yy, orderh, orderk)
    local l, m, Hx, Hy, comm;
    comm := 0;
    if (orderh = 0 or orderk = 0) then
    comm := 0;
    else
     for l from 1 to orderh  do
    for m from 1 to orderk  do
    Hx := myHx(xx, xy, l, m - 1);
   comm := comm + Comm2(Hx, Comm2Hx(xx, xy, yx, yy, orderh - l, orderk - m + 1), h, k,
        orderh, orderk);
    collect(comm, {h, k}, distributed);
    end do;
    end do;
    end if;
    end proc:
```
Comm3Hy picks the terms of order [orderh] in h and [orderk] in k of [Theta^v(yx,yy),[Theta^v(yx,yy), Theta^h(xx,xy)]]
```
> Comm3Hy := proc(xx, xy, yx, yy, orderh, orderk)
    local l, m, Hx, Hy, comm;
     comm := 0;
   if (orderh = 0 or orderk = 0) then
    comm := 0;
    else
    for l from 1 to orderh  do
    for m from 1 to orderk  do
    Hy := myHy(yx, yy, l - 1, m);
   comm := comm + Comm2(Hy, Comm2Hy(yx, yy, xx, xy, orderh - l + 1, orderk - m), h, k,
        orderh, orderk);
    collect(comm, {h, k}, distributed);
    end do;
    end do;
    end if
    end proc:
```

Comm4Hyxxy picks the terms of order [orderh] in h and [orderk] in k of [Theta^v(yx,yy),[Theta^h(xx,xy),[Theta^h(xx,xy),Theta^v(yx,yy)]]]

```
> Comm4Hyxxy := proc(xx, xy, yx, yy, orderh, orderk)
  local l, m, Hy, comm;
   comm := 0;
  for l from 1 to orderh do
  for m from 1 to orderk do
  Hy := myHy(yx, yy, l - 1, m);
  comm := comm + Commutator(Hy, Comm3Hx(xx, xy, yx, yy, orderh - l + 1, orderk - m));
  collect(comm, {h, k}, distributed);
  end do;
  end do:
  end proc:
```

Comm4Hxyyx picks the terms of order [orderh] in h and [orderk] in k of [Theta^h(xx,xy),[Theta^v(yx,yy),[Theta^v(xy,yy),Theta^h(xx,xy)]]]

```
> Comm4Hxyyx := proc(xx, xy, yx, yy, orderh, orderk)
  local l, m, Hx, comm;
   comm := 0;
  for l from 1 to orderh do
  for m from 1 to orderk do
  Hx := myHx(xx, xy, l, m - 1);
  comm := comm + Commutator(Hx, Comm3Hy(xx, xy, yx, yy, orderh - l, orderk - m + 1));
  collect(comm, {h, k}, distributed);
  end do;
  end do:
  end proc:
```

Comm4Hyyyx picks the terms of order [orderh] in h and [orderk] in k of [Theta^v(yx,yy),[Theta^v(yx,yy),[Theta^v(xy,yy),Theta^h(xx,xy)]]]

```
> Comm4Hyyyx := proc(xx, xy, yx, yy, orderh, orderk)
  local l, m, Hx, Hy, comm;
  comm := 0;
 if (orderh = 0 or orderk = 0) then
  comm := 0;
  else
 for l from 1 to orderh do
  for m from 1 to orderk do
   Hy := myHy(yx, yy, l - 1, m);
 comm := comm + Commutator(Hy, Comm3Hy(xx, xy, yx, yy, orderh - l + 1, orderk - m));
  collect(comm, {h, k}, distributed);
  end do;
  end do;
  end if;
  end proc:
```

Comm4Hxxxy picks the terms of order [orderh] in h and [orderk] in k of [Theta^h(xx,xy),[Theta^h(xx,xy),[Theta^h(xx,xy),Theta^v(yx,yy)]]]

```
> Comm4Hxxxy := proc(xx, xy, yx, yy, orderh, orderk)
   local l, m, Hx, comm;
   comm := 0;
 if (orderh = 0 or orderk = 0) then
 comm := 0;
 else
```

```
    for l from 1 to orderh do
    for m from 1 to orderk do
    Hx := myHx(xx, xy, l, m - 1);
    comm := comm + Commutator(Hx, Comm3Hx(xx, xy, yx, yy, orderh - l, orderk - m + 1));
    collect(comm, {h, k}, distributed);
    end do;
    end do:
    end if;
    end proc:
```

Comm4Hxyxy picks the terms of order [orderh] in h and [orderk] in k of [Theta^h(xx,xy),[Theta^v(yx,yy),[Theta^h(xx,xy),Theta^v(yx,yy)]]]

```
> Comm4Hxyxy :=proc(xx, xy, yx, yy, orderh, orderk)
    local l, m, Hx, comm;
    comm := 0;
    if (orderh = 0 or orderk = 0) then
    comm := 0;
    else
    for l from 1 to orderh do
    for m from 1 to orderk do
     Hx := myHx(xx, xy, l, m - 1);
    comm := comm + Commutator(Hx,-Comm3Hy(xx, xy, yx, yy, orderh - l, orderk - m + 1));
    collect(comm, {h, k}, distributed);
    end do;
    end do;
    end if;
    end proc:
```

Comm4Hyxyx picks the terms of order [orderh] in h and [orderk] in k of [Theta^v(yx,yy),[Theta^h(xx,xy),[Theta^v(yx,yy),Theta^h(xx,xy)]]]

```
> Comm4Hyxyx :=proc(xx, xy, yx, yy, orderh, orderk)
     local l, m, Hy, comm;
    comm := 0;
    if (orderh = 0 or orderk = 0) then
    comm := 0;
    else
    for l from 1 to orderh do
    for m from 1 to orderk do
     Hy := myHy(xx, xy, l - 1, m);
    comm := comm + Commutator(Hy,-Comm3Hx(xx, xy, yx, yy, orderh - l + 1, orderk - m));
    collect(comm, {h, k}, distributed);
    end do;
    end do;
    end if;
    end proc:
```

Comm5Hyyyyx picks the terms of order [orderh] in h and [orderk] in k of [Theta^v,[Theta^v(yx,yy),[Theta^v(yx,yy),[Theta^v(xy,yy),Theta^h(xx,xy)]]]]

```
> Comm5Hyyyyx := proc(xx, xy, yx, yy, orderh, orderk)
    local l, m, Hx, Hy, comm;
    comm := 0;
    if (orderh = 0 or orderk = 0) then
    comm := 0;
    else
```

```
    for l from 1 to orderh  do
    for m from 1 to orderk  do
     Hy := myHy(yx, yy, l − 1, m);
    comm := comm + Commutator(Hy, Comm4Hyyyx(xx, xy, yx, yy, orderh − l + 1, orderk − m));
     collect(comm, {h, k}, distributed);
     end do;
     end do;
     end if;
     end proc:
```

Comm5Hxxxxy picks the terms of order [orderh] in h and [orderk] in k of [Theta^h,[Theta^h(xx,xy),[Theta^h(xx,xy),[Theta^h(xx,xy),Theta^v(yx,yy)]]]]

```
> Comm5Hxxxxy := proc(xx, xy, yx, yy, orderh, orderk)
    local l, m, Hx, comm;
    comm := 0;
    if (orderh = 0 or orderk = 0) then
    comm := 0;
    else
    for l from 1 to orderh  do
    for m from 1 to orderk  do
    Hx := myHx(xx, xy, l, m − 1);
    comm := comm + Commutator(Hx, Comm4Hxxxy(xx, xy, yx, yy, orderh − l, orderk − m + 1));
     collect(comm, {h, k}, distributed);
     end do;
     end do;
     end if;
     end proc:
```

Comm5Hxyyyx picks the terms of order [orderh] in h and [orderk] in k of [Theta^h(xx,xy),[Theta^v(yx,yy),[Theta^v(yx,yy),[Theta^v(yx,yy),Theta^h(xx,xy)]]]]

```
> Comm5Hxyyyx := proc(xx, xy, yx, yy, orderh, orderk)
    local l, m, Hx, Hy, comm;
    comm := 0;
    if (orderh = 0 or orderk = 0) then
    comm := 0;
    else
    for l from 1 to orderh  do
    for m from 1 to orderk  do
     Hx := myHx(xx, xy, l, m − 1);
    comm := comm + Commutator(Hx, Comm4Hyyyx(xx, xy, yx, yy, orderh − l, orderk − m + 1));
     collect(comm, {h, k}, distributed);
     end do;
     end do;
     end if;
     end proc:
```

Comm5Hyxxxy picks the terms of order [orderh] in h and [orderk] in k of [Theta^v(yx,yy),[Theta^h(xx,xy),[Theta^h(xx,xy),[Theta^h(xx,xy),Theta^v(yx,yy)]]]]

```
> Comm5Hyxxxy := proc(xx, xy, yx, yy, orderh, orderk)
     local l, m, Hx, comm, Hy;
    comm := 0;
    if (orderh = 0 or orderk = 0) then
    comm := 0;
    else
```

```
    for l from 1 to orderh  do
    for m from 1 to orderk  do
    Hy := myHy(xx, xy, l − 1, m);
    comm := comm + Commutator(Hy, Comm4Hxxxy(xx, xy, yx, yy, orderh − l + 1, orderk − m));
     collect(comm, {h, k}, distributed);
     end do;
     end do;
     end if;
     end proc:
```

Comm5Hyxyxy picks the terms of order [orderh] in h and [orderk] in k of [Theta^v(yx,yy),[Theta^h(xx,xy),[Theta^v(yx,yy),[Theta^h(xx,xy),Theta^v(yx,yy)]]]]

