Lie-group methods

Arieh Iserles, Hans Munthe-Kaas, Syvert Nørsett, Antonella Zanna

Lie-group methods

Arieh Iserles
Department of Applied Mathematics and Theoretical Physics,
University of Cambridge, England
Email: a.iserles@damtp.cam.ac.uk

Hans Z. Munthe-Kaas
Department of Computer Science,
University of Bergen, Norway
Email: hans@ii.uib.no

Syvert P. Nørsett
Institute of Mathematics,
Norwegian University of Science and Technology
Trondheim, Norway
Email: norsett@math.ntnu.no

Antonella Zanna
Department of Computer Science,
University of Bergen, Norway
Email: anto@ii.uib.no

Many differential equations of practical interest evolve on Lie groups or on manifolds acted upon by Lie groups. The retention of Lie-group structure under discretization is often vital in the recovery of qualitatively-correct geometry and dynamics and in the minimisation of numerical error. Having introduced requisite elements of differential geometry, this paper surveys the novel theory of numerical integrators that respect Lie-group structure, highlighting theory, algorithmic issues and a number of applications.

CONTENTS

1. Numerical analysts in Plato's temple

"Ageometretos medeis eisito": let nobody enter who does not understand geometry. These were the words written at the entrance to Plato's Temple of the Muses. Are numerical analysts welcome in Plato's temple?

Historically, the answer is negative. Computational mathematics is all about rendering mathematical phenomena in an algorithmic form, amenable to sufficiently precise, affordable and robust number crunching. A mathematical phenomenon can be approached in one of two ways: either by exploring its *qualitative* features (which, to a large extent, are synonymous with geometry or, at the very least, can be formulated in geometric terminology) or by approximating its *quantitative* character. Although only purists reside completely at either end of the spectrum, it is fair to point out that numerical analysis, by its very 'rules of engagement', is what 'quantitative mathematics' is all about. Ask a numerical analyst "How good is the solution?" and the likely answer will address itself to a subtly different question, "How small is the magnitude of the error?".

In principle, the emphasis on quantitative aspects in mathematical computing served it well. It is hard to imagine modern technological civilisation without the multitude of silent computer programs in the background, flying the airplanes, predicting the weather, making sense of CAT scans, controlling robots, identifying fingerprints, keeping reactions from running away and predicting the behaviour of stock markets. This is the success story of numerical analysis, of this 'quantitative number crunching', and nothing should be allowed to obscure it. So, perhaps if we are doing so well everywhere else, we might cede Plato's temple to our more

'pure' brethren and sisters: let them engage in sterile intellectual discourse while we change the world!

The main contention of this review and of the emerging discipline of *geometric integration* is that this approach, although tempting, is at best incomplete, at worst badly misguided. The history is not just a heroic tale of numerical algorithms fleshing out mathematical concepts as numbers and graphs. Progress has always occurred along parallel, intertwined tracks: *both* better theoretical understanding of qualitative attributes of a mathematical construct *and* its better computation. The airplane-flying, weather-predicting and CAT-scanning programs can do their job only because they deliver an answer that explains in a satisfactory manner qualitative features, as well as producing the 'right' numbers! Indeed, an artificial dichotomy of quantitative and qualitative aspects of mathematical research is in our opinion misleading and it serves ill mathematical and applied communities alike.

On the one hand, computation tells pure mathematics *what to prove.* Phenomena are often initially identified when observed under discretization and subsequently subjected to the full rigour of mathematical analysis. A familiar case in point is the discovery of solitons in the solution of the Korteweg–de Vries equation by Zabusky and Kruskal (1965), an event which launched a whole new mathematical discipline, other examples abound. Indeed, we are so used to rely on the computer as a laboratory of pure mathematics that it is difficult to imagine the heroic work of Gaston Julia (1918) on the geometry of fractals while bearing in mind that he has had no access to computers, never able to easily calculate a sequence of rational iterations or to visualise a fractal on a computer monitor!

On the other hand, qualitative analysis tells computation, quite literally, *what to compute.* Every seasoned numerical analyst knows that the procedure of 'discretise everything in sight and throw it on a computer' works only with toy problems. The more we know about the qualitative behaviour of the underlying mathematical construct, the more we can identify the right computational approach, concentrate resources at the right place, focus on features that influence more the quantitative behaviour and, by the conclusion of the computation, have well-founded expectation that the graph on the computer monitor corresponds to a genuine solution of the problem in hand.

Moreover, consumers of numerical calculations are not interested just in numbers, graphs and impressive visualisation. Very often it is the qualitative features, most conveniently phrased in the language of geometry, that draw genuine interest in applications: periodicity, chaoticity, conservation of energy or angular momentum, reduction to lower-dimensional manifolds, symmetry, reversibility, ...

The contention of this review is not just that the contribution of geometry to computation in the special case of time-evolving systems of differential equations is absolutely crucial, but that the terminology of differential geometry, and in particular Lie groups, creates the right backdrop to this process. The following example will help to elucidate this point, while serving as a convenient introduction to the theme of this paper.

Let us denote by S_N the set of all $N \times N$ real symmetric matrices and consider the solution of the *isospectral flow*

$$Y' = B(Y)Y - YB(Y), \quad t \geq 0, \qquad Y(0) = Y_0 \in \mathrm{S}_N, \tag{1.1}$$

where the (sufficiently smooth) function B maps S_N to $N \times N$ real skew-symmetric matrices. The solution itself remains in S_N for all $t \geq 0$. Such flows occur in a variety of applications. Perhaps the earliest (and the best known) is the *Toda lattice* of material points subjected to nearest-neighbour interaction. It has been demonstrated by Flaschka that, in the case of an exponential interaction potential, the underlying Hamiltonian system can be rendered in the form (1.1), where Y is tridiagonal and B maps

$$\begin{bmatrix} \alpha_1 & \beta_1 & 0 & \cdots & 0 \\ \beta_1 & \alpha_2 & \beta_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \beta_{N-2} & \alpha_{N-1} & \beta_{N-1} \\ 0 & \cdots & 0 & \beta_{N-1} & \alpha_N \end{bmatrix} \quad \text{to} \quad \begin{bmatrix} 0 & \beta_1 & 0 & \cdots & 0 \\ -\beta_1 & 0 & \beta_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -\beta_{N-2} & 0 & \beta_{N-1} \\ 0 & \cdots & 0 & -\beta_{N-1} & 0 \end{bmatrix}$$

(Toda 1981). Another important application of (1.1) is to *Lax pairs* in fluid dynamics, whence S_N need be replaced by a suitable function space, B is a differential operator and the outcome is a partial differential equation of a hyperbolic type (Toda 1981). Before we mention another application of isospectral flows, we need to single out their most remarkable qualitative feature which, coincidentally, explains their name: as the time evolves, the *eigenvalues* of $Y(t)$ stay put! Upon a moment's reflection, this renders such flows interesting in the context of numerical algebra. Indeed, the classical QR algorithm is intimately related to sampling the solution of (1.1) at unit intervals (Deift, Nanda and Tomei 1983). Many other iterative algorithms can be phrased in this terminology and, perhaps more importantly, many interesting algorithms rely on this construct in the first place. Pride of place belongs here to methods for the *inverse eigenvalue problem:* seeking a matrix of a given structure that possesses a specified set of eigenvalues (or singular values). Such problems are important in a wide range of applications, ranging from the theory of vibrations to control theory, tomography, system identification, geophysics, all the way to particle physics. Isospectral flows are a common denominator to perhaps the most powerful approach toward the design of practical algorithms for the inverse eigenvalue problem, which has been pioneered in the main by Chu (1998). Suppose that we are seeking a matrix in a class $\mathcal{T} \subset \mathrm{S}_N$ with the eigenvalues $\boldsymbol{\eta} \in \mathbb{R}^N$. Often it is possible to design a matrix function B so that attractive fixed points of (1.1) lie in $\mathcal{T}$. In that case, letting $Y(0) = \operatorname{diag} \boldsymbol{\eta}$ results in a flow that converges to the solution of the inverse eigenvalue problem.

As an example of such a procedure we mention the inverse eigenvalue problem

for *Toeplitz matrices.* Thus, $\mathcal{T}$ consists of symmetric $N \times N$ Toeplitz matrices:

$$X \in \mathcal{T} \qquad \Leftrightarrow \qquad x_{k,l} = t_{|k-l|}, \quad k, l = 1, 2, \ldots, N,$$

where $t_0, t_1, \ldots, t_{N-1}$ are arbitrary real numbers. Such problems are important in the design of control systems but, remarkably, even the very existence of a solution has been until very recently an open problem, which has been answered by Landau (1994) in a beautiful, yet non-constructive, existence proof. Following (Chu 1993, Trench 1997), we let

$$b_{k,l}(Y) = \begin{cases} y_{k,l-1} - y_{k+1,l}, & 1 \le k < l \le N, \\ 0, & 1 \le k = l \le N, \\ y_{k+1,l} - y_{k,l-1}, & 1 \le l < k \le N \end{cases}$$

be a *Toeplitz annihilator.* Note that B is indeed skew symmetric and that $B(Y) = \boldsymbol{O}$ for $Y \in \mathcal{T}$: thus, a solution is a fixed point.

While remarking that many important questions with regard to the convergence of the above algorithm are still wide open, we should draw the reader's attention to a crucial observation. For numerical purposes, sooner or later we must replace (1.1) by a computational time-stepping scheme. Will such a scheme respect the eigenstructure of Y? This is not simply an optional extra since the whole point of the exercise is to evaluate an answer in $\mathcal{T}$. Yet, as proved in (Calvo, Iserles and Zanna 1997), the most popular numerical methods, multistep and Runge–Kutta schemes, do not respect isospectral structure and they fail to converge to the correct element of $\mathcal{T}$: the error on the eigenvalues, of the same order of magnitude as the error in the numerical trajectory itself, is unacceptable.

An alternative, proposed by Calvo et al. (1997), is to observe that all the elements of the *isospectral manifold*

$$\mathcal{I}(\boldsymbol{\eta}) = \{X \in \mathrm{S}_N \,:\, \sigma(X) = \boldsymbol{\eta}\},$$

where $\sigma(X)$ denotes the spectrum of X, can be written in the form $X = QY_0Q^{\mathrm{T}}$, where $Y_0 = \operatorname{diag} \boldsymbol{\eta}$ is our initial condition and $Q \in \mathrm{SO}(N)$, the set of all $N \times N$ real orthogonal matrices with unit determinant. The main idea is to seek, in place of (1.1), a differential equation that is satisfied by $Q(t)$ in the representation

$$Y(t) = Q(t)Y_0Q(t)^{\mathrm{T}}, \quad t \ge 0. \tag{1.2}$$

It is easy to ascertain that this equation has the form

$$Q' = B(QY_0Q^{\mathrm{T}})Q, \quad t \ge 0, \qquad Q(0) = I \tag{1.3}$$

and that, provided we can solve it while retaining orthogonality, we can easily recover the solution of the original isospectral flow.

To illustrate our point, let us consider a simple numerical experiment. We choose

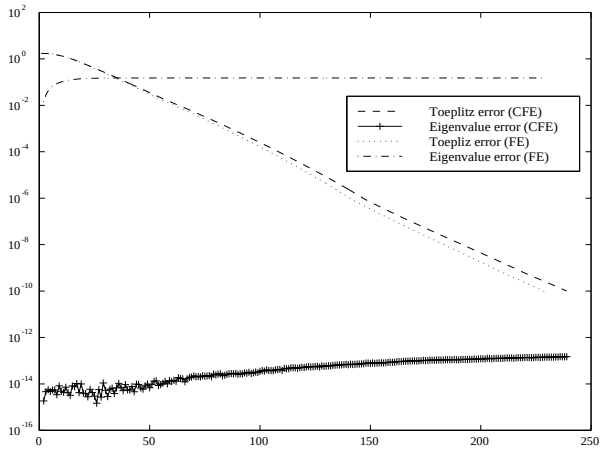


Fig. 1.1. Toeplitz error and error on the eigenvalues versus number of iterations for a 5×5 symmetric inverse eigenvalue problem when solved with the Forward Euler scheme (FE) and the Cayley-based Forward Euler (CFE). Although both methods converge asymptotically to a Toeplitz matrix, the error on the eigenvalues of CFE stays within machine accuracy while the error of FE is completely determined by the choice of integration step size.

$\boldsymbol{\eta} = [1, 2, 3, 4, 5]^{\mathrm{T}}$, and solve the symmetric inverse eigenvalue problem (1.1), where the matrix function B is the Toeplitz annihilator introduced above. We consider first the standard Forward Euler (FE) scheme,

$$Y_{n+1} = Y_n + h[B(Y_n), Y_n], \qquad n \in \mathbb{Z}^+,$$

with initial condition $Y_0 = \mathrm{diag}(\boldsymbol{\eta})$ and step size $h = \frac{1}{10}$. The Toeplitz error, $\|B(Y_n)\|_2$, is plotted in Figure 1.1 as a dotted line. Clearly, as n tends to infinity, the Toeplitz error becomes progressively smaller and Y_n tends to a Toeplitz matrix, as the theory predicted. Do the eigenvalues stay put? The answer is negative, since the FE scheme is not isospectral, as proven in Calvo et al. (1997). Therefore the error on the eigenvalues after the first step is of the same order of the error of FE and is carried along the whole integration (dash-dotted line in Figure 1.1).

Consider next the iteration

$$\begin{aligned} Q_{n+1} &= \left(I - \tfrac{1}{2}hB(Y_n)\right)^{-1}\left(I + \tfrac{1}{2}hB(Y_n)\right), \\ Y_{n+1} &= Q_{n+1} Y_n Q_{n+1}^{\mathrm{T}}, \end{aligned} \qquad n \in \mathbb{Z}^+,$$

which is equivalent to solving (1.3) with a modified version of the Forward Euler scheme based on the Cayley expansion (note that Q_{n+1} is orthogonal) in tandem

with a similarity transformation for the update Y_{n+1}. This scheme, to which we will refer as CFE (Cayley-type Forward Euler), is explicit, has the same order of accuracy and requires only slightly more computations than the more classical FE. However, unlike FE, it is isospectral *by design* and preserves the eigenvalues to machine accuracy. In conclusion, FE tends to a Toeplitz matrix with wrong eigenvalues, while CFE, a simple modification of FE that instead preserves the qualitative features of the flow, tends to a Toeplitz matrix with the right eigenvalues. The Toeplitz and the eigenvalue error of CFE are displayed in Figure 1.1 and correspond to the dashed and 'plus' lines respectively.

The set $\mathrm{SO}(N)$ in which the matrix Q of the above example evolves is an instance of a *Lie group,* a concept that will be described and debated in great detail in Section 2, while (1.2) and (1.3) are special instances of a *group action* and a *Lie-group equation.*

Let us comment briefly on the contents of this review. In Section 2 we have assembled the common mathematical denominator underlying this paper: elements of differential geometry, Lie groups and algebras, homogeneous spaces and differential equations evolving on such objects. Section 3 is devoted to Runge–Kutta–Munthe-Kaas schemes, the most natural approach to Lie-group solvers in our setting. In Section 4 we describe expansions, originally due to Magnus and to Fer, which can be converted into interesting computational tools. Section 5 is concerned with a make-or-break issue for many Lie-group methods, multivariate quadrature of multilinear forms over polytopes. We demonstrate there that some very technical tools from Lie-algebra theory can be used to a great effect in reducing the numerical cost. Lie-group methods are typically based on local imposition of a convenient coordinate system in the group. In Section 6 we debate less conventional choices of the coordinate map, which are suitable for important Lie groups and equations of practical interest. The theme of Section 7 is time-symmetric methods that, by design, exhibit many favourable features, while the concern of Section 8 is the practical approximation of a matrix exponential from a Lie algebra, so that the result lies in the right Lie group. Stability issues are addressed in Section 9, while Section 10 reviews practical issues of implementation and error control and introduces the `DiffMan` package. A sample of the many applications of Lie-group methods is presented in Section 11. Finally, in Appendix A we list practical Lie-group methods, while Appendix B displays useful explicit formulae for integration in SO(3), perhaps the single Lie group with most important relevance to problems in science and engineering.

The purpose of this survey is not to cover the entire corpus of Lie-group methods but to present a unified introduction to a young discipline that is likely to undergo many exciting further developments. We have omitted many interesting methods and papers, with due apologies to their authors, to keep our narrative more focused and clear. Only the future can tell which methods and techniques will survive.

It is vital throughout the paper to distinguish what exactly is the type of objects under consideration. We will be often mixing in our formulae elements of Lie

groups and Lie algebras, scalars, matrices and vectors. To assist the reader, we have adhered to a consistent naming convention:

- Elements in a Lie algebra are denoted by the Roman letters $a, b, \ldots, h$ and $A, B, \ldots, H$ and by the Greek letters $\alpha, \beta, \ldots, \xi$ and Δ, Θ, Ξ.
- Elements in a Lie group are denoted by the Roman letters $p, q, \ldots, z$ and $P, Q, \ldots, Z$ and by the remaining Greek letters: $\pi, \rho, \ldots, \omega$ and $\Upsilon, \Phi, \Psi, \Omega$.
- Elements in an abstract construct (e.g., an abstract Lie group) are denoted mostly with lower-case letters. However, as soon as we are concerned with specific representation, we reserve lower-case letters for scalars, upper case for matrices and lower-case, boldfaced letters for vectors.
- Special elements deserve special names. Thus, $\boldsymbol{I}$ is the identity in a Lie group, while $\boldsymbol{O}$ is the zero of a Lie algebra. A generic Lie group will be denoted by $\mathcal{G}$ and a generic Lie algebra by $\mathfrak{g}$. In general, we reserve Gothic font for Lie algebras.
- As in all naming conventions, we make obvious compromises with standard mathematical practice and common sense. Proper names remain unchanged: thus, $\sin t$ is the familiar sine function, not an element of a group, $\{\mathrm{B}_k\}_{k\in\mathbb{Z}^+}$ are Bernoulli numbers, Φ is the set of roots of a Lie algebra, h is the step size of a time-stepping algorithm and so on. We also employ a plethora of integration variables, summation indices, constants etc. Occasionally variables evolve in structures which are neither Lie groups nor Lie algebras, e.g. the isospectral manifold $\mathcal{I}(\boldsymbol{\eta})$. In all these cases, which should be obvious from the context, we use *ad hoc* notation.

2. Theory and background

Lie groups and Lie algebras are mathematical objects which have originated in the seminal work of Sophus Lie (1842–1899) on solving differential equations by quadrature, using symmetry methods. Originally these concepts were quite concrete, related to flows of differential equations on $\mathbb{R}^N$. Early in the twentieth century an abstract view of Lie group theory emerged, commencing from the work of Elie Cartan on the classification of Lie algebras. The advantage of abstract formulation is that it simplifies mathematical analysis, and hence this presentation has become dominant throughout mathematical literature. However, the abstract theory is concentrating on understanding mathematical structures rather than exposing applications in solving differential equations. Hence it is not at all clear to most applied mathematicians that Lie groups are really very useful objects also in applied and computational mathematics, and it might be difficult to find the motivation to learn an abstract theory.

We believe that the original idea of arriving at Lie algebras via continuous actions on a domain should be an excellent starting point for computationally-oriented mathematicians, and in fact for many applications it is important to keep this view in focus. In this presentation we will commence from this perspective and gradually

move towards somewhat more abstract formulations. Eventually, in Section 2.5 we will return to concrete matrix formulation, concentrating on the numerical solution of matrix differential equations of the form

$$Y' = A(t, Y)Y, \quad t \geq 0, \qquad Y(0) = Y_0,$$

where Y and $A(t, Y)$ are $N \times N$ matrices. It turns out that all our solution techniques can be derived in this concrete matrix setting and without major modifications they can be applied to more general situations. All the algorithms of this paper will be derived within the matrix framework, and hence the theory from this point and up to Section 2.5 might be read in a relaxed manner, without the need to master all the details at first reading.

Numerical integration of ordinary differential equations (ODEs) is traditionally concerned with solving initial value problems evolving on $\mathbb{R}^N$,

$$\boldsymbol{y}' = \boldsymbol{f}(t, \boldsymbol{y}), \quad t \geq 0, \qquad \boldsymbol{y}(0) = \boldsymbol{y}_0, \qquad \boldsymbol{y}(t) \in \mathbb{R}^N,$$

where $\boldsymbol{f}$ is a vector field on $\mathbb{R}^+ \times \mathbb{R}^N$. Well-known numerical integrators, such as Runge–Kutta and multistep methods, advance a time-stepping procedure by adding vectors in $\mathbb{R}^N$,

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\boldsymbol{a}_n,$$

where $\boldsymbol{a}_n = \boldsymbol{a}_n(h, \boldsymbol{y}_n, \ldots)$ is computed by the given numerical method and h is the time step. One might say that *classical integrators are formulated using a set of 'basic motions' given by translations on $\mathbb{R}^N$ to advance the numerical solution.*

A major motivation for Lie group methods is the possibility of replacing the domain $\mathbb{R}^N$ with more general configuration spaces and replacing translations on $\mathbb{R}^N$ by more general families of 'basic motions' on the domain. For example, if $\boldsymbol{y}(t)$ is a vector known to evolve on a sphere, one might consider rotations $\boldsymbol{y}_{n+1} = Q_n\boldsymbol{y}_n$, where Q_n is an orthogonal matrix, as basic motions. We have already encountered another example, isospectral flows (1.1). In that case $Y(t)$ evolves on the isospectral manifold $\mathcal{I}(\sigma(Y_0))$ and this is the right configuration space. Moreover, the natural 'basic motions' are (1.2) and they rest upon the fact that any two elements in $\mathcal{I}(\sigma(Y_0))$ are similar via an orthogonal matrix. A generalisation of the last example does not require any more that Y_0 is symmetric and that the matrix function B is skew symmetric: any nonsingular initial value and sufficiently smooth matrix function will do. The flow remains isospectral but the elements of the configuration space are no longer orthogonally similar. The representation (1.2) is valid, however, if we allow $Q(t)$ to range across all possible nonsingular real matrices and replace Q^{T} with Q^{-1}.

An important reason why a manifold, rather than the entire $\mathbb{R}^N$, is a suitable configuration space is that it often expresses crucial geometric attributes of the underlying differential system, e.g. conservation laws, symmetries or symplectic structure. As will be seen in the sequel, an added bonus of this approach is that it

frequently leads to interesting numerical advantages, in particular to slower error accumulation.

In seeking abstractions and generalisations of classical numerical methods it is important to bear in mind abstractions in pure-mathematical treatment of differential equations. However, the transition from pure to computational mathematics is not straightforward. Whenever a pure mathematician says "There exists an animal such that...", an applied mathematician must add questions like "Can we compute this animal efficiently?" and "How can we represent it in software?".

2.1. Vector fields and flows on manifolds

A cornerstone of all abstract mathematical presentations of differential equations is the concept of *differential manifolds* as the definition of domains on which differential equations evolve. A good general introduction to manifolds and to differential geometry are (Abraham and Marsden 1978) and (Guillemin and Pollack 1974).

Intuitively one should think of a d-dimensional manifold as being a smooth domain which in a (small) neighbourhood of any point 'looks like' $\mathbb{R}^d$, but typically looks different globally. It is known that any d-dimensional manifold can be *represented* as a d-dimensional surface embedded in $\mathbb{R}^N$ for some $N \geq d$. This is a conceptually very useful view of manifolds. It is, however, also important to know that all geometric properties of manifolds exist independently of any particular representation[1].

It is possible to discuss these properties in a *coordinate-free* language, which is independent of any particular representation. Although this abstract presentation is mathematically very elegant and it provides vital clues toward writing good software in an object-oriented language (Engø, Marthinsen and Munthe-Kaas 1999), it requires quite a bit of work to present and comprehend. The advantage of the abstract language is that it focuses on the essential structures. However, in most of the applications we will deal with, the manifolds exist naturally as surfaces embedded in $\mathbb{R}^N$, and furthermore it is fully possible to understand and use the numerical techniques we will discuss in this paper without knowing the abstract theory of manifolds. We have therefore decided to base our discussion on the following very concrete definition.

Definition 2.1 A d-dimensional manifold $\mathcal{M}$ is a d-dimensional smooth surface $\mathcal{M} \subset \mathbb{R}^N$ for some $N \geq d$.

It should be made crystal clear that all the numerical techniques we are about to present rely solely on those properties of $\mathcal{M}$ that exist independently of any particular embedding in $\mathbb{R}^N$. We believe that a reader with knowledge of coordinate-free

[1] Existence independently of representations should in fact be taken as the very *definition* of a geometric property.

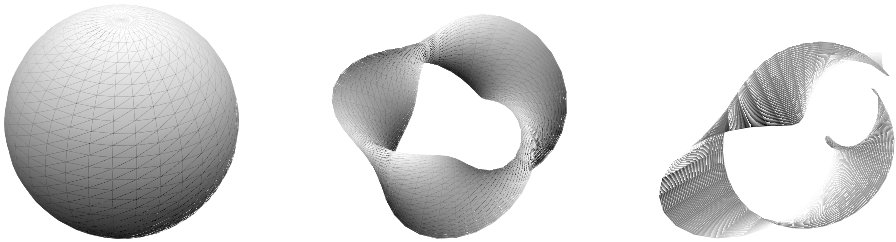


Fig. 2.1. Examples of manifolds embedded in $\mathbb{R}^3$: a sphere, a doubly-twisted Möbius-strip-like torus and a twisted ribbon.

presentations will have no difficulty whatsoever in translating the algorithms and results to a more general setting.

Example 2.1 It is easy to construct examples of manifolds by a number of ways. An example of abstract definition is the specification of an *atlas* of local coordinate charts. However, given our focus on 'concrete' manifolds, we present examples already embedded in $\mathbb{R}^N$.

- Any smooth surface will do and few familiar examples are displayed in Figure 2.1. Smoothness is important: the torus

$$\{(\cos\psi + \rho\cos\theta, \sin\psi + \rho\sin\theta, \rho\cos\theta) \,:\, 0 \leq \psi, \theta \leq 2\pi\}$$

  is a manifold for $\rho \in (0,1)$ but not when $\rho = 1$, because of a singularity at the origin.
- An important representation of a manifold is as a smooth subset of solutions of a smooth algebraic equation, $g(\boldsymbol{x}) = 0$. Thus, for example, $g(\boldsymbol{x}) = \|\boldsymbol{x}\|_2^2 - 1$ defines a unit sphere.
- The algebraic-equation representation is of direct relevance to geometric integration, since conservation laws and integrals of differential systems are nothing else but algebraic equations constant along the solution trajectory. Thus, for example, a Hamiltonian system with Hamiltonian energy $H(\boldsymbol{p}, \boldsymbol{q})$ evolves on the manifold $\{H(\boldsymbol{p}, \boldsymbol{q}) = H(\boldsymbol{p}(0), \boldsymbol{q}(0)) \,:\, \boldsymbol{p}, \boldsymbol{q} \in \mathbb{R}^N\}$.
- The set $\mathrm{O}(N)$ of all $N \times N$ orthogonal matrices is a manifold, since $X \in \mathrm{O}(N)$ is equivalent to $g(X) = \|X^{\mathrm{T}}X - \boldsymbol{I}\|_2^2 = 0$. So is also $\mathrm{SL}(N)$, the set of all $N \times N$ matrices with unit determinant, since $g(X) = \det X - 1$ is a smooth function.
- A *Stiefel manifold* is the set of all real $M \times N$ matrices X such that $X^{\mathrm{T}}X = \boldsymbol{I}$: typically $M > N$, such matrices are 'long and skinny' and $XX^{\mathrm{T}} \neq \boldsymbol{I}$.
- A *Grassmann manifold* is a Stiefel manifold, equivalenced by $\mathrm{O}(M)$. In other words, we identify X_1, X_2 such that $X_1^{\mathrm{T}}X_1 = X_2^{\mathrm{T}}X_2 = \boldsymbol{I}$ if there exists an

orthogonal $M \times M$ matrix Q such that $X_1 = QX_2$. An alternative interpretation of a Grassmann manifold is as the set of all N-dimensional subspaces of $\mathbb{R}^M$.

The single most important property of a manifold is the existence of *tangents* to the manifold in any point $p \in \mathcal{M}$. If we think of the manifold as a surface in $\mathbb{R}^N$, then a tangent at p can be defined as a vector $\boldsymbol{a}$ such that $\operatorname{dist}(p + \varepsilon \boldsymbol{a}, \mathcal{M}) = \mathcal{O}(\varepsilon^2)$. This construction of a tangent is relying on the embedding of $\mathcal{M}$ in $\mathbb{R}^N$, on the linear-space structure of $\mathbb{R}^N$ and even on the metric structure of $\mathbb{R}^N$. An alternative way to define tangents is by differentiating a curve. This approach has the advantage of making no use of the embedding of $\mathcal{M}$ in $\mathbb{R}^N$ and hence makes sense also on general manifolds.

Definition 2.2 Let $\mathcal{M}$ be a d-dimensional manifold and suppose that $\rho(t) \in \mathcal{M}$ is a smooth curve such that $\rho(0) = p$. A *tangent vector* at p is defined as

$$\boldsymbol{a} = \left. \frac{\mathrm{d}\rho(t)}{\mathrm{d}t} \right|_{t=0}.$$

The set of all tangents at p is called the *tangent space at* p and denoted by $\mathrm{T}\mathcal{M}|_p$. It has the structure of a d-dimensional linear space: if $\boldsymbol{a}, \boldsymbol{b} \in \mathrm{T}\mathcal{M}|_p$ then $\boldsymbol{a}+\boldsymbol{b} \in \mathrm{T}\mathcal{M}|_p$ and $\alpha\, \boldsymbol{a} \in \mathrm{T}\mathcal{M}|_p$ for any real α. The collection of all tangent spaces at all points $p \in \mathcal{M}$ is called the *tangent bundle* of $\mathcal{M}$ and denoted by $\mathrm{T}\mathcal{M} = \bigcup_{p \in \mathcal{M}} \mathrm{T}\mathcal{M}|_p$.

Note that whereas it is fine to add tangents *based at the same point*, there is in general no rule for adding tangent vectors based at different points. Thus, to specify a tangent completely we need to provide both the basepoint p and the tangent itself. Hence $\mathrm{T}\mathcal{M}$ is a $2d$-dimensional space, with elements $(p, \boldsymbol{a})$ consisting of all possible tangents $\boldsymbol{a}$ for all possible basepoints p.

Definition 2.3 A (tangent) *vector field on* $\mathcal{M}$ is a smooth function $F : \mathcal{M} \to \mathrm{T}\mathcal{M}$ such that $F(p) \in \mathrm{T}\mathcal{M}|_p$ for all $p \in \mathcal{M}$. The collection of all vector fields on $\mathcal{M}$ is denoted by $\mathfrak{X}(\mathcal{M})$.