```
> Comm5Hyxyxy := proc(xx, xy, yx, yy, orderh, orderk)
    local l, m, Hy, comm;
    comm := 0;
    if (orderh = 0 or orderk = 0) then
    comm := 0;
    else
    for l from 1 to orderh  do
     for m from 1 to orderk  do
      Hy := myHy(yx, yy, l − 1, m);
    comm := comm + Commutator(Hy, Comm4Hxyxy(xx, xy, yx, yy, orderh − l + 1, orderk − m));
     collect(comm, {h, k}, distributed);
     end do;
     end do;
     end if;
     end proc:
```

Comm5Hxyxyx picks the terms of order [orderh] in h and [orderk] in k of [Theta^h(xx,xy),[Theta^v(yx,yy),[Theta^h(xx,xy),[Theta^v(yx,yy),Theta^h(xx,xy)]]]]

```
> Comm5Hxyxyx := proc(xx, xy, yx, yy, orderh, orderk)
    local l, m, Hx, comm;
    comm := 0;
    if (orderh = 0 or orderk = 0) then
    comm := 0;
    else
    for l from 1 to orderh  do
     for m from 1 to orderk  do
      Hx := myHx(xx, xy, l, m − 1);
    comm := comm + Commutator(Hx, Comm4Hyxyx(xx, xy, yx, yy, orderh − l, orderk − m + 1));
     collect(comm, {h, k}, distributed);
     end do;
     end do;
     end if;
     end proc:
```

Loop uses the routines defined above to compute the terms of order [orderh] in h and [orderk] in k of (rho^gamma1(x1,y1)-rho^gamma2(x1,y1))

```
> Loop := proc(orderh, orderk)
    global x0, y0, x1, y1;
    HX00 := myHx(x0, y0, orderh, orderk);
    HX01 := myHx(x0, y1, orderh, orderk);
    HY00 := myHy(x0, y0, orderh, orderk);
    HY10 := myHy(x1, y0, orderh, orderk);
```

$$COMM21 := Comm2Hx(x0, y1, x0, y0, orderh, orderk);$$
$$COMM22 := Comm2Hy(x1, y0, x0, y0, orderh, orderk);$$

$$COMM31 := Comm3Hx(x0, y1, x0, y0, orderh, orderk);$$
$$COMM32 := Comm3Hy(x0, y1, x0, y0, orderh, orderk);$$
$$COMM33 := Comm3Hy(x0, y0, x1, y0, orderh, orderk);$$
$$COMM34 := Comm3Hx(x0, y0, x1, y0, orderh, orderk);$$

$$COMM41 := Comm4Hyxxy(x0, y1, x0, y0, orderh, orderk);$$
$$COMM42 := Comm4Hxyyx(x0, y0, x1, y0, orderh, orderk);$$

$$COMM511 := Comm5Hyyyyx(x0, y1, x0, y0, orderh, orderk);$$
$$COMM521 := Comm5Hxxxxy(x0, y1, x0, y0, orderh, orderk);$$
$$COMM531 := Comm5Hxyyyx(x0, y1, x0, y0, orderh, orderk);$$
$$COMM541 := Comm5Hyxxxy(x0, y1, x0, y0, orderh, orderk);$$
$$COMM551 := Comm5Hyxyxy(x0, y1, x0, y0, orderh, orderk);$$
$$COMM561 := Comm5Hxyxyx(x0, y1, x0, y0, orderh, orderk);$$
$$COMM512 := Comm5Hxxxxy(x0, y0, x1, y0, orderh, orderk);$$
$$COMM522 := Comm5Hyyyyx(x0, y0, x1, y0, orderh, orderk);$$
$$COMM532 := Comm5Hyxxxy(x0, y0, x1, y0, orderh, orderk);$$
$$COMM542 := Comm5Hxyyyx(x0, y0, x1, y0, orderh, orderk);$$
$$COMM552 := Comm5Hxyxyx(x0, y0, x1, y0, orderh, orderk);$$
$$COMM562 := Comm5Hyxyxy(x0, y0, x1, y0, orderh, orderk);$$

$$LOOP1 := HX01 + HY00 + \frac{1}{2}\cdot COMM21 + \frac{1}{12}\cdot(COMM31 + COMM32) - \frac{1}{24}\cdot COMM41 - \frac{1}{720}\cdot(COMM511 + COMM521) + \frac{1}{360}\cdot(COMM531 + COMM541) + \frac{1}{120}\cdot(COMM551 + COMM561);$$

$$LOOP2 := HX00 + HY10 + \frac{1}{2}\cdot COMM22 + \frac{1}{12}\cdot(COMM33 + COMM34) - \frac{1}{24}\cdot COMM42 - \frac{1}{720}\cdot(COMM512 + COMM522) + \frac{1}{360}\cdot(COMM532 + COMM542) + \frac{1}{120}\cdot(COMM552 + COMM562);$$

$$LOOP := \frac{(LOOP1 - LOOP2)}{h^{orderh}\cdot k^{orderk}} : simplify(\%);$$

end proc:

Warning, `HX00` is implicitly declared local to procedure `Loop`
Warning, `HX01` is implicitly declared local to procedure `Loop`
Warning, `HY00` is implicitly declared local to procedure `Loop`
Warning, `HY10` is implicitly declared local to procedure `Loop`
Warning, `COMM21` is implicitly declared local to procedure `Loop`
Warning, `COMM22` is implicitly declared local to procedure `Loop`
Warning, `COMM31` is implicitly declared local to procedure `Loop`
Warning, `COMM32` is implicitly declared local to procedure `Loop`
Warning, `COMM33` is implicitly declared local to procedure `Loop`

```
Warning, `COMM34` is implicitly declared local to procedure
`Loop`
Warning, `COMM41` is implicitly declared local to procedure
`Loop`
Warning, `COMM42` is implicitly declared local to procedure
`Loop`
Warning, `COMM511` is implicitly declared local to procedure
`Loop`
Warning, `COMM521` is implicitly declared local to procedure
`Loop`
Warning, `COMM531` is implicitly declared local to procedure
`Loop`
Warning, `COMM541` is implicitly declared local to procedure
`Loop`
Warning, `COMM551` is implicitly declared local to procedure
`Loop`
Warning, `COMM561` is implicitly declared local to procedure
`Loop`
Warning, `COMM512` is implicitly declared local to procedure
`Loop`
Warning, `COMM522` is implicitly declared local to procedure
`Loop`
Warning, `COMM532` is implicitly declared local to procedure
`Loop`
Warning, `COMM542` is implicitly declared local to procedure
`Loop`
Warning, `COMM552` is implicitly declared local to procedure
`Loop`
Warning, `COMM562` is implicitly declared local to procedure
`Loop`
Warning, `LOOP1` is implicitly declared local to procedure
`Loop`
Warning, `LOOP2` is implicitly declared local to procedure
`Loop`
Warning, `LOOP` is implicitly declared local to procedure `Loop`
```

Define the "zero-curvature" relation (red R in the table in the text)

```
> myR := convert(diff(Qx(x0, y0), y0) - diff(Qy(x0, y0), x0) - (Qy(x0, y0)·Qx(x0, y0)
      - Qx(x0, y0)·Qy(x0, y0)), D);
```

$$myR := \mathrm{D}_2(Qx)(x0, y0) - \mathrm{D}_1(Qy)(x0, y0) - Qy(x0, y0)\ Qx(x0, y0) + Qx(x0, y0)\ Qy(x0, y0) \quad (1)$$

This is the substitution to rewrite in terms of R

```
> D2Qxsub := ((D[1])(Qy))(x0, y0) + Qy(x0, y0) Qx(x0, y0) - Qx(x0, y0) Qy(x0, y0) + R(x0,
      y0);
```

$$D2Qxsub := \mathrm{D}_1(Qy)(x0, y0) + Qy(x0, y0)\ Qx(x0, y0) - Qx(x0, y0)\ Qy(x0, y0) + R(x0, y0) \quad (2)$$

Order 2 terms (h*k)

```
> Loop(1, 1);
```

$$\mathrm{D}_2(Qx)(x0, y0) + \frac{[Qx(x0, y0), Qy(x0, y0)]_-}{2} - \mathrm{D}_1(Qy)(x0, y0) - \frac{[Qy(x0, y0), Qx(x0, y0)]_-}{2} \quad (3)$$

```
> subs(D[2](Qx)(x0, y0) = D2Qxsub, Loop(1, 1)) : Simplify(%);
```

$$R(x0, y0) \quad (4)$$

```
> Loop(1, 1) - myR : simplify(%);
```

$$0 \tag{5}$$

Order 3 terms (h^2*k)

```
> Loop(2, 1);
```

$$\frac{\mathrm{D}_{1,2}(Qx)(x0,y0)}{2} - \frac{[Qx(x0,y0), \mathrm{D}_2(Qx)(x0,y0)]_-}{4} - \frac{[\mathrm{D}_2(Qx)(x0,y0), Qx(x0,y0)]_-}{4} + \frac{[\mathrm{D}_1(Qx)(x0,y0), Qy(x0,y0)]_-}{4} - \frac{\mathrm{D}_{1,1}(Qy)(x0,y0)}{2} - \frac{[\mathrm{D}_1(Qy)(x0,y0), Qx(x0,y0)]_-}{2} - \frac{[Qy(x0,y0), \mathrm{D}_1(Qx)(x0,y0)]_-}{4} \tag{6}$$

```
> subs((D[1, 2](Qx))(x0, y0) = convert(diff(D2Qxsub, x0), D), Loop(2, 1)) : Simplify(%);
```

$$\frac{\mathrm{D}_1(R)(x0,y0)}{2} \tag{7}$$

Order 4 terms (h^3*k,h^2*k^2)

```
> Loop(3, 1) - 1/6 · convert(diff(myR, x0, x0), D) - 1/12 · Commutator(convert(diff(myR, x0),
      D), Qx(x0, y0)) + 1/12 · Commutator(myR, convert(diff(Qx(x0, y0), x0), D)) :
  Simplify(%);
```

$$0 \tag{8}$$

```
> Loop(2, 2) - 1/4 · convert(diff(myR, x0, y0), D) - 1/6 · Commutator(myR, Commutator(Qx(x0,
      y0), Qy(x0, y0))) - 1/12 · Commutator(Qy(x0, y0), Commutator(myR, Qx(x0, y0))) + 1/4
  · Commutator(myR, D[1](Qy)(x0, y0)) : Simplify(%);
```

$$0 \tag{9}$$

Order 5 terms (h^4*k,h^3*k^2)

```
> Loop(4, 1) - (1/24) * convert(diff(myR, `$`(x0, 3)), D) - (1/24)
    * Commutator(convert(diff(myR, `$`(x0, 2)), D), Qx(x0, y0)) + (1/24)
    * Commutator(myR, convert(diff(Qx(x0, y0), `$`(x0, 2)), D)) : Simplify(%);
```

$$0 \tag{10}$$

```
> Loop(3, 2) - (1/12) · convert(diff(myR, x0$2, y0), D) + (1/24) · Commutator(Qx(x0, y0),
    convert(diff(myR, x0, y0), D)) - (1/24) · Commutator(convert(diff(Qx(x0, y0), x0), D),
    convert(diff(myR, y0), D)) + (1/24) · Commutator(Qx(x0, y0), Commutator(Qy(x0, y0),
    convert(diff(myR, x0), D))) + (1/12) · Commutator(myR, convert(diff(myR, x0), D))
  - (1/12) · Commutator(convert(diff(Qy(x0, y0), x0), D), convert(diff(myR, x0), D))
  - (1/6) · Commutator(convert(diff(Qy(x0, y0), x0$2), D), myR) + (1/24)
  · Commutator(Qy(x0, y0), Commutator(convert(diff(Qx(x0, y0), x0), D), myR)) + (1/24)
```

$$\cdot Commutator(convert(diff(Qy(x0, y0), x0), \mathrm{D}), Commutator(Qx(x0, y0), myR)) - \left(\frac{1}{8}\right)$$
$$\cdot Commutator(Commutator(convert(diff(Qy(x0, y0), x0), \mathrm{D}), Qx(x0, y0)), myR) - \left(\frac{1}{8}\right)$$
$$\cdot Commutator(Commutator(Qy(x0, y0), convert(diff(Qx(x0, y0), x0), \mathrm{D})), myR) :$$
$$simplify(\%);$$
$$0 \qquad \textbf{(11)}$$

Example with $SL(2)$ - part 1

The following MATLAB code relates to Example 1 in Section 4.3.2 and can be found at the following link: https://zenodo.org/record/3661107/files/new_ex_both.m?download=1. It requires the *Diffman* package,[17],available at http://www.diffman.no. We think that the most efficient way to help the reader in understanding the following code is through commenting it. We tried to add some pseudocode, but as there are no routines and most of the commands are quite intuitive, it would have mimicked very closely the original code. Therefore, the code has been thoroughly commented, in order not to leave any doubts. The other Matlab codes presented in this section are analogous and the reader can refer to the comments on the following code in case there were any doubts. The two main changes are the Lie group object that are initialised and the way the solutions and errors are plotted.

```
clear all;
close all;

%initial conditions for the moving frame and initialisation of the
 discrete grid and auxiliary variables
rho0x=[1/2,sqrt(3)/2; -sqrt(3)/2, 1/2];
double x0;
double x1;
double y0;
double y1;
double h;
x0=3; %here we specify the parameters for the grid
x1=4;
y0=3;
y1=4;
h=0.01; %h and hy are the step sizes in the x and y direction
 respectively
hy=0.01;
[X,Y]=meshgrid(x0:h:x1,y0:hy:y1);   %create the discrete grid
N=round((x1-x0)/h); %the grid is made by (N+1)*(M+1) points
M=round((y1-y0)/hy);
sol=zeros(M+1,N+1); %these are auxiliary variables used to store the
 solution and
sol1=zeros(M+1,N+1);%perform checks on it
solrho=zeros(2,2,N+1,N+1);
comp=zeros(N+1,2);
detrho=zeros(N+1,N+1);


%This part of the code takes an initial condition for the moving frame
%and solve the equation Y'=AY for Y on the line given by y=y0
dminit; %Initialise the Diffman toolbox
y=hmlie(lgsl(2));   %Create an object in SL(2)
setdata(y,rho0x);   %Set the initial condition to rho0x
vf=vectorfield(y);  %Initialise a vector field object
seteqntype(vf,'L'); %Set the differential equation type to linear
fid1 = fopen( 'vfsl2x.m', 'wt' );   %Define the vector field
fprintf( fid1, 'function [ la ] = vfsl2x( x,y ); la=liealgebra(y);
  dat=[0, -1; (4*x.^3+3.*%f-4)./(6*x.^5), 0]; setdata(la,dat);
 end',y0);
fclose(fid1);
setfm2g(vf,'vfsl2x');   %Set the vector field defined in the step
 above as the A in Y'=AY
ts=tsmagnus;    %Specify the timestepper we want to use is the Magnus
 Expansion one
setmethod(ts,'M6a');    %Specify the order of the Magnus expansion to
 6 (highest possible here)
f=flow; %Initialise a flow object
setcoordinate(ts,'exp');    %Specify we want to use the exponential
 map to pass from the Lie algebra to the Lie group
setvectorfield(f,vf);   %Construct the equation Y'=AY
```

```
settimestepper(f,ts);   %Specify we want to solve the equation
 constructed in the step above with a Magnus expansion of order 6
curve=f(y,x0,x1,h); %Solve the equation with the parameters specified
 above in the interval [x0,x1]
dati4=getdata(curve.y); %Obtain the solution as a vector

%Once we have obtained the solution, which is the moving frame, we can
%compute the minimisers as indicated at the end of section 4.3.1
for i=1:N+1
sol(1,i)=squeeze(-dati4(1,2,i)./dati4(1,1,i));
detrho(1,i)=det(dati4(:,:,i)); % a check on the solution belonging to
 SU(2) at all points
end

%As we solved for rho on y=y0, we use that solution as an initial
 condition
%to solve for rho on the lines given by x=x_i, where x_i are all the
 points
%included in the grid between x0 and x1 included
u=hmlie(lgsl(2)); %create an element of SL(2)
for j=1:N+1
setdata(u,dati4(:,:,j)); %set the initial data to rho(x_i,y0) found in
 the previous step
vf=vectorfield(u);  %initialise a vector field object
seteqntype(vf,'L'); %set the equation type to linear
fid = fopen( 'vfsl2y.m', 'wt' );    %define the vector field as an
 element of the Lie algebra
fprintf( fid, 'function [ la ] = vfsl2y( t,u ); la=liealgebra(u);
 dat=[1./(2*(3.*t-4)), (%f+ (%f-1) *%f)./(3*t-4); -(4*(%f+ (%f-1) *
%f).^3+3*t-4)./(6*(3.*t-4).*(%f+ (%f-1) *%f).^4), -1./(2*(3*t-4))];
 setdata(la,dat); return',x0, j,h, x0, j,h, x0, j,h);
fclose(fid);
setfm2g(vf,'vfsl2y');   %associate the defined vector field to the
 initialised vector field object
ts=tsmagnus;    %specify we want to use a Magnus expansion time
 stepper
setmethod(ts,'M6a'); %specify we want to use a sixth-order Magnus
 expansion
f=flow; %initialise a flow object
setcoordinate(ts,'exp'); %specify we want to use the exp map to go
 from the Lie algebra to the Lie group
setvectorfield(f,vf); %construct the equation Y'=AY
settimestepper(f,ts); %specify we want to solve the equation with the
 chosen time stepper
curve=f(u,y0,y1,hy);    %solve the equation in the interval [y0,y1] on
 the line x=x_i
dati3=getdata(curve.y); %obtain the solution as a vector


%Once we have the moving frame we can compute the minisers as
 specified at
%the end of Section 4.3.1 as follows
for k=2:M+1
sol(k,j)=squeeze(-dati3(1,2,k)./dati3(1,1,k));
```

```
detrho(k,j)=det(dati3(:,:,k)); %a check on the determinant
end
end


%As we computed the moving frame, and the minisers, solving the
%differential equation first on y=y0 and then on x=x_i, now we first
 solve
%the Lie group differential equation on the line x=x0 and then we use
 that
%solution to solve the DE for the moving frame on y=y_i, where y_i are
 all
%the points in the grid between y0 and y1 included.