Addition and scalar multiplication of vector fields are defined pointwise in a natural way as $(F+G)(p) = F(p) + G(p)$ and $(\alpha F)(p) = \alpha(F(p))$. If $F, G \in \mathfrak{X}(\mathcal{M})$ then also $F + G \in \mathfrak{X}(\mathcal{M})$ and $\alpha F \in \mathfrak{X}(\mathcal{M})$ for all real α.

Definition 2.4 Let F be a tangent vector field on $\mathcal{M}$. By a *differential equation (evolving) on* $\mathcal{M}$ we mean a differential equation of the form

$$\boldsymbol{y}' = F(\boldsymbol{y}), \quad t \geq 0, \qquad \boldsymbol{y}(0) \in \mathcal{M}, \tag{2.1}$$

where $F \in \mathfrak{X}(\mathcal{M})$. Whenever convenient, we allow F in (2.1) to be a function of time, $F = F(t, \boldsymbol{y})$. The *flow* of F is the solution operator $\Psi_{t,F} : \mathcal{M} \to \mathcal{M}$ such that

$$\boldsymbol{y}(t) = \Psi_{t,F}(\boldsymbol{y}_0)$$

solves (2.1).

Note that we can find the vector field F from $\Psi_{t,F}$ by differentiation,

$$F(\boldsymbol{y}) = \frac{\mathrm{d}}{\mathrm{d}t}\Psi_{t,F}(\boldsymbol{y})\bigg|_{t=0}.$$

F is often called the *infinitesimal generator* of the flow $\Psi_{t,F}$.

By reparametrizing time (or scaling the vector field) we can see that the flow operator satisfies the identity

$$\Psi_{\alpha,F} = \Psi_{1,\alpha F}. \tag{2.2}$$

The task of computing the flow of a given vector field is often called the *exponentiation of the vector field*. We will occasionally employ the notation

$$\Psi_{1,F} \equiv \exp(F) \qquad \Leftrightarrow \qquad \Psi_{t,F} \equiv \exp(tF).$$

Computation of a flow is a particular example of an *exponential map*. We will return to a more general definition of exponential maps later. The notation Ψ will be used when we want to emphasise that we are discussing flows of vector fields, and exp whenever the map can be conveniently given the more general interpretation.

Recall our goal of integrating (2.1) numerically, using a chosen set of 'basic motions' to advance the numerical solution. If the analytical solution evolves on $\mathcal{M}$ it is natural to choose a set of basic motions which are everywhere tangent to $\mathcal{M}$, which will produce a numerical solution also evolving on $\mathcal{M}$. It is useful to consider also these basic motions as flows of a finite or infinite collection of vector fields, $B_1, B_2, \ldots$. If we want an efficient solver, we must be able to compute the flow, i.e. exponentiate these B_is, efficiently.

From the standpoint of geometry, numerical integration of ODEs is concerned with the task of approximating the exponential of a general vector field F by exponentials originating in a family of simpler vector fields $B_1, B_2, \ldots$.

Example 2.2 Let $\mathcal{M} = \mathbb{R}^N$ and let $T_{\boldsymbol{a}}$ stand for the constant vector field, $T_{\boldsymbol{a}}(\boldsymbol{y}) = \boldsymbol{a}$ for some vector $\boldsymbol{a} \in \mathbb{R}^N$. The flow of $T_{\boldsymbol{a}}$ is translation along $\boldsymbol{a}$,

$$\Psi_{t,T_{\boldsymbol{a}}}(\boldsymbol{y}_0) = \boldsymbol{y}_0 + t\boldsymbol{a}.$$

The set of all translations can be obviously used to advance the numerical solution in any desired direction on $\mathbb{R}^N$. Note that translations *commute*,

$$\Psi_{t_1,T_{\boldsymbol{a}}} \circ \Psi_{t_2,T_{\boldsymbol{b}}}(\boldsymbol{y}_0) = \boldsymbol{y}_0 + t_1\boldsymbol{a} + t_2\boldsymbol{b} = \Psi_{t_2,T_{\boldsymbol{b}}} \circ \Psi_{t_1,T_{\boldsymbol{a}}}(\boldsymbol{y}_0).$$

Generally flows do not commute. We will later see that a major difference between traditional numerical integrators and Lie-group methods is that the former are

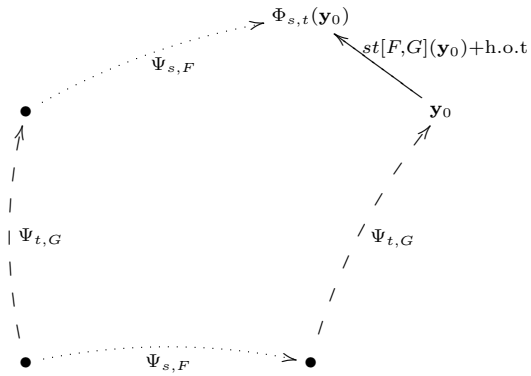


Fig. 2.2. A geometric interpretation of the commutator.

based on a set of commuting flows whereas the latter allow more general flows as basic movements to advance the solution.

The degree to which the flows of two vector fields fail to commute is measured by the *commutator* of the two vector fields. Consider two general vector fields F and G and their flows $\Psi_{s,F}$ and $\Psi_{t,G}$. To investigate commutativity, we form the following composition of the flows,

$$\Phi_{s,t} = \Psi_{s,F} \circ \Psi_{t,G} \circ \Psi_{-s,F} \circ \Psi_{-t,G} \equiv \exp(sF) \circ \exp(tG) \circ \exp(-sF) \circ \exp(-tG) \quad (2.3)$$

(see Figure 2.2). Obviously the flows commute if $\Phi_{s,t}(\boldsymbol{y}) = \boldsymbol{y}$ for all $s, t \geq 0$ and all $\boldsymbol{y} \in \mathbb{R}^N$. However, this is a nonlinear condition that in many cases may be difficult to compute or verify. One of the fundamental ideas due to Sophus Lie was the observation that such nonlinear conditions can be turned into equivalent *linear infinitesimal* conditions. To accomplish this we want to linearise $\Phi_{s,t}$ for small s and t. Since $\Phi_{0,t}(\boldsymbol{y}) = \Phi_{s,0}(\boldsymbol{y}) = \boldsymbol{y}$ for all $\boldsymbol{y}$, we must have

$$\Phi_{s,t}(\boldsymbol{y}) = \boldsymbol{y} + stH(\boldsymbol{y}) + \mathcal{O}(s^2 t) + \mathcal{O}(st^2) \quad (2.4)$$

for some vector field H. This vector field H is called the *commutator*, or the *Jacobi bracket* of F and G, and it is written as $H = [F, G]$.

Lemma 2.1 Given two vector fields F, G on $\mathbb{R}^N$, the commutator $H = [F, G]$ can be computed componentwise at a given point $\boldsymbol{y} \in \mathbb{R}^N$ as

$$H_i(\boldsymbol{y}) = \sum_{j=1}^{N} \left\{ G_j(\boldsymbol{y}) \frac{\partial F_i(\boldsymbol{y})}{\partial y_j} - F_j(\boldsymbol{y}) \frac{\partial G_i(\boldsymbol{y})}{\partial y_j} \right\}. \quad (2.5)$$

(Note that many authors define the commutator with an opposite sign. The commutator as given here is often called *the (-)Jacobi bracket*.)

Proof. From (2.3) and (2.4) we get

$$H = \frac{\partial^2}{\partial s \partial t} \exp(sF) \circ \exp(tG) \circ \exp(-sF) \circ \exp(-tG) \Big|_{s=t=0}.$$

Since $\frac{\partial}{\partial s} \exp(sF) \circ \exp(tG) \circ \exp(-sF) \big|_{s=0} = 0$, this simplifies to

$$H = [F, G] = \frac{\partial^2}{\partial s \partial t} \exp(sF) \circ \exp(tG) \circ \exp(-sF) \Big|_{s=t=0}. \tag{2.6}$$

Neglecting higher order terms in s and t, Euler's integration scheme yields

$$\Psi_{s,F} \circ \Psi_{t,G} \circ \Psi_{-s,F}(\boldsymbol{y}) = \boldsymbol{y} - sF(\boldsymbol{y}) + tG(\boldsymbol{y}_1) + sF(\boldsymbol{y}_2) + \text{h. o. t},$$

where

$$\boldsymbol{y}_1 = \boldsymbol{y} - sF(\boldsymbol{y}) \ , \quad \boldsymbol{y}_2 = \boldsymbol{y} - sF(\boldsymbol{y}) + tG(\boldsymbol{y}_1).$$

Hence (2.6) implies that

$$H_i(\boldsymbol{y}) = \frac{\partial}{\partial s} G_i(\boldsymbol{y}_1) + \frac{\partial}{\partial t} F_i(\boldsymbol{y}_2) \Big|_{s=t=0} = \sum_{j=1}^{N} \left\{ -\frac{\partial G_i(\boldsymbol{y})}{\partial y_j} F_j(\boldsymbol{y}) + \frac{\partial F_i(\boldsymbol{y})}{\partial y_j} G_j(\boldsymbol{y}) \right\}.$$

□

We will now review the most salient properties of the commutator.

Lemma 2.2 If $F, G \in \mathfrak{X}(\mathcal{M})$ then $H = [F, G] \in \mathfrak{X}(\mathcal{M})$.

Proof. The function

$$\rho(t) = \exp(\sqrt{t}F) \circ \exp(\sqrt{t}G) \circ \exp(-\sqrt{t}F) \circ \exp(-\sqrt{t}G)(\boldsymbol{y}_0)$$

is a curve that evolves on $\mathcal{M}$. The tangent defined by this curve is $[F, G](\boldsymbol{y}_0)$. □

Dividing the polygon of Figure 2.2 into infinitesimally small rectangles, we can verify that

Lemma 2.3 Two flows $\Psi_{s,F}$ and $\Psi_{t,G}$ commute if and only if $[F, G] = 0$.

From (2.5) one may prove the following important features of the commutator which should be familiar in the special case (which we will encounter soon again) of a commutator of two matrices.

Lemma 2.4 The commutator of vector fields satisfies the identities

$$\begin{aligned}
[F, G] &= -[G, F] && \text{(skew symmetry)}, && (2.7)\\
[\alpha F, G] &= \alpha [F, G] \ \text{ for } \alpha \in \mathbb{R}, && && (2.8)\\
[F + G, H] &= [F, H] + [G, H] && \text{(bilinearity)}, && (2.9)\\
0 &= [F, [G, H]] + [G, [H, F]] + [H, [F, G]] && \text{(Jacobi's identity)}. && (2.10)
\end{aligned}$$

Example 2.3 Let L_A denote the linear vector field on $\mathbb{R}^N$, given by some matrix A, that is $L_A(\boldsymbol{y}) = A\boldsymbol{y}$. The solution of the linear equation $\boldsymbol{y}' = A\boldsymbol{y}$ is given as

$$\boldsymbol{y}(t) = \sum_{j=0}^{\infty} \frac{(tA)^j}{j!} \boldsymbol{y}_0 = \operatorname{expm}(tA)\boldsymbol{y}_0,$$

where expm denotes the classical matrix exponential. Hence

$$\Psi_{t,L_A}(\boldsymbol{y}_0) \equiv \exp(tL_A)(\boldsymbol{y}_0) = \operatorname{expm}(tA)\boldsymbol{y}_0.$$

(This motivates the name 'exponentiation' for computing the flow.) Now let us compute the commutator of two linear vector fields from (2.5),

$$\begin{aligned}
[L_A, L_B]_i(\boldsymbol{y}) &= \sum_{j,k,l} \left(B_{j,k} y_k \frac{\partial A_{i,l} y_l}{\partial y_j} - A_{j,k} y_k \frac{\partial B_{i,l} y_l}{\partial y_j} \right) \\
&= \sum_{j,k} \left(A_{i,j} B_{j,k} - B_{i,j} A_{j,k} \right) y_k.
\end{aligned}$$

Thus

$$[L_A, L_B] = L_C \quad \text{where} \quad C = AB - BA, \tag{2.11}$$

the familiar definition of a commutator from linear algebra. Note that linear vector fields constitute a complete family of vector fields, closed under commutators and linear combinations, $L_A + L_B = L_{A+B}$ and $\alpha L_A = L_{\alpha A}$.

In applications of Lie group integrators to partial differential equations (PDEs) it is often useful to consider a more general version of (2.5). Let y be a point in a (finite or infinite-dimensional) linear space $\mathcal{M}$. By a vector field F on $\mathcal{M}$ we mean some operator (linear or nonlinear) such that the (ordinary or partial) differential equation

$$\frac{\partial y}{\partial t} = F(y)$$

is well defined. An infinite-dimensional example is a parabolic PDE, where y belongs to some function space on a domain and F is a spatial differentiation operator, e.g. $F(y) = \nabla^2 y$. If F and G are two vector fields on $\mathcal{M}$ then

$$[F, G](y) = \frac{\partial}{\partial s} [F(y + sG(y)) - G(y + sF(y))] \Big|_{s=0}. \tag{2.12}$$

The proof is very similar to that of Lemma 2.1 and it is straightforward to verify that in the finite-dimensional case (2.12) reduces to (2.5). Note that if F and G are linear operators then (2.12) yields immediately $[F, G](y) = F(G(y)) - G(F(y))$, as we saw in Example 2.3.

Example 2.4 Let $\boldsymbol{y} \in \mathbb{R}^N$ and consider the set of all *affine linear vector fields*

$$F_{(A,\boldsymbol{a})}(\boldsymbol{y}) = A\boldsymbol{y} + \boldsymbol{a}$$

where A is an $N \times N$ matrix and $\boldsymbol{a} \in \mathbb{R}^N$. Let us compute the commutator of two such vector fields $F_{(A,\boldsymbol{a})}$ and $F_{(B,\boldsymbol{b})}$. By inserting these vector fields in (2.12) we obtain

$$\begin{aligned}\big[F_{(A,\boldsymbol{a})}, F_{(B,\boldsymbol{b})}\big](\boldsymbol{y}) &= \frac{\partial}{\partial s}\{A(\boldsymbol{y} + s(B\boldsymbol{y} + \boldsymbol{b})) + \boldsymbol{a} - B(\boldsymbol{y} + s(A\boldsymbol{y} + \boldsymbol{a})) - \boldsymbol{b}\}\Big|_{s=0} \\ &= AB\boldsymbol{y} + A\boldsymbol{b} - BA\boldsymbol{y} - B\boldsymbol{a}.\end{aligned}$$

Thus

$$\big[F_{(A,\boldsymbol{a})}, F_{(B,\boldsymbol{b})}\big] = F_{(C,\boldsymbol{c})}, \quad \text{where} \quad (C,\boldsymbol{c}) = (AB - BA, A\boldsymbol{b} - B\boldsymbol{a}). \tag{2.13}$$

We note that the set of all affine linear vector fields is yet another example of a collection of vector fields closed under linear combination and commutation.

2.2. Lie algebras, Lie groups and Lie group actions

A problem, fundamental to numerical analysis of differential equations on manifolds, is *to determine the set of all possible flows that can be obtained by composing a given set of basic flows.* If we restrict the discussion to 'sufficiently small t', important information is provided by the so-called *BCH formula.*

Theorem 2.5. (Baker–Campbell–Hausdorff) For sufficiently small $t \geq 0$ we have

$$\exp(tF) \circ \exp(tG) = \exp(tH),$$

where $H = \mathrm{bch}(F, G)$ can be constructed from iterated commutators of F and G. The first few terms are

$$H = F + G + \tfrac{1}{2}t[F,G] + \tfrac{1}{12}t^2\left([F,[F,G]] + [G,[G,F]]\right) + \mathcal{O}(t^3).$$

Higher order terms can be obtained by recursion (Varadarajan 1984).

Definition 2.5 A *Lie algebra of vector fields* is a collection of vector fields which is closed under linear combination and commutation. In other words, letting $\mathfrak{g}$ denote the Lie algebra,

$$\begin{aligned} B \in \mathfrak{g} \quad &\Rightarrow \quad \alpha B \in \mathfrak{g} \text{ for all } \alpha \in \mathbb{R}. \\ B_1, B_2 \in \mathfrak{g} \quad &\Rightarrow \quad B_1 + B_2, [B_1, B_2] \in \mathfrak{g} \end{aligned}$$

Given a collection of vector fields $\boldsymbol{B} = \{B_1, B_2, \ldots\}$, the least Lie algebra of vector fields containing $\boldsymbol{B}$ is called *the Lie algebra generated by* $\boldsymbol{B}$.

We arrive at the following conclusion. Let $\mathfrak{g}$ be the Lie algebra generated by the set $\boldsymbol{B} = \{B_1, B_2, \ldots\}$ of vector fields. For small t, the combination of flows of vector fields in $\boldsymbol{B}$ yields the flow of a vector field in $\mathfrak{g}$. Furthermore, the flow of any vector field in $\mathfrak{g}$ can, provided $t \geq 0$ is small enough, be approximated arbitrarily well by composing flows of vector fields in $\boldsymbol{B}$. Thus *the Lie algebra contains (for small $t \geq 0$) all the information about composition of flows.*

Until now we have discussed vector fields and flows *on a manifold*. Yet, whether we wish engage in mathematical analysis or produce software for solving ODEs, it is natural to ask *are there important properties of vector fields and flows that can be specified (and possibly programmed) independently of which manifold they are acting upon?* This question will lead us to the abstract definition of Lie algebras and Lie groups. The 'glue' that connects an abstract Lie algebra to concrete vector fields on a manifold is called a *Lie algebra homomorphism*, and abstract Lie groups are connected to flows on a manifold via a *Lie group action*. To elucidate this state of affairs, let us commence with an important example.

Example 2.5 In Example 2.3 we have shown that there exists natural correspondence between $N \times N$ matrices and linear vector fields defined on $\mathbb{R}^N$. We can illustrate this as

$$\begin{aligned} A &\mapsto L_A \\ \alpha A &\mapsto \alpha L_A \\ A + B &\mapsto L_A + L_B \\ AB - BA &\mapsto [L_A, L_B] \end{aligned}$$

A linear subspace of matrices closed under *matrix commutation*, $[A, B] \equiv AB - BA$, is called a *matrix Lie algebra*. The arrow is an example of a Lie algebra homomorphism, a linear map between two Lie algebras which preserves commutators.

Even the computation of flows of linear vector fields and compositions of such flows can be transformed into linear algebra operations. To achieve this, we must specify how a given $N \times N$ matrix P corresponds to a motion on our domain $\mathbb{R}^N$. The simplest possible choice is motions by matrix–vector products. Define thus the map

$$\Lambda(P, \boldsymbol{y}) = P\boldsymbol{y}.$$

Identifying a matrix P with the motion $\Lambda(P, \cdot\,)$ leads to the correspondence

$$\begin{aligned} P &\mapsto \Lambda(P, \cdot\,) \\ \operatorname{expm}(sA) &\mapsto \Lambda(\operatorname{expm}(sA), \cdot\,) = \Psi_{s,L_A} \\ \operatorname{expm}(sA)\operatorname{expm}(tB) &\mapsto \Lambda(\operatorname{expm}(sA)\operatorname{expm}(tB), \cdot\,) = \Psi_{s,L_A} \circ \Psi_{t,L_B}. \end{aligned}$$

Note that the latter of these identifications rely on the associative property of the

map Λ,

$$\Lambda(P, \Lambda(R, \boldsymbol{y})) = \Lambda(PR, \boldsymbol{y}).$$

Motivated by this example, we now proceed to precise mathematical definition of the underlying concepts in a more abstract setting.

Definition 2.6 A *Lie algebra* is a linear space V equipped with a *Lie bracket*, a bilinear, skew-symmetric mapping

$$[\,\cdot\,,\,\cdot\,] : V \times V \to V$$

that obeys identities (2.7–10) from Lemma 2.4.

Definition 2.7 A *Lie algebra homomorphism* is a linear map between two Lie algebras, $\varphi : \mathfrak{g} \to \mathfrak{h}$, satisfying the identity

$$\varphi([v, w]_{\mathfrak{g}}) = [\varphi(v), \varphi(w)]_{\mathfrak{h}}, \qquad v, w \in \mathfrak{g}.$$

An invertible homomorphism is called an *isomorphism.*

Definition 2.8 A *Lie group* is a differential manifold $\mathcal{G}$ equipped with a product $\cdot : \mathcal{G} \times \mathcal{G} \to \mathcal{G}$ satisfying

$p \cdot (q \cdot r) = (p \cdot q) \cdot r \quad \forall\, p, q, r \in \mathcal{G}$	(associativity),
$\exists \boldsymbol{I} \in \mathcal{G}$ such that $\boldsymbol{I} \cdot p = p \cdot \boldsymbol{I} = p \quad \forall\, p \in \mathcal{G}$	(identity element),
$\forall p \in \mathcal{G}\ \exists\ p^{-1} \in \mathcal{G}$ such that $p^{-1} \cdot p = \boldsymbol{I}$	(inverse),
The maps $(p, r) \mapsto p \cdot r$ and $p \mapsto p^{-1}$ are smooth functions	(smoothness).

Definition 2.9 An *action* of a Lie group $\mathcal{G}$ on a manifold $\mathcal{M}$ is a smooth map $\Lambda : \mathcal{G} \times \mathcal{M} \to \mathcal{M}$ satisfying

$$\begin{aligned} \Lambda(\boldsymbol{I}, \boldsymbol{y}) &= \boldsymbol{y} \quad \forall \boldsymbol{y} \in \mathcal{M} \\ \Lambda(p, \Lambda(r, \boldsymbol{y})) &= \Lambda(pr, \boldsymbol{y}) \quad \forall p, r \in \mathcal{G},\ \boldsymbol{y} \in \mathcal{M}. \end{aligned} \tag{2.14}$$

If this relation does hold only in a local sense, for all elements p and r *sufficiently close to the identity* $\boldsymbol{I} \in \mathcal{G}$, we say that Λ is *local action*.

2.3. From finite to infinitesimal and back

Given a set of flows on a domain we can find their vector fields by differentiation. From the discussion above we know that if the flows are closed under composition then the vector fields are closed under Jacobi brackets and linear combinations, hence form a Lie algebra. On the other hand, provided we know the vector fields, we may recover the corresponding flows by integrating differential equations. Similar correspondence between the finite and the infinitesimal is fundamental in abstract

Lie theory. We will review here a number of basic results, interpreting them in the terminology of group actions on a manifold. Let $\mathcal{G}$ be a Lie group, acting on a manifold $\mathcal{M}$ through $\Lambda : \mathcal{G} \times \mathcal{M} \to \mathcal{M}$ and let $\rho(t) \in \mathcal{G}$ be a curve such that $\rho(0) = \boldsymbol{I}$, the identity of $\mathcal{G}$. This curve produces a flow $\Lambda(\rho(t), \cdot\,)$ on $\mathcal{M}$ and by differentiation we find a vector field

$$F(y) = \left.\frac{\mathrm{d}}{\mathrm{d}t}\Lambda(\rho(t), y)\right|_{t=0}.$$

The collection of all such vector fields forms a Lie algebra. Note that in order to produce F it is only necessary to know the tangent to $\rho(t)$ at $t = 0$. Thus the set of all tangents at identity can be endowed with a structure of a Lie algebra.

Definition 2.10 The Lie algebra $\mathfrak{g}$ of a Lie group $\mathcal{G}$ is defined as the linear space of all tangents to $\mathcal{G}$ at the identity $\boldsymbol{I}$. The Lie bracket in $\mathfrak{g}$ is defined as

$$[a, b] = \left.\frac{\partial^2}{\partial s \partial t}\rho(s)\sigma(t)\rho(-s)\right|_{s=t=0} \tag{2.15}$$

where $\rho(s)$ and $\sigma(t)$ are two smooth curves on $\mathcal{G}$ such that $\rho(0) = \sigma(0) = \boldsymbol{I}$, $\rho'(0) = a$ and $\sigma'(0) = b$.

Note that the bracket defined in (2.15) is essentially the same as (2.6). From this it is straightforward to verify that the correspondence between elements in $\mathfrak{g}$ and vector fields on $\mathcal{M}$ is an algebra homomorphism.

Lemma 2.6 Let $\lambda_* : \mathfrak{g} \to \mathfrak{X}(\mathcal{M})$ be defined as

$$\lambda_*(a)(y) = \left.\frac{\mathrm{d}}{\mathrm{d}s}\Lambda(\rho(s), y)\right|_{s=0}, \tag{2.16}$$

where $\rho(s)$ is a curve in $\mathcal{G}$ such that $\rho(0) = \boldsymbol{I}$ and $\rho'(0) = a$. Then λ_* is a linear map between Lie algebras such that

$$[a, b]_{\mathfrak{g}} = [\lambda_*(a), \lambda_*(b)]_{\mathfrak{X}(\mathcal{M})}.$$

Thus we can go from the finite to the infinitesimal (from groups and group actions to algebras and algebra homomorphisms) by differentiation. To do the opposite and move from the infinitesimal to the finite, we must somehow compute flows of vector fields. A discussion of this process leads to the general definition of the exponential mapping.

Suppose $\mathcal{G}$ is a Lie group with Lie algebra $\mathfrak{g}$ and let $\Lambda : \mathcal{G} \times \mathcal{M} \to \mathcal{M}$ be a group action. A given fixed element $a \in \mathfrak{g}$ corresponds to a vector field $\lambda_*(a) \in \mathfrak{X}(\mathcal{M})$. We want to compute the flow of this field. Let us first assume for simplicity that $\mathcal{G}$ is a matrix group.

Lemma 2.7 For a fixed $A \in \mathfrak{g}$ the flow of $\lambda_*(A)$, i.e. the solution of

$$y'(t) = \lambda_*(A)(y(t)) \quad \text{for} \quad y(0) = y_0 \in \mathcal{M}$$

can be expressed in the form

$$y(t) = \Lambda(S(t), y_0)$$

where the curve $S(t) \in \mathcal{G}$ satisfies the matrix differential equation

$$S'(t) = AS, \quad t \geq 0, \qquad S(0) = \boldsymbol{I},$$

which has the explicit solution

$$S(t) = \operatorname{expm}(tA), \qquad t \geq 0.$$

Proof. We assume that $y(t) = \Lambda(S(t), y_0)$. Differentiation results in $y'(t) = \partial_1\Lambda(S'(t), y_0)$, where ∂_1 is the derivative with respect to the first argument. On the other hand, to compute $\lambda_*(A)$ we pick a curve $R(s) \in \mathcal{G}$ such that $R'(0) = A$ and $R(0) = \boldsymbol{I}$. From (2.14) we get

$$\begin{aligned} y'(t) &= \lambda_*(A)(y(t)) = \frac{\partial}{\partial s}\Lambda(R(s), \Lambda(S(t), y_0))\Big|_{s=0} \\ &= \frac{\partial}{\partial s}\Lambda(R(s)S(t), y_0)\Big|_{s=0} = \partial_1\Lambda(R'(0)S(t), y_0) = \partial_1\Lambda(AS(t), y_0). \end{aligned}$$

Thus

$$S'(t) = AS(t).$$

Obviously $S(0) = \boldsymbol{I}$. The explicit solution is easily verified. □

Lemma 2.7 holds unaltered for a general group $\mathcal{G}$ if we define the product of an element of an algebra $a \in \mathfrak{g}$ with an element of a group $\sigma \in \mathcal{G}$ as

$$a\sigma \equiv \frac{\mathrm{d}}{\mathrm{d}s}\rho(s)\sigma\Big|_{s=0}, \tag{2.17}$$

where $\rho(s) \in \mathcal{G}$ is a smooth curve such that $\rho'(0) = a$ and $\rho(0) = \boldsymbol{I}$. We also define the exponential mapping so that the flow of $\lambda_*(a)$ is of the form $\Lambda(\exp(ta), \cdot)$:

Definition 2.11 Let $\mathcal{G}$ be a Lie group and $\mathfrak{g}$ its Lie algebra. The exponential mapping $\exp : \mathfrak{g} \to \mathcal{G}$ is defined as $\exp(a) = \sigma(1)$ where $\sigma(t) \in \mathcal{G}$ satisfies the differential equation

$$\sigma'(t) = a\sigma(t), \qquad \sigma(0) = \boldsymbol{I}.$$

These definitions lead to the following general form of Lemma 2.7.

Theorem 2.8 Let $\Lambda : \mathcal{G} \times \mathcal{M} \to \mathcal{M}$ be a group action and $\lambda_* : \mathfrak{g} \to \mathfrak{X}(\mathcal{M})$ the corresponding Lie algebra homomorphism (2.16). For any $a \in \mathfrak{g}$ the flow of the vector field $F = \lambda_*(a)$, i.e. the solution of the equation

$$y'(t) = F(y(t)) = \lambda_*(a)(y(t)), \quad t \geq 0, \qquad y(0) = y_0 \in \mathcal{M},$$

is given as

$$y(t) = \Lambda(\exp(ta), y_0).$$

Let us at this point make a small detour to introduce the *adjoint representation* which is fundamental in many contexts. By splitting (2.15) into two smaller steps, we obtain:

Definition 2.12 Let $p \in \mathcal{G}$ and let $\sigma(t)$ be a smooth curve on $\mathcal{G}$ such that $\sigma(0) = \boldsymbol{I}$ and $\sigma'(0) = b \in \mathfrak{g}$. The *adjoint representation* is defined as

$$\mathrm{Ad}_p(b) = \left. \frac{\mathrm{d}}{\mathrm{d}t} p\sigma(t)p^{-1} \right|_{t=0} \tag{2.18}$$

The derivative of Ad with respect to the first argument is denoted ad. Let $\rho(s)$ be a smooth curve on $\mathcal{G}$ such that $\rho(0) = \boldsymbol{I}$ and $\rho'(0) = a$. Definition 2.10 now yields:

$$\mathrm{ad}_a(b) \equiv \left. \frac{\mathrm{d}}{\mathrm{d}s} \mathrm{Ad}_{\rho(s)}(b) \right|_{s=0} = [a, b]. \tag{2.19}$$

The following formulae show that Ad is both a linear group action (of $\mathcal{G}$ on $\mathfrak{g}$) and also that for a fixed argument p it is a Lie-algebra isomorphism of $\mathfrak{g}$ onto itself:

$$\mathrm{Ad}_p(a) \in \mathfrak{g} \qquad \text{for all } p \in \mathcal{G},\ a \in \mathfrak{g}. \tag{2.20}$$

$$\mathrm{Ad}_p \circ \mathrm{Ad}_q = \mathrm{Ad}_{pq} \tag{2.21}$$

$$\mathrm{Ad}_p(a + b) = \mathrm{Ad}_p(a) + \mathrm{Ad}_p(b) \tag{2.22}$$

$$\mathrm{Ad}_p([a, b]) = [\mathrm{Ad}_p(a), \mathrm{Ad}_p(b)] \tag{2.23}$$

Note that according to (2.22) both Ad_p and ad_a are linear in their second argument, hence they may be regarded as matrices acting on the linear space $\mathfrak{g}$. This gives meaning to the following important formula relating Ad, ad and the exponential mapping:

$$\mathrm{Ad}_{\exp(a)} = \mathrm{expm}(\mathrm{ad}_a). \tag{2.24}$$

2.4. Differential equations on manifolds

We wish to return to general differential equations on manifolds, as given in (2.1). In order to construct and implement numerical solvers for this equation, we require a concrete way of representing the vector field $F(y)$. Herewith we describe a very general approach presented in (Munthe-Kaas and Zanna 1997).