rho0y=[1/2,sqrt(3)/2; -sqrt(3)/2, 1/2]; %specify an initial condition
 for rho(x0,y0)
yy=hmlie(lgsl(2)); %initialise an object in SL(2)
setdata(yy,rho0y); %set the initial data to rho0y
vf=vectorfield(yy); %initialise a vector field object
seteqntype(vf,'L'); %set the differential equation type to linear
fid1 = fopen( 'vfsl2y1.m', 'wt' ); %define the vector field
fprintf( fid1, 'function [ la ] = vfsl2y1( t,yy ); la=liealgebra(yy);
 dat=[1./(2*(3.*t-4)), %f./(3*t-4); -(4*%f.^3+3*t-4)./(6*(3*t-4).*
%f.^4), -1./(2*(3*t-4))]; setdata(la,dat); return',x0,x0,x0);
fclose(fid1);
setfm2g(vf,'vfsl2y1'); %assigned the defined vector field to the
 initialised vector field object
ts=tsmagnus; %select a Magnus expansion time stepper
setmethod(ts,'M6a'); %choose a sixth-order Magnus expansion
f=flow; %initialise a flow object
setcoordinate(ts,'exp'); %specify we want to use exp map to pass from
 the Lie algebra to the Lie group
setvectorfield(f,vf); % construct the differential equation
settimestepper(f,ts); % specify we want to solve the differential
 equation with the chosen time stepper
curve=f(yy,y0,y1,hy);%solve the differential equation
dati=getdata(curve.y);%obtain the solution as a vector
for i=1:M+1 % obtain the minimisers as specified at the end of Section
 4.3.1
sol1(i,1)=squeeze(-dati(1,2,i)./dati(1,1,i));
detrho1(i,1)=det(dati(:,:,i)); %check the determinant of the moving
 frame is 1 at every point
end

%As we solved for the line x=x0, we use that solution to solve the
%Lie group differential equation on the lines y=y_i, where y_i are all
 the
%points in the grid between y0 and y1 included.
u1=hmlie(lgsl(2));  %initialise an element of SL(2)
for j=1:M+1
setdata(u1,dati(:,:,j));    %set the initial condition of the SL(2)
 element to the respective element of the solution found in the
 previous step
vf=vectorfield(u1); %initialise a vector field object
```

```
seteqntype(vf,'L'); %Set the equation type to linear
fid = fopen( 'vfsl2x1.m', 'wt' );   %define the vector field A in
 Y'=AY
fprintf( fid, 'function [ la ] = vfsl2x1( x,u1 ); la=liealgebra(u1);
  dat=[0, -1; (4*x.^3+3*(%f+(%f-1)*%f)-4)./(6*x.^5), 0];
 setdata(la,dat); end',y0, j,h);
fclose(fid);
setfm2g(vf,'vfsl2x1');
ts=tsmagnus; %choose a Magnus expansion time stepper
setmethod(ts,'M6a');   %specify we want a sixth-order Magnus expansion
f=flow; %initialise a flow object
setcoordinate(ts,'exp'); %use exponential map to go to the Lie group
 from the Lie algebra
setvectorfield(f,vf); %build the equation Y'=AY
settimestepper(f,ts); %choose to solve the equation with the specified
 time stepper
curve=f(u1,x0,x1,h); %solve the equation for the moving frame Y
dati1=getdata(curve.y); %store the solution into a vector
for k=2:N+1 %compute the minimisers as shown at the end of Section
 4.3.1
sol1(j,k)=squeeze(-dati1(1,2,k)./dati1(1,1,k));
detrho1(j,k)=det(dati1(:,:,k));
end
end

%After all the computations have been performer, now it is time to
 plot the
%relevant figures
% figure();
% surf(X,Y,sol); %plot rho^{\gamma_1}
% hold on;
% surf(X,Y,sol1); %plot rho^{\gamma_2} on the same figure as the
 previous plot
% colormap summer;
% xlabel('x') % x-axis label
% ylabel('y') % y-axis label
% zlabel('u') % z-axis label
% hold off;
% figure();
% surf(X,Y,abs(sol-sol1)); %plot the absolute value of the pointwise
 difference of the two solutions


  DiffMan Version 2.01 is initialized - 2012.05.01

  Please report any problems and/or bugs to:
       help@diffman.no

  For more information and how to get started, try:
      >> dmtutorial
      >> dmhelp
      >> demo
```

Published with MATLAB® R2019a

Example with $SL(2)$ - part 2

The following MATLAB code relates to Example 2 in Section 4.3.2 and can be found at the following link: https://zenodo.org/record/3661107/files/sigma_x_new_v2.m?download=1. It requires the *Diffman* package,[17], available at http://www.diffman.no.

```
clear all;
close all;

%initial condition and initialisation of grid and variables
rho0x=[1/2,sqrt(3)/2; -sqrt(3)/2, 1/2];
double x0;
double x1;
double y0;
double y1;
double h;
x0=3;
x1=4;
y0=3;
y1=4;
h=0.01;
hy=0.01;
[X,Y]=meshgrid(x0:h:x1,y0:hy:y1);
N=round((x1-x0)/h);
M=round((y1-y0)/hy);
sol=zeros(M+1,N+1);
sol1=zeros(M+1,N+1);
solrho=zeros(2,2,N+1,N+1);
comp=zeros(N+1,2);
detrho=zeros(N+1,N+1);


%solve for the line y=y0 given an initial data rho(xo,yo)
dminit;
y=hmlie(lgsl(2));
setdata(y,rho0x);
vf=vectorfield(y);
seteqntype(vf,'L');
fid1 = fopen( 'vfsl2x.m', 'wt' );
fprintf( fid1, 'function [ la ] = vfsl2x( x,y ); la=liealgebra(y);
  dat=[0, -1; (4*x.^3+3.*%f-4)./(6*x.^5), 0]; setdata(la,dat);
 end',y0);
fclose(fid1);
setfm2g(vf,'vfsl2x');
ts=tsmagnus;
setmethod(ts,'M6a');
f=flow;
setcoordinate(ts,'exp')
setvectorfield(f,vf);
settimestepper(f,ts);
curve=f(y,x0,x1,h);
dati4=getdata(curve.y);
for i=1:N+1
sol(1,i)=squeeze(-dati4(1,2,i)./dati4(1,1,i));
detrho(1,i)=det(dati4(:,:,i));
end
```

```
%solve for rho(xi,y) given as initial data rho(xi,y0) computed in the
%previous step
u=hmlie(lgsl(2));
for j=1:N+1
setdata(u,dati4(:,:,j));
vf=vectorfield(u);
seteqntype(vf,'L');
fid = fopen( 'vfsl2y.m', 'wt' );
fprintf( fid, 'function [ la ] = vfsl2y( t,u ); la=liealgebra(u);
 dat=[1./(2*(3.*t-4)), (%f+ (%f-1) *%f)./(3*t-4); -(4*(%f+ (%f-1) *
%f).^3+3*t-4)./(6*(3.*t-4).*(%f+ (%f-1) *%f).^4), -1./(2*(3*t-4))];
 setdata(la,dat); return',x0, j,h, x0, j,h, x0, j,h);
fclose(fid);
setfm2g(vf,'vfsl2y');
ts=tsmagnus;
setmethod(ts,'M6a');
f=flow;
setcoordinate(ts,'exp')
setvectorfield(f,vf);
settimestepper(f,ts);
curve=f(u,y0,y1,hy);
dati3=getdata(curve.y);
for k=2:M+1
sol(k,j)=squeeze(-dati3(1,2,k)./dati3(1,1,k));
detrho(k,j)=det(dati3(:,:,k));
end
end


rho0y=[1/2,sqrt(3)/2; -sqrt(3)/2, 1/2];


yy=hmlie(lgsl(2));
%solve for the line y=y0 given an initial data rho(xo,yo)
setdata(yy,rho0y);
vf=vectorfield(yy);
seteqntype(vf,'L');
fid1 = fopen( 'vfsl2y1.m', 'wt' );
fprintf( fid1, 'function [ la ] = vfsl2y1( t,yy ); la=liealgebra(yy);
 dat=[1./(2*(3.*t-4)), %f./(3*t-4); -(4*%f.^3+3*t-4)./(6*(3*t-4).*
%f.^4), -1./(2*(3*t-4))]; setdata(la,dat); return',x0,x0,x0);
fclose(fid1);
setfm2g(vf,'vfsl2y1');
ts=tsmagnus;
setmethod(ts,'M6a');
f=flow;
setcoordinate(ts,'exp')
setvectorfield(f,vf);
settimestepper(f,ts);
curve=f(yy,y0,y1,hy);
dati=getdata(curve.y);
for i=1:M+1
sol1(i,1)=squeeze(-dati(1,2,i)./dati(1,1,i));
detrho1(i,1)=det(dati(:,:,i));
```

```
end


%solve for rho(xi,y) given as initial data rho(xi,y0) computed in the
%previous step
u1=hmlie(lgsl(2));
for j=1:M+1
setdata(u1,dati(:,:,j));
vf=vectorfield(u1);
seteqntype(vf,'L');
fid = fopen( 'vfsl2x1.m', 'wt' );
fprintf( fid, 'function [ la ] = vfsl2x1( x,u1 ); la=liealgebra(u1);
  dat=[0, -1; (4*x.^3+3*(%f+(%f-1)*%f)-4)./(6*x.^5), 0];
 setdata(la,dat); end',y0, j,h);
fclose(fid);
setfm2g(vf,'vfsl2x1');
ts=tsmagnus;
setmethod(ts,'M6a');
f=flow;
setcoordinate(ts,'exp')
setvectorfield(f,vf);
settimestepper(f,ts);
curve=f(u1,x0,x1,h);
dati1=getdata(curve.y);
for k=2:N+1
sol1(j,k)=squeeze(-dati1(1,2,k)./dati1(1,1,k));
detrho1(j,k)=det(dati1(:,:,k));
end
end


% diff=abs(sol-sol1);
% figure;
% surf(X,Y,sol);
% hold on;
% surf(X,Y,sol1);
% hold off;
% figure;
% surf(X,Y,diff);
% xlabel('x') % x-axis label
% ylabel('y') % y-axis label
% zlabel('|u1-u2|') % z-axis label


  DiffMan Version 2.01 is initialized - 2012.05.01

  Please report any problems and/or bugs to:
       help@diffman.no

  For more information and how to get started, try:
      >> dmtutorial
      >> dmhelp
      >> demo
```

Published with MATLAB® R2019a

Example with $SU(2)$

The following MATLAB code relates to Section 4.3.1 and can be found at the following link: https://zenodo.org/record/3661107/files/su_2_2nd_ex_su2.m?download=1. It requires the *Diffman* package,[17], available at http://www.diffman.no.

```
clear all;
close all;

%initial condition and initialisation of grid and variables
rho0x=[-1/3+1i*1/4,1/2-1i*sqrt(83)/12;-1/2-1i*sqrt(83)/12,-1/3-1i*1/4];
x0=0;
x1=1;
y0=0;
y1=1;
h=0.01;
hy=0.01;
[X,Y]=meshgrid(x0:h:x1,y0:hy:y1);
N=floor((x1-x0)/h);
M=floor((y1-y0)/hy);
solu=zeros(M+1,N+1);
solv=zeros(M+1,N+1);
solu1=zeros(M+1,N+1);
solv1=zeros(M+1,N+1);
solrho=zeros(2,2,M+1,N+1);
solrho1=zeros(2,2,M+1,N+1);
comp=zeros(N+1,2);
detrho=zeros(M+1,N+1);



%solve for rho on the line y=y0 given an initial data rho(x0,y0)
dminit;
y=hmlie(lgsu(2));
setdata(y,rho0x);
vf=vectorfield(y);
seteqntype(vf,'L');
fid1 = fopen( 'vfsu2x.m', 'w' );
fprintf( fid1, 'function [ la ] = vfsu2x( x,y ); la=liealgebra(y);
 dat=[-1i*%f.^3,0;0, 1i*%f.^3]; setdata(la,dat); end',y0,y0);
fclose(fid1);
setfm2g(vf,'vfsu2x');
ts=tsmagnus;
setmethod(ts,'M6a');
f=flow;
setcoordinate(ts,'exp')
setvectorfield(f,vf);
settimestepper(f,ts);
curve=f(y,x0,x1,h);
dati=getdata(curve.y);
for j=1:N+1
    solrho(:,:,1,j)=dati(:,:,j);
    solu(1,j)=real(dati(1,1,j))-1i*imag(dati(1,1,j));
    solv(1,j)=real(dati(1,2,j))-1i*imag(dati(1,2,j));
    detrho(1,j)=det(dati(:,:,j));
end
```

```
%solve for rho(x,y) given as initial data rho(x,y0) computed in the
%previous step
u=hmlie(lgsu(2));
for j=1:N+1
setdata(u,dati(:,:,j));
vf=vectorfield(u);
seteqntype(vf,'L');
string1=['function [ la ] = vfsu2y( t,u ); la=liealgebra(u);' ...
   'qy11=-1i.*(3.*(%f+(%f-1)*%f).*t.^2 - t.^2);'...
   'qy12=-(t + 5).*sin(2.*t.^3.*(%f+(%f-1)*%f)) -
 t.^4.*cos(2.*t.^3.*(%f+(%f-1)*%f))./3 - (-t.^4.*sin(2.*t.^3.*(%f
+(%f-1)*%f))./3 + (t + 5).*cos(2.*t.^3.*(%f+(%f-1)*%f))).*1i;'...
   'qy21=(t + 5).*sin(2.*t.^3.*(%f+(%f-1)*%f)) +
 t.^4.*cos(2.*t.^3.*(%f+(%f-1)*%f))./3 - (-t.^4.*sin(2.*t.^3.*(%f
+(%f-1)*%f))./3 + (t + 5).*cos(2.*t.^3.*(%f+(%f-1)*%f))).*1i;'...
   'qy22=(3.*(%f+(%f-1)*%f).*t.^2 - t.^2).*1i;'...
   'dat=[qy11,qy12; qy21,qy22]; setdata(la,dat); end'];
fid = fopen( 'vfsu2y.m', 'w' );
fprintf( fid,
 string1,x0,j,h,x0,j,h,x0,j,h,x0,j,h,x0,j,h,x0,j,h,x0,j,h,x0,j,h,x0,j,h);
fclose(fid);
setfm2g(vf,'vfsu2y');
ts=tsmagnus;
setmethod(ts,'M6a');
f=flow;
setcoordinate(ts,'exp')
setvectorfield(f,vf);
settimestepper(f,ts);
curve=f(u,y0,y1,hy);
dati1=getdata(curve.y);
for k=2:M+1
    solrho(:,:,k,j)=dati1(:,:,k);
    solu(k,j)=real(dati1(1,1,k))-1i*imag(dati1(1,1,k));
    solv(k,j)=real(dati1(1,2,k))-1i*imag(dati1(1,2,k));
    detrho(k,j)=det(dati1(:,:,k));
end
end

rho0y=rho0x;

%solve for the line x=x0 given an initial data rho(x0,y0)
yy=hmlie(lgsu(2));
setdata(yy,rho0y);
vf=vectorfield(yy);
seteqntype(vf,'L');
string2=['function [ la ] = vfsu2y1 ( t,yy ); la=liealgebra(yy);' ...
   'qy11=-1i.*(3.*(%f).*t.^2 - t.^2);'...
   'qy12=-(t + 5).*sin(2.*t.^3.*(%f)) - t.^4.*cos(2.*t.^3.*(%f))./3 -
 (-t.^4.*sin(2.*t.^3.*(%f))./3 + (t + 5).*cos(2.*t.^3.*(%f))).*1i;'...
   'qy21=(t + 5).*sin(2.*t.^3.*(%f)) + t.^4.*cos(2.*t.^3.*(%f))./3 -
 (-t.^4.*sin(2.*t.^3.*(%f))./3 + (t + 5).*cos(2.*t.^3.*(%f))).*1i;'...
   'qy22=(3.*(%f).*t.^2 - t.^2).*1i;'...
   'dat=[qy11,qy12; qy21,qy22]; setdata(la,dat); end'];
```

```
fid2 = fopen( 'vfsu2y1.m', 'w' );
fprintf( fid2, string2,x0,x0,x0,x0,x0,x0,x0,x0,x0,x0);
fclose(fid2);
setfm2g(vf,'vfsu2y1');
ts=tsmagnus;
setmethod(ts,'M6a');
f=flow;
setcoordinate(ts,'exp')
setvectorfield(f,vf);
settimestepper(f,ts);
curve=f(yy,y0,y1,hy);
dati=getdata(curve.y);
for j=1:M+1
    solrho1(:,:,j,1)=dati(:,:,j);
    solu1(j,1)=real(dati(1,1,j))-1i*imag(dati(1,1,j));
    solv1(j,1)=real(dati(1,2,j))-1i*imag(dati(1,2,j));
    detrho1(j,1)=det(dati(:,:,j));
end


%solve for rho(x,y) given as initial data rho(x0,y) computed in the
%previous step
u1=hmlie(lgsu(2));
for j=1:M+1
setdata(u1,dati(:,:,j));
vf=vectorfield(u1);
seteqntype(vf,'L');
fid1 = fopen( 'vfsu2x1.m', 'w' );
fprintf( fid1, 'function [ la ] = vfsu2x1( x,y ); la=liealgebra(y);
  dat=[-1i*(%f+(%f-1)*%f).^3,0;0, 1i*(%f+(%f-1)*%f).^3];
 setdata(la,dat); end',y0,j,h,y0,j,h);
fclose(fid1);
setfm2g(vf,'vfsu2x1');
ts=tsmagnus;
setmethod(ts,'M6a');
f=flow;
setcoordinate(ts,'exp')
setvectorfield(f,vf);
settimestepper(f,ts);
curve=f(u1,x0,x1,h);
dati1=getdata(curve.y);
for k=2:N+1
    solrho1(:,:,j,k)=dati1(:,:,k);
    solu1(j,k)=real(dati1(1,1,k))-1i*imag(dati1(1,1,k));
    solv1(j,k)=real(dati1(1,2,k))-1i*imag(dati1(1,2,k));
    detrho1(j,k)=det(dati1(:,:,k));
end
end


for j=1:M+1
    for l=1:N+1
        diffrho(:,:,j,l)=solrho(:,:,j,l)-solrho1(:,:,j,l);
        normrho(j,l)=norm(solrho(:,:,j,l));
```

```
        normrho1(j,l)=norm(solrho1(:,:,j,l));
        normdiffrho(j,l)=norm(diffrho(:,:,j,l));
    end
end
% figure();
% surf(X,Y,normdiffrho);
% xlabel('x') % x-axis label
% ylabel('t') % y-axis label
% zlabel('rho^{gamma1}-rho^{gamma2}') % z-axis label
% figure();
% surf(X,Y,imag(solu));
% xlabel('x') % x-axis label
% ylabel('t') % y-axis label
% zlabel('Im(solu)') % z-axis label
```

```
  DiffMan Version 2.01 is initialized - 2012.05.01

  Please report any problems and/or bugs to:
       help@diffman.no

  For more information and how to get started, try:
      >> dmtutorial
      >> dmhelp
      >> demo
```

Published with MATLAB® R2019a

Example with $SE(2)$ - Maple part

The following Maple code is an example of how to use *Indiff*, [41], in order to compute the curvature matrices in the case of the standard action of $SE(2)$ on a pair of evolving curves. The code has been fully commented so that the reader could also work out the details for other examples if needed. The code can be found at the following link: https://zenodo.org/record/3661107/files/curvmatrixse2_surfaces.mw?download=1.