Assumption 2.1 Given a differential equation $y'(t) = F(t, y)$ on a manifold $\mathcal{M}$, we assume the existence of a Lie algebra $\mathfrak{g}$, a Lie algebra homomorphism $\lambda_* : \mathfrak{g} \to \mathfrak{X}(\mathcal{M})$ and a function $a : t \times \mathcal{M} \to \mathfrak{g}$ such that the equation can be written in the form

$$y'(t) = \lambda_*(a(t, y))(y) \tag{2.25}$$

If λ_* is known from the context and no confusion is likely, we will usually write equation (2.25) in the shorthand form

$$y'(t) = a(t, y)y.$$

In many important examples the function a depends only on t and not on y. These equations, $y'(t) = a(t)y$, are called equations of Lie type, or *linear-type* Lie-group equations. Some of the algorithms to be presented in the sequel are aimed at the general equation (2.25), while others are aimed at exploiting the special structure of linear equations.

Given an equation to be solved, an important challenge is to find a 'good' homomorphism λ_*. It is not difficult to see that *any* differential equation can be written in the form (2.25). We might for example let $\mathfrak{g} = \mathfrak{X}(\mathcal{M})$ and choose λ_* as the identity map, which would trivially render any equation in this form. However, in order to construct practical solution algorithms we need to make some additional assumptions about $\mathfrak{g}$. We will usually assume that either *all the elements* of $\mathfrak{g}$, or at least *a particular basis* of $\mathfrak{g}$ can be exponentiated efficiently. To achieve this, one might embed $\mathcal{M}$ in a linear space $\mathbb{R}^n$, and let $\mathfrak{g}$ be the set of all translations on $\mathbb{R}^n$, since translations are trivial to exponentiate. This choise will, however, fail to capture much of the structure of the equations to be solved. In fact we will see that for this choice most of our numerical solution techniques will reduce to classical Runge–Kutta methods. The task of finding a 'good action' is in many respects similar to the task of finding a good preconditioner in the theory of iterative methods for solving linear algebraic equations $A\boldsymbol{x} = \boldsymbol{b}$. In both cases we want to find some approximation to our original equation which is both simple to solve and which captures some important structural feature of the equation. The two extreme choices, on the one hand $\mathfrak{g} = \mathfrak{X}(\mathcal{M})$ and on the other hand $\mathfrak{g}$ as a set of all translations on $\mathbb{R}^n$, are similar to preconditioning a linear system with the matrix A itself or on the other hand choosing the identity matrix as preconditioner.

Let us now examine briefly a number of examples of equations presented in the form of Assumption 2.1. A useful approach to find a good action is the following:

1 Given a differential equation written in a familiar form, look at the terms and see if it is possible to find related equations which are simpler to integrate.

2 Check that the family of simpler equations forms a Lie algebra. Find a suitable representation $\mathfrak{g}$ for the algebra and the corresponding homomorphism λ_*.

3 Check that the original equation can be written in the form (2.25). This is not possible only in the case where there exist some points on $\mathcal{M}$ where the

vector fields in the Lie algebra do not generate the direction of the original equation. In this case one must search for a larger Lie algebra.

Example 2.6 (Orthogonal matrix flows) Matrix differential equations of the form

$$Y' = A(t,Y)Y, \quad t \geq 0, \qquad Y(0) = Y_0 \in \mathrm{O}(N), \tag{2.26}$$

where $A : \mathbb{R}^+ \times \mathrm{O}(N) \to \mathfrak{so}(N)$, are called *orthogonal flows* – we have already encountered such a flow in (1.3). It is well known that the exact solution $Y(t)$ is an orthogonal matrix for all $t \geq 0$ (Dieci, Russell and van Vleck 1994). A simpler family of equations is given by

$$Y' = CY, \quad t \geq 0, \qquad Y(0) = Y_0 \in \mathrm{O}(N), \tag{2.27}$$

where C is any constant matrix in $\mathfrak{so}(N)$. The solution of (2.27) is given as

$$Y(t) = \operatorname{expm}(tC)Y_0, \quad t \geq 0.$$

From (2.11) we find that if $F(Y) = CY$ and $G(Y) = DY$ then $[F,G](Y) = (CD - DC)Y$. Thus, the family of simple vector fields is a Lie algebra isomorphic to $\mathfrak{so}(N)$. We have $\lambda_*(C)(Y) = CY$, hence (2.26) is in the form (2.25) if $a(t,Y) = A(t,Y)$. Note that also the flow of (2.27) is orthogonal, hence any numerical method based on the composition of such flows will yield a solution which retains orthogonality.

Example 2.7 (Isospectral matrix flows) We have already encountered such flows in Section 1, in (1.1). With slightly greater generality, we write them in the form

$$Y' = B(t,Y)Y - YB(t,Y), \quad t \geq 0, \qquad Y(0) = Y_0 \in \mathrm{S}_N, \tag{2.28}$$

where $B : \mathbb{R}^+ \times \mathrm{S}_N \to \mathfrak{so}(N)$. The analytical solution $Y(t)$ is a family of matrices with eigenvalues invariant under the flow. A simpler family of equations is given by

$$Y'(t) = CY - YC, \quad t \geq 0, \qquad Y(0) = Y_0 \in \mathrm{S}_N, \tag{2.29}$$

where again $C \in \mathfrak{so}(N)$ is constant. The solution of this equation is given explicitly in the form

$$Y(t) = \operatorname{expm}(tC)Y_0 \operatorname{expm}(-tC), \quad t \geq 0. \tag{2.30}$$

We may now proceed, exactly like in the previous example, to find the brackets of such vector fields. An alternative route, already anticipated in (1.2), is to note that the basic flows in (2.30) are given by the orthogonal matrix group $\mathrm{O}(N)$ acting on $Y \in \mathrm{S}_N$. We have the action $\Lambda : \mathrm{O}(N) \times \mathrm{S}_N \to \mathrm{S}_N$ given by

$$\Lambda(Q,X) = QXQ^{\mathrm{T}} \quad \text{for any} \quad Q \in \mathrm{O}(N).$$

By differentiation, similarly to Lemma 2.6, we find that $\lambda_*(C)(Y) = CY - YC$. Hence (2.28) is in the form (2.25) if $a(t,Y) = B(t,Y)$. The basic flow (2.30) is isospectral since it is a similarity transformation, hence also any numerical method based on this flow is automatically isospectral.

Note that, essentially, these two examples are identical, except for the action involved. Hence in a computer implementation we might reuse the implementation of the Lie algebra and the Lie group, while simply changing the action: there is no need to develop separate computational approach to orthogonal and isospectral flows! This illustrates the importance of working with abstractly-defined groups and algebras rather than tying these concepts to flows and vector fields on particular manifolds.

Example 2.8 (ODEs on $\mathbb{R}^N$) Consider an ordinary differential equation in the familiar form required by all classical numerical integrators,

$$\boldsymbol{y}'(t) = \boldsymbol{g}(t, \boldsymbol{y}), \quad t \geq 0, \qquad \boldsymbol{y}(0) = \boldsymbol{y}_0 \in \mathbb{R}^N, \tag{2.31}$$

where $\boldsymbol{g} : \mathbb{R} \times \mathbb{R}^N \to \mathbb{R}^N$. A simple family of basic flows is given by translations, e.g.

$$\boldsymbol{z}' = \boldsymbol{a},$$

where $\boldsymbol{a} \in \mathbb{R}^N$ is constant. The algebra of these flows can be identified with $\mathbb{R}^N$ and, since translations commute, we obtain the trivial bracket $[\boldsymbol{a}, \boldsymbol{b}] = 0$ for all $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^N$. The algebra homomorphism is given here simply as the identity map, hence the equation is in the form (2.25) if $a(t, \boldsymbol{y}) = \boldsymbol{g}(t, \boldsymbol{y})$. This choice yields nothing new compared to classical integration schemes.

A more interesting choice is choosing linear vector fields as basic flows,

$$\boldsymbol{z}' = A\boldsymbol{z}.$$

We already know from Example (2.3) that the Lie algebra of these is $\mathfrak{gl}(N)$, the set of all $N \times N$ matrices with the usual matrix bracket. However, these flows cannot produce everywhere all possible tangent directions. Indeed, at $\boldsymbol{y} = \boldsymbol{0}$ these flows cannot produce *any* nonzero tangent. Therefore, using this action, it is generally not possible to write (2.31) in the form (2.25).

We might finally consider a combination of translations and linear maps, i.e. the set of all *affine* linear maps given by equations of the form

$$\boldsymbol{z}' = A\boldsymbol{z} + \boldsymbol{b} \qquad \text{for} \qquad A \in \mathfrak{gl}(N), \quad \boldsymbol{b} \in \mathbb{R}^N.$$

The Lie algebra can be identified with all pairs $(A, \boldsymbol{b})$ where A is a matrix and $\boldsymbol{b}$ a vector and we have already seen in Example (2.4) that the bracket is given as $[(A, \boldsymbol{b}), (C, \boldsymbol{d})] = (AC - CA, A\boldsymbol{d} - C\boldsymbol{b})$. The algebra homomorphism is

$$\lambda_*(A, \boldsymbol{b})(\boldsymbol{y}) = A\boldsymbol{y} + \boldsymbol{b}.$$

In this situation there are many possible choices of a function $a(t, \boldsymbol{y})$ such that (2.31) acquires the form (2.25). No matter what we pick as the matrix part, it is always possible to adjust the vector so that

$$\lambda_*(a(t, \boldsymbol{y}))(\boldsymbol{y}) = \boldsymbol{g}(t, \boldsymbol{y}).$$

A natural choice is *local linearization,* namely letting the matrix part be the Jacobian of $\boldsymbol{g}$ at $\boldsymbol{y}$,

$$J\boldsymbol{g}(t,\boldsymbol{y})_{i,j} = \frac{\partial \boldsymbol{g}_i(t,\boldsymbol{y})}{\partial y_j}.$$

The resulting $a : \mathbb{R} \times \mathcal{M} \to \mathfrak{g}$ is

$$a(t,\boldsymbol{y}) = (J\boldsymbol{g}(t,\boldsymbol{y}),\, \boldsymbol{g}(t,\boldsymbol{y}) - J\boldsymbol{g}(t,\boldsymbol{y})\boldsymbol{y})\,.$$

Possible advantages of using affine motions to advance the solution of stiff ODEs are a subject of ongoing research.

Example 2.9. (ODEs on a sphere) Many mechanical problems involve rotations in a 3-space. In Appendix B we list useful formulae for fast computations in SO(3). As a simple example we will consider a motion on the surface of a sphere.

$$\boldsymbol{y}'(t) = \boldsymbol{a}(\boldsymbol{y}(t)) \times \boldsymbol{y}(t), \tag{2.32}$$

where $\boldsymbol{a}, \boldsymbol{y} \in \mathbb{R}^3$. If $\|\boldsymbol{y}(0)\|_2 = 1$ then $\boldsymbol{y}(t)$ evolves on the unit sphere $\mathbb{S}^2 \subset \mathbb{R}^3$. A simpler system that also evolves on $\mathbb{S}^2$ is given by

$$\boldsymbol{y}'(t) = \boldsymbol{c} \times \boldsymbol{y}(t), \tag{2.33}$$

for any fixed vector $\boldsymbol{c}$. To compute the commutator of two equations of the form (2.33), we employ the *hat map* (B.1) taking a 3-vector $\boldsymbol{c}$ to a 3×3 matrix $\widehat{\boldsymbol{c}}$, such that $\boldsymbol{c}\times\boldsymbol{y} = \widehat{\boldsymbol{c}}\boldsymbol{y}$. Using ($B$.2) we see that the commutator of the vector fields $\boldsymbol{c}\times\boldsymbol{y}$ and $\boldsymbol{d}\times\boldsymbol{y}$ is given by $(\boldsymbol{c}\times\boldsymbol{d})\times\boldsymbol{y}$. Thus the Lie algebra in this example may be identified with $(\mathbb{R}^3, \times)$, the real 3-space with the Lie bracket

$$[\boldsymbol{c}, \boldsymbol{d}] = \boldsymbol{c}\times\boldsymbol{d}$$

given by a vector product. The simplified equation (2.33) is a matrix equation $\boldsymbol{y}' = \widehat{\boldsymbol{c}}\boldsymbol{y}$ with solution

$$\boldsymbol{y}(t) = \operatorname{expm}(t\widehat{\boldsymbol{c}})\boldsymbol{y}_0,$$

where $\operatorname{expm}(t\widehat{\boldsymbol{c}})$ can be computed fast using the Rodrigues formula (B.10).

Example 2.10. (Parabolic PDEs) The final example in this section is chosen to illustrate the diversity of problems that may be tackled with the machinery of Lie-group methods. This example involves infinite-dimensional Lie algebras, a topic that is technically more demanding than the finite-dimensional case. There are several ways to circumvent the mathematical problems, we may either discuss the problem after it has been discretised in space (and has become finite dimensional), or we may plunge straight ahead using the available techniques, disregarding possible mathematical difficulties. We will henceforth follow the latter approach, being aware that resulting algorithms must be verified by other means.

Suppose that we wish to integrate a parabolic PDE with coefficients varying in space and time, e.g. the *heat equation*

$$\frac{\partial u(t,\boldsymbol{x})}{\partial t} = \nabla \cdot (\mu(\boldsymbol{x})\nabla u(t,\boldsymbol{x}))\,,$$

where u is the temperature and μ the heat conductivity of the material. With greater generality, consider equations of the form

$$\frac{\partial u(t,\boldsymbol{x})}{\partial t} = \mathcal{L}(u), \tag{2.34}$$

where $\mathcal{L}$ is an elliptic operator. To simplify the discussion, we wish to neglect boundary conditions, so suppose that u is defined on the unit square with periodic boundary conditions. Thus, u should be thought of as a 'point' on the infinite-dimensional manifold $\mathcal{M} = C^\infty(\mathbb{T})$, the collection of all smooth functions on a torus. It is well known that (classical) explicit integrators for parabolic PDEs are typically *stiff* and stability analysis leads to severe step size restriction: $\Delta t < c(\Delta x)^2$ for explicit finite-difference methods. A family of simpler equations, which can be solved exactly, explicitly and very efficiently with the *Fast Fourier Transform (FFT)* is the set of all parabolic equations with constant coefficients of the form

$$\frac{\partial u(t,\boldsymbol{x})}{\partial t} = \bar{\mu}\nabla^2 u(t,\boldsymbol{x}),$$

where $\bar{\mu}$ is constant. However, just like in Example 2.8, these equations cannot move an arbitrary point $u \in \mathcal{M}$ in an arbitrary direction. (In other words, they do not define a *transitive action* on $\mathcal{M}$). Hence, we enlarge the family of simplified equations by adding an inhomogeneous term,

$$\frac{\partial u(t,\boldsymbol{x})}{\partial t} = \bar{\mu}\nabla^2 u(t,\boldsymbol{x}) + b(x), \tag{2.35}$$

where $b \in C^\infty(\mathbb{T})$. Also this equation is easy to solve using FFTs. Letting the flows of (2.35) define our group action on $\mathcal{M}$, we see that the corresponding Lie algebra $\mathfrak{g}$ can be identified with pairs $(\bar{\mu}, b)$. The Lie-algebra action is given by

$$\lambda_*((\bar{\mu}, b))(u) = \bar{\mu}\nabla^2 u + b.$$

Using this action we see that any equation of the form (2.34) can be cast into the form (2.25) by choosing the function $a : \mathbb{R}\times\mathcal{M} \to \mathfrak{g}$ as

$$a(u) = (\bar{\mu}, \mathcal{L}(u) - \bar{\mu}\nabla^2 u) \tag{2.36}$$

for some choice of $\bar{\mu}$. For example, if $\mathcal{L}(u) = \nabla \cdot (\mu(\boldsymbol{x})\nabla u(t,\boldsymbol{x}))$, we would let $\bar{\mu}$ be some averaged value of $\mu(x)$.

In order to define the whole structure of $\mathfrak{g}$, we need to determine the Lie bracket. Let F and G be two vector fields on $\mathcal{M}$ defined at a point u as

$$\begin{aligned} F(u) &= \bar{\mu}\nabla^2 u + f(x) \\ G(u) &= \bar{\nu}\nabla^2 u + g(x). \end{aligned}$$

Using (2.12), we obtain

$$\begin{aligned} [F,G](u) &= \frac{\partial}{\partial s}\left[\bar{\mu}\nabla^2(u + s(\bar{\nu}\nabla^2 u + g)) + f - \bar{\nu}\nabla^2(u + s(\bar{\mu}\nabla^2 u + f)) - g\right]\Big|_{s=0} \\ &= 0 \cdot \nabla^2 u + \bar{\mu}\nabla^2 g - \bar{\nu}\nabla^2 f. \end{aligned}$$

Thus, the bracket on $\mathfrak{g} = \mathbb{R} \times C^\infty(T)$ is

$$[(\bar{\mu}, f), (\bar{\nu}, \mathfrak{g})] = (0, \bar{\mu}\nabla^2 g - \bar{\nu}\nabla^2 f).$$

It is interesting to note that also the bracket can be computed efficiently using FFTs.

This type of equations is considered in greater detail and various numerical examples are given in (Munthe-Kaas and Lodden n.d.). It turns out that, at least in some cases, it is possible to construct explicit integrators based on this action that are not subjected to any step size restriction involving the spatial discretization Δx. Thus, the methods are stable regardless of how spatial discretization is chosen. This is a topic of ongoing research, and many aspects of these integrators are currently incompletely understood.

We will return to more examples and to numerical experiments in Section 11. Before discussing numerical algorithms, we need to study some important properties of the exponential map.

2.5. Much Ado about something

In this section we have emphasised a general view of differential equations on manifolds, based on Lie groups *acting* on manifolds. This outlook is important not just for the sake of mathematical beauty or abstraction but, as we hope to have persuaded the reader, also from the point of view of applications and computation. However, insofar as clarity of exposition is concerned, it is often better to restrict ourselves to the far simpler, familiar and more intuitive matrix theory.

In fact, it turns out that for all the algorithms that we present in this paper it is quite straightforward to translate results derived in matrix setting to the more general setting of local Lie-group actions on some domain. The following theorem of Ado underscores the importance of studying the matrix case (Olver 1995, Varadarajan 1984).

Theorem 2.9. (Ado's theorem) Any finite-dimensional Lie algebra is isomorphic to a subalgebra of the matrix algebra $\mathfrak{gl}(N)$ for some $N \geq 1$.

Although similar result does not hold for all finite-dimensional Lie groups, it is true that whenever we are given a finite-dimensional *local* Lie group action, we can always find an equivalent local action by a matrix Lie group. More information on these topics can be found in (Olver 1995) and (Varadarajan 1984).

Aware of the danger of rules-of-a-thumb being mathematically imprecise, it is nonetheless worthwhile to summarise these results as follows.

For practically any concept in general Lie theory there exists a corresponding concept within matrix Lie theory. Vice versa, practically any result which holds in the matrix case remains valid within the general Lie theory.

The above remark is even more important in numerical context, since computation always takes place in a finite-dimensional setting. Even if the original equation evolves on an infinite-dimensional manifold, its practical computation must sooner-or-later involve a discretization to a finite-dimensional formulation.

At this point it is time to wake up the readers who have surfed through the general theory in a relaxed manner. We will restate the main definitions in the concrete form in which they appear within the matrix theory. It is worthwhile to compare these definitions to the corresponding general definitions above.

Definition 2.13 A real *matrix Lie group* is a smooth subset $\mathcal{G} \subseteq \mathbb{R}^{n\times n}$, closed under matrix products and matrix inversion. We let $\boldsymbol{I} \in \mathcal{G}$ denote the identity matrix.

Definition 2.14 The Lie algebra $\mathfrak{g}$ of a matrix Lie group $\mathcal{G}$ is the linear subspace $\mathfrak{g} \subseteq \mathbb{R}^{n\times n}$ consisting of all matrices of the form

$$\mathfrak{g} = \left\{ A \in \mathbb{R}^{n\times n} \,:\, A = \left.\frac{\mathrm{d}\rho(s)}{\mathrm{d}s}\right|_{s=0} \right\},$$

where $\rho(s) \in \mathcal{G}$ is a smooth curve such that $\rho(0) = \boldsymbol{I}$. The space $\mathfrak{g}$ is closed under matrix additions, scalar multiplication and the matrix commutator

$$[A, B] = AB - BA. \tag{2.37}$$

Complex matrix Lie groups and algebras are defined similarly.

The time has come to introduce some of the main *dramatis personae* of our survey, concrete examples of Lie groups and algebras. In each case it is easy to verify that all the axioms of a group or an algebra, as the case might be, are fulfilled, and we leave this as an exercise to the reader.

- The set of all real $N \times N$ nonsingular matrices is a (multiplicative) Lie group, the *general linear group* $\mathrm{GL}(N)$. The corresponding Lie algebra is the set $\mathbb{R}^{N\times N}$ of all $N \times N$ real matrices which, in keeping with our terminology, we denote by $\mathfrak{gl}(N)$.

  The general linear group and algebra can be defined over other fields than

$\mathbb{R}$, in which case we communicate this in the second argument. For example, $\mathrm{GL}(N;\mathbb{C})$ consists of all nonsingular $N \times N$ complex matrices.

- All members of $\mathrm{GL}(N)$ with unit determinant form the *special linear group* $\mathrm{SL}(N)$. Its Lie algebra, $\mathfrak{sl}(N)$, consists of all matrices in $\mathfrak{gl}(N)$ with zero trace.

- $N \times N$ real orthogonal matrices form the *orthogonal group* $\mathrm{O}(N)$, whose Lie algebra $\mathfrak{so}(N)$ consists of $N \times N$ skew-symmetric matrices.

  The set $\mathrm{SO}(N) = \mathrm{SL}(N) \cap \mathrm{O}(N)$, consisting of $N \times N$ real orthogonal matrices with unit determinant, is the *special orthogonal group.* Its Lie algebra is $\mathfrak{so}(N)$, whch we have just encountered. This is not contradictory: we never claimed that two different Lie groups must have different Lie algebras! As a matter of fact, more is true: If $\mathcal{G}$ is a Lie group and $\mathcal{G}_{\boldsymbol{Id}}$ its connected component such that $\boldsymbol{I} \in \mathcal{G}_{\boldsymbol{Id}}$ (precisely the situation with $\mathrm{O}(N)$ and $\mathrm{SO}(N)$, respectively) then they produce the same Lie algebra.

- The set of all $(2N) \times (2N)$ real matrices X such that $XJX^{\mathrm{T}} = J$, where

$$J = \begin{bmatrix} \boldsymbol{O}_N & \boldsymbol{I}_N \\ -\boldsymbol{I}_N & \boldsymbol{O}_N \end{bmatrix},$$

  is the *symplectic group* and denoted by $\mathrm{Sp}(N)$. (The Jacobian of the flow of a Hamiltonian ODE system evolves in $\mathrm{Sp}(N)$.) The corresponding Lie algebra, $\mathfrak{sp}(N)$, consists of $F \in \mathfrak{gl}(2N)$ such that $FJ + JF^{\mathrm{T}} = \boldsymbol{O}$.

- All the matrices $X \in \mathrm{SL}(4)$ such that $XJX^{\mathrm{T}} = J$, where

$$J = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix},$$

  form the *Lorenz group* $\mathrm{SO}(3,1)$. Its Lie algebra $\mathfrak{so}(3,1)$ is made out of all $F \in \mathfrak{gl}(4)$ such that $FJ + JF^{\mathrm{T}} = \boldsymbol{O}$.

- As an example of complex Lie groups, we mention the *unitary group* $\mathrm{U}(N;\mathbb{C})$ of all $N \times N$ complex unitary matrices: $X \in \mathrm{U}(N;\mathbb{C})$ if and only if $XX^{\mathrm{H}} = \boldsymbol{I}$. The Lie algebra corresponding to $\mathrm{U}(N;\mathbb{C})$ is the set $\mathfrak{u}(N;\mathbb{C})$ of all skew-Hermitian matrices in $\mathfrak{gl}(N;\mathbb{C})$.

  The unitary group should not be confused with $\mathrm{O}(N;\mathbb{C})$, the group of all $N \times N$ complex orthogonal matrices, whose Lie algebra is $\mathfrak{so}(N;\mathbb{C})$.

- Similarly to the case of $\mathrm{O}(N)$ and $\mathrm{SO}(N)$, we obtain the *special unitary group* intersecting $\mathrm{U}(N;\mathbb{C})$ with $\mathrm{SL}(N;\mathbb{C})$. Its Lie algebra, $\mathfrak{su}(N;\mathbb{C})$, is made out of $N \times N$ complex skew-Hermitian matrices with zero trace.

Definition 2.15 A differential equation on a matrix Lie group is an equation of the form

$$Y' = A(t, Y)Y, \quad t \geq 0, \qquad Y(0) \in \mathcal{G}, \tag{2.38}$$

where $A : \mathbb{R} \times \mathcal{G} \to \mathfrak{g}$ and AY is the usual matrix product between $A \in g$ and $Y \in \mathcal{G}$.

The reader may verify that this is the special case of the general form of a differential equation on a manifold, given in Assumption 2.1, where $\mathcal{M} = \mathcal{G}$ is a matrix Lie group and the action Λ is taken to be the left (matrix) multiplication in $\mathcal{G}$,

$$\Lambda(R, Y) = RY.$$

Using (2.16) we find

$$\lambda_*(A)(Y) = AY.$$

Since $\mathfrak{g}$ is defined as the collection of *all* tangent directions at $\boldsymbol{I} \in \mathcal{G}$ and matrix multiplication by Y is an invertible mapping, we see that any tangent at Y can be written in the form AY and all differential equations on $\mathcal{G}$ can be written in the form (2.38).

Definition 2.16 The exponential mapping $\operatorname{expm} : \mathfrak{g} \to \mathcal{G}$ is defined as

$$\operatorname{expm}(A) = \sum_{j=0}^{\infty} \frac{A^j}{j!}. \tag{2.39}$$

Note that $\operatorname{expm}(\boldsymbol{O}) = \boldsymbol{I}$, and that for A sufficiently near $\boldsymbol{O} \in \mathfrak{g}$ the exponential has a smooth inverse given by the matrix logarithm $\operatorname{logm} : \mathcal{G} \to \mathfrak{g}$.

Definition 2.17 The *adjoint representation*, Ad, and its derivative, ad, are defined as

$$\operatorname{Ad}_P(A) = PAP^{-1} \tag{2.40}$$

$$\operatorname{ad}_A(B) = AB - BA = [A, B]. \tag{2.41}$$

Table 2.1. *Correspondence between the matrix case and general Lie theory*

Matrix case	**General case**
AY, $A \in \mathfrak{g}$, $Y \in \mathcal{G}$	$\lambda_*(a)(y)$, $a \in \mathfrak{g}$, $y \in \mathcal{M}$
$Y' = A(Y, t)Y$	$y' = \lambda_*(a(y, t))(y)$
RY, $R, Y \in \mathcal{G}$	$\Lambda(r, y)$, $r \in \mathcal{G}$, $y \in \mathcal{M}$
$\operatorname{expm}(A) = \sum_{j=0}^{\infty} A^j/j!$	Definition 2.11
$[A, B] = AB - BA$	Definition 2.10
PAP^{-1}, $P \in \mathcal{G}$, $A \in \mathfrak{g}$	Definition 2.12

It is easy to verify that (2.20–23) hold in the matrix case, while (2.24) is far from being an obvious identity even in that case.

Table 2.1 summarises the correspondence between the matrix case and the general case.

2.6. The differential of the exponential map

We have introduced the exponential mapping $\exp : \mathfrak{g} \to \mathcal{G}$ as the fundamental solution of the equation $y' = ay$, or more explicitly in the matrix case as $\exp A = \sum_{j=0}^{\infty} A^j/j!$. For development of numerical algorithms it is essential to discuss the derivative of the exponential map. This will first be used to deduce an infinitesimal version of the BCH formula and in the following chapters to derive a variety of different numerical algorithms for differential equations on Lie groups.

To simplify the exposition we will restrict the proofs to matrix theory. The results are, however, valid in an abstract setting. The reader is referred to (Varadarajan 1984) for proofs in a general context.

Given a scalar function $a(t) \in \mathbb{R}$, the derivative of the exponential is given as $\mathrm{d}\exp(a(t))/\mathrm{d}t = a'(t)\exp(a(t))$. One might have hoped for a similar result also when $A(t)$ is a matrix. However, due to the fact that in general $[A, A'] \neq \boldsymbol{O}$, this is not the case and we must correct this formula. Note that $\mathrm{d}\exp(A(t))/\mathrm{d}t$ must be tangent to $\mathcal{G}$ in the point $P(t) = \exp(A(t))$. We have seen in Section 2.5 that *any* such tangent can be written as $C(t)P(t)$, where $C(t) \in \mathfrak{g}$. Furthermore, general properties of $\mathrm{d}/\mathrm{d}t$ imply that $B(t)$ must depend only on $A(t)$ and $A'(t)$, and that the dependence on A' is linear. This function is denoted by dexp.