```
> restart: read "D:/indiff-src-2";read
  "C:/Users/hausl/Dropbox/uni/Indiff/indiff-src-2"; #initialising
  the Indiff package
```

Error, unable to read `D:/indiff-src-2`
Error, (in with) package Groebner does not export normalf
Error, (in with) package Groebner does not export gsolve
Error, (in with) package Groebner does not export inter_reduce

[*jacobian*]

[*transpose*]

[*inverse*]

Error, (in with) package Groebner does not export gbasis
Error, (in with) package Groebner does not export termorder

[*Ore_to_DESol*, *Ore_to_RESol*, *Ore_to_diff*, *Ore_to_shift*, *annihilators*, *applyopr*, *diff_algebra*, *dual_algebra*, *dual_polynomial*, *poly_algebra*, *qshift_algebra*, *rand_skew_poly*, *reverse_algebra*, *reverse_polynomial*, *shift_algebra*, *skew_algebra*, *skew_elim*, *skew_gcdex*, *skew_pdiv*, *skew_power*, *skew_prem*, *skew_product*] **(1)**

#Specifying the indepent variables (tau is a dummy variable used for the application to the Calculus of Variations)

```
> vars:=[s,t,tau];
```

$$vars := [s, t, \tau] \quad \textbf{(2)}$$

#specifying the dependent variables

```
> ukns:=[x,u];
```

$$ukns := [x, u] \quad \textbf{(3)}$$

#group parameters. In this case we are considering SE(2), so three parameters

```
> GroupP:=[theta,a,b];
```

$$GroupP := [\theta, a, b] \quad \textbf{(4)}$$

#The matrix of infinitesimals of the action

```
> XiPhis:=Matrix([[0,0,0,-u(s,t,tau),x(s,t,tau)],[0,0,0,1,0],[0,0,
  0,0,1]]);
```

$$XiPhis := \begin{bmatrix} 0 & 0 & 0 & -u(s,t,\tau) & x(s,t,\tau) \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(5)}$$

#The normalisation equations used to define the moving frame. In this case they read, from left to right, as x=0, u=0, u_s=0.

```
> Neqs:=[In[x,[]],In[u,[]],In[u,[1]]];
```

$$Neqs := [In_{x,[\,]}, In_{u,[\,]}, In_{u,[1]}] \quad \textbf{(6)}$$

#choose the order for the system reduction

```
> HNI([[1,2,3],[x,u]],ttdeg);
```

$$\{In_{u,[\,]}, In_{u,[1]}, In_{x,[\,]}\} \quad \textbf{(7)}$$

This command returns the error matrix K defined in Section 2.2.3, from which we can construct the curvature matrices. It contains the generating differential invariants

```
> K:=Kmat()
```

$$K := \begin{bmatrix} \frac{In_{u,[1,1]}}{In_{x,[1]}} & In_{x,[1]} & 0 \\ \frac{In_{u,[1,2]}}{In_{x,[1]}} & In_{x,[2]} & In_{u,[2]} \\ \frac{In_{u,[1,3]}}{In_{x,[1]}} & In_{x,[3]} & In_{u,[3]} \end{bmatrix} \quad (8)$$

#Next three commands define a basis for the Lie algebra se(2)

```
> gtheta:=Matrix([[0,-1,0],[1,0,0],[0,0,0]]);
```

$$gtheta := \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (9)$$

```
> ga:=Matrix([[0,0,1],[0,0,0],[0,0,0]]);
```

$$ga := \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (10)$$

```
> gb:=Matrix([[0,0,0],[0,0,1],[0,0,0]]);
```

$$gb := \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad (11)$$

#Following 3 commands construct the curvature matrices given a basis of the Lie algebra and the matrix K, as shown in Theorem 2.2.34.

```
> Qs:=-K[1,1]*gtheta-K[1,2]*ga-K[1,3]*gb;
```

$$Qs := \begin{bmatrix} 0 & \frac{In_{u,[1,1]}}{In_{x,[1]}} & -In_{x,[1]} \\ -\frac{In_{u,[1,1]}}{In_{x,[1]}} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (12)$$

```
> Qt:=-K[2,1]*gtheta-K[2,2]*ga-K[2,3]*gb;
```

$$Qt := \begin{bmatrix} 0 & \frac{In_{u,[1,2]}}{In_{x,[1]}} & -In_{x,[2]} \\ -\frac{In_{u,[1,2]}}{In_{x,[1]}} & 0 & -In_{u,[2]} \\ 0 & 0 & 0 \end{bmatrix} \quad (13)$$

```
> Qtau:=-K[3,1]*gtheta-K[3,2]*ga-K[3,3]*gb;
```

$$Qtau := \begin{bmatrix} 0 & \frac{In_{u,\,[1,\,3]}}{In_{x,\,[1]}} & -In_{x,\,[3]} \\ -\frac{In_{u,\,[1,\,3]}}{In_{x,\,[1]}} & 0 & -In_{u,\,[3]} \\ 0 & 0 & 0 \end{bmatrix} \tag{14}$$

#The following 3 commands check that Theorem 2.2.35 is satisfied

```
> map(Idiff,Qs,2)-map(Idiff,Qt,1)-Qt.Qs+Qs.Qt: map(simplify,%);
```

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{15}$$

```
> map(Idiff,Qtau,2)-map(Idiff,Qt,3)-Qt.Qtau+Qtau.Qt: map(simplify,
  %);
```

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{16}$$

```
> map(Idiff,Qs,3)-map(Idiff,Qtau,1)-Qtau.Qs+Qs.Qtau: map(simplify,
  %);
```

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{17}$$

Use Theorem 2.2.35 to find the syzygies between the generating differential invariants

```
> Qt.Qs-Qs.Qt+map(Diff,Qt,s): map(simplify,%);
```

$$\begin{bmatrix} \frac{\mathrm{d}}{\mathrm{d}s}\,0 & \frac{\partial}{\partial s}\left(\frac{In_{u,\,[1,\,2]}}{In_{x,\,[1]}}\right) & \frac{In_{u,\,[1,\,1]}\,In_{u,\,[2]}}{In_{x,\,[1]}} + \frac{\partial}{\partial s}\left(-In_{x,\,[2]}\right) \\ \frac{\partial}{\partial s}\left(-\frac{In_{u,\,[1,\,2]}}{In_{x,\,[1]}}\right) & \frac{\mathrm{d}}{\mathrm{d}s}\,0 & In_{u,\,[1,\,2]} - \frac{In_{u,\,[1,\,1]}\,In_{x,\,[2]}}{In_{x,\,[1]}} + \frac{\partial}{\partial s}\left(-In_{u,\,[2]}\right) \\ \frac{\mathrm{d}}{\mathrm{d}s}\,0 & \frac{\mathrm{d}}{\mathrm{d}s}\,0 & \frac{\mathrm{d}}{\mathrm{d}s}\,0 \end{bmatrix} \tag{18}$$

```
> map(Diff,Qs,t)
```

$$\begin{bmatrix} \frac{\mathrm{d}}{\mathrm{d}t}\,0 & \frac{\partial}{\partial t}\left(\frac{In_{u,\,[1,\,1]}}{In_{x,\,[1]}}\right) & \frac{\partial}{\partial t}\left(-In_{x,\,[1]}\right) \\ \frac{\partial}{\partial t}\left(-\frac{In_{u,\,[1,\,1]}}{In_{x,\,[1]}}\right) & \frac{\mathrm{d}}{\mathrm{d}t}\,0 & \frac{\mathrm{d}}{\mathrm{d}t}\,0 \\ \frac{\mathrm{d}}{\mathrm{d}t}\,0 & \frac{\mathrm{d}}{\mathrm{d}t}\,0 & \frac{\mathrm{d}}{\mathrm{d}t}\,0 \end{bmatrix} \tag{19}$$

A couple of examples of how to simplify expressions in Qt.Qs-Qs.Qt+map(Diff,Qt,s) and rewrite them in terms of only the generating differential invariants

```
> Idiff(In[u, [2]],1)
```

$$\frac{-In_{u,\,[1,\,1]}\,In_{x,\,[2]}+In_{u,\,[1,\,2]}\,In_{x,\,[1]}}{In_{x,\,[1]}} \tag{20}$$

```
> Idiff(In[x, [1]],3)
```

$$In_{x,\,[1,\,3]} \tag{21}$$

Example with $SE(2)$ - Matlab part

The following MATLAB code relates to Section 4.3.3 and can be found at the following link: https://zenodo.org/record/3661107/files/se_2_2D_kappa_t_GL3.m?download=1. It requires the *Diffman* package,[17], available at http://www.diffman.no.