Definition 2.18 The differential of the exponential mapping is defined as the 'right trivialised' tangent of the exponential map, i.e. as a function $\mathrm{dexp} : \mathfrak{g}\times\mathfrak{g} \to \mathfrak{g}$ such that

$$\frac{\mathrm{d}}{\mathrm{d}t}\exp(A(t)) = \mathrm{dexp}_{A(t)}(A'(t))\exp(A(t)). \tag{2.42}$$

Just like the functions Ad_A and ad_A defined in (2.40) and (2.41) respectively, also dexp_A is linear in its second argument for a fixed A. Hence we may regard all these as being matrices acting on $\mathfrak{g}$. In fact dexp_A is an analytic function of the matrix transformation ad_A:

$$\mathrm{dexp}_A = \frac{\mathrm{expm}(\mathrm{ad}_A) - \boldsymbol{I}}{\mathrm{ad}_A}. \tag{2.43}$$

This formula should be read as a power series in the following manner: Since

$$\frac{\mathrm{e}^x - 1}{x} = 1 + \tfrac{1}{2!}x + \tfrac{1}{3!}x^2 + \tfrac{1}{4!}x^3 \cdots + \tfrac{1}{(j+1)!}x^j + \cdots,$$

we obtain

$$\begin{aligned}\operatorname{dexp}_A(C) &= C + \tfrac{1}{2!}[A,C] + \tfrac{1}{3!}[A,[A,C]] + \tfrac{1}{4!}[A,[A,[A,C]]] + \cdots \\ &= \sum_{j=0}^{\infty} \frac{1}{(j+1)!} \operatorname{ad}_A^j C.\end{aligned} \tag{2.44}$$

The fact that dexp_A is an analytic function in ad_a makes it easy to invert the matrix dexp_A simply by inverting the analytic function,

$$\operatorname{dexp}_A^{-1} = \frac{\operatorname{ad}_A}{\operatorname{expm}(\operatorname{ad}_A) - \boldsymbol{I}}. \tag{2.45}$$

Recall that

$$\frac{x}{\mathrm{e}^x - 1} = 1 - \tfrac{1}{2}x + \tfrac{1}{12}x^2 - \tfrac{1}{720}x^4 + \cdots = \sum_{j=0}^{\infty} \frac{\mathrm{B}_j}{j!} x^j,$$

where B_j are *Bernoulli numbers* (Abramowitz and Stegun 1970). Thus,

$$\operatorname{dexp}_A^{-1}(C) = C - \tfrac{1}{2}[A,C] + \tfrac{1}{12}[A,[A,C]] + \cdots = \sum_{j=0}^{\infty} \frac{\mathrm{B}_j}{j!} \operatorname{ad}_A^j(C). \tag{2.46}$$

Note that, except for B_1, all odd-indexed Bernoulli numbers vanish. Hence $\operatorname{dexp}_A^{-1} + \frac{1}{2}\operatorname{ad}_A$ is an even function of ad_A. We have based the formulae here on *right trivialisations,* i.e. tangents at a point $P \in G$ being written as CP, $C \in \mathfrak{g}$. It is equally possible to derive formulae based on left trivialisations, tangents written in the form $\tilde{C}P$. If $PC = \tilde{C}P$, we observe that $\tilde{C} = P^{-1}CP = \operatorname{Ad}_{P^{-1}}(C)$. Using (2.24), we compute the left-trivialised formulae as

$$\operatorname{Ad}_{\exp(-A)} \operatorname{dexp}_A = \exp(\operatorname{ad}_{-A}) \frac{\exp(\operatorname{ad}_A) - \boldsymbol{I}}{\operatorname{ad}_A} = \frac{\boldsymbol{I} - \exp(\operatorname{ad}_{-A})}{\operatorname{ad}_A} = \operatorname{dexp}_{-A}.$$

Hence

$$\frac{\mathrm{d}}{\mathrm{d}t} \exp(A(t)) = \operatorname{dexp}_{A(t)}(A'(t)) \exp(A(t)) = \exp(A(t)) \operatorname{dexp}_{-A(t)}(A'(t)). \tag{2.47}$$

Thus we can arrive at the left versions by changing the sign of every commutator. Note that $\operatorname{dexp}_A^{-1}(C)$ and $\operatorname{dexp}_{-A}^{-1}(C)$ differ only in the sign of the term $\pm\frac{1}{2}[A,C]$.

The definition of dexp in Definition 2.18 can be generalised to any smooth function $\psi : \mathfrak{g} \to \mathcal{G}$:

Definition 2.19 Given a smooth function $\psi : \mathfrak{g} \to \mathcal{G}$, we define *the right trivialized tangent* of ψ as the function $\mathrm{d}\psi : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$ defined such that

$$\frac{\mathrm{d}}{\mathrm{d}t} \psi(A(t)) = \mathrm{d}\psi_{A(t)}(A'(t)) \psi(A(t)). \tag{2.48}$$

The function $\mathrm{d}\psi$ is always linear in the second argument, A'.

Let us now apply dexp and dexp^{-1} to obtain a differential equation for the BCH formula. Going back to Theorem 2.5, we define a function $\mathrm{bch} : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$ such that

$$\mathrm{expm}(\,\mathrm{bch}\, AB) = \,\mathrm{expm}(A)\,\mathrm{expm}(B).$$

We may compute $C = \,\mathrm{bch}(A, B)$ by integrating a differential equation. Let $C(t) = \mathrm{bch}(tA, B)$. Clearly $C(0) = B$, and we seek $C(1)$. Writing

$$\mathrm{expm}(C(t)) = \,\mathrm{expm}(tA)\,\mathrm{expm}(B),$$

we find by differentiation that

$$\begin{aligned} & \mathrm{dexp}_{Ct}(C'(t))\exp(C(t)) = \,\mathrm{dexp}_{tA}(A)\exp(tA)\exp(B) = A\exp(C(t)) \\ \Rightarrow \quad & C'(t) = \,\mathrm{dexp}^{-1}_{C(t)}(A). \end{aligned}$$

We have proved: the following result.

Lemma 2.10 The function $C = \,\mathrm{bch}(A, B)$ can be computed by integrating the differential equation

$$C'(t) = \,\mathrm{dexp}^{-1}_{C(t)}(A), \quad 0 \le t \le 1, \qquad C(0) = B, \tag{2.49}$$

from $t = 0$ to $t = 1$.

In this light the dexp^{-1} function may be regarded as a kind of 'infinitesimal BCH generator'. A more concise presentation of this idea is given in (Engø 1998). A symbolic algorithm to compute the BCH from this formula can be found in (Munthe-Kaas and Owren 1999).

2.7. Crouch–Grossman methods

The discipline of Lie-group methods owes a great deal to the pioneering work of Peter Crouch and his coworkers, who were the first to introduce in a systematic, mathematically-sophisticated manner ODE methods that evolve on manifolds. It is interesting to note that their work was primarily motivated by problems in robotics and control theory.

The main algorithm originating in this circle of ideas is the method of *rigid frames* of Crouch and Grossman (1993). It has been originally stated in a more general formalism of differential equations evolving on manifolds. To fit the method into our narrative and to simplify its exposition we restrict our discussion to Lie-group equations.

In essence, the Crouch–Grossman approach is an attempt to apply a Runge–Kutta method to (2.1) (or, in a Lie-group context, to (2.38)) by repeatedly freezing and thawing coefficients and keeping the flow in the correct configuration space. The solution of a 'frozen' Lie-group equation,

$$Y' = A(\tilde{t}, \tilde{Y})Y \quad t \ge t_*, \qquad Y(t_*) = Y_*,$$

is simply $\operatorname{expm}((t-t_*)A(\tilde{t},\tilde{Y}))Y_*$. Freezing (2.38) at t_n and letting $Y_{n+1} = \operatorname{dexp}(hA(t_n,Y_n))Y_n$, where $Y_m \approx Y(t_m)$, $t_m = mh$, results in a first-order method of very little merit. This can be remedied in the following procedure. We choose constants c_k, b_l, $a_{k,l}$, $1 \le l < k \le \nu$ and let

$$\left.\begin{aligned} X_k &= \mathrm{e}^{ha_{k,k-1}F_{k-1}}\mathrm{e}^{ha_{k,k-2}F_{k-2}}\cdots\mathrm{e}^{ha_{k,1}F_1}Y_n \\ F_k &= A(t_n + c_k h, X_k), \\ Y_{n+1} &= \mathrm{e}^{hb_\nu F_\nu}\mathrm{e}^{hb_{\nu-1}F_{\nu-1}}\cdots\mathrm{e}^{hb_1F_1}Y_n. \end{aligned}\right\} \qquad k = 1,2,\ldots,\nu, \tag{2.50}$$

In other words, we model the solution as a product of ν 'frozen' steps. Note that $X_k \in \mathcal{G}$, $F_k \in \mathfrak{g}$, hence, as required, $Y_{n+1} \in \mathcal{G}$.

The method (2.50) might appear to the initiated as a 'Lie-group version' of a Runge–Kutta scheme, an analogy that we have deliberately reinforced by employing notation that will be reserved in the sequel to RK schemes. Yet, order conditions are considerably more challenging than in the classical RK case: they are nonlinear in the weights $b_1, b_2, \ldots, b_\nu$ and there are more of them!

Moderate headway can be made by elementary means and a great deal of algebra. Thus, Crouch and Grossman (1993) have derived three-stage methods of order three, for example

$$\begin{aligned} X_1 &= Y_n, & F_1 &= A(t_n, X_1), \\ X_2 &= \mathrm{e}^{\frac{3}{4}hF_1}Y_n, & F_2 &= A(t_n + \tfrac{3}{4}h, X_2), \\ X_3 &= \mathrm{e}^{\frac{17}{108}hF_2}\mathrm{e}^{\frac{119}{216}hF_1}Y_n, & F_3 &= A(t_n + \tfrac{17}{24}h, X_3), \\ Y_{n+1} &= \mathrm{e}^{\frac{13}{51}hF_3}\mathrm{e}^{-\frac{2}{3}hF_2}\mathrm{e}^{\frac{24}{17}hF_1}Y_n. \end{aligned}$$

Inquiry into higher-order methods, though, requires more than algebra and elbow grease. The situation is further complicated by the fact that (2.50) is typically formulated in considerably more abstract manner, in a manifold setting: this does not make order analysis any simpler!

The order of classical RK methods is nowadays determined by a method due to Butcher (1963), which identifies expansion terms of both the exact and the approximate solution with rooted trees. Remarkably, similar approach can be generalised to Crouch–Grossman methods and this has been accomplished by Owren and Marthinsen (1999*b*). Details of their work are outside the scope of this survey and we refer the readers to the primary source. Let us just mention that, unlike the classical RK case, there are no fourth-order methods of this kind with four stages and $\nu = 5$ is required. Moreover, Owren and Marthinsen (1999*b*) extended (2.50) to *implicit* methods, whereby $a_{k,l}$ is given for all $k,l = 1,2,\ldots,\nu$ and

$$X_k = \mathrm{e}^{ha_{k,\nu}F_\nu}\mathrm{e}^{ha_{k,\nu-1}F_{\nu-1}}\cdots\mathrm{e}^{ha_{k,1}F_1}Y_n, \qquad k = 1,2,\ldots,\nu.$$

This allows for better order/stages ratio but the downside is the need to solve

nonlinear equations in Lie groups. Classical methods, e.g. the Newton–Raphson technique, are of little use here since, unless iterated to convergence, they are unlikely to deliver a solution which resides in the Lie group $\mathcal{G}$. Recently, however, Owren and Welfert (1996) developed two variants of Newton's method that always evolve in $\mathcal{G}$. This brings implicit versions of (2.50) within the realm of computation, although they are expensive.

The mainstream of research into Lie-group methods has moved in the last few years away from the Crouch–Grossman approach. The main reason is that the RK-MK methods, the theme of the next section, provide a considerably more convenient, intuitive and easy-to-analyse means of translating Runge–Kutta formalism to a Lie-group setting. Yet, it would be unfair to pronounce Crouch–Grossman methods as unviable or of purely historical interest. Firstly, in their more general setting, Crouch–Grossman methods can be made (with some effort!) to evolve on arbitrary smooth manifolds, while the scope of RK-MK is restricted to group actions. Secondly, in the present stage in the lifetime of geometric integration we are denied the comfort of discarding lines of inquiry simply because of our current, incomplete understanding.

3. Runge–Kutta on manifolds and RK-MK

In this section we describe a class of numerical integration schemes for computing (2.38), or more generally (2.25). The methods have become known under the name of *RK-MK-type* schemes. We will later see that they might just as well be called *integration schemes based on canonical coordinates of the first kind.* These methods were originally developed in a series of four papers, (Munthe-Kaas 1995, Munthe-Kaas and Zanna 1997, Munthe-Kaas 1998) and (Munthe-Kaas 1999).

A major motivation behind the first of these papers was an attempt to understand and specify the basic operations underlying classical Runge–Kutta methods for integration of differential equations. Abstract specifications of mathematical structures are fundamental in theoretical computer science as a tool for structuring software. An *object-oriented* program consists of a collection of program modules which interact in a well-specified manner. A module could e.g. represent some mathematical structure, like a linear space, a Lie algebra or a Lie group. The basic idea of object-oriented programming is that particular *representations* of the mathematical structure to be modelled should be *hidden* within the program module, and that interactions between different program modules should be independent of particular representations. Although this approach to programming has been very successful for discrete problems, considerably less has been done within areas of continuous mathematics, such as integration of differential equations. Much insight about the important underlying structures can be gained by studying 'pure' mathematical definitions, since these focus more on *what* the essential mathematical structures are, rather than on *how* they can be represented. Thus it is natural, for example, to specify that domains of differential equations should be differential

manifolds and that the right hand side of the equation should be a vector field on the manifold.

Seen from this perspective, classical integration schemes such as Runge–Kutta methods contain 'type errors' in their formulation. As an example, consider the trapezoidal rule

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \frac{h}{2}[\boldsymbol{f}(\boldsymbol{y}_n) + \boldsymbol{f}(\boldsymbol{y}_{n+1})].$$

All the operations are valid if $\boldsymbol{y}$ and f are vectors. However, if $\boldsymbol{y}$ is a point on a manifold and $\boldsymbol{f}$ a vector field, then this expression involves addition of tangent vectors at different base points, and also the addition of a point on a manifold and a tangent to the manifold, both being invalid operations in the context of general manifolds.

In (Munthe-Kaas 1995) classical Runge–Kutta methods are reformulated using coordinate-independent operations on a Lie group. It is shown there that the Butcher theory for order conditions of Runge–Kutta methods (Butcher 1963, Hairer, Nørsett and Wanner 1993) can be reformulated in a geometrical language, replacing the 'Butcher trees' with commutators in a Lie algebra. The outcome is a so-called *Lie–Butcher theory.* Although the resulting algorithms respect Lie-group structure, they can, in the simplest version, attain at most order two for a general non-commutative Lie group. In the sequel, (Munthe-Kaas 1998), the Lie–Butcher order theory is improved, order conditions derived to arbitrary order in general Lie groups and explicit methods of Runge–Kutta type presented up to order four. The paper (Munthe-Kaas and Zanna 1997) generalises the theory from equations on Lie groups to equations evolving on more general manifolds acted upon by a Lie group. In the final of these papers, (Munthe-Kaas 1999), it is shown that similar methods of arbitrarily high order can be constructed and analysed in a relatively simple manner, without employing the Lie–Butcher theory.

In this section our goal is to arrive at the main ideas of the algorithms while employing a minimal amount of formal theory. We have therefore decided not to discuss the general Lie–Butcher theory since this would require a significant amount of Lie theory, beyond what we have already introduced in Section 2. The interested reader is referred to (Munthe-Kaas 1998) for details on Lie–Butcher series. We will continue along the lines of (Munthe-Kaas 1999), but instead of proofs for the general equation (2.25) we restrict the discussion mainly to the simple matrix case (2.38).

We have seen that classical Runge–Kutta methods are valid and 'type-correct' only for differential equations evolving in a linear space V, since then the configuration space and the space of vector fields coincide. If the analytical solution of the differential equation evolves on some linear subspace $W \subset V$, then it is easy to show that a consistent numerical integrator will also evolve on W (up to a departure due to roundoff errors). On the other hand, if the differential equation evolves on some nonlinear manifold embedded in V, it is much more difficult to devise numerical algorithms that stay on the right submanifold. It is well known that traditional ν-stage Runge–Kutta methods preserve quadratic submanifolds if the coefficients

satisfy the condition

$$b_k b_l = b_k a_{k,l} + b_l a_{l,k}, \qquad k, l = 1, 2, \ldots, \nu$$

(Cooper 1987), while Calvo et al. (1997) show that this condition is also necessary. In the same paper it is also shown that it is in general impossible to devise classical Runge–Kutta methods that preserve arbitrary cubic submanifolds. Linear multistep methods or truncated Taylor expansions cannot in general preserve even quadratic submanifolds (Iserles 1997).

In the case of equations on a Lie group $\mathcal{G}$, recall that the local structure in a neighbourhood of any point can be described by the Lie algebra $\mathfrak{g}$, which is a *linear space.* Even better, if $\mathcal{H}$ is a Lie subgroup of $\mathcal{G}$, then there exists a (linear!) subalgebra $\mathfrak{h}$ of $\mathfrak{g}$ describing the local structure of $\mathcal{H}$. Given a differential equation evolving on $\mathcal{H}$, it is in general impossible to devise classical integration scheme that will evolve on $\mathcal{H}$. On the other hand, if an equation is evolving on $\mathfrak{h}$, so will almost *any* reasonable numerical integrator. It thus seems to be a good idea to try to solve a differential equation in the Lie algebra rather than in the Lie group!

Given the equation $Y' = A(t, Y)Y$, $Y(0) = Y_0$, we call the map

$$\mathfrak{g} \ni A \mapsto \operatorname{expm}(A)Y_0 \in \mathcal{G} \tag{3.1}$$

canonical coordinates of the first kind (Varadarajan 1984). This defines a smooth invertible map between a neighbourhood of $\boldsymbol{O} \in \mathfrak{g}$ and a neighbourhood of $Y_0 \in G$. We say that these coordinates are *centred* about $Y_0 \in \mathcal{G}$. A crucial step is 'pulling back' the equation from $\mathcal{G}$ to $\mathfrak{g}$ using this map.

Lemma 3.1 For small $t \geq 0$ the solution of (2.38) is given by

$$Y(t) = \operatorname{expm}(\Theta(t))Y_0.$$

where $\Theta \in \mathfrak{g}$ satisfies the differential equation

$$\Theta'(t) = \operatorname{dexp}^{-1}_{\Theta(t)}(A(t, Y)) \qquad \Theta(0) = \boldsymbol{O} \tag{3.2}$$

and the dexp^{-1} operator has been defined in (2.46).

Proof. Differentiation of $Y(t) = \exp(\Theta(t))Y_0$ yields

$$Y'(t) = \operatorname{dexp}_{\Theta(t)}(\Theta'(t)) \exp(\Theta(t))Y_0 = \operatorname{dexp}_{\Theta(t)}(\Theta'(t))Y(t).$$

The lemma follows from $Y'(t) = A(t, Y)Y(t)$. □

Equation (3.2) is absolutely crucial to the entire business of Lie-group methods. It has been originally stated by Felix Hausdorff (1906), although some attribute it to John Edward Campbell, who might have published it few years earlier. (The names of both, together with Henry Frederick Baker, have been immortalised in the BCH

formula, cf. Theorem 2.5.) The corresponding result for the general case (2.25) is given in (Munthe-Kaas 1999).

Note that the proof uses no other property of the exponential mapping than the definition of dexp. Hence the argument can be easily generalised, replacing exp with a general coordinate map ψ and dexp with $\mathrm{d}\psi$ as discussed in Definition 2.19. This will be used in Section 6.

The simplest and most natural solution strategy is to apply a classically-formulated Runge–Kutta method to (3.2), *rather than to the original equation* (2.38). At each step we choose a coordinate system of the form (3.1), centred at the last known point Y_n. Let us consider briefly the details of this algorithm. Recall that a ν-stage Runge–Kutta method is defined by constants $\{a_{k,l}\}_{k,l=1}^{\nu}$, $\{b_l\}_{l=1}^{\nu}$, $\{c_k\}_{k=1}^{\nu}$, usually written as a *Butcher tableau*

$$\begin{array}{c|cccc} c_1 & a_{1,1} & a_{1,2} & \cdots & a_{1,\nu} \\ c_2 & a_{2,1} & a_{2,2} & \cdots & a_{2,\nu} \\ \vdots & \vdots & \vdots & & \vdots \\ c_\nu & a_{\nu,1} & a_{\nu,2} & \cdots & a_{\nu,\nu} \\ \hline & b_1 & b_2 & \cdots & b_\nu \end{array}$$

(Hairer et al. 1993). Applied to a standard vector equation $\boldsymbol{y}' = \boldsymbol{f}(t, \boldsymbol{y})$, a single step of length h from $\boldsymbol{y}_n$ to $\boldsymbol{y}_{n+1}$ is given by

$$\left.\begin{aligned} \boldsymbol{\theta}_k &= \boldsymbol{y}_n + \sum_{l=1}^{\nu} a_{k,l} \boldsymbol{f}_l, \\ \boldsymbol{f}_k &= h\boldsymbol{f}(t_n + c_k h, \boldsymbol{\theta}_k), \end{aligned}\right\} \quad k = 1, \ldots, \nu, \qquad \boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \sum_{l=1}^{\nu} b_l \boldsymbol{f}_l. \tag{3.3}$$

Applying this scheme to (3.2), we obtain the *RK-MK algorithm.* The following equations describes a single RK-MK step from $Y_n \in \mathcal{G}$ to $Y_{n+1} \in \mathcal{G}$:

$$\left.\begin{aligned} \Theta_k &= \sum_{l=1}^{\nu} a_{k,l} F_l, \\ A_k &= hA(t_n + c_k h,\ \mathrm{expm}(\Theta_k) Y_n), \\ F_k &= \mathrm{dexp}_{\Theta_k}^{-1}(A_k), \end{aligned}\right\} \quad k = 1, \ldots, \nu, \qquad \begin{aligned} \Theta &= \sum_{l=1}^{\nu} b_l F_l, \\ Y_{n+1} &= \mathrm{expm}(\Theta) Y_n. \end{aligned} \tag{3.4}$$

The same algorithm also integrates the general equation (2.28), provided we replace $\mathrm{expm}(\Theta) Y_n$ with its general form $\Lambda(\exp(\Theta), Y_n)$.

In order to complete this algorithm, we need to provide practical means for computing $\mathrm{dexp}^{-1}_{\Theta_k}(A_k)$. In some cases there exists fast direct algorithms for this, see Appendix B. Note that even if dexp^{-1} is approximated, the resulting algorithm will evolve on the correct manifold. In general one may use the expansion (2.46), truncated to the order of the underlying Runge–Kutta scheme, and the resulting algorithm will obtain the same order as the underlying Runge–Kutta scheme, while staying on the correct manifold. For high order methods, a significant number of commutators must be computed if dexp^{-1} is computed using (2.46). In Section 5.3 the structure of so called *free Lie algebras* is used to dramatically reduce the number of commutators.

Examples of specific methods based on (3.4) feature in Appendix A, where, in a more general setting, it is redesignated as $(A.1)$. Here we just stress again the main difference between (3.3) and (3.4): the latter acts in the Lie algebra $\mathfrak{g}$, which is a linear space, thereby respecting Lie-group structure.

4. Magnus and Fer expansions

There are several possible points of departure for our description of Magnus and Fer expansions. Perhaps the simplest and the most intuitive is the scalar linear differential equation

$$y' = a(t)y, \quad t \geq 0, \qquad y(0) = y_0.$$

Its solution, $y(t) = \exp\left(\int_0^t a(\xi)\mathrm{d}\xi\right) y_0$, is familiar to all well-trained mathematics undergraduates. Bearing in mind that the definition of the exponential can be easily extended from scalars to matrices, one might have perhaps hoped that its obvious generalisation,

$$\mathrm{expm}\left(\int_0^t A(\xi)\mathrm{d}\xi\right) Y_0, \tag{4.1}$$

is the solution of the matrix linear system

$$Y' = A(t)Y, \quad t \geq 0, \qquad Y(0) = Y_0. \tag{4.2}$$

Unless $A(t_1)$ and $A(t_2)$ commute with each other for all $t_1, t_2 \geq 0$, this is, unfortunately, misplaced hope. Before offering possible remedies to this state of affairs, we mention that if $Y_0 \in \mathcal{G}$, a Lie group, and A lies in its Lie algebra $\mathfrak{g}$ (whence (4.2) is a Lie-group equation) then (4.1) evolves in $\mathcal{G}$. Bearing in mind the advisability of respecting Lie-group structure, we need to 'correct' (4.1) without losing this feature.

Two possible remedies suggest themselves. Firstly, we may seek a correction $\Delta(t)$ evolving *in the Lie algebra* $\mathfrak{g}$ so that

$$Y(t) = \mathrm{expm}\left(\int_0^t A(\xi)\mathrm{d}\xi + \Delta(t)\right) Y_0.$$

Alternatively, we may correct with $V(t)$ *in the Lie group* $\mathcal{G}$,

$$Y(t) = \operatorname{expm}\left(\int_0^t A(\xi)\mathrm{d}\xi\right) V(t).$$

This gives rise to *Magnus* and *Fer* expansions, respectively.

Both Magnus (1954) and Fer (1958) expansions originated within a non-numerical context and they have found extensive use, e.g. in mathematical physics, quantum chemistry, control theory and stochastic differential equations as a perturbative tool in the investigation of linear systems (4.2). Fashioning them into a numerical weapon is nontrivial and will occupy us in this and next sections.

4.1. Magnus expansions and rooted trees

Our point of departure is the *dexpinv equation* (2.46) which we recall for completeness of exposition: the solution of (4.2) can be written in the form $Y(t) = \operatorname{expm}(\Theta(t))Y_0$, $t \geq 0$, where

$$\Theta' = \operatorname{dexp}_\Theta^{-1} A = \sum_{k=0}^{\infty} \frac{\mathrm{B}_k}{k!} \operatorname{ad}_\Theta^k A, \quad t \geq 0, \qquad \Theta(0) = \boldsymbol{O}, \tag{4.3}$$

$\{\mathrm{B}_k\}_{k\in\mathbb{Z}^+}$ being Bernoulli numbers. As a first step, we attempt to solve (4.3) by Picard iteration,

$$\begin{aligned} \Theta^{[0]}(t) &\equiv \boldsymbol{O}, \\ \Theta^{[m+1]}(t) &= \int_0^t \operatorname{dexp}_{\Theta^{[m]}(\xi)}^{-1} A(\xi)\mathrm{d}\xi = \sum_{k=0}^{\infty} \frac{\mathrm{B}_k}{k!} \int_0^t \operatorname{ad}_{\Theta^{[m]}(\xi)}^k A(\xi)\mathrm{d}\xi, \quad m = 0, 1, \ldots. \end{aligned}$$

Rearranging terms for simplicity, we obtain

$$
\begin{aligned}
\Theta^{[1]}(t) &= \int_0^t A(\xi_1)\mathrm{d}\xi_1, \\
\Theta^{[2]}(t) &= \int_0^t A(\xi_1)\mathrm{d}\xi_1 - \tfrac{1}{2}\int_0^t \left[\int_0^{\xi_1} A(\xi_2)\mathrm{d}\xi_2, A(\xi_1)\right]\mathrm{d}\xi_1 \\
&\quad + \tfrac{1}{12}\int_0^t \left[\int_0^{\xi_1} A(\xi_2)\mathrm{d}\xi_2, \left[\int_0^{\xi_1} A(\xi_2)\mathrm{d}\xi_2, A(\xi_1)\right]\right]\mathrm{d}\xi_1 + \cdots \\
\Theta^{[3]}(t) &= \int_0^t A(\xi_1)\mathrm{d}\xi_1 - \tfrac{1}{2}\int_0^t \left[\int_0^{\xi_1} A(\xi_2)\mathrm{d}\xi_2, A(\xi_1)\right]\mathrm{d}\xi_1 \\
&\quad + \tfrac{1}{12}\int_0^t \left[\int_0^{\xi_1} A(\xi_2)\mathrm{d}\xi_2, \left[\int_0^{\xi_1} A(\xi_2)\mathrm{d}\xi_2, A(\xi_1)\right]\right]\mathrm{d}\xi_1 \\
&\quad + \tfrac{1}{4}\int_0^t \left[\int_0^{\xi_1}\left[\int_0^{\xi_2} A(\xi_3)\xi_3, A(\xi_2)\right]\mathrm{d}\xi_2, A(\xi_1)\right]\mathrm{d}\xi_1 \\
&\quad - \tfrac{1}{24}\int_0^t \left[\int_0^{\xi_1}\left[\int_0^{\xi_2} A(\xi_3)\mathrm{d}\xi_3, \left[\int_0^{\xi_2} A(\xi_3)\mathrm{d}\xi_3, A(\xi_2)\right]\right]\mathrm{d}\xi_2, A(\xi_1)\right]\mathrm{d}\xi_1 \\
&\quad - \tfrac{1}{24}\int_0^t \left[\int_0^{\xi_1}\left[\int_0^{\xi_2} A(\xi_3)\mathrm{d}\xi_3, A(\xi_2)\right]\mathrm{d}\xi_2, \left[\int_0^{\xi_1} A(\xi_2)\mathrm{d}\xi_2, A(\xi_1)\right]\right]\mathrm{d}\xi_1 \\
&\quad - \tfrac{1}{24}\int_0^t \left[\int_0^{\xi_1} A(\xi_2)\mathrm{d}\xi_2, \left[\int_0^{\xi_1}\left[\int_0^{\xi_2} A(\xi_3)\mathrm{d}\xi_3, A(\xi_2)\right]\mathrm{d}\xi_2, A(\xi_1)\right]\right]\mathrm{d}\xi_1 + \cdots
\end{aligned}
$$

and so on. The Picard theorem implies that $\Theta(t) = \lim_{m\to\infty}\Theta^{[m]}(t)$ exists in a suitably small neighbourhood of the origin and the above first few iterations indicate that it can be expanded as a linear combination of terms that are composed from *integrals* and *commutators* acting recursively on the matrix A. This is the *Magnus expansion*

$$\Theta(t) = \sum_{k=0}^{\infty} H_k(t), \tag{4.4}$$

where each H_k is a linear combination of terms that include exactly $k+1$ integrals (or – and later we will see that it boils down to the same thing – k commutators).

Thus,

$$
\begin{aligned}
H_0(t) &= \int_0^t A(\xi_1)\mathrm{d}\xi_1, \\
H_1(t) &= -\tfrac{1}{2}\int_0^t \left[\int_0^{\xi_1} A(\xi_2)\mathrm{d}\xi_2, A(\xi_1)\right]\mathrm{d}\xi_1, \\
H_2(t) &= \tfrac{1}{12}\int_0^t \left[\int_0^{\xi_1} A(\xi_2)\mathrm{d}\xi_2, \left[\int_0^{\xi_1} A(\xi_2)\mathrm{d}\xi_2, A(\xi_1)\right]\right]\mathrm{d}\xi_1 \\
&\quad + \tfrac{1}{4}\int_0^t \left[\int_0^{\xi_1}\left[\int_0^{\xi_2} A(\xi_3)\mathrm{d}\xi_3, A(\xi_2)\right]\mathrm{d}\xi_2, A(\xi_1)\right]\mathrm{d}\xi_1, \\
H_3(t) &= -\tfrac{1}{24}\int_0^t \left[\int_0^{\xi_1}\left[\int_0^{\xi_2} A(\xi_3)\mathrm{d}\xi_3, \left[\int_0^{\xi_2} A(\xi_3)\mathrm{d}\xi_3, A(\xi_2)\right]\right]\mathrm{d}\xi_2, A(\xi_1)\right]\mathrm{d}\xi_1 \\
&\quad - \tfrac{1}{24}\int_0^t \left[\int_0^{\xi_1}\left[\int_0^{\xi_2} A(\xi_3)\mathrm{d}\xi_3, A(\xi_2)\right]\mathrm{d}\xi_2, \left[\int_0^{\xi_1} A(\xi_2)\mathrm{d}\xi_2, A(\xi_1)\right]\right]\mathrm{d}\xi_1 \\
&\quad - \tfrac{1}{24}\int_0^t \left[\int_0^{\xi_1} A(\xi_2)\mathrm{d}\xi_2, \left[\int_0^{\xi_1}\left[\int_0^{\xi_2} A(\xi_3)\mathrm{d}\xi_3, A(\xi_2)\right]\mathrm{d}\xi_2, A(\xi_1)\right]\right]\mathrm{d}\xi_1 \\
&\quad - \tfrac{1}{8}\int_0^t \left[\int_0^{\xi_1} A(\xi_2)\mathrm{d}\xi_2, \left[\int_0^{\xi_1} A(\xi_2)\mathrm{d}\xi_2, \left[\int_0^{\xi_1} A(\xi_2)\mathrm{d}\xi_2, A(\xi_1)\right]\right]\right]\mathrm{d}\xi_1.
\end{aligned}
$$

It is possible to derive the next few sets H_k with enough perseverance and perhaps a little help from a computer algebra package. Yet, it is evident that the terms are becoming increasingly complex. A considerably more transparent form of the Magnus expansion, amenable for easy recursive derivation and easier discussion of computational issues, can be obtained by associating each term in the expansion with a *rooted binary tree,* an approach that has been pioneered by Iserles and Nørsett (1999).

Let us recall briefly relevant terminology of graph theory (Harary 1969).

- Let $\boldsymbol{V} = \{v_1, v_2, \ldots, v_r\}$ be a finite set of distinct *vertices* and $\boldsymbol{E} \in \boldsymbol{V} \times \boldsymbol{V}$ a set of *edges.* (The edges (v_i, v_j) and (v_j, v_i) are identified with each other.) We say that $\boldsymbol{G} = \langle \boldsymbol{V}, \boldsymbol{E} \rangle$ is a *graph.*
- The ordered set $\{(v_{s_l}, v_{t_l}) : l = 1, 2, \ldots, r\}$ of edges is a *path* from $v_i \in \boldsymbol{V}$ to $v_j \in \boldsymbol{V}$, $i \neq j$, if $s_1 = i$, $t_l = s_{l+1}$, $l = 1, 2, \ldots, r-1$ and $t_r = j$.
- The graph is said to be *connected* if there is a path between any two distinct vertices. It is a *tree* if exactly one path links every two vertices.
- A *rooted tree* is the pair $\boldsymbol{T} = (\boldsymbol{G}, w)$, where $\boldsymbol{G}$ is a tree and $w \in \boldsymbol{V}$ is its *root.* There exists natural partial order on $\boldsymbol{T}$: we say that $v_i \prec v_j$ if v_i precedes v_j in the unique path extending from the root w to v_j. In that case v_i is the

ancestor of v_j, while v_j is the *successor* of v_i. (Thus, the root is the ancestor of all vertices in $\boldsymbol{V} \setminus \{w\}$.)

- If $v_i \prec v_j$ and there is no $v_k \in \boldsymbol{V}$ such that $v_i \prec v_k \prec v_j$, we say that v_i is the *parent* of v_j (most graph-theory texts adopt a more sexist definition) and v_j the *child* of v_i. Childless vertices are called *leaves.*
- If each vertex in a rooted tree has at most two children, $\boldsymbol{T}$ is called a *binary tree.* If each vertex has either exactly two children or is a leaf, $\boldsymbol{T}$ is said to be a *strictly binary tree.*

It is always worthwhile to draw a pictorial representation of a graph, whereby edges are merely undirected lines extending between vertices, which are denoted by black discs. The graph-theoretical convention is that the root is always placed at the bottom, although computer scientists occasionally defy gravity by locating it at the top. We will follow the mathematical convention.

We commence our investigation of the Magnus expansion by rewriting (4.4) in the form

$$\Theta(t) = \sum_{k=0}^{\infty} H_k(t) = \sum_{k=0}^{\infty} \sum_{\tau \in \mathbb{T}_k} \alpha(\tau) \int_0^t \mathcal{C}_\tau(\xi) \mathrm{d}\xi, \quad t \geq 0, \tag{4.5}$$

Thus, each $\mathcal{C}_\tau$ for $\tau \in \mathbb{T}_k$ is made out of exactly k integrals and k commutators, while each $\alpha(\tau)$ is a scalar constant. Recalling how the expansion has been obtained from Picard's iteration, we observe that the terms $\mathcal{C}_\tau$ can be obtained by just two *composition rules:*

1 The index set $\mathbb{T}_0$ is a simpleton, $\mathbb{T}_0 = \{\tau_\circ\}$, say, and $\mathcal{C}_{\tau_\circ}(t) = A(t)$;

2 If $\tau_1 \in \mathbb{T}_{m_1}$ and $\tau_2 \in \mathbb{T}_{m_2}$ then there exists $\tau \in \mathbb{T}_{m_1+m_2+1}$ such that

$$\mathcal{C}_\tau(t) = \left[\int_0^t \mathcal{C}_{\tau_1}(\xi) \mathrm{d}\xi, \mathcal{C}_{\tau_2}(t) \right]. \tag{4.6}$$

Although this observation makes the expansion somewhat more transparent, much greater transparency is obtained by identifying the index sets $\mathbb{T}_k$ with subsets of binary rooted trees, in a manner that makes the above composition rules stand out pictorially. We express the relationship between the index τ and the term $\mathcal{C}_\tau(t)$ by the map $\tau \rightsquigarrow \mathcal{C}_\tau(t)$.

1 We let $\mathbb{T}_0$ consist of the single rooted tree with one vertex, $\bullet$, and

$$\bullet \rightsquigarrow A(t).$$

2 Suppose that $\mathbb{T}_{m_1} \ni \tau_1 \rightsquigarrow \mathcal{C}_{\tau_1}(t)$ and $\mathbb{T}_{m_2} \ni \tau_2 \rightsquigarrow \mathcal{C}_{\tau_2}(t)$. Then

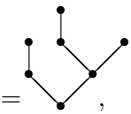
$$\mathbb{T}_{m_1+m_2+1} \ni \begin{array}{c} \tau_1 \\ | \\ \bullet \\ \diagup \quad \diagdown \\ \bullet \qquad \tau_2 \end{array} \rightsquigarrow \left[\int_0^t \mathcal{C}_{\tau_1}(\xi) \mathrm{d}\xi, \mathcal{C}_{\tau_2}(t) \right]. \tag{4.7}$$

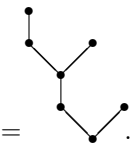
Thus, (4.6) is 'coded' in graph terminology by denoting integration by adding a root to a tree, while commutation is denoted by joining two trees with a common root. It is possible to show that all the terms in the Magnus expansion can be obtained in this manner (Iserles and Nørsett 1999).

To derive $\mathbb{T}_1$ we have just one option, $m_1 = m_2 = 0$, and the outcome is a single tree,

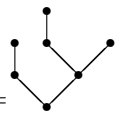
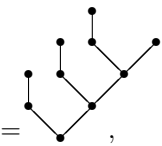
$\tau_1 = \bullet,$ $\tau_2 = \bullet$ $\Rightarrow$ $\tau =$  .

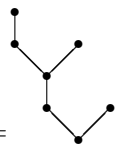
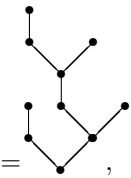
There are two possibilities in $\mathbb{T}_2$, namely $m_1 = 0$, $m_2 = 1$ and $m_1 = 1$, $m_2 = 0$. They yield

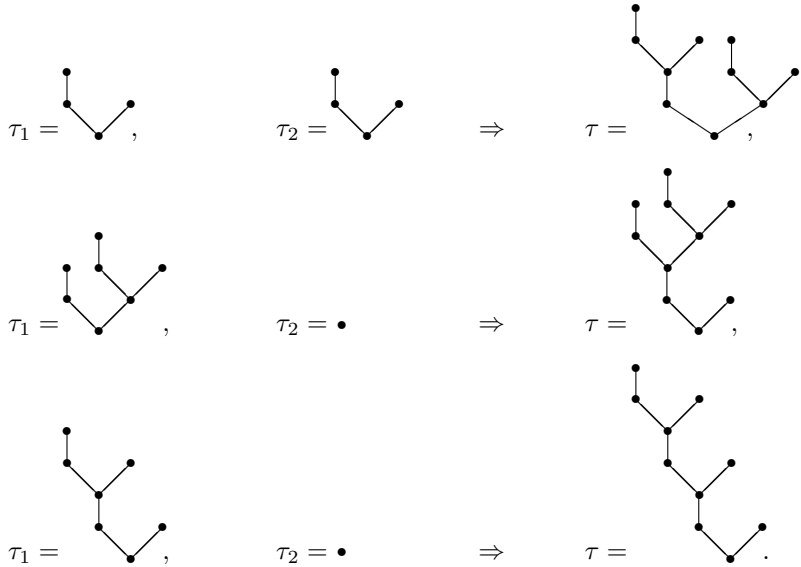
$\tau_1 = \bullet,$ $\tau_2 =$  $\Rightarrow$ $\tau =$  ,

$\tau_1 =$  , $\tau_2 = \bullet$ $\Rightarrow$ $\tau =$  .

Next, we construct $\mathbb{T}_3$:

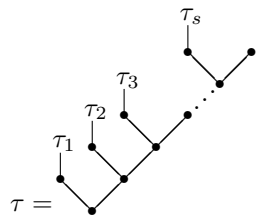
$\tau_1 = \bullet,$ $\tau_2 =$  $\Rightarrow$ $\tau =$  ,

$\tau_1 = \bullet,$ $\tau_2 =$  $\Rightarrow$ $\tau =$  ,



The principle should be quite clear by now, as should be the correspondence between trees and expansion terms which, indeed, can be 'read' directly from the graph. Thus, the last tree can be 'recited' as "the integral of A, commuted with A, integrated, commuted with A, integrated and commuted with A".

We now have a recursive algorithm to derive the $\mathcal{C}_\tau$s but recall that, to complete our description of (4.5), we also require the constants $\alpha(\tau)$. Fortunately, also this can be deduced from the tree formalism. We thus commence by letting $\alpha(\bullet) = 1$ and continue by observing that each tree in $\cup_{k\in\mathbb{N}}\mathbb{T}_k$ can be written in a unique form as



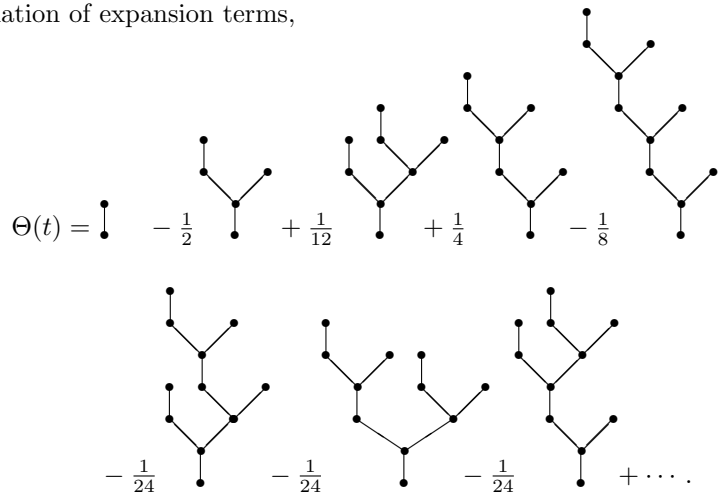
(4.8)

for some $s \geq 1$. We can assume by induction that $\alpha(\tau_i)$ are already known for $i = 1, 2, \ldots, s$. In that case it has been proved in (Iserles and Nørsett 1999) that

$$\alpha(\tau) = \frac{\mathrm{B}_s}{s!} \prod_{i=1}^{s} \alpha(\tau_i). \tag{4.9}$$

Note that $\mathrm{B}_{2m+1} = 0$ for $m \geq 1$. This implies that many (although, unfortunately, not too many...) terms in (4.5) vanish.

We can now write the first terms of the Magnus expansion in a tree formalism, using the convention that linear combination of trees corresponds to a linear combination of expansion terms,



(4.10)

Note that the coefficient corresponding to the last tree in $\mathbb{T}_3$ vanishes, this being a consequence of (4.9) and of $\mathrm{B}_3 = 0$.

It is relatively easy to continue the expansion to higher-order terms. Moreover, it is easy to prove that

$$\tau \in \mathbb{T}_k \qquad \Rightarrow \qquad \int_0^t \mathcal{C}_\tau(\xi)\mathrm{d}\xi = \mathcal{O}\big(t^{k+1}\big)\,, \qquad k \in \mathbb{Z}^+ \tag{4.11}$$

for every sufficiently-smooth matrix function A. On the face of it, this gives a handy device to truncate the Magnus expansion to obtain an approximant of given order – we will bother later about the calculation of multivariate integrals. This, however, is grossly misleading, since the naive estimate (4.11) can be improved a very great deal: far fewer terms are required!

4.2. Convergence of the Magnus expansion

Before we proceed to improve the estimate (4.11) and even consider the question of designing a realistic numerical algorithms based on the Magnus expansion, we need to examine the issue of convergence.

It has been proved in (Iserles and Nørsett 1999) that convergence takes place for sufficiently small $t \geq 0$, but the result was unduly pessimistic. A considerably better (and in a well-defined sense optimal) estimate has been obtained by Blanes, Casas, Oteo and Ros (1998). Herewith we present briefly a short and elegant proof due to Moan (1998).

Theorem 4.1 Suppose that the Lie algebra $\mathfrak{g}$ is equipped with the norm $\|\cdot\|$.

The Magnus expansion (4.4) absolutely converges in this norm for every $t \geq 0$ such that

$$\int_0^t \|A(\xi)\| \mathrm{d}\xi \leq \int_0^{2\pi} \frac{\mathrm{d}\xi}{4 + \xi[1 - \cot(\xi/2)]} \approx 1.086868702. \tag{4.12}$$

Proof. Integrating (4.3) and taking norms, we have by the triangle inequality and the trivial bound $\|\mathrm{ad}_B^k A\| \leq (2\|B\|)^k \|A\|$ that

$$\begin{aligned} \|\Theta(t)\| &= \left\| \int_0^t \mathrm{dexp}^{-1}_{\Theta(\xi)} A(\xi) \mathrm{d}\xi \right\| \leq \int_0^t \| \mathrm{dexp}^{-1}_{\Theta(\xi)} A(\xi) \| \mathrm{d}\xi \\ &\leq \int_0^t \sum_{k=0}^{\infty} \frac{|\mathrm{B}_k|}{k!} (2\|\Theta(\xi)\|)^k \|A(\xi)\| \mathrm{d}\xi = \int_0^t g(2\|\Theta(\xi)\|) \|A(\xi)\| \mathrm{d}\xi, \end{aligned}$$

where

$$g(x) = 2 + \frac{x}{2}\left(1 - \cot\frac{x}{2}\right).$$

We now use a Bihari-type inequality from (Moan 1998): suppose that $h, g, v \in \mathrm{C}(0, t^*)$ are positive and that g is nondecreasing. Then

$$h(t) \leq \int_0^t g(h(\xi)) v(\xi) \mathrm{d}\xi, \qquad t \in (0, t^*)$$

implies that

$$h(t) \leq \tilde{g}^{-1}\left(\int_0^t v(\xi) \mathrm{d}\xi \right), \quad t \in (0, t^{**}), \qquad \text{where} \qquad \tilde{g}(x) = \int_0^x \frac{\mathrm{d}\xi}{g(\xi)}$$

and $t^{**} \in (0, t^*]$ is such that $\tilde{g}\left(\int_0^t v(\xi) \mathrm{d}\xi \right)$ is bounded in $(0, t^{**})$. In our case $h(t) = 2\|\Theta(t)\|$, $v(t) = \|A(t)\|$ and $g(t)$ are all positive and the latter is nondecreasing for $t \in (0, 2\pi)$. Therefore,

$$\|\Theta(t)\| \leq \tfrac{1}{2}\tilde{g}^{-1}\left(\int_0^t \|A(\xi)\| \mathrm{d}\xi \right)$$

and $\|\Theta(t)\|$ is bounded, provided that $\tilde{g}\left(\int_0^t \|A(\xi)\| \mathrm{d}\xi \right)$ is bounded. The latter holds as long as condition (4.12) is satisfied. □

The condition of Theorem 4.1 has been recently improved for a more relaxed convergence framework. Moan (n.d.) proved that the Magnus expansion *converges in norm* for all $t \in (0, t^*)$ with regard to the Euclidean norm,

$$\lim_{m \to \infty} \left\| \Theta(t) - \sum_{k=0}^{m} H_m(t) \right\|_2 = 0,$$

provided that

$$\int_0^{t^*} \|A(\xi)\|_2 \mathrm{d}\xi < \pi. \tag{4.13}$$

Magnus expansion cannot be expected to converge *always*. In its numerical implementation, this means that the expansion (like any other numerical method for ODEs) need be advanced in a time-stepping fashion, rather than being applied globally. Yet, the condition of Theorem 4.1 is not unduly restrictive and (4.13) is even less so. It might be problematic for stiff systems, a subject that has not received much attention in the study of Lie-group methods. An important exception is the use of Magnus expansions in the calculation of Sturm–Liouville spectra, where an elegant device allows to integrate the equations way beyond the formal upper bound (4.12) (cf. Section 11.2 and (Moan 1998)).

4.3. Power of a tree and time symmetry

Following Iserles, Nørsett and Rasmussen (1998), we say that a tree $\tau \in \cup_{k\in\mathbb{Z}^+} \mathbb{T}_k$ is of *power* m if $m \geq 0$ is the least integer such that

$$\mathcal{C}_\tau(t) = \mathcal{O}(t^m)$$

for all sufficiently smooth matrix functions A. Letting $\mathbb{F}_m$ be the set of all trees of power m and truncating the Magnus expansion,

$$\Theta_p(t) = \sum_{m=0}^{p-1} \sum_{\tau\in\mathbb{F}_m} \alpha(\tau) \int_0^t \mathcal{C}_\tau(\xi)\mathrm{d}\xi, \tag{4.14}$$

it is trivial to verify that $\Theta_p(t) = \Theta(t) + \mathcal{O}\big(t^{p+1}\big)$ and we have an order-p approximant.

We already know from (4.11) that $\tau \in \mathbb{T}_k$ implies $\tau \in \mathbb{F}_m$ for some $m \geq k$. How large can m get, though? The main mechanism that increases order is commutation. Thus, suppose that

$$\mathcal{C}_{\tau_i}(t) = D_i t^{m_i} + E_i t^{m_i+1} + F_i t^{m_i+2} + \cdots, \qquad i = 1, 2.$$

Then

$$\begin{aligned}
&\left[\int_0^t \mathcal{C}_{\tau_1}(\xi)\mathrm{d}\xi, \mathcal{C}_{\tau_2}(t)\right] \\
&= \frac{1}{m_1+1}[D_1, D_2]t^{m_1+m_2+1} + \left(\frac{1}{m_1+1}[D_1, E_2] + \frac{1}{m_1+2}[E_1, D_2]\right) t^{m_1+m_2+2} \\
&\quad + \left(\frac{1}{m_1+1}[D_1, F_2] + \frac{1}{m_1+2}[E_1, E_2] + \frac{1}{m_1+3}[F_1, D_2]\right) t^{m_1+m_2+3} + \cdots.
\end{aligned}$$

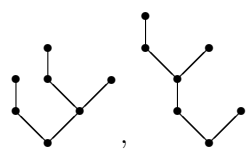
In general, $[D_1, D_2] \neq \boldsymbol{O}$, hence the new term resides in $\mathbb{F}_{m_1+m_2+1}$. However, cancellation takes place in the important special case $\tau_1 = \tau_2$, whence

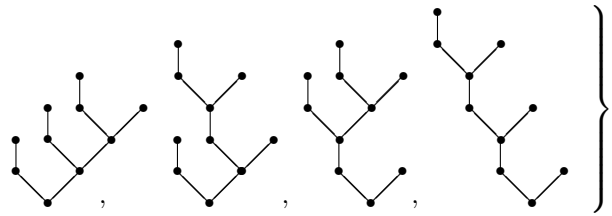
$$\begin{aligned} & \left[\int_0^t \mathcal{C}_{\tau_1}(\xi)\mathrm{d}\xi, \mathcal{C}_{\tau_1}(t)\right] \\ = \quad & \frac{1}{(m_1+1)(m_1+2)}[D_1, E_1]t^{2m_1+2} + \frac{2}{(m_1+1)(m_1+3)}[D_1, F_1]t^{2m_1+3} + \cdots \end{aligned}$$

and the new term is in $\mathbb{F}_{2m_1+2}$, a gain of one unit.

While $\mathbb{F}_0 = \{\bullet\}$, we observe that $\mathbb{F}_1 = \emptyset$ and

$\mathbb{F}_2 =$ 

$\mathbb{F}_3 =$ 

$\mathbb{F}_4 =$ 

In general, the number of terms counted according to power is significantly smaller than similar enumeration by the number of commutators. It is possible to prove that

$$\limsup_{k\to\infty}(\#\mathbb{T}_k)^{1/k} = 4, \qquad \limsup_{m\to\infty}(\#\mathbb{F}_m)^{1/m} \approx 3.11674 \tag{4.15}$$

(Iserles and Nørsett 1999, Iserles et al. 1998). As an example of the reduction in cardinality, compare $\#\mathbb{T}_6 = 132$ with $\#\mathbb{F}_6 = 21$.

Using the 'truncation by power' (4.14) is thus aptly justified. However, before we rush to pronounce this as the 'correct' truncated Magnus expansion, we need to pay attention to yet another device that reduces the number of terms in the expansion.

Let Φ_t be the *flow* corresponding to the differential equation (4.2), $\Phi_t(Y_0) = Y(t)$. It is obvious that the flow is *time symmetric,* $\Phi_{-t} \circ \Phi_t = \mathrm{Id}$, since integrating from 0 to t and back to 0 returns us to the original initial value. What is far less obvious, yet has been proved in (Iserles et al. 1998), is that the truncation by power (4.14) respects time symmetry. In other words, let

$$\tilde{\Phi}_t(Y_0) = \mathrm{e}^{\Theta_p(t)}Y_0, \qquad t \geq 0.$$

Then $\tilde{\Phi}_{-t} \circ \tilde{\Phi}_t = \mathrm{Id}$. This is remarkable, since any analytic time-symmetric map S_t can be represented in the form $S_t = \mathrm{e}^{F_t}$ where the map F_t is expandable in *odd powers of* t *only* (Hairer et al. 1993). This fits our framework perfectly.

Theorem 4.2 The function Θ_p can be expanded in odd powers of t and

$$\Theta_{2q-1}(t) = \Theta(t) + \mathcal{O}\big(t^{2q+1}\big)\,, \qquad q \in \mathbb{N}.$$

Therefore, truncating by power with odd p leads to a gain of an extra unit of order! This is a critical observation which leads to substantial savings in high-order Magnus expansions.

It is important to realise that it is not true that individual elements in H_{2q-1}, $q \geq 2$, are $\mathcal{O}\big(t^{2q+1}\big)$: it is their linear combination that knocks out the $\mathcal{O}\big(t^{2q}\big)$ term!

We may now re-examine the expansion (4.10), truncating by power and identifying the order. Reverting from trees to standard notation, we have

$$\begin{aligned}
\Theta(t) \;=\; & \int_0^t A(\xi)\mathrm{d}\xi \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{order 2} \quad (4.16)\\
& - \tfrac{1}{2}\int_0^t\int_0^{\xi_1} [A(\xi_2), A(\xi_1)]\mathrm{d}\boldsymbol{\xi} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{order 4} \quad (4.17)\\
& + \tfrac{1}{12}\int_0^t\int_0^{\xi_1}\int_0^{\xi_1} [A(\xi_2), [A(\xi_3), A(\xi_1)]]\mathrm{d}\boldsymbol{\xi}\\
& + \tfrac{1}{4}\int_0^t\int_0^{\xi_1}\int_0^{\xi_2} [[A(\xi_3), A(\xi_2)], A(\xi_1)]\mathrm{d}\boldsymbol{\xi}\\
& - \tfrac{1}{24}\int_0^t\int_0^{\xi_1}\int_0^{\xi_1}\int_0^{\xi_3} [A(\xi_2), [[A(\xi_4), A(\xi_3)], A(\xi_1)]]\mathrm{d}\boldsymbol{\xi}\\
& - \tfrac{1}{24}\int_0^t\int_0^{\xi_1}\int_0^{\xi_2}\int_0^{\xi_2} [[A(\xi_3), [A(\xi_4), A(\xi_2)]], A(\xi_1)]\mathrm{d}\boldsymbol{\xi}\\
& - \tfrac{1}{8}\int_0^t\int_0^{\xi_1}\int_0^{\xi_2}\int_0^{\xi_3} [[[A(\xi_4), A(\xi_3)], A(\xi_2)], A(\xi_1)]\mathrm{d}\boldsymbol{\xi} \quad \ldots\text{order 6} \quad (4.18)\\
& + \cdots
\end{aligned}$$

Very often entries of A and its commutators can be integrated explicitly, e.g. when they are polynomials or trigonometric functions. In that case the truncated Magnus expansions (4.16–18), say, can be computed explicitly. However, a more comprehensive numerical approach to Magnus expansions requires the computation of multivariate integrals. Although at the first glance this may appear as a very formidable task, it turns out that the special structure of 'Magnus integrals' renders them amenable to very effective and affordable numerical treatment. We defer the discussion of this issue to Section 5.

4.4. Fer expansions

At the first instance, we wish to represent the solution of (4.2) in the form

$$Y(t) = \mathrm{e}^{\int_0^t A(\xi)\mathrm{d}\xi} V(t), \qquad t \geq 0. \tag{4.19}$$

Direct differentiation yields

$$\begin{aligned} V' &= \left[\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{e}^{-\int_0^t A(\xi)\mathrm{d}\xi}\right] Y + \mathrm{e}^{-\int_0^t A(\xi)\mathrm{d}\xi} Y' \\ &= \left[\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{e}^{-\int_0^t A(\xi)\mathrm{d}\xi}\right] \mathrm{e}^{\int_0^t A(\xi)\mathrm{d}\xi} V + \mathrm{e}^{-\int_0^t A(\xi)\mathrm{d}\xi} A \mathrm{e}^{\int_0^t A(\xi)\mathrm{d}\xi} V \\ &= \left[\mathrm{Ad}_{\,\mathrm{expm}\left(-\int_0^t A(\xi)\mathrm{d}\xi\right)} A(t) - \mathrm{dexp}_{\,\mathrm{expm}\left(-\int_0^t A(\xi)\mathrm{d}\xi\right)} A(t)\right] V. \end{aligned}$$

Recalling from (2.24) and (2.43) that

$$\begin{aligned} \mathrm{Ad}_{\,\mathrm{expm}(E)} D &= \mathrm{e}^{\mathrm{ad}_E} D, \\ \mathrm{dexp}_E D &= \frac{\mathrm{e}^{\mathrm{ad}_E} - \boldsymbol{I}}{\mathrm{ad}_E} D, \end{aligned}$$

we deduce that the correction term V itself obeys a linear differential equation,

$$V' = \left[\frac{(\boldsymbol{I} + \mathrm{ad}_E)\,\mathrm{e}^{-\mathrm{ad}_E} - \boldsymbol{I}}{\mathrm{ad}_E} D\right] V, \quad t \geq 0, \qquad V(0) = Y_0,$$

where

$$D = A(t), \qquad E = \int_0^t A(\xi)\mathrm{d}\xi.$$

This is indeed the main step in constructing the *Fer expansion:* the correction V in (4.19) satisfies the equation

$$V' = \left[\sum_{k=1}^{\infty} (-1)^k \frac{k}{(k+1)!} \mathrm{ad}^k_{\int_0^t A(\xi)\mathrm{d}\xi} A(t)\right] V, \quad t \geq 0, \qquad V(0) = Y_0. \tag{4.20}$$

The idea is now to iterate (4.20). Thus, we let $B_0 = A$ and generate the sequence $\{B_m\}_{m\in\mathbb{Z}^+}$ recursively, where

$$B_m(t) = \sum_{k=1}^{\infty} (-1)^k \frac{k}{(k+1)!} \mathrm{ad}^k_{\int_0^t B_{m-1}(\xi)\mathrm{d}\xi} B_{m-1}(t), \qquad t \geq 0,\ m \in \mathbb{N}. \tag{4.21}$$

The *Fer expansion* of the solution of (4.2) is

$$Y(t) = \mathrm{e}^{\int_0^t B_0(\xi)\mathrm{d}\xi} \mathrm{e}^{\int_0^t B_1(\xi)\mathrm{d}\xi} \mathrm{e}^{\int_0^t B_2(\xi)\mathrm{d}\xi} \cdots Y_0, \qquad t \geq 0. \tag{4.22}$$

This expansion was introduced by Fer (1958) who, remarkably, did not recognise that it respects Lie-group structure. It was rediscovered by Iserles (1984) in a numerical context but, again, Lie groups were not mentioned. Finally, Zanna (1996) recognised (4.22) as a Lie-group solver. In (Zanna and Munthe-Kaas 1997) it is shown that this expansion can be understood as a version of so-called Lie reduction. From this it follows that if $\mathfrak{g}$ is *solvable* (cf. Section 6.5) then the expansion in (4.2) is exact for a *finite* product of exponentials.