```
clear all;
close all;

%initial condition and initialisation of grid and variables
rho0x=[1/2, -sqrt(3)/2, -0.4; sqrt(3)/2, 1/2, 0.2; 0,0,1];
x0=1;
x1=2;
y0=1;
y1=2;
h=0.01;
hy=0.01;
[X,Y]=meshgrid(x0:h:x1,y0:hy:y1);
N=floor((x1-x0)/h);
M=floor((y1-y0)/hy);
rho=zeros(3,3,M+1,N+1);
rho1=zeros(3,3,M+1,N+1);
detrho=zeros(N+1,N+1);


%solve for rho on the line y=y0 given an initial data rho(x0,y0)
dminit;
y=hmlie(lggl(3));
setdata(y,rho0x);
vf=vectorfield(y);
seteqntype(vf,'L');
fid1 = fopen( 'vfse3x.m', 'w' );
fprintf( fid1, 'function [ la ] = vfse3x( x,y ); la=liealgebra(y);
  dat=[0,-4./(x + %f),-1;4./(x + %f),0,0;0,0,0]; setdata(la,dat);
 end',y0,y0);
fclose(fid1);
setfm2g(vf,'vfse3x');
ts=tsmagnus;
setmethod(ts,'M6a');
f=flow;
setcoordinate(ts,'exp')
setvectorfield(f,vf);
settimestepper(f,ts);
curve=f(y,x0,x1,h);
dati=getdata(curve.y);
for i=1:N+1
    rho(:,:,1,i)=dati(:,:,i);
end


%solve for rho(x,y) given as initial data rho(x,y0) computed in the
%previous step
u=hmlie(lggl(3));
for j=1:N+1
setdata(u,rho(:,:,1,j));
vf=vectorfield(u);
seteqntype(vf,'L');
string1=['function [ la ] = vfse3y( t,u ); la=liealgebra(u);' ...
```

```
   'qy11=0;'...
   'qy12=(17*(%f+(%f-1).*%f) + 17*t - 4)./((%f+(%f-1).*%f) + t);'...
   'qy13=-cos(4*log((%f+(%f-1).*%f) + t)) + sin(4*log((%f+(%f-1).*%f)
 + t)) - 1 + 4*(%f+(%f-1).*%f) + 4*t;'...
   'qy21=-(17*(%f+(%f-1).*%f) + 17*t - 4)./((%f+(%f-1).*%f) + t);'...
   'qy22=0;'...
   'qy23=-(%f+(%f-1).*%f) - t - sin(4*log((%f+(%f-1).*%f) + t)) -
 cos(4*log((%f+(%f-1).*%f) + t));'...
   'qy31=0;'...
   'qy32=0;'...
   'qy33=0;'...
   'dat=[qy11,qy12,qy13;qy21,qy22,qy23;qy31,qy32,qy33];
 setdata(la,dat); end'];
fid = fopen('vfse3y.m','w');
fprintf(fid,string1,x0,j,h,x0,j,h,x0,j,h,x0,j,h,x0,j,h,x0,j,h,x0,j,h,x0,j,h,x0,j,h
fclose(fid);
setfm2g(vf,'vfse3y');
ts=tsmagnus;
setmethod(ts,'M6a');
f=flow;
setcoordinate(ts,'exp')
setvectorfield(f,vf);
settimestepper(f,ts);
curve=f(u,y0,y1,hy);
dati=getdata(curve.y);
for k=2:M+1
    rho(:,:,k,j)=dati(:,:,k);
end
end


%solve for the line x=x0 given an initial data rho(x0,y0)
yy=hmlie(lggl(3));
setdata(yy,rho0x);
vf=vectorfield(yy);
seteqntype(vf,'L');
string2=['function [ la ] = vfse3y1 ( t,yy ); la=liealgebra(yy);' ...
   'qy11=0;'...
   'qy12=(17*(%f) + 17*t - 4)./((%f) + t);'...
   'qy13=-cos(4*log((%f) + t)) + sin(4*log((%f) + t)) - 1 + 4*(%f) +
 4*t;'...
   'qy21=-(17*(%f) + 17*t - 4)./((%f) + t);'...
   'qy22=0;'...
   'qy23=-(%f) - t - sin(4*log((%f) + t)) - cos(4*log((%f) + t));'...
   'qy31=0;'...
   'qy32=0;'...
   'qy33=0;'...
   'dat=[qy11,qy12,qy13;qy21,qy22,qy23;qy31,qy32,qy33];
 setdata(la,dat); end'];
fid2 = fopen( 'vfse3y1.m', 'w' );
fprintf( fid2, string2,x0,x0,x0,x0,x0,x0,x0,x0,x0,x0);
fclose(fid2);
setfm2g(vf,'vfse3y1');
ts=tsmagnus;
```

```
setmethod(ts,'M6a');
f=flow;
setcoordinate(ts,'exp')
setvectorfield(f,vf);
settimestepper(f,ts);
curve=f(yy,y0,y1,hy);
dati=getdata(curve.y);
for i=1:M+1
    rho1(:,:,i,1)=dati(:,:,i);
end


%solve for rho(x,y) given as initial data rho(x0,y) computed in the
%previous step
u1=hmlie(lggl(3));
for j=1:M+1
setdata(u1,rho1(:,:,j,1));
vf=vectorfield(u1);
seteqntype(vf,'L');
fid1 = fopen( 'vfse3x1.m', 'w' );
fprintf( fid1, 'function [ la ] = vfse3x1( x,y ); la=liealgebra(y);
  dat=[0,-4./(x + (%f+(%f-1)*%f)),-1;4./(x + (%f+(%f-1)*
%f)),0,0;0,0,0]; setdata(la,dat); end',y0,j,h,y0,j,h);
fclose(fid1);
setfm2g(vf,'vfse3x1');
ts=tsmagnus;
setmethod(ts,'M6a');
f=flow;
setcoordinate(ts,'exp')
setvectorfield(f,vf);
settimestepper(f,ts);
curve=f(u1,x0,x1,h);
dati=getdata(curve.y);
for k=2:N+1
    rho1(:,:,j,k)=dati(:,:,k);
end
end




for j=1:M+1
    for l=1:N+1
        diffrho(:,:,j,l)=rho(:,:,j,l)-rho1(:,:,j,l);
        normrho(j,l)=norm(rho(:,:,j,l));
        normrho1(j,l)=norm(rho1(:,:,j,l));
        normdiffrho(j,l)=norm(diffrho(:,:,j,l));
        solx(j,l)=rho(1,2,j,l).*rho(2,3,j,l)-
rho(1,3,j,l).*rho(1,1,j,l);
        solx1(j,l)=rho1(1,2,j,l).*rho1(2,3,j,l)-
rho1(1,3,j,l).*rho1(1,1,j,l);
```

```
        solu(j,l)=-
rho(2,3,j,l).*rho(1,1,j,l)+rho(1,3,j,l).*rho(2,1,j,l);
        solu1(j,l)=-
rho1(2,3,j,l).*rho1(1,1,j,l)+rho1(1,3,j,l).*rho1(2,1,j,l);
    end
end


% figure();
% surf1=surf(X,Y,normdiffrho);
% xlabel('s') % x-axis label
% ylabel('t') % y-axis label
% zlabel('rho^{gamma1}-rho^{gamma2}') % z-axis label
% figure();
% surf2=surf(solx,solu,Y);
% set(surf2,'LineStyle','none')
% xlabel('x') % x-axis label
% ylabel('u') % y-axis label
% zlabel('t') % z-axis label
```

```
  DiffMan Version 2.01 is initialized - 2012.05.01

  Please report any problems and/or bugs to:
       help@diffman.no

  For more information and how to get started, try:
      >> dmtutorial
      >> dmhelp
      >> demo
```

Published with MATLAB® R2019a

C. Appendix to Chapter 5

Example 1

The following MATLAB code relates to Section 5.4.1. The code has been thoroughly commented to provide a better readability for the reader. The Matlab file can be found at https://zenodo.org/record/3661107/files/RZSZ_explicit_v11.m?download=1.

```
clear all;

%initial conditions and auxiliary variables
a=0; %initial point
b=1; %endpoint
n=250; %number of points in [a,b]
s=linspace(a,b,n+1);
thetam1=0.001; %initial conditions for theta
theta0=0.002;
theta1=0.0025;
theta=zeros(n+1,1);
theta(1)=thetam1;
theta(2)=theta0;
theta(3)=theta1;
ell=zeros(n+1,1); %initial conditions for \ell
ell(1)=0.0015;
ell(2)=0.0013;
dt=zeros(n+1,1); %initialising \Delta \Theta
dt(1)=theta(2)-theta(1);
dt(2)=theta(3)-theta(2);
Z=zeros(2,1,n+1); %this is the vector called W in section 5.4.2
x0=0; %we can choose arbitrarily where do we want the minimisers (x,u)
 to start
u0=0;
x=zeros(n+1,1); %initialising the minimisers
u=zeros(n+1,1);
x(1)=x0;
u(1)=u0; %setting the initial point for the minimisers

Rdt0=[cos(dt(2)) -sin(dt(2)); sin(dt(2)) cos(dt(2))]; %rotation matrix
 for dt0

z1=2/ell(2)*(sin(dt(1))*cos(dt(1))/ell(1)-sin(dt(2))*cos(dt(2))/
ell(2));  %first Euler-Lagrange equation
z2=-sin(dt(2))^2/ell(2)^2;    %second Euler-Lagrange equation
Z(:,:,1)=[z1 ; z2]; % updating the vector Z (vector W in Section
 5.4.2)
Z(:,:,2)=Rdt0*Z(:,:,1); %computing the next step
z1=Z(1,1,2);
z2=Z(2,1,2);
syms l dtheta %initialising symbolic variables. These are the unknowns
 we need to find to compute the next point of the solution
d=0; %two parameters that keep track of where is the solution and in
 which direction are we going
scen=0;

for k=3:n+1
     if abs(dt(k-1))<1e-7 %we need to distinguish this case as the
 solver would pick a zero solution if we're too close to 0
         if scen==3 %parameters that keep track of which quadrant are
 we solving the equations in
             scen=4;
```