The first step in a numerical implementation of the Fer expansion (4.22) is truncation of the infinite product. To this end it is vital to recognise the rate of decay of the matrices B_m. Assuming that $B_m(t) = \mathcal{O}(t^{p_m})$, $m \in \mathbb{Z}^+$, it is easy to verify from (4.21) that $p_m = 2p_{m-1} + 2$. This, in tandem with $p_0 = 0$, yields $p_m = 2^{m+1} - 2$ and we deduce that

$$\mathrm{e}^{\int_0^t B_0(\xi)\mathrm{d}\xi}\mathrm{e}^{\int_0^t B_1(\xi)\mathrm{d}\xi}\cdots\mathrm{e}^{\int_0^t B_{s-1}(\xi)\mathrm{d}\xi}Y_0 = Y(t) + \mathcal{O}\Big(t^{2^{s+1}-1}\Big). \tag{4.23}$$

Thus, we obtain an approximant of order $2^{s+1} - 2$: the order grows exponentially with s (Iserles 1984)!

Of course, if we are interested in an order-p Fer approximant to the solution of (4.2), there is no need to carry out the summation in (4.21) *ad infinitum*. Systematic analysis of order conditions necessary for the formation of Fer approximants of various orders has been carried out by Zanna (1998) using the same binary rooted trees that we have already encountered in our analysis of the Magnus expansion. Identifying A with the single-vertex tree, we have the following explicit form of a Fer approximant of any given order:

Order 2: $s = 1$,

$B_0(t)$: •.

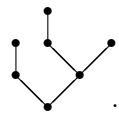
Order 3: $s = 2$,

$B_0(t)$: •,

$B_1(t)$: $\frac{1}{2}$  .

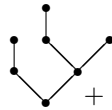
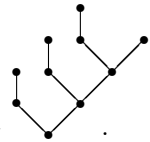
Order 4: $s = 2$,

$B_0(t)$: •,

$B_1(t)$: $\frac{1}{2}$  + $\frac{1}{3}$  .

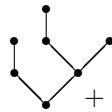
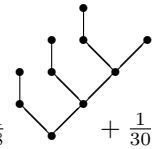
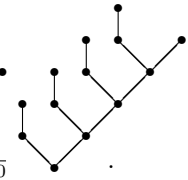
Order 5: $s = 2$,

$B_0(t)$: •,

$B_1(t)$: $\frac{1}{2}$  + $\frac{1}{3}$  + $\frac{1}{8}$  .

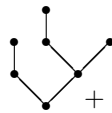
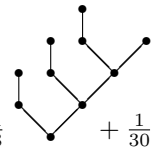
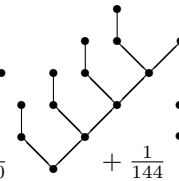
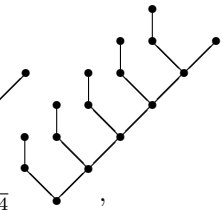
Order 6: $s = 2$,

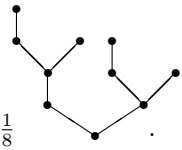
$B_0(t)$: •,

$B_1(t)$: $\frac{1}{2}$  + $\frac{1}{3}$  + $\frac{1}{8}$  + $\frac{1}{30}$  .

Order 7: $s = 3$,

$B_0(t)$: •,

$B_1(t)$: $\frac{1}{2}$  + $\frac{1}{3}$  + $\frac{1}{8}$  + $\frac{1}{30}$  + $\frac{1}{144}$  ,

$B_2(t)$: $\frac{1}{8}$  .

and so on. Note that just a subset of 'Magnus trees' occurs in the expansion. This representation of expansion terms as linear combination of trees is central to the application of the multivariate quadrature algorithms of the next section and derivation of practical numerical methods for the Fer expansion.

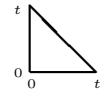
5. Quadrature and graded algebras

5.1. Multivariate quadrature over polytopes

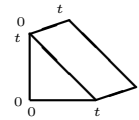
Casting our eyes again over the Magnus method (4.16–18) and considering its numerical implementation, let us discuss the computation of the first four integrals, noting that each need be carried out over a different polytope:

$$I_1(t) = \int_0^t A(\xi)\mathrm{d}\xi \qquad \text{over the line segment}$$

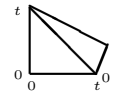
$$I_2(t) = \int_0^t \int_0^{\xi_1} [A(\xi_2), A(\xi_1)]\mathrm{d}\boldsymbol{\xi} \qquad \text{over the triangle}$$



$$I_3(t) = \int_0^t \int_0^{\xi_1} \int_0^{\xi_1} [A(\xi_2), [A(\xi_3), A(\xi_1)]]\mathrm{d}\boldsymbol{\xi} \qquad \text{over the prism}$$



$$I_4(t) = \int_0^t \int_0^{\xi_1} \int_0^{\xi_2} [[A(\xi_3), A(\xi_2)], A(\xi_1)]\mathrm{d}\boldsymbol{\xi} \qquad \text{over the pyramid}$$



Unless we can replace all these integrals by affordable and accurate quadrature, the Magnus method (and by the same token the Fer method) of non-trivial order is of little but theoretical value. Yet, multivariate quadrature is notoriously expensive in terms of function evaluations (Cools 1997). Fortunately, the special nature of integrals occurring within the context of Magnus and Fer expansions renders them particularly suitable for numerical quadrature with a remarkably small number of function evaluations (Iserles and Nørsett 1999).

We commence by observing that, time-stepping with step $h > 0$, each Magnus or Fer expansion term is of the form

$$I(h) = \int_{\mathcal{S}} \boldsymbol{L}(A(\xi_1), A(\xi_2), \ldots, A(\xi_s))\mathrm{d}\boldsymbol{\xi}, \tag{5.1}$$

where $\boldsymbol{L}$ is a *multilinear form,* while $\mathcal{S}$ is a polytope of a special form,

$$\mathcal{S} = \{\boldsymbol{\xi} \in \mathbb{R}^s \,:\, \xi_1 \in [0,h],\ \ \xi_l \in [0,\xi_{m_l}],\ l = 2,3,\ldots,s\},$$

where $m_l \in \{1,2,\ldots,l-1\}$, $l = 2,3,\ldots,s$. Thus, for example,

$$\int_0^t \int_0^{\xi_1} \int_0^{\xi_1} [A(\xi_2), [A(\xi_3), A(\xi_1)]]\mathrm{d}\boldsymbol{\xi} \qquad \text{and} \qquad \int_0^t \int_0^{\xi_1} \int_0^{\xi_2} [[A(\xi_3), A(\xi_2)], A(\xi_1)]\mathrm{d}\boldsymbol{\xi}$$

yield $s = 3$ and

$$\boldsymbol{L}(E_1, E_2, E_3) = [E_2, [E_3, E_1]], \qquad m_2 = 1,\ m_3 = 1$$

and

$$\boldsymbol{L}(E_1, E_2, E_3) = [[E_3, E_2], E_1], \qquad m_2 = 1,\ m_3 = 2$$

respectively.

Following Iserles and Nørsett (1999), we propose to discretise $I(h)$ as follows: Choose ν distinct quadrature points, $c_1, c_2, \ldots, c_\nu \in [0,1]$, evaluate $A_k = hA(c_k h)$,

$k = 1, 2, \ldots, \nu$ and form the quadrature

$$K(h) = \sum_{\boldsymbol{k} \in \mathrm{C}_s^\nu} b_{\boldsymbol{k}} \boldsymbol{L}(A_{k_1}, A_{k_2}, \ldots, A_{k_s}), \tag{5.2}$$

where C_s^ν is the set of all combinations of length s from the set $\{1, 2, \ldots, \nu\}$. The weights are

$$b_{\boldsymbol{k}} = \int_{\tilde{\mathcal{S}}} \prod_{i=1}^{s} \ell_{k_i}(\xi_i) \mathrm{d}\boldsymbol{\xi}, \tag{5.3}$$

where

$$\tilde{\mathcal{S}} = \{\boldsymbol{\xi} \in \mathbb{R}^s \,:\, \xi_1 \in [0,1],\ \ \xi_l \in [0, \xi_{m_l}],\ l = 2, 3, \ldots, s\}$$

is the polytope $\mathcal{S}$ scaled to the unit cube and

$$\ell_j(x) = \prod_{\substack{i=1 \\ i \neq j}}^{\nu} \frac{x - c_j}{c_i - c_j}, \qquad j = 1, 2, \ldots, \nu,$$

are the familiar *cardinal polynomials of Lagrange's interpolation.* Note that (5.3) are *interpolatory weights:* they follow naturally by substituting the interpolation polynomial at the quadrature points,

$$\tilde{A}(t) = h^{-1} \sum_{k=1}^{\nu} \ell_k\left(\frac{t}{h}\right) A_k \tag{5.4}$$

in place of $A(t)$ in (5.1) and carrying out the integration explicitly. Conceptually, (5.2) recycles ν function values at all possible combinations at the s 'slots' of the multilinear function $\boldsymbol{L}$.

How well does the quadrature (5.2) approximate the integral (5.1)? The answer is straightforward in the case $s = 1$, $\boldsymbol{L}(E) = E$, since we recover standard univariate interpolatory quadrature, which is of order $\nu + m$, where $m \geq 0$ is the largest integer so that

$$\int_0^1 \xi^{i-1} c(\xi) \mathrm{d}\xi = 0, \quad i = 1, 2, \ldots, m \qquad \text{where} \qquad c(t) = \prod_{k=1}^{\nu} (t - c_k) \tag{5.5}$$

is the *collocation polynomial.*

Theorem 5.1 The orthogonality condition (5.5) implies that the quadrature rule (5.2) is of order $\nu + m$ for all polytopes $\mathcal{S}$ and all multilinear forms $\boldsymbol{L}$. In particular, if $c_1, c_2, \ldots, c_\nu$ are the roots of the *Legendre polynomial* P_ν, shifted to the interval $[0, 1]$ (*Gauss–Legendre points*), then the quadrature is of order 2ν.

We do not propose to present the proof from (Iserles and Nørsett 1999), which is long, technical and not particularly illuminating. In the sequel, in the context of Magnus methods for nonlinear Lie-group equations, we describe a much clearer argument due to Zanna (1999) that explains why a suitable linear combination of quadratures (5.2) approximates the truncated Magnus expansion to the order reported in Theorem 5.1. Instead, we present an example, the quadrature of the four integrals that have opened this subsection using (5.2) at three Gauss–Legendre points.

Letting $\nu = 3$, hence order six, we have

$$c_1 = \tfrac{1}{2} - \tfrac{\sqrt{15}}{10}, \qquad c_2 = \tfrac{1}{2}, \qquad c_3 = \tfrac{1}{2} + \tfrac{\sqrt{15}}{10},$$

whence

$$\begin{aligned}
\ell_1(x) &= \frac{5+\sqrt{15}}{6} - \frac{10+\sqrt{15}}{3}x + \frac{10}{3}x^2,\\
\ell_2(x) &= -\frac{2}{3} + \frac{20}{3}x - \frac{20}{3}x^2,\\
\ell_3(x) &= \frac{5-\sqrt{15}}{6} - \frac{10-\sqrt{15}}{3}x + \frac{10}{3}x^2.
\end{aligned}$$

Insofar as the univariate integral I_1 is concerned, we have the familiar Gauss–Legendre quadrature,

$$I_1(h) \approx \tfrac{1}{18}(5A_1 + 8A_2 + 5A_3),$$

while, after taking into account skew-symmetry of commutators, the quadrature of the planar integral is

$$I_2(h) \approx \tfrac{\sqrt{15}}{54}(2[A_1, A_2] + [A_1, A_3] + 2[A_2, A_3]].$$

Insofar as the two cubic integrals are concerned, we are interested (cf. (4.18)) in their linear combination which, after a great deal of simplification, yields

$$\begin{aligned}
\tfrac{1}{12}I_3(h) + \tfrac{1}{4}I_4(h) \;\approx\; & \tfrac{1}{27216}(94[A_1,[A_1,A_2]] + 45[A_1,[A_1,A_3]]\\
& + 194[A_1,[A_2,A_3]] - 152[A_2,[A_1,A_2]] + 152[A_2,[A_2,A_3]]\\
& - 194[A_3,[A_1,A_2]] - 45[A_3,[A_1,A_3]] - 94[A_3,[A_2,A_3]]).
\end{aligned} \tag{5.6}$$

Before any attempts are made either to anoint (5.2) as a new wonder-quadrature or to expand efforts to optimise its calculation, we hasten to say that it is suboptimal! Indeed, it is an immediate consequence of Theorem 4.2 that $\frac{1}{12}I_3(h)+\frac{1}{4}I_4(h) = \mathcal{O}\big(h^4\big)$, rather than $\mathcal{O}\big(h^3\big)$. Moreover, some commutators can be expressed in terms of other commutators. Although all this can be done on an *ad hoc* basis, it is significantly better and more efficient to understand this phenomenon mathematically and apply the fruits of our understanding not just to truncated Magnus method but also to other Lie-group solvers.

Quadrature (5.2) scores exceedingly well in terms of function evaluations: *the number of function evaluations required to compute all the integrals in a Magnus expansion to requisite order is the same as the cost of the corresponding univariate Gauss–Legendre quadrature!* The trade-off, though, is that this approach requires a very large volume of linear-algebra calculations, since the number of all combinations is inordinately large. Fortunately, the cost of linear algebra can be reduced a very great deal by exploiting the theory of free Lie-algebras. This salutary example of a mathematical theory purer than a driven snow finding a very practical application in the design of numerical algorithms is told in the remainder of this section.

5.2. The self-adjoint basis

The first step in the effort to improve the multivariate quadrature formula (5.2) is a change of basis. As suggested first by Munthe-Kaas and Owren (1999), we choose $c_1 < c_2 < \cdots c_\nu$ *symmetric with respect to* $\frac{1}{2}$ and replace the function values $A_1, A_2, \ldots, A_\nu$ with the solution of the *Vandermonde system*

$$\sum_{l=1}^{\nu} (c_k - \tfrac{1}{2})^{l-1} B_l = A_k, \qquad k = 1, 2, \ldots, \nu. \tag{5.7}$$

We say that $\{B_1, B_2, \ldots, B_\nu\}$ is a *self-adjoint basis* and note for future reference that Gauss–Legendre points in $[0, 1]$, being symmetric with respect to $\frac{1}{2}$, lead to such basis.

Proposition 5.2 Given a sufficiently smooth matrix function A, it is true that $B_l = c_l h^l A^{(l-1)}(\frac{1}{2}h) + \mathcal{O}\big(h^{l+1}\big)$, where $c_l \neq 0$ is a scalar constant, $l = 1, 2, \ldots, \nu$. Moreover, each $h^{-l} B_l$ can be expanded in even powers of h.

It follows from the proposition that the interpolating polynomial can be written in the form

$$\tilde{A}(t) = h^{-1} \sum_{l=0}^{\nu-1} B_l (\tfrac{t}{h} - \tfrac{1}{2})^l. \tag{5.8}$$

Substituting this into (5.1) allows us to rephrase the quadrature formula (5.2) in a considerably more convenient form,

$$\bar{K}(h) = \sum_{\boldsymbol{l} \in \mathrm{C}_s^\nu} \bar{b}_{\boldsymbol{l}} \boldsymbol{L}(B_{l_1}, B_{l_2}, \ldots, B_{l_s}), \tag{5.9}$$

where

$$\bar{b}_{\boldsymbol{l}} = \int_{\bar{S}} \prod_{i=1}^{s} (\xi_i - \tfrac{1}{2})^{l_i - 1} \mathrm{d}\boldsymbol{\xi}. \tag{5.10}$$

Proposition 5.3. (Munthe-Kaas and Owren (1999)) Suppose that a linear combination of integrals $I(h)$ can be expanded in odd powers of h. Then so can be the linear combination of quadratures (5.9).

Recall from Section 4.3 that Magnus series, truncated by power, conform with the assumptions of Proposition 5.3: at a stroke, roughly half commutators go away. Yet, this is but the first of three important savings that are a consequence of the change of a basis.

An element $F \in \mathfrak{g}$ constructed from the basis terms $B_1, B_2, \ldots, B_\nu$ by the standard Lie-algebra operations of linear combination and commutation is said to be of *grade* m if $F = \mathcal{O}(h^m)$ for all sufficiently smooth matrix functions A. This is denoted by $\omega(F) = m$. We note from Proposition 5.2 that $\omega(B_l) = l$, $l = 1, 2, \ldots, \nu$. Moreover, the grade is inherited under commutation, e.g. $\omega([B_k, B_l]) = \omega(B_k) + \omega(B_l)$ and, with greater generality,

$$\omega(\boldsymbol{L}(B_{l_1}, B_{l_2}, \ldots, B_{l_s})) = |\boldsymbol{l}| = \sum_{i=1}^{\nu} l_i.$$

By the definition of the grade, this is equivalent to

$$\boldsymbol{L}(B_{l_1}, B_{l_2}, \ldots, B_{l_s}) = \mathcal{O}\Big(h^{|\boldsymbol{l}|}\Big).$$

Thus, as long as we are interested in an order-p quadrature, we can discard higher-order terms in (5.10). The outcome is

$$\hat{K}(h) = \sum_{\boldsymbol{l} \in \hat{\mathrm{C}}_s^{\nu,p}} \bar{b}_{\boldsymbol{l}} \boldsymbol{L}(B_{l_1}, B_{l_2}, \ldots, B_{l_s}), \tag{5.11}$$

where $\hat{\mathrm{C}}_s^{\nu,p} \subseteq \mathrm{C}_s^{\nu}$ such that

$$\boldsymbol{l} \in \hat{\mathrm{C}}_s^{\nu,p} \qquad \Leftrightarrow \qquad |\boldsymbol{l}| \le p.$$

Let us recall the four integrals from Section 5.1. We presently obtain using (5.11) the following order-6 quadrature formulae using Gauss–Legendre points with $\nu = 3$. In line with (5.7), we let $B_1 = A_2$, $B_2 = \frac{\sqrt{15}}{10}(A_3 - A_1)$ and $B_3 = \frac{20}{3}(A_3 - 2A_2 + A_1)$.

$$\begin{aligned}
I_1(h) &\approx B_1 + \tfrac{1}{12} B_3, \\
I_2(h) &\approx \tfrac{1}{6}[B_2, B_1] - \tfrac{1}{120}[B_3, B_2], \\
I_3(h) &\approx -\tfrac{1}{8}[B_1, [B_2, B_1]] + \tfrac{1}{80}[B_2, [B_2, B_1]] - \tfrac{1}{120}[B_1, [B_3, B_1]] \\
&\quad + \tfrac{1}{480}[B_1, [B_3, B_2]] + \tfrac{1}{240}[B_2, [B_3, B_1]] - \tfrac{7}{480}[B_3, [B_2, B_1]] \\
&\quad - \tfrac{1}{160}[B_1, [B_3, B_1]], \\
I_4(h) &\approx \tfrac{1}{24}[B_1, [B_2, B_1]] + \tfrac{1}{80}[B_2, [B_2, B_1]] - \tfrac{1}{120}[B_1, [B_3, B_1]] \\
&\quad - \tfrac{1}{1440}[B_1, [B_3, B_2]] - \tfrac{1}{720}[B_2, [B_3, B_1]] + \tfrac{7}{1440}[B_3, [B_2, B_1]] \\
&\quad + \tfrac{1}{480}[B_1, [B_3, B_1]].
\end{aligned}$$

Moreover, consistently with Proposition 5.3, we have

$$\tfrac{1}{12}I_3(h) + \tfrac{1}{4}I_4(h) \approx \tfrac{1}{240}[B_2,[B_2,B_1]] - \tfrac{1}{360}[B_1,[B_3,B_1]].$$

Just two terms survive!

Taken together, throwing away terms of high enough grade and the removal of terms with even $|\boldsymbol{l}|$ after summation removes a high proportion of commutators. Having said this, the most important feature of the self-adjoint basis that allows us to reduce the computational cost has not been mentioned mentioned yet!

We have already exploited skew symmetry of the commutator in the derivation and 'beautification' of our integration formulae. This is a fairly transparent procedure. However, let us recall that the commutator is also subject to the *Jacobi identity* (2.10). This allows for a very powerful mechanism to express commutators as linear combinations of other commutators and leads to results that are of importance not just to Magnus expansions, but also to RK-MK methods and the evaluation of the BCH formula. This is the theme of the next subsection.

5.3. Free Lie algebras

In the previous subsections we have seen how to construct an approximation to the solution Y of the Lie-group differential equation $Y' = A(t)Y$ by means of linear combinations of matrices $G_1, G_2, \ldots, G_\nu \in \mathfrak{g}$, whereby the G_i are either 'samples' of the matrix function $A(t)$, in which case $G_i = hA(c_i h)$, or terms of the self-adjoint basis, in which case the G_is coincide with the matrices B_i of Section 5.2.

It is clear that, if we want to make the best out of the properties of the commutator (skew-symmetry and Jacobi identity) it is useful to depart from specific representations (A_is and B_is) and treat the G_is as abstract objects in an abstract algebra $\mathfrak{g}$ that embodies the structure that is common to all Lie algebras but nothing more. This is the main idea behind *free Lie algebras*, a formalisation of a Lie algebra whose terms can be generated by means of brackets of pairwise elements and such that there are no reducing mechanisms other than skew-symmetry and the Jacobi identity of the commutator.

More precisely, the following definition formalises the concept of free Lie algebras presented above (Munthe-Kaas and Owren 1999) .

Definition 5.1 Let I be a set of indices, either finite or countable. A Lie algebra $\mathfrak{g}$ is *free* over the set I if

i) For every index $i \in I$ there exists $G_i \in \mathfrak{g}$;
ii) For any Lie algebra $\mathfrak{h}$ and any function $i \in I \mapsto H_i \in \mathfrak{h}$ there exists a unique Lie algebra homomorphism $\pi : \mathfrak{g} \to \mathfrak{h}$ such that $\pi(G_i) = H_i$ for all $i \in I$.

Moreover, $\mathcal{S} = \{G_i\}_{i\in I}$ is called the *set of generators* of the free Lie algebra $\mathfrak{g}$.

In our exposition it is useful to think of a free Lie algebra (FLA) as a linear space and to describe it in terms of a basis. One of the most popular is a *Hall basis* $\mathcal{H}$ that

contains the generators, $\mathcal{S} \subseteq \mathcal{H}$ (Bourbaki 1975). All elements of $\mathcal{H}$ are produced by recursive commutation of generators. We can associate a *length* function to each element in the following fashion: $\ell(G_i) = 1$ for $G_i \in \mathcal{S}$ and $\ell(H) = \ell(H_1) + \ell(H_2)$ for all $H \in \mathcal{H} \setminus \mathcal{S}$, where $H = [H_1, H_2]$. Intuitively, we may say that the length of H corresponds to how many commutators of generators are needed to construct H. In other words, the length function merely counts commutators.

The Hall basis $\mathcal{H}$ can be endowed with a total ordering defined recursively on the length ℓ of its elements. In general, we say that $G \prec H$ if $\ell(G) < \ell(H)$. If $\ell(G) = \ell(H)$ then $G \prec H$ if G precedes H in lexicographic order. Moreover, to take into account skew-symmetry and the Jacobi identity, we require that

- elements of length two $[G_i, G_j]$ are included in $\mathcal{H}$ if $G_i \prec G_j$;
- elements of length greater or equal to three are included in $\mathcal{H}$ if they are of the form $[H_i, [H_j, H_k]]$, with $H_i, H_j, H_k, [H_j, H_k] \in \mathcal{H}$ and moreover $H_j \preceq H_i \prec [H_j, H_k]$.

An example of the first terms of a Hall basis generated by three elements G_1, G_2, G_3 is given by

$$
\begin{array}{lllll}
G_1, & G_2, & G_3, & [G_1, G_2], & [G_1, G_3], \\
[G_2, G_3], & [G_1, [G_1, G_2]], & [G_1, [G_1, G_3]], & [G_2, [G_1, G_2]], & [G_2, [G_1, G_3]], \\
[G_2, [G_2, G_3]], & [G_3, [G_1, G_2]], & [G_3, [G_1, G_3]], & [G_3, [G_2, G_3]], & \ldots.
\end{array}
$$

We shall not go into details of algorithmic construction of the Hall basis, which can be found in (Bourbaki 1975, Munthe-Kaas and Owren 1999). It is interesting, however, to mention how fast the number of elements of the Hall basis grows: assuming that I is finite and consists of ν indices (equivalently, $\mathcal{S}$ consists of ν generators), the linear subspace of terms of length exactly equal to m has dimension

$$\rho_m = \frac{1}{m} \sum_{d|m} \mu(d) \nu^{m/d},$$

the sum being carried over all integers d dividing m, a result known as *Witt's formula*. The function μ is the *Möbius function,* defined as follows: assume that d can be factorized as $d = d_1^{n_1} d_2^{n_2} \cdots d_q^{n_q}$, with each d_i a prime number and $n_i \geq 1$. Then

$$\mu(d) = \begin{cases} 1, & d = 1, \\ (-1)^q, & n_i = 1 \text{ for all } i = 1, 2, \ldots, q, \\ 0. & \text{otherwise.} \end{cases}$$

The number ρ_m grows quite fast, as illustrated in Table 5.1 for $\nu = 3$.

Why is all this relevant to our discussion? Let us represent $G_1 = A_1, G_2 = A_2, \ldots, G_\nu = A_\nu$, where $A_i = hA(c_i h)$, for $i = 1, \ldots, \nu$. Since $A_i = \mathcal{O}(h)$, a term containing exactly m commutators corresponds to a combination that is at least of order $\mathcal{O}(h^m)$. Thus, in order to have a numerical approximation of order

Table 5.1. *Dimension of linear spaces of words of length equal to m in the Hall basis generated by G_1, G_2 and G_3.*

m	1	2	3	4	5	6	7	8	9	10
ρ_m	3	3	8	18	48	116	312	810	2184	5580

p, the number of linearly-independent terms that we need to take into account is bounded by $\sum_{m=1}^{p} \rho_m$ or, because of Theorem 4.2, by $\sum_{m=1}^{p-1} \rho_m$ if the c_i are symmetrically distributed with respect to $\frac{1}{2}$. For instance, for a sixth-order Gauss–Legendre scheme ($\nu = 3$), this leads to 80 linearly independent terms of the Hall basis, not counting the number of commutators involved!

The growth of ρ_m can be reduced introducing a *grading* ω of the FLA $\mathfrak{g}$, as suggested by Munthe-Kaas and Owren (1999). Assume that

$$\omega(G_i) = \omega_i \in \mathbb{N}, \qquad G_i \in \mathcal{S}, \quad i = 1, 2, \ldots, \nu.$$

The grading propagates in a natural manner in the Hall basis $\mathcal{H}$: for all $H \in \mathcal{H}$ of the form $H = [H_1, H_2]$ we let

$$\omega(H) = \omega(H_1) + \omega(H_2).$$

A consequence of the grading is that the Hall basis $\mathcal{H}$ splits into a disjoint union of sets of grade m,

$$\mathcal{H} = \bigcup_{m=1}^{\infty} \mathcal{H}_m, \qquad \mathcal{H}_m = \{H \in \mathcal{H} : \omega(H) = m\},$$

consequently $\mathfrak{g}$ becomes a direct sum of subspaces,

$$\mathfrak{g} = \coprod_{m=1}^{\infty} \mathfrak{g}_m, \qquad \mathfrak{g}_m = \operatorname{span} \mathcal{H}_m,$$

and we say that $\mathfrak{g}$ is a *graded FLA algebra*.

The fundamental result on the dimension of $\mathfrak{g}_m$ is due to Munthe-Kaas and Owren (1999).

Theorem 5.4 Let $\mathfrak{g}$ be the graded FLA generated by $\mathcal{S} = \{G_1, G_2, \ldots, G_\nu\}$, with grades $\omega_1, \omega_2, \ldots, \omega_\nu$ respectively. Denote by $\lambda_1, \lambda_2, \ldots, \lambda_r$ the roots of the rth degree polynomial

$$p(z) = 1 - \sum_{i=1}^{\nu} z^{\omega_i}, \qquad r = \max_{1 \le i \le \nu} \omega_i.$$

Then

$$\dim \mathfrak{g}_m = \bar{\rho}_m = \frac{1}{m} \sum_{d|m} \left(\sum_{i=1}^{r} \lambda_i^{m/d} \right) \mu(d). \tag{5.12}$$

To illustrate the benefits of the grading, Table 5.2 displays $\dim \mathfrak{g}_m$, for $m = 1, 2, \ldots, 10$ for a graded algebra generated by G_1, G_2, G_3 with weights $\omega_1 = 1, \omega_2 = 2, \omega_3 = 3$. Comparison with Table 5.1 reveals a significant reduction in the number of linearly independent terms. The results have been obtained using the MATLAB package *DiffMan*, which will be further discussed in Section 10.2 (Engø et al. 1999).

Table 5.2. *Dimension of linear spaces of terms of weight exactly equal to m (top) and weight at most m (bottom) in the Hall basis generated by G_1, G_2 and G_3 with weights 1, 2 and 3 respectively.*

m	1	2	3	4	5	6	7	8	9	10
$\bar{\rho}_m$	1	1	2	2	4	5	10	15	26	42
$\sum_{i=1}^{m} \bar{\rho}_i$	1	2	4	6	10	15	25	40	66	108

Linking again with the theory of Section 5.1–2, the case of a graded algebra with generators $G_1, \ldots, G_\nu$ and weights $1, 2, \ldots, \nu$, corresponds to the realization

$$G_i = \frac{h^i}{(i-1)!} A^{(i-1)}(\xi_i), \qquad i = 1, 2, \ldots, \nu,$$

where $A^{(i-1)}(\xi_i)$ is the $(i-1)$th derivative of the function A for some $\xi_i \in (0, h)$. The weight ω_i merely indicates that G_i is an $\mathcal{O}(h^i)$ term. Moreover,

$$\tilde{A}(t) = h^{-1} \sum_{i=1}^{\nu} G_i (\tfrac{t}{h})^i,$$

is exactly the collocation polynomial (5.4), the information about the nodes c_i being hidden in the G_is. The equivalence is revealed by means of the Vandermonde transformation

$$A_i = \sum_{j=1}^{\nu} c_j^{j-1} G_j, \qquad i = 1, \ldots, \nu.$$

This procedure applies to all kinds of collocation, whether with symmetric nodes or otherwise. In particular we deduce from Table 5.2 that with three collocation points is is possible to obtain order six with at most 15 terms! A substantial saving, compared with 80 using an ungraded algebra. . . .

To reduce further the dimension of the graded FLA, we exploit the argument

of Theorem 5.2, assuming that the collocation points are symmetric in [0,1] with respect to $\frac{1}{2}$. We construct

$$\tilde{A}(t) = h^{-1} \sum_{i=1}^{\nu} G_i (\tfrac{t}{h} - \tfrac{1}{2})^i,$$

where $G_i \equiv B_i$ are the *self-adjoint basis* introduced in Section 5.2. Theorem 5.2 implies that *only terms with odd grades need be considered!* As an example, if $\mathcal{S} = \{G_1, G_2, G_3\}$, with weights $1, 2$ and 3 respectively, then the growth of the dimension of the graded FLA is given by

m	1	2	3	4	5	6	7	8	9	10
$\sum_{\substack{i=1 \\ i \text{ odd}}}^{m} \bar{\rho}_i$	1	1	3	3	7	7	11	11	37	37

and comparison with Table 5.2 is remarkable. In particular, we deduce that for methods based on three symmetric collocation points (for instance a sixth-order Gauss–Legendre), we need at most seven terms of the graded Hall basis $\mathcal{H}$. Letting $G_i = B_i$, the seven terms are

$$B_1,\ B_3,\ [B_1, B_2],\ [B_2, B_3],\ [B_1, [B_1, B_3]],\ [B_2, [B_1, B_2]],\ [B_1, [B_1, [B_1, B_2]]].$$

Bounds on the number of independent terms for different orders are given below for methods based on Gauss–Legendre quadrature. The sharpest bound corresponds to the case of a graded FLA including only odd terms.

ν (stages)	1	2	3	4	5
2ν (order of the method)	2	4	6	8	10
$\sum_{i=1}^{2\nu-1} \rho_i$	1	5	80	3304	> 10000
$\sum_{i=1}^{2\nu-1} \bar{\rho}_i$	1	3	10	33	111
$\sum_{\substack{i=1 \\ i \text{ odd}}}^{2\nu-1} \bar{\rho}_i$	1	2	7	22	73

It is important to note that

$$\sum_{i=1}^{p-1} \bar{\rho}_{i \text{ odd}} \qquad \left(\sum_{i=1}^{p} \rho_i \quad \text{respectively} \right)$$

in the case of the self-adjoint (non self-adjoint resp.) basis is an *upper bound* on the number of linearly-independent commutators required for a method of order p.

We have seen the advantage of changing the basis in the case of linear equations $Y' = A(t)Y$. Similar savings can be also achieved in the case of explicit RK-MK methods for the general equation $Y' = A(t, Y)Y$. By combining the stage values A_i computed by the algorithm, we seek linear combinations B_i of highest-possible grade. However, in order to obtain an explicit method, we must require that B_i are related to A_i by a triangular matrix. Optimal combinations can be found using linear algebra and the theory of B-series. The details can be found in (Munthe-Kaas and Owren 1999). It turns out that in this case it is in general not possible to change basis in such a way that $\omega(B_i) = i$: the weights grow slower. The method (*A*.7) originates in the classical four-stage RK4 scheme. The resulting basis B_i has the grading $1, 2, 3, 3$, and the scheme has just two commutators. Similar savings have been realised for the seven-stage DOPRI5(4) method in (Munthe-Kaas and Owren 1999).

5.4. Reducing further the number of commutators

The theory of free Lie algebras allows us to derive an upper bound on the number of linearly independent terms required to obtain numerical methods of given order, taking into account skew-symmetry and the Jacobi identity of the commutator. It also provides algorithms to compute the requisite pattern of dependency, e.g. in terms of the Hall basis.

Although the theory of free Lie algebras estimates the numbers of commutators for a method of order p, this by no means indicates the *least* number of commutators required for a method of a given order.

For general $N \times N$ matrices, the computation of the commutator is an $\mathcal{O}(N^3)$ operation, a cost that quickly adds up when we consider methods of order three and higher.

At present there is no theory that systematically reduces the number of commutators to minimum. However, we shall present a technique, due to Blanes, Casas and Ros (1999) that gives, true for today, the least number of commutators for methods based on Gauss–Legendre and Newton–Cotes quadratures up to order ten. Before proceeding further, it is important to remark that the content of this section applies to linear Lie group problems $Y' = A(t)Y$ solved with Magnus-type/RK-MK methods based on symmetric nodes in $[0, 1]$. It is convenient to illustrate the procedure with an example. Let us thus construct a sixth-order Gauss–Legendre method based on the collocation nodes

$$c_1 = \tfrac{1}{2} - \tfrac{\sqrt{15}}{10}, \qquad c_2 = \tfrac{1}{2}, \qquad c_3 = \tfrac{1}{2} + \tfrac{\sqrt{15}}{10}.$$

Using the toolbox of graded algebras, we obtain

$$\begin{aligned}\Theta \;=\; & B_1 + \tfrac{1}{12}B_3 - \tfrac{1}{12}[B_1, B_2] + \tfrac{1}{240}[B_2, B_3] + \tfrac{1}{360}[B_1, [B_1, B_3]] \\ & - \tfrac{1}{240}[B_2, [B_1, B_2]] + \tfrac{1}{720}[B_1, [B_1, [B_1, B_2]]].\end{aligned}$$

Since in the self-adjoint basis $B_i = \mathcal{O}(h^i)$, note that Θ includes terms with odd powers of h only and, in this form, can be evaluated by computing just seven commutators.

Let us focus on the portion of Θ consisting of single commutators,

$$C_2 = -\tfrac{1}{12}[B_1, B_2] + \tfrac{1}{240}[B_2, B_3].$$

The first fundamental observation is that terms of the form $[B_i, B_j]$ can only be obtained if one of the indices is even and the other is odd. Therefore, we look for a linear combination

$$[b_1 B_1 + b_3 B_3, b_2 B_2] \tag{5.13}$$

for some real coefficients b_i, that equals C_2. This can be achieved by choosing for instance $b_1 = -\frac{1}{12}, b_3 = -\frac{1}{240}$ and $b_2 = 1$. Note that computing (5.13) requires one commutator only instead of two. In general, given $B_1, B_2, \ldots, B_\nu$, terms of the form

$$\sum_{i=1}^{\nu-1} \sum_{j=i+1}^{\nu} k_{i,j}[B_i, B_j]$$

may be replaced by a *single* commutator,

$$\left[\sum_{i=1}^{\nu/2} b_{2i-1} B_{2i-1}, \sum_{j=1}^{\nu/2} b_{2j} B_{2j}\right]$$

provided that the coefficients $\{b_i\}_{1\le i\le \nu}$ and $\{k_{i,j}\}_{1\le i<j\le \nu}$ are compatible up to the order of the method.

Next, let us consider terms with double commutators. Let us focus on

$$-\tfrac{1}{240}[B_2, [B_1, B_2]].$$

Since $C_2 = -\frac{1}{12}[B_1, B_2] + \mathcal{O}(h^5)$, evaluating the term $-\frac{1}{20}[B_2, C_2]$ in place of $-\frac{1}{240}[B_2, [B_1, B_2]]$ amounts to the calculation of just one commutator. Note that $-\frac{1}{20}[B_2, C_2] = -\frac{1}{240}[B_2, [B_1, B_2]] + \mathcal{O}(h^7)$, hence the approximation retains the odd-power of h expansion and the error introduced is subsumed in the local truncation error of the method.

Finally, let us consider the combination

$$\tfrac{1}{360}[B_1, [B_1, B_3]] + \tfrac{1}{720}[B_1, [B_1, [B_1, B_2]]]. \tag{5.14}$$

Clearly, $[B_1, B_3]$ is not obtained from (5.13) using the linearity of the bracket, and needs to be taken into account. However, $[B_1, B_2]$ conforms with (5.13) and we can replace it with C_2, introducing only odd-powered error in h which is also subsumed in the local truncation error. In summary, (5.14) can be replaced by the term

$$C_3 = [B_1, [B_1, \tfrac{1}{360} B_3 - \tfrac{1}{60} C_2]]$$

which requires the computation of two commutators. Therefore, Θ can be computed in the form

$$\Theta = B_1 + \tfrac{1}{12} B_3 + C_2 + C_3,$$

requiring four commutators only.

It is difficult to formalise the last two steps, involving two or more commutators. The reduction in the number of commutators has, in present state of knowledge, to be done on a case-by-case analysis.

Some examples of methods for linear problems obtained by means of graded algebras, made more economic with the technique of Blanes et al. (1999), are described in Appendix A.2.

5.5. Nonlinear problems: collocation methods

Magnus and Fer expansions presented in Section 4 have been designed for linear problems, when the matrix function A depends on time only. When $A = A(t, Y)$, though, multivariate integrals appearing in (5.1) depend on $A_k = hA(c_k h, Y(c_k h))$, namely also on the value of the unknown variable Y at quadrature points. Assume that the quadrature points $c_1, c_2, \dots c_\nu$ obey the orthogonality conditions (5.5) for some $m \le \nu$, hence the corresponding univariate interpolatory quadrature has order $p = \nu + m$. Since each A_k is a multiple of h, it is clear that the values A_k can be replaced by $hA(c_k h, X_k)$, where X_k is an approximation to $Y(c_k h)$ of order at least $p - 1$. Following this point of view, it is possible to derive $X_k \approx Y(c_k h)$ with a numerical method of order $p - 1$. For instance, using an RK-MK method of order $p - 1$ and a Magnus (or Fer) expansion of order p is a Lie-group equivalent of a *predictor–corrector* method in classical numerical ODE theory.

A more elegant approach, due to Zanna (1999), is to construct suitable approximants to X_ks using directly the underlying principle of collocation methods. Proceeding as in Section 5.1, we replace the function $A(t, Y)$ by its Lagrangian interpolating polynomial

$$\tilde{A}(t, Y) = h^{-1} \sum_{k=1}^{\nu} \ell_k \left(\frac{t}{h} \right) A_k$$

where $A_k = hA(c_k h, X_k)$, $k = 1, 2, \dots, \nu$. The corresponding *dexpinv equation* is integrated by means of Picard iterations *á la* Section 4, obtaining the usual order trees for the Magnus and the Fer expansion. The main difference with the linear case is that now the same order trees are also employed to evaluate the integration coefficients for the internal stages X_k. At each internal stage we need to calculate quadratures of the form (5.2),

$$K_l(h) = \sum_{\boldsymbol{j} \in \mathrm{C}_s^\nu} a_{l;\boldsymbol{j}} \boldsymbol{L}(A_{j_1}, A_{j_2}, \dots, A_{j_s}),$$

where C_s^ν is the set of all combinations of length s from the set $\{1, 2, \dots, \nu\}$. The

integration weights are different from those in (5.2) and are defined as

$$a_{k;\boldsymbol{j}} = \int_{\tilde{\mathcal{S}}_k} \prod_{i=1}^{s} \ell_{j_i}(\xi_i) \mathrm{d}\boldsymbol{\xi},$$

where

$$\tilde{\mathcal{S}}_k = \{\boldsymbol{\xi} \in \mathbb{R}^s \,:\, \xi_1 \in [0, c_k],\ \ \xi_l \in [0, \xi_{m_l}],\ l = 2, 3, \ldots, s\}$$

is the polytope $\tilde{\mathcal{S}}$ scaled to the $[0, c_k]$-cube instead of the unit cube. The weights $b_{\boldsymbol{j}}$ are recovered by substituting $c_k = 1$.

Theorem 5.5 Let $c_1, c_2, \ldots, c_\nu$ be ν collocation nodes and let $p = \nu + m$, where m is the largest index such that (5.5) is satisfied. Assume that the Magnus or Fer expansion is truncated to include all trees of power $q \le p$ for the evaluation of Y_{n+1}, and of power $q \le p - 1$ for the intermediate stages. Then the resulting scheme has order p.

We sketch the main idea and refer the reader to (Zanna 1999) for details. The starting point is the *Alekseev–Gröbner lemma*, a nonlinear version of the variation of constants formula whose proof can be found in (Nørsett and Wanner 1981), stating that, if $\boldsymbol{y}$ is the solution of the differential equation $\boldsymbol{y}' = \boldsymbol{f}(t, \boldsymbol{y})$ with initial condition $\boldsymbol{y}_0$, and if $\boldsymbol{w}(t)$ is a C^1 approximation to $\boldsymbol{y}$ such that $\boldsymbol{w}(t_0) = \boldsymbol{y}_0$, then

$$\boldsymbol{y}(t) - \boldsymbol{w}(t) = \int_{t_0}^{t} \Phi(t, \xi, \boldsymbol{w}(\xi))[\boldsymbol{f}(\xi, \boldsymbol{w}(\xi)) - \boldsymbol{w}'(\xi)]\mathrm{d}\xi,$$

where $\Phi(t, \xi, \boldsymbol{w}(\xi))$ is the partial derivative of the the solution passing through $(\xi, \boldsymbol{w}(\xi))$ with respect to the initial condition $\boldsymbol{w}(\xi)$. In the usual classical collocation setting for RK methods, $\boldsymbol{w}$ corresponds to the case when the function $\boldsymbol{f}$ is replaced by a collocation polynomial: at the nodes it is true that $\boldsymbol{f}(t_n + c_k h, \boldsymbol{w}(t_n + c_k h)) - \boldsymbol{w}'(t_n + c_k h)) = \boldsymbol{0}$, hence the error reduces solely to quadrature error.

In our Lie-group setting the main difference consists in the fact that only the function $A(t, Y)$ is collocated, and not the whole right-hand side $A(t, Y)Y$. However, this can be viewed as a collocation method in the algebra $\mathfrak{g}$, where classical analysis remains valid.

A typical example of such collocation methods is the fourth-order scheme

$$\begin{aligned}
X_1 &= Y_n, \\
A_1 &= hA(t_n, X_1), \\
X_2 &= \operatorname{expm}\{\tfrac{5}{24}A_1 + \tfrac{1}{3}A_2 - \tfrac{1}{24}A_3 - \tfrac{1}{2}(\tfrac{11}{240}[A_1, A_2] + \tfrac{5}{576}[A_1, A_3] \\
&\qquad + \tfrac{1}{72}[A_2, A_3])\}Y_n, \\
A_2 &= hA(t_n + \tfrac{1}{2}h, X_2), \\
X_3 &= \operatorname{expm}\{\tfrac{1}{6}A_1 + \tfrac{2}{3}A_2 + \tfrac{1}{6}A_3 - \tfrac{1}{2}(\tfrac{2}{15}[A_1, A_2] + \tfrac{1}{30}[A_1, A_3] \\
&\qquad + \tfrac{2}{15}[A_2, A_3])\}Y_n, \\
A_3 &= hA(t_n + h, X_3), \\
Y_{n+1} &= \operatorname{expm}\{\tfrac{1}{6}A_1 + \tfrac{2}{3}A_2 + \tfrac{1}{6}A_3 - \tfrac{1}{2}(\tfrac{2}{15}[A_1, A_2] + \tfrac{1}{30}[A_1, A_3] \\
&\qquad + \tfrac{2}{15}[A_2, A_3])\}Y_n
\end{aligned}$$

for $n \in \mathbb{Z}^+$, with the *Gauss–Lobatto* quadrature points $c_1 = 0$, $c_2 = \frac{1}{2}$ and $c_3 = 1$ (Zanna 1998). Note that the coefficients of the A_i's are the classical Runge–Kutta coefficients of Lobatto collocation scheme with the same quadrature points (Hairer et al. 1993), while the coefficients of the commutator terms $[A_i, A_j]$ are evaluated by integrating

$$a_{k;i,j} - a_{k;j,i} = \int_0^{c_k}\int_0^{\xi_1} \ell_i(\xi_1)\ell_j(\xi_2)\mathrm{d}\xi_2\mathrm{d}\xi_1 - \int_0^{c_k}\int_0^{\xi_1} \ell_j(\xi_1)\ell_i(\xi_2)\mathrm{d}\xi_2\mathrm{d}\xi_1,$$

a term corresponding to the power-three tree in the Magnus expansion (cf. Section 4).

A useful formula for evaluating the $a_{k;i,j}$s corresponding to the power-three tree is given by

$$\{a_{k;i,j}\}_{i,j=1}^{\nu} = a(c_k), \qquad k = 1, 2, \ldots, \nu, \tag{5.15}$$

where $a(\theta)$ is the $\nu \times \nu$ matrix function of the scalar argument θ defined as

$$a(\theta) = V^{-\mathrm{T}} T(\theta) H J T(\theta) V^{-1},$$

where $J = \operatorname{diag}(1, \frac{1}{2}, \frac{1}{3}, \ldots \frac{1}{\nu})$, $T(\theta) = \operatorname{diag}(\theta, \theta^2, \ldots, \theta^\nu)$, V is the Vandermonde matrix

$$V = \begin{bmatrix} 1 & c_1 & \cdots & c_1^{\nu-1} \\ 1 & c_2 & \cdots & c_2^{\nu-1} \\ \vdots & \vdots & & \vdots \\ 1 & c_\nu & \cdots & c_\nu^{\nu-1} \end{bmatrix},$$

and finally H is the Hilbert matrix with entries $H_{i,j} = \frac{1}{i+j}$, $i, j = 1, 2, \ldots, \nu$.

This formula is reminiscent of the matrix representation of the standard Runge–Kutta matrix of a collocation method (Nørsett and Wanner 1981). This is not a coincidence: the methods of Zanna (1998) generalise the concept of collocation to the special multivariate integrals that occur in Magnus or Fer expansions.

6. Alternative coordinates

All the methods so far, whether applied to Lie groups or in a homogeneous-space setting, have been based on the exponential map. In other words, we represented the solution as an exponential (RK-MK and Magnus methods) or as a product of exponentials (Crouch–Grossman and Fer methods). It is entirely legitimate to query to which extent this renders such methods unduly expensive and non-competitive.

Sometimes the exact computation of the exponential is easy: a case in point is the application of Magnus expansions to the computation of Sturm–Liouville spectra in Section 11.2, since the exponential of an element in $\mathfrak{sl}(2)$ can be evaluated exactly with great ease. In other cases the exponential can be replaced by a suitable approximation $\delta : \mathfrak{g} \to \mathcal{G}$, $\delta(z) \approx \mathrm{e}^z$. This is the case with quadratic Lie groups: A Lie group G is *quadratic* if

$$\mathcal{G} = \{X \in \mathrm{GL}(N) \,:\, XPX^{\mathrm{T}} = P\}, \tag{6.1}$$

where $P \in \mathrm{GL}(N)$ is a given matrix. Many Lie groups that appear in applications are of this kind, for example $\mathrm{O}(N)$, $\mathrm{Sp}(N)$ and $\mathrm{O}(N, M)$. Moreover, some complex groups can be brought into this framework by replacing the transpose $^{\mathrm{T}}$ by the Hermitian (i.e., conjugate) transpose $^{\mathrm{H}}$.

The Lie algebra of the quadratic Lie group (6.1) is

$$\mathfrak{g} = \{B \in \mathfrak{gl}(N) \,:\, BP + PB^{\mathrm{T}} = O\} \tag{6.2}$$

and it is easy to prove that $\delta : \mathfrak{g} \to \mathcal{G}$ whenever $\delta(z) = \mathrm{e}^{\gamma(z)}$ and γ is an odd function, analytic in the neighbourhood of the origin (Celledoni and Iserles 1998). An important case occurs when $\gamma(z) = \log q(z) - \log q(-z)$, where q is a polynomial, and it leads to rational functions $\delta(z) = q(z)/q(-z)$. In particular, *diagonal Padé approximants* to the exponential are of this form (Baker 1975) and they can be applied very effectively in place of the exponential.

Yet, for some algebras there exists no analytic nonconstant function $\delta : \mathfrak{g} \to \mathcal{G}$ except for the (scaled) exponential. This, in particular, is the case with $\mathrm{SL}(N)$ (Kang and Zai-jiu 1995). In yet other cases, although we may replace the exponential with, say, a Padé approximant with impunity, the sheer size of the system renders this impractical when the number of variables is large. In that case there exist two possibilities. Firstly, we may endeavour to approximate the exponential of a matrix by some nonstandard means while keeping the outcome in a Lie group. This is the theme of Section 8. In the present section we consider another approach, which disposes of the dexpinv equation (2.46) altogether.

The research into 'alternative coordinates' is in a fairly preliminary stage and just two surrogates to the dexpinv equation have been identified so far, the Cayley transform and canonical coordinates of the second kind. We consider them in detail in the remainder of this section.

6.1. The Cayley transform and RK–Cayley methods

Let $\mathcal{G}$ be a *quadratic* Lie group. The main idea is to replace the exponential with the *Cayley transform.* Thus, given the Lie-group equation $Y' = A(t, Y)Y$, where $A : \mathbb{R}^+ \times \mathcal{G} \to \mathfrak{g}$, we seek a solution in the form

$$Y(t) = \mathrm{cay}[\Delta(t)]Y_0 = [I - \tfrac{1}{2}\Delta(t)]^{-1}[I + \tfrac{1}{2}\Delta(t)]Y_0, \qquad t \geq 0, \tag{6.3}$$

and at the first instance seek a differential equation for Δ. More generally, we may solve the homogeneous-space equation (2.26) replacing the exponential with the Cayley action but, for the sake of simplicity, the discussion is restricted to the 'straight' Lie-group case.

It is important to realise that our approach has no connection whatsoever with approximating the exponential. True, $\delta(z) = (1+\frac{1}{2}z)/(1-\frac{1}{2}z)$ is a special case of a Padé approximant to the exponential, $\delta(z) = \mathrm{expm}(z) + \mathcal{O}(z^3)$, but this is entirely coincidental: as a matter of fact, we could have replaced, at the cost of slightly more complicated coefficients, the number $\frac{1}{2}$ with an arbitrary nonzero constant. So far, everything is exact and no approximation has taken place.

It is an easy exercise, left to the reader, to ascertain that the function Δ in (6.3) obeys the differential equation

$$\Delta' = \mathrm{dcay}_\Delta^{-1} A(\mathrm{cay}(\Delta)Y_0, t) = A - \tfrac{1}{2}[\Delta, A] - \tfrac{1}{4}\Delta A \Delta, \quad t \geq 0, \qquad \Delta(0) = \boldsymbol{O}. \tag{6.4}$$

Note the presence of the term $\Delta A \Delta$ in the above equation. In general, we cannot expect such a term to reside in $\mathfrak{g}$, but quadratic Lie algebras (6.2) are an exception. For future reference we note that the more general *symmetric triple product* $[\![D, E, F]\!]_3 = DEF + FED$ resides in $\mathfrak{g}$ for all $D, E, F \in \mathfrak{g}$ and quadratic Lie groups $\mathfrak{g}$. (The reason for this notation will be clear in Section 6.3.) Applying (6.2) thrice,

$$P(DEF)^{\mathrm{T}} = PF^{\mathrm{T}}E^{\mathrm{T}}D^{\mathrm{T}} = -FPE^{\mathrm{T}}D^{\mathrm{T}} = FEPD^{\mathrm{T}} = -FEDP.$$

Therefore

$$\begin{aligned} & (DEF + FED)P + P(DEF + FED)^{\mathrm{T}} \\ = \; & (DEF + FED)P - (FED + DEF)P = O \end{aligned}$$

and indeed $[\![D, E, F]\!]_3 \in \mathfrak{g}$. This confirms that no illicit terms have crept into (6.4) and Δ' evolves in $\mathfrak{g}$. (To be more precise, it evolves in $\mathrm{T}\mathfrak{g}$, except that the latter can be identified with $\mathfrak{g}$.)

The simplest implementation of (6.3) to quadratic Lie-group solvers is by employing Runge–Kutta methods in the Lie algebra, similarly to RK-MK, except that expm and dexp^{-1} in (3.4) need be replaced by cay and dcay^{-1} respectively. This has been accomplished systematically by Engø (1998).

6.2. Cayley expansions

Proceeding like in Section 4.1 and subjecting (6.4) to Picard iteration, we observe that the solution Δ can be expanded quite similarly to the Magnus expansion of the dexpinv equation (4.3),

$$\begin{aligned}\Delta(t) &= \int_0^t A(\xi)\mathrm{d}\xi - \tfrac{1}{2}\int_0^t\int_0^{\xi_1}[A(\xi_2),A(\xi_1)]\mathrm{d}\xi \\ &\quad + \tfrac{1}{4}\int_0^t\int_0^{\xi_1}\int_0^{\xi_2}[[A(\xi_3),A(\xi_2)],A(\xi_1)]\mathrm{d}\xi_3\mathrm{d}\xi_2\mathrm{d}\xi_1 \\ &\quad - \tfrac{1}{4}\int_0^t\int_0^{\xi_1}\int_0^{\xi_1}A(\xi_2)A(\xi_1)A(\xi_3)\mathrm{d}\xi_3\mathrm{d}\xi_2\mathrm{d}\xi_1 + \cdots .\end{aligned}$$

We seek to expand Δ, in greater generality, in a manner similar to (4.5),

$$\Delta(t) = \sum_{k=0}^{\infty}\sum_{\tau\in\mathbb{S}_k}\alpha(\tau)D_\tau(t), \qquad t\geq 0, \tag{6.5}$$

where each D_τ for $\tau\in\mathbb{S}_k$ is made out of exactly $k+1$ integrals. (Note that, unlike (4.5), the 'exterior' integral is already included in the expansion term – intuitively speaking, $D_\tau(t) = \int_0^t C_\tau(\xi)\mathrm{d}\xi$. This makes the notation somewhat simpler and more transparent.) Following upon the construction of Iserles (1999*b*), we identify three composition rules that are needed to assemble the terms in (6.5).

1 $\mathbb{S}_0 = \{\tau_\circ\}$, and
$$D_{\tau_\circ}(t) = \int_0^t A(\xi)\mathrm{d}\xi;$$

2 If $\tau_1\in\mathbb{S}_{k-1}$, $k\geq 1$, then there exists $\tau\in\mathbb{S}_k$ such that
$$D_\tau(t) = \int_0^t\left[\int_0^t D_{\tau_1}(\xi),A(\xi)\right]\mathrm{d}\xi; \tag{6.6}$$

3 If $k\geq 2$ and $\tau_1\in\mathbb{S}_{k-j}$, $\tau_2\in\mathbb{S}_{j-1}$ for some $1\leq j\leq k$ then there exists $\tau\in\mathbb{S}_k$ such that
$$D_\tau(t) = \int_0^t D_{\tau_1}(\xi)A(\xi)D_{\tau_2}(\xi)\mathrm{d}\xi. \tag{6.7}$$

Note that the outcome resides in the Lie algebra, as long as $\alpha(\tilde{\tau}) = \alpha(\tau)$ where τ has been given in (6.7) and

$$D_{\tilde{\tau}}(t) = \int_0^t D_{\tau_2}(\xi)A(\xi)D_{\tau_1}(\xi)\mathrm{d}\xi$$

is the *conjugate term* of $D_{\tilde{\tau}}(t)$. (The existence of such a term is assured by the third composition rule.)

Similarly to the association between rooted binary trees and terms in the Magnus expansion, we wish to render the structure of the above composition rules clearer by using graph theory. The presence of three, rather then two, composition rules makes this goal different and 'plain' rooted binary trees are no longer adequate for the task in hand. Instead, following Iserles (1999*b*), we employ rooted *bicolour* binary trees: each vertex can be of one of two colours, black or white. The composition rules are interpreted in the following manner, borrowing as much as possible from the construction in Section 4.

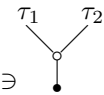
1 $\mathbb{S}_0 = \{$ $\}$ and

$$\text{  } \rightsquigarrow \int_0^t A(\xi)\mathrm{d}\xi;$$

2 If $\mathbb{S}_{k-1} \ni \tau_1 \rightsquigarrow D_{\tau_1}(t)$ then (6.6) corresponds to

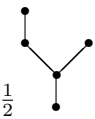
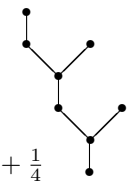
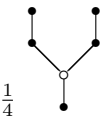
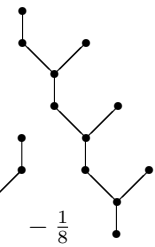
$$\mathbb{S}_k \ni \text{  } \rightsquigarrow \int_0^t [D_{\tau_1}(\xi), A(\xi)]\mathrm{d}\xi;$$

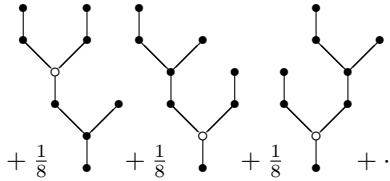
3 Letting $\mathbb{S}_{k-j} \ni \tau_1 \rightsquigarrow D_{\tau_1}(t)$ and $\mathbb{S}_j \ni \tau_1 \rightsquigarrow D_{\tau_2}(t)$, (6.7) corresponds to

$$\mathbb{S}_k \ni \text{  } \rightsquigarrow D_\tau(t) = \int_0^t D_{\tau_1}(\xi)A(\xi)D_{\tau_2}(\xi)\mathrm{d}\xi.$$

Unlike the case of the Magnus expansion, the derivation of the coefficients $\alpha(\tau)$ is straightforward and does not require any recursion. Given $\tau \in \cup\,\mathbb{S}_k$, we denote by $\gamma(\tau)$ the number of *white* nodes therein. It is possible to prove that $\alpha(\tau) = (-1)^{k+\gamma(\tau)}2^{-k}$ and the outcome is the *Cayley expansion*

$$\Delta(t) = \sum_{k=0}^{\infty} \frac{(-1)^k}{2^k} \sum_{\tau\in\mathbb{S}_k} (-1)^{\gamma(\tau)} D_\tau(t) \tag{6.8}$$

$$= \text{} \quad -\tfrac{1}{2} \text{  } \quad +\tfrac{1}{4} \text{  } \quad -\tfrac{1}{4} \text{  } \quad -\tfrac{1}{8} \text{  }$$



Note that the last two trees above are conjugate and that they have the same weights. This is true in general, since if τ and $\tilde{\tau}$ are conjugate then they have the same number of white vertices, $\gamma(\tau) = \gamma(\tilde{\tau})$. Therefore, conjugate trees translate into (scaled) symmetric triple products and we stay safely within the Lie algebra.

Absolute convergence of the Cayley expansion (6.8) was proved in (Iserles 1999*b*) for $t \in (0, t^*)$, provided that $\int_0^t \|A(\xi)\| \mathrm{d}\xi < 2$, a result that can be somewhat improved for some norms and Lie algebras. However, inasmuch as convergence in norm with respect to $\|\cdot\|_2$ is concerned, the Magnus-expansion condition (4.13) of Moan (n.d.) remains valid in the present setting.