```
        elseif scen==1
                scen=5;
        end
   eq1=Z(1,1,k-1)*l^2-l*sin(2*dt(k-1))/ell(k-1)+2*dtheta; %E-L
equation 1
   eq2=-dtheta^2+Z(2,1,k-1)*l^2; %E-L equation 2. We're solving for
dtheta and l
   eq=[eq1 eq2];
   initguess=[ell(k-1) -20*dt(k-1)]; %initial guess for the solver
   if mod(d,2)==0
      sol=vpasolve(eq,[l dtheta], [1e-6  1e-2;-0.7 -1e-8]); %solution
if we're going lockwise
   else
      sol=vpasolve(eq,[l dtheta], [1e-6  1e-2;1e-8 2.7]); %solution
if we're going anticlockwise
   end
   dt(k)=sol.dtheta;%storing the solution
   ell(k)=sol.l;
   d=d+1;
   elseif z2<0 %same as before, for z2<0 (which means we are in the
upper half plane)
   scen=1;
   eq1=Z(1,1,k-1)*l^2-l*sin(2*dt(k-1))/ell(k-1)+sin(2*dtheta);
   eq2=-sin(dtheta)^2-Z(2,1,k-1)*l^2;
   initguess=[ell(k-1) dt(k-1)];
   if dt(k-1)<0
      initguess=[ell(k-1) -dt(k-1)];
   end
   sol=vpasolve(eq,[l dtheta],[1e-6  1e-2;1e-6 2]);
   dt(k)=sol.dtheta;
   ell(k)=sol.l;
   elseif z2>0 %same as before, for z2>0 (which means we are in the
lower half plane)
     if scen==5
        scen=5;
      else scen=3;
       end
   eq1=Z(1,1,k-1)*l^2-l*sin(2*dt(k-1))/ell(k-1)+sin(2*dtheta);
   eq2=-sin(dtheta)^2+Z(2,1,k-1)*l^2;
   eq=[eq1 eq2];
   initguess=[ell(k-1) dt(k-1)];
   if dt(k-1)>0 && scen~=4 && scen~=3
   initguess=[ell(k-1) -20*dt(k-1)];
   end
   sol=vpasolve(eq,[l dtheta],initguess);
   dt(k)=sol.dtheta;
   ell(k)=sol.l;
   end

   if     scen==1 %advancing the solution if we are going
anticlockwise
   Z(:,:,k)=[cos(dt(k)) -sin(dt(k)); sin(dt(k))
cos(dt(k))]*[Z(1,1,k-1);Z(2,1,k-1)];
```

```
    elseif scen==3 || scen==4 %advancing the solution if we are going
 clockwise
    Z(:,:,k)=[cos(dt(k)) sin(dt(k)); -sin(dt(k))
 cos(dt(k))]*[Z(1,1,k-1);Z(2,1,k-1)];
    end
    z1=Z(1,1,k);
    z2=Z(2,1,k);
end

for l=2:n-1
    theta(l+2)=dt(l+1)+theta(l+1); %extract the angles theta as we
 have computed the difference theta(l+2)-theta(l+1)=dtheta(l+1)
end

for j=2:n+1
    x(j)=x(j-1)+cos(theta(j-1))*ell(j-1); %computing the minimisers
    u(j)=u(j-1)-sin(theta(j-1))*ell(j-1);
end

for m=1:n+1
    normZ(m)=norm(Z(1:2,1,m)); %check the norm of Z (we have been
 rotating Z, so the norm should stay constant)
end


%conservationb laws
c=zeros(1,3,n+1);
Ad=zeros(3,3,n+1); %initialise the adjoint representation of the
 moving frame
for k=3:n
%v1,v2 and v3 are the three components of the vector of invariants
 appearing in the
%conservation laws
v1(k)=-cos(dt(k-1))*sin(dt(k-1))^2/ell(k-1)^2+sin(dt(k-1))/
ell(k-1)*(sin(2*dt(k-2))/ell(k-2)-sin(2*dt(k-1))/ell(k-1));
v2(k)=-sin(dt(k-1))^3/ell(k-1)^2-cos(dt(k-1))/
ell(k-1)*(sin(2*dt(k-2))/ell(k-2)-sin(2*dt(k-1))/ell(k-1));
v3(k)=-sin(2*dt(k-1))/ell(k-1);
%defining the moving frame
Ad(:,:,k)=[cos(theta(k)),-sin(theta(k)),-sin(theta(k))*x(k)-
cos(theta(k))*u(k);sin(theta(k)),cos(theta(k)),cos(theta(k))*x(k)-
sin(theta(k))*u(k);0,0,1];
v=[v1(k),v2(k),v3(k)];
c(1,:,k)=v*Ad(:,:,k); %computing the conservation laws
end

for m=10:n-3 %computing the difference between the m-th and 0th
 elements of the cons laws
   c1diff(m)= abs(c(1,1,m)-c(1,1,10));
   c2diff(m)= abs(c(1,2,m)-c(1,2,10));
   c3diff(m)= abs(c(1,3,m)-c(1,3,10));
end
```

```
%  figure(); %plot the evolution of the difference in the conservation
 laws
%  subplot(2,2,1);
%  plot(x(5:end),c1diff);
%  xlabel('t');
%  ylabel('|c_1(t)-c_1(t_0)|');
%  subplot(2,2,2);
%  plot(x(5:end),c2diff);
%  xlabel('t');
%  ylabel('|c_2(t)-c_2(t_0)|');
%  subplot(2,2,3);
%  plot(x(5:end),c3diff);
%  xlabel('t');
%  ylabel('|c_3(t)-c_3(t_0)|')
%  sgtitle('Conservation Laws')
%  figure();
%  plot(x,u); %plot the minimisers as (x,u)
%  xlabel('x');
%  ylabel('u');
%  title('Solution')
```

Published with MATLAB® R2019a

Example 2

The following MATLAB code relates to Section 5.4.2. The code has been thoroughly commented to provide a better readability for the reader. The Matlab file can be found at https://zenodo.org/record/3661107/files/example_constrained_1.m?download=1.

```
clear all;
%setting initial conditions
a=0;
b=1;
n=250;  %number of points
h=0.0015;
s=linspace(a,b,n+1);
thetam1=0.001;  %initial conditions for the angle
theta0=0.002;
theta1=0.0025;
theta=zeros(n+1,1);
theta(1)=thetam1;
theta(2)=theta0;
theta(3)=theta1;
lambda=zeros(n+1,1);
lambda(2)=0.0015;
dt=zeros(n+1,1);
dt(1)=theta(2)-theta(1);    %defining \Delta \Theta
dt(2)=theta(3)-theta(2);
Z=zeros(2,1,n+1);
x0=0;
u0=0;
x=zeros(n+1,1);
u=zeros(n+1,1);
x(1)=x0;
u(1)=u0;
dt1=zeros(n+1,1);
syms dtheta;

Rdt0=[cos(dt(3)) -sin(dt(3)); sin(dt(3)) cos(dt(3))]; %Rotation matrix

z1=2*(sin(dt(1))*cos(dt(1))-sin(dt(2))*cos(dt(2)))/h^2;  %Euler-
Lagrange equation 1
z2=lambda(2)*h-sin(dt(2))^2/h^2;  %Euler-Lagrange equation 2
Z(:,:,1)=[z1 ; z2];
Z(:,:,2)=Rdt0*Z(:,:,1); %advancing the solution

for k=3:n+1
    dt(k)=asin(sin(2*dt(k-1))-Z(1,1,k-1)*h^2)/2;  %in this case we can
 explicitely solve for dt
    lambda(k)=Z(2,1,k-1)+sin(dt(k))^2/h^2; %and for lambda too
    Z(:,:,k)=[cos(dt(k)) -sin(dt(k)); sin(dt(k))
 cos(dt(k))]*Z(:,:,k-1); %advancing the solution
end

for l=2:n-2
    theta(l+2)=dt(l+1)+theta(l+1); %extracting the thetas from the dt
end

for j=1:n %constructing the minimisers (x,u)
    x(j+1)=x(j)+cos(theta(j))*h;
    u(j+1)=u(j)-sin(theta(j))*h;
```

```
end

for m=1:n+1
    normZ(m)=norm(Z(1:2,1,m)); %check the norm of Z stayed constant
end

%conservation laws
Ad=zeros(3,3,n+1); %initialising the moving frame
for k=3:n
    %v1,v2 and v3 are the three invariant components of the vector of
    %invariants in the conservation laws
v1(k)=-cos(dt(k-1))*(sin(dt(k-1))^2/h^2-lambda(k-1))+sin(dt(k-1))/
h*(sin(2*dt(k-2))/h-sin(2*dt(k-1))/h);
v2(k)=-sin(dt(k-1))*(sin(dt(k-1))^2/h^2-lambda(k-1))-cos(dt(k-1))/
h*(sin(2*dt(k-2))/h-sin(2*dt(k-1))/h);
v3(k)=-sin(2*dt(k-1))/h;
%defining the moving frame
Ad(:,:,k)=[cos(theta(k)),-sin(theta(k)),-sin(theta(k))*x(k)-
cos(theta(k))*u(k);sin(theta(k)),cos(theta(k)),cos(theta(k))*x(k)-
sin(theta(k))*u(k);0,0,1];
v=[v1(k),v2(k),v3(k)];
c(:,1,k)=v*Ad(:,:,k); %computing the conservation laws
end



for m=10:n-3 %computing the difference between the m-th and 0-th
 elements of the conservation laws
   c1diff(m)= abs(c(1,1,m)-c(1,1,10));
   c2diff(m)= abs(c(2,1,m)-c(2,1,10));
   c3diff(m)= abs(c(3,1,m)-c(3,1,10));
end

%  figure(); %plot the conservation laws
%  subplot(2,2,1);
%  plot(s(5:end),c1diff);
%  xlabel('t');
%  ylabel('|c_1(t)-c_1(t_0)|');
%  subplot(2,2,2);
%  plot(s(5:end),c2diff);
%  xlabel('t');
%  ylabel('|c_2(t)-c_2(t_0)|');
%  subplot(2,2,3);
%  plot(s(5:end),c3diff);
%  xlabel('t');
%  ylabel('|c_3(t)-c_3(t_0)|')
%  sgtitle('Conservation Laws')
%  figure();
%  plot(x,u);
%  xlabel('x'); %plot the minimisers as (x,u)
%  ylabel('u');
%  title('Solution')
```

Published with MATLAB® R2019a