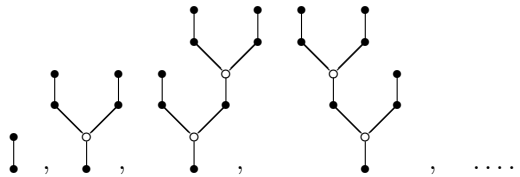
Similarly to the case of Magnus expansions, it makes sense to truncate the series (6.8) *by power,*

$$\Delta(t) \approx \sum_{m=0}^{p-1} \sum_{\tau \in \mathbb{G}_m} \frac{(-1)^{\beta(\tau)+\gamma(\tau)}}{2^{\beta(\tau)}} D_\tau(t), \tag{6.9}$$

where $\beta(\tau) + 1$ is the number of integrals in D_τ (in other words, $\tau \in \mathbb{S}_{\beta(\tau)}$), while $\mathbb{G}_m$ stands for the set of trees of *power* m,

$$\tau \in \mathbb{G}_m \qquad \Longleftrightarrow \qquad D_\tau(t) = \mathcal{O}\big(t^{m+1}\big)$$

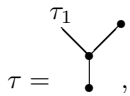
for all sufficiently-smooth matrix functions A. The mechanism that allows m to exceed $\beta(\tau)$ is subtly different from that of Magnus expansions (4.14) since, except for the second tree in the expansion (6.9), we never encounter an instance of $D_\tau(t) = \int [D_{\tau_1}, D'_{\tau_1}]$. Instead, we say that a tree is *basic* if it has no black nodes with two children (equivalently, if the corresponding expansion term contains no commutators). The first few basic trees are



Each basic tree has an even number of vertices and it is easy to verify that if $\tau \in \mathbb{S}_{2m}$ and τ is basic then

$$A(t) = A_0 + \mathcal{O}(t) \qquad \Rightarrow \qquad D_\tau(t) = c A_0^{2m+1} t^{2m+1} = \mathcal{O}\big(t^{2m+2}\big)$$

where $c \neq 0$ is scalar. In other words, $\tau \in \mathbb{G}_{2m}$ and nothing is gained. However, as soon as we form the tree



where τ_1 is basic, it is trivial to notice that $\tau \in \mathbb{G}_{2m+2}$, a 'gain' of one unit in power. Needless to say, this gain is inherited each time τ features as a component of a larger tree.

Truncating by power economises on the number of components: it has been proved in (Iserles 1999*b*) that

$$\limsup_{k\to\infty}(\#\mathbb{S}_k)^{1/k} = 3 \qquad \text{and} \qquad \limsup_{m\to\infty}(\#\mathbb{G}_m)^{1/m} = 2.69805\ldots$$

(in either case there is substantial saving in comparison with the Magnus expansion, cf. (4.15)). However, a very important feature of Magnus expansions is unfortunately lost: *The Cayley expansion* (6.9), *truncated by power, is no longer time symmetric!* (We should perhaps emphasise that the role of time symmetry survives when the exponential is replaced with the Cayley transform: it still implies even order.) In other words, if we truncate the Cayley expansion (with all the integrals evaluated exactly) so that $p = 3$, say, in (6.9), the order will be just three.

Similarly to (4.16–18), we conclude by presenting Cayley expansions in standard notation, rather than in a tree terminology.

$$\begin{aligned}\Delta(t) \;=\; & \int_0^t A(\xi)\mathrm{d}\xi \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{order 2} \\ & -\tfrac{1}{2}\int_0^t\int_0^{\xi_1}[A(\xi_2),A(\xi_1)]\mathrm{d}\boldsymbol{\xi} \\ & +\tfrac{1}{4}\int_0^t\int_0^{\xi_1}\int_0^{\xi_2}[[A(\xi_3),A(\xi_2)],A(\xi_1)]\mathrm{d}\boldsymbol{\xi} \quad \ldots\ldots\ldots\ldots\ldots\ldots\text{order 3}\end{aligned}$$

$$
\begin{aligned}
&- \tfrac{1}{4}\int_0^t\int_0^{\xi_1}\int_0^{\xi_1} A(\xi_2)A(\xi_1)A(\xi_3)\mathrm{d}\boldsymbol{\xi} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{order 4}\\
&- \tfrac{1}{8}\int_0^t\int_0^{\xi_1}\int_0^{\xi_2}\int_0^{\xi_3} [[[A(\xi_4),A(\xi_3)],A(\xi_2)],A(\xi_1)]\mathrm{d}\boldsymbol{\xi} \quad \ldots\ldots\ldots\ldots \text{order 5}\\
&+ \tfrac{1}{8}\int_0^t\int_0^{\xi_1}\int_0^{\xi_2}\int_0^{\xi_2} [A(\xi_3)A(\xi_2)A(\xi_4),A(\xi_1)]\mathrm{d}\boldsymbol{\xi}\\
&+ \tfrac{1}{8}\int_0^t\int_0^{\xi_1}\int_0^{\xi_2}\int_0^{\xi_1} [A(\xi_3),A(\xi_2)]A(\xi_1)A(\xi_4)\mathrm{d}\boldsymbol{\xi}\\
&+ \tfrac{1}{8}\int_0^t\int_0^{\xi_1}\int_0^{\xi_1}\int_0^{\xi_3} A(\xi_2)A(\xi_1)[A(\xi_4),A(\xi_3)]\mathrm{d}\boldsymbol{\xi}\\
&+ \tfrac{1}{16}\int_0^t\int_0^{\xi_1}\int_0^{\xi_2}\int_0^{\xi_3}\int_0^{\xi_4} [[[[A(\xi_5),A(\xi_4)],A(\xi_3)],A(\xi_2)],A(\xi_1)]\mathrm{d}\boldsymbol{\xi}\\
&- \tfrac{1}{16}\int_0^t\int_0^{\xi_1}\int_0^{\xi_2}\int_0^{\xi_3}\int_0^{\xi_1} [[A(\xi_4),A(\xi_3)],A(\xi_2)]A(\xi_1)A(\xi_5)\mathrm{d}\boldsymbol{\xi}\\
&- \tfrac{1}{16}\int_0^t\int_0^{\xi_1}\int_0^{\xi_1}\int_0^{\xi_3}\int_0^{\xi_4} A(\xi_2)A(\xi_1)[[A(\xi_5),A(\xi_4)],A(\xi_3)]\mathrm{d}\boldsymbol{\xi} \quad \ldots \text{order 6}\\
&+ \cdots .
\end{aligned}
$$

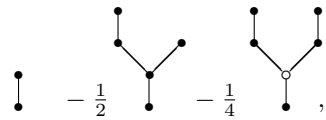
The above expansion underscores the importance of time symmetry in reducing the number of terms: compare the order-6 truncation with (4.18). We hasten to reassure the disappointed reader that not all is lost: time symmetry and even order will be regained in the next subsection.

6.3. Quadrature of the Cayley expansion and hierarchical algebras

In principle, the terms in the Cayley expansion (6.9) can be approximated exactly like 'Magnus integrals', since they are all consistent with (5.1): integrals of a multilinear form $\boldsymbol{L}$ over a polytope $\mathcal{S}$. The theory of Section 5.1–2 is robust enough to cater for symmetric triple products, not just commutators.

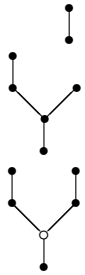
Thus, to obtain a third-order method we truncate by power,

$$\Delta(t) \;\approx\; \tfrac{}{}\;\; -\tfrac{1}{2}\;\; -\tfrac{1}{4}\;\;, \tag{6.10}$$



and replace integrals by quadrature using a self-adjoint basis. More specifically, we evaluate hA at the Gauss–Legendre points $(\frac{1}{2} \pm \frac{\sqrt{3}}{6})h$, denote these function values by A_1, A_2 and let $B_1 = \frac{1}{2}(A_1 + A_2)$, $B_2 = \sqrt{3}(A_2 - A_1)$. The relevant fourth-order

quadratures are



$$: \qquad B_1,$$

$$: \qquad -\tfrac{1}{6}[B_2, B_1],$$

$$: \qquad \tfrac{1}{3}B_1^3 + \tfrac{1}{12}B_1B_2B_1 - \tfrac{1}{24}B_2B_1^2 - \tfrac{1}{24}B_1^2B_2$$

but we can throw away the last three terms with complete impunity: after all, we want a third-order method! The outcome is

$$B_1 + \tfrac{1}{12}[B_2, B_1] - \tfrac{1}{12}B_1^3. \tag{6.11}$$

Just to be on the safe side, we expand the solution, only to find that, lo and behold, the order of (6.11) is *four*. Not the order of (6.10), we hasten to say: the miracle has occurred just as the integrals have been replaced by quadrature!

This is not a serendipitous coincidence. *Quadrature recovers time symmetry, thereby boosting the order of an odd-order truncation* (6.9) (Iserles 1999*b*). Thus, herewith for example a sixth-order method, where B_0, B_1, B_2 have been obtained from order-six Gauss–Legendre quadrature:

$$\begin{aligned} &B_1 + \tfrac{1}{12}B_3 + \tfrac{1}{12}[B_2, B_1] - \tfrac{1}{12}B_1^3 - \tfrac{1}{240}[B_3, B_2] - \tfrac{1}{240}[[B_3, B_1], B_1] \\ &- \tfrac{1}{240}[[B_2, B_1], B_2] - \tfrac{1}{48}B_1B_3B_1 - \tfrac{3}{320}[B_2, B_1^3] + \tfrac{7}{960}B_1[B_2, B_1]B_1 \\ &+ \tfrac{1}{960}[[[B_2, B_1], B_1], B_1] + \tfrac{1}{120}B_1^5. \end{aligned} \tag{6.12}$$

Having hopefully learnt something from our analysis of discretised Magnus expansions in Section 5, our next question is whether all the terms in (6.12) are necessary or can we perhaps replace some with linear combinations of other terms. In other words, we wish to repeat here the discussion from Section 5.3, except that in the present situation we should reckon with two operations: commutation and the symmetric triple product, the latter characteristic of quadratic Lie algebras. Wishing to derive the dimension of linear spaces of *graded free algebras,* along the lines of Section 5.3, a natural temptation is to express symmetric triple products in terms of commutators, but this soon leads to mushrooming complexity. More effective approach is described in (Iserles and Zanna 2000).

Let $(\mathfrak{g}, +)$ be an Abelian group over a field of zero characteristic and introduce a countable family of m-nary operations

$$[\![\,\cdot, \ldots, \cdot\,]\!]_m : \overbrace{\mathfrak{g} \times \mathfrak{g} \times \cdots \times \mathfrak{g}}^{m \text{ times}} \to \mathfrak{g}, \qquad m \in \mathbb{N},$$

which is subject to the following three axioms:

1 Alternate symmetry: For all $F_1, F_2, \ldots, F_m \in \mathfrak{g}$

$$[\![F_1, F_2, \ldots, F_m]\!]_m + (-1)^m [\![F_m, F_{m-1}, \ldots, F_1]\!]_m = \boldsymbol{O};$$

2 Multilinearity: $[\![F_1, F_2, \ldots, F_m]\!]_m$ is linear in each of its m arguments;

3 Hierarchy condition: For all $F_1, \ldots, F_m, E_1, \ldots, E_s \in \mathfrak{g}$ and $1 \leq l \leq m$ it is true that

$$\begin{aligned} & [\![F_1, \ldots, F_{l-1}, [\![E_1, \ldots, E_s]\!]_s, F_{l+1}, \ldots, F_m]\!]_m \\ = \; & [\![F_1, \ldots, F_{l-1}, E_1, \ldots, E_s, F_{l+1}, \ldots, F_m]\!]_{m+s-1} \\ & - (-1)^s [\![F_1, \ldots, F_{l-1}, E_s, \ldots, E_1, F_{l+1}, \ldots, F_m]\!]_{m+s-1}. \end{aligned}$$

The triple $(\mathfrak{g}, +, \{[\![\cdots]\!]_m\}_{m\in\mathbb{N}})$ has been called by Iserles and Zanna (2000) a *hierarchical algebra.* It is easy to see that each hierarchical algebra is a Lie algebra (with the commutator defined as $[\,\cdot\,,\,\cdot\,] = [\![\,\cdot\,,\,\cdot\,]\!]_2$, while each quadratic Lie algebra is hierarchical with

$$[\![F_1, \ldots, F_m]\!]_m = F_1 F_2 \cdots F_m - (-1)^m F_m F_{m-1} \cdots F_1, \qquad F_1, \ldots, F_m \in \mathfrak{g}.$$

We now proceed along the same path as Section 5.3, choose a set $G_1, G_2, \ldots, G_\nu$ of generators and define a *free hierarchical algebra (FHA)* similarly to Definition 5.1. We endow the generators with grading ω and extend it to FHA in a natural manner,

$$\omega([\![H_{i_1}, H_{i_2}, \ldots, H_{i_r}]\!]_r = \sum_{k=1}^{r} \omega(H_k).$$

One should not take the analogy with FLA too far, since FHA require subtly different approach. At the heart of the discussion of Section 5.3 it the fact that, using for example the Hall basis, we can express every element of a FLA as a linear combination of *primitive* terms of the form

$$[G_{i_1}, [G_{i_1}, [\ldots, [G_{i_{r-1}}, G_{i_r}] \cdots]]].$$

In the case of FHA Iserles and Zanna (2000) prove that the primitive 'building blocks' can be chosen of the form

$$[\![G_{i_1}, G_{i_2}, \ldots, G_{i_r}]\!]_r.$$

The method of proof is constructive, repeatedly using the three axioms. Anticipating future discussion, we exemplify it with one of the terms in the sixth-order Cayley expansion (6.12), assuming that $\nu \geq 2$:

$$\begin{aligned} [\![[\![[\![G_2, G_1]\!]_2, G_1]\!]_2, G_1]\!]_2 & \overset{\text{Axiom 3}}{=} [\![[\![G_2, G_1]\!]_2, G_1, G_1]\!]_3 - [\![G_1, [\![G_2, G_1]\!]_2, G_1]\!]_3 \\ & \overset{\text{Axiom 3}}{=} ([\![G_2, G_1, G_1, G_1]\!]_4 - [\![G_1, G_2, G_1, G_1]\!]_4) \\ & \quad - ([\![G_1, G_2, G_1, G_1]\!]_4 - [\![G_1, G_1, G_2, G_1]\!]_4) \\ & \overset{\text{Axiom 1}}{=} [\![G_2, G_1, G_1, G_1]\!]_4 - 3[\![G_1, G_2, G_1, G_1]\!]_4. \end{aligned}$$

Let $\mathfrak{g}_m$ be the set of all the grade-m elements in the FHA $\mathfrak{g}$. Similarly to FLA, we can express $\mathfrak{g}$ as a direct sum of $\mathfrak{g}_m$ for $m \in \mathbb{N}$. The dimension of each $\mathfrak{g}_m$ has been characterised in (Iserles and Zanna 2000).

Theorem 6.1 Let $\mathfrak{g}$ be the graded FHA generated by $\boldsymbol{S} = \{G_1, G_2, \ldots, G_\nu\}$, with grades $\omega_1, \omega_2, \ldots, \omega_\nu$ respectively. Denote by $\lambda_1, \lambda_2, \ldots, \lambda_r$ the roots of the rth degree polynomial

$$p(z) = 1 - \sum_{i=1}^{\nu} z^{\omega_i}, \qquad r = \max_{1 \le i \le \nu} \omega_i,$$

and assume that they are all distinct. Then

$$\begin{aligned} \bar{\sigma}_{2m} &= \tfrac{1}{2} \sum_{l=1}^{r} \frac{\lambda_l^{-m-1}}{p'(\lambda_l)} \left\{ 2 - \lambda_l^{-m} - \tfrac{1}{2}[p(\lambda_l^{1/2}) + p(-\lambda_l^{1/2})] \right\} \\ \bar{\sigma}_{2m+1} &= \tfrac{1}{2} \sum_{l=1}^{r} \frac{\lambda_l^{-m-3/2}}{p'(\lambda_l)} \left\{ -\lambda_l^{-m-1/2} + \tfrac{1}{2}[p(\lambda_l^{1/2}) - p(-\lambda_l^{1/2})] \right\}, \end{aligned} \tag{6.13}$$

where $\bar{\sigma}_m = \dim \mathfrak{g}_m$, $m \in \mathbb{N}$.

The proof is long and technical. Its main step is in demonstrating that

$$\sum_{m=1}^{\infty} t^m \dim \mathfrak{g}_m = \tfrac{1}{2} \left[\frac{1}{p(t)} + \frac{p(t)}{p(t^2)} \right].$$

Table 6.1. *Dimensions* (6.13) *of graded FHA for* $\nu = 3$ *in two cases:* $\omega_i \equiv 1$ *and* $\omega_i = i$.

	m	1	2	3	4	5	6	7	8	9	10
$\omega_i \equiv 1$	$\bar{\sigma}_m$	3	3	18	36	135	351	1134	3240	9963	29403
$\omega_i = i$	$\bar{\sigma}_m$	1	1	3	3	8	11	25	39	80	134

Comparison with Tables 5.1–2 demonstrates that the dimension of graded subspaces of FHA grows larger than in the FLA case. This is hardly a surprise. A Lie algebra is closed with respect to just one binary operation, commutation, in addition to the usual linear-space operations. A hierarchical algebra, however, is closed with respect to a countable number of operations! On the face of it, there are infinitely more ways of forming terms in FHA. Fortunately, the hierarchy conditions mean that the operations are interconnected and the growth in dimension is not as bad as we might have expected.

The ratio of the dimensions $\bar{\rho}_m$ and $\bar{\sigma}_m$ from (5.12) and (6.13) respectively can

be determined asymptotically. The dominant zero of polynomial p, λ_1, say, is in $(1,\infty)$ and simple. Iserles and Zanna (2000) proved that

$$\frac{\bar\sigma_m}{\bar\rho_m} = -\frac{\lambda_1 m}{2p'(\lambda_1)}[1+o(1)], \qquad m \gg 1.$$

The method of proof of Theorem 6.1 is constructive and it naturally leads to a basis and to algorithmic means of its construction. We refer the reader to (Iserles and Zanna 2000) for details, here just presenting the results for $\nu = 2$ and the grades $\omega(G_i) = i$. The basis of $\mathfrak{g}_m$ is denoted by $\mathscr{B}_m$.

$$\begin{aligned}
\mathscr{B}_1 :&\quad \{[\![G_1]\!]_1\},\\
\mathscr{B}_2 :&\quad \{[\![G_2]\!]_1\},\\
\mathscr{B}_3 :&\quad \{[\![G_3]\!]_1, [\![G_1,G_2]\!]_2, [\![G_1,G_1,G_1]\!]_3\},\\
\mathscr{B}_4 :&\quad \{[\![G_1,G_3]\!]_2, [\![G_1,G_1,G_2]\!]_3, [\![G_1,G_2,G_1]\!]\}\\
\mathscr{B}_5 :&\quad \{[\![G_2,G_3]\!]_2, [\![G_1,G_2,G_2]\!]_3, [\![G_1,G_1,G_3]\!]_3, [\![G_2,G_1,G_2]\!]_3,\\
&\qquad [\![G_1,G_3,G_1]\!]_3, [\![G_1,G_1,G_1,G_2]\!]_4,]G_1,G_1,G_2,G_1]\!]_4, [\![G_1,G_1,G_1,G_1,G_1[\![_5\}.
\end{aligned}$$

We conclude by going back to the sixth-order Cayley expansion (6.12) and representing it in the FHA basis. As before, we let $G_i = B_i$ and $\omega_i = i$. The outcome, after long but straightforward algebra, is

$$\begin{aligned}
&[\![B_1]\!]_1 \quad \ldots\ldots \text{grade 1}\\
&+\tfrac{1}{12}[\![B_3]\!]_1 - \tfrac{1}{12}[\![B_1,B_2]\!]_2 - \tfrac{1}{24}[\![B_1,B_1,B_1]\!]_3 \quad \ldots\ldots \text{grade 3}\\
&+\tfrac{1}{240}[\![B_2,B_3]\!]_2 + \tfrac{1}{240}[\![B_1,B_2,B_2]\!]_3 - \tfrac{1}{240}]B_1,B_1,B_3]\!]_3 - \tfrac{1}{240}[\![B_2,B_1,B_2]\!]_3\\
&-\tfrac{1}{160}[\![B_1,B_3,B_1]\!]_3 + \tfrac{1}{120}[\![B_1,B_1,B_1,B_2]\!]_4 - \tfrac{1}{240}[\![B_1,B_1,B_2,B_1]\!]_4\\
&+\tfrac{1}{240}[\![B_1,B_1,B_1,B_1,B_1]\!]_5 \quad \ldots\ldots \text{grade 5}
\end{aligned}$$

Note that, thanks to time symmetry, only odd-grade elements enter the expansion.

6.4. Canonical coordinates of the second kind I: A naive approach

The main idea of the present section is to consider alternatives to the standard exponential map $\exp : \mathfrak{g} \to \mathcal{G}$, which we can write in the form

$$\mathfrak{g} \ni \sum_{k=1}^{d} \theta_k C_k \sim (\theta_1,\theta_2,\ldots,\theta_d) \to \exp\left(\sum_{k=1}^{d}\theta_k C_k\right) \in \mathcal{G}, \tag{6.14}$$

where $d = \dim \mathfrak{g}$ and $\boldsymbol{C} = \{C_1, C_2, \ldots, C_d\}$ is a basis of the Lie algebra. The map (6.14) induces (at least locally, near the identity) a coordinate system in the Lie group which has been termed by Varadarajan (1984) the *canonical coordinates of the first kind.* An alternative to (6.14) (which, incidentally, explains why we have

insisted to write it in such a strange form) are the *canonical coordinates of the second kind (CCSK),*

$$\mathfrak{g} \ni \sum_{k=1}^{d} \theta_k C_k \sim (\theta_1, \theta_2, \ldots, \theta_d) \to \mathrm{e}^{\theta_1 C_1} \mathrm{e}^{\theta_2 C_2} \cdots \mathrm{e}^{\theta_d C_d} \in \mathcal{G}. \tag{6.15}$$

Why should we consider (6.15)? On the face of it, we have replaced a single exponential with d exponentials (and the whole exercise becomes really interesting when $d \gg 1$!), hardly a sensible point of departure. However, as long as $\boldsymbol{C}$ is appropriately chosen, the computation of each $\mathrm{expm}(\theta_k C_k)$ can be exceedingly cheap and, moreover, the approach lends itself naturally to the exploitation of sparsity: as long as we can expect that there should be no component in the C_k direction, say (or that it is suitably small), we can drop the relevant exponential from the product.[2] A useful analogy is the distinction between Householder reflections and Givens rotations in numerical algebra.

It is possible to approach the issue of CCSK within the context of this survey from two distinctive points of view. Although ultimately they are closely related, they follow different philosophy, the first 'naive' and the other more mathematically sophisticated. This subsection is devoted to the more 'naive' approach, which associates CCSK with *splittings.*

Splitting methods have rich history throughout numerical analysis of differential equations and they are exceedingly useful in geometric integration, e.g. in the computational of Hamiltonian systems (Sanz Serna and Calvo 1994, Yoshida 1990) and in the recovery of integrals and conservation laws (McLachlan, Quispel and Robidoux 1998). Yet, the splitting of a flow into components *corresponding to elements of a basis* allows a significant enhancement of the technique. For simplicity, let us assume that we are solving the linear Lie-group equation (4.2), namely $Y' = A(t)Y$, $t \geq 0$. We express the solution in the form

$$Y(t) = \mathrm{e}^{\theta_1(t) C_1} \mathrm{e}^{\theta_2(t) C_2} \cdots \mathrm{e}^{\theta_d(t) C_d} Y_0, \qquad t \geq 0, \tag{6.16}$$

where $\theta_1, \theta_2, \ldots, \theta_d$ are scalar functions. It has been proved by Wei and Norman (1964) that such functions always exist locally (and, in the case of solvable Lie algebras or for 2×2 real matrices, globally). The exact derivation of $\theta_1, \theta_2, \ldots, \theta_d$ is, needless to say, impossible in general, otherwise we could have written down the solution of (4.2) explicitly! Instead, we replace the θ_ks with *polynomials,* which are chosen so as to match suitable order conditions at $t = 0$.

Letting $t = 0$ in (6.16), we note that $\theta_k(0) = 0$, $k = 1, 2, \ldots, d$. To obtain more

[2] We return to this point in far greater detail in Section 8.

useful order conditions we differentiate Y. Brief algebra confirms that

$$\begin{aligned} A(t) &= Y'(t)Y^{-1}(t) \\ &= \sum_{k=1}^{d} \theta_i'(t) \mathrm{e}^{\theta_1(t)C_1} \cdots \mathrm{e}^{\theta_{k-1}(t)C_{k-1}} C_k \mathrm{e}^{-\theta_{k-1}(t)C_{k-1}} \cdots \mathrm{e}^{-\theta_1(t)C_1} \\ &= \sum_{k=1}^{d} \theta_k'(t) \mathrm{Ad}_{\,\mathrm{expm}[\theta_1(t)C_1]} \cdots \mathrm{Ad}_{\,\mathrm{expm}[\theta_{k-1}(t)C_{k-1}]} C_k. \end{aligned} \tag{6.17}$$

Letting $t = 0$ in (6.17) we obtain the first-order condition

$$A(0) = \sum_{k=1}^{d} \theta_k'(0) C_k.$$

Recalling that $A(0) \in \mathfrak{g}$, we can expand it in the elements of $\boldsymbol{C}$ and this yields $\theta_k'(0)$ explicitly.

Higher-order conditions can be obtained by differentiating (6.17) and massaging the formulae with a great deal of (fairly unpleasant) algebra. Thus, for example,

$$\begin{aligned} A' \;=\; & \sum_{k=1}^{d} \theta_k'' \mathrm{Ad}_{\mathrm{e}^{\theta_1 C_1}} \cdots \mathrm{Ad}_{\mathrm{e}^{\theta_{k-1} C_{k-1}}} C_k \\ & + \sum_{k=1}^{d} \sum_{l=1}^{k-1} \theta_k' \theta_l' \mathrm{Ad}_{\mathrm{e}^{\theta_1 C_1}} \cdots \mathrm{Ad}_{\mathrm{e}^{\theta_l C_l}} [C_l, \mathrm{Ad}_{\mathrm{e}^{\theta_{l+1} C_{l+1}}} \cdots \mathrm{Ad}_{\mathrm{e}^{\theta_{k-1} C_{k-1}}} C_k] \end{aligned}$$

and, letting $t = 0$, we have

$$\sum_{k=1}^{d} \theta''(0) C_k = A'(0) - \sum_{k=1}^{d} \sum_{l=1}^{k-1} \theta_k'(0) \theta_l'(0) [C_l, C_k].$$

Recall that $\boldsymbol{C}$ is a basis of $\mathfrak{g}$, hence there exist scalars $c_{k,l}^j$ such that

$$[C_k, C_l] = \sum_{j=1}^{d} c_{k,l}^j C_j, \qquad k, l = 1, 2, \ldots, d.$$

They are called the *structure constants* of $\mathfrak{g}$ and play an important role in the theory of Lie algebras (Olver 1995, Varadarajan 1984). Using structure constants and observing that $A'(0) \in \mathfrak{g}$ can be expanded in elements of $\boldsymbol{C}$, we obtain

$$\sum_{k=1}^{d} \theta_k''(0) C_k = A'(0) + \sum_{j=1}^{d} \sum_{k=1}^{d} \sum_{l=1}^{k-1} \theta_k'(0) c_{k,l}^j \theta_l'(0) C_j, \tag{6.18}$$

hence second-order conditions.

Typically, the dimension d is quite large, for example $\dim \mathfrak{so}(N) = \frac{1}{2}(N-1)N$ and $\dim \mathfrak{sl}(N) = N^2 - 1$. Thus, in principle it might be costly to evaluate $\theta_k''(0)$, $k = 1, 2, \ldots, d$ in (6.18). Higher-order conditions are substantially costlier still. Yet, the cost can be reduced a very great deal by the right choice of the basis $\boldsymbol{C}$.

The most suitable basis $\boldsymbol{C}$ is provided by a *root-space decomposition* of the (non-nilpotent) Lie algebra $\mathfrak{g}$. Deferring our discussion of this construct to the next subsection, we describe in a more nontechnical setting the special case of $\mathfrak{so}(N)$. To this end we choose the basis

$$\boldsymbol{C} = \{C_{k,l} = \boldsymbol{e}_k \boldsymbol{e}_l^{\mathrm{T}} - \boldsymbol{e}_l \boldsymbol{e}_k^{\mathrm{T}} \,:\, 1 \le k < l \le N\},$$

where $\boldsymbol{e}_j \in \mathbb{R}^N$ is the jth unit vector. Note that

$$B \in \mathfrak{so}(N) \qquad \Rightarrow \qquad B = \sum_{k=1}^{N-1} \sum_{l=k+1}^{N} b_{k,l} C_{k,l}$$

and that $U = \operatorname{expm}(tC_{k,l})$ is a rigid rotation in the (k,l) plane: it coincides with the identity matrix, except for

$$\begin{bmatrix} u_{k,k} & u_{k,l} \\ u_{l,k} & u_{l,l} \end{bmatrix} = \begin{bmatrix} \cos t & \sin t \\ -\sin t & \cos t \end{bmatrix}.$$

Therefore, multiplying a matrix with $\operatorname{expm}(tC_{k,l})$ is cheap. Moreover, it is easy to verify that

$$[C_{r,s}, C_{k,l}] = \begin{cases} C_{k,s}, & r = l,\ s \ne k, \\ C_{l,r}, & r \ne l,\ s = k, \\ C_{r,k}, & r \ne k,\ s = l, \\ C_{s,l}, & r = k,\ s \ne l, \\ \boldsymbol{O}, & \text{otherwise}, \end{cases} \qquad r < s,\ \ k < l,$$

where we identify $C_{i,j}$ with $-C_{j,i}$ for $i > j$. The condition (6.18) simplifies to

$$\theta_{k,l}''(0) = a_{k,l}'(0) - \sum_{i=1}^{N} a_{k,i}(0) a_{i,l}(0), \qquad 1 \le k < l \le N,$$

where $A(t) = \sum_{k=1}^{N-1} \sum_{l=k+1}^{N} \alpha_{k,l}(t) C_{k,l}$. This requires $\mathcal{O}\big(N^3\big)$ flops altogether, in comparison with $\mathcal{O}\big(N^6\big)$ if sparsity of structure constants is disregarded. Similar construction can be applied in other Lie algebras.

In principle, we can go on to derive $\theta_k^{(i)}(0)$ for $i = 0, 1, \ldots, p$, but this procedure, even while utilising root-space decomposition, becomes progressively more expensive for larger orders p. Herewith we present a device which, to our knowledge, is

new and which allows one to obtain order p while computing one less derivative. Observing that (6.16) is sensitive to the ordering of the basis, the main idea is to alternate the order of elements of $\boldsymbol{C}$ while time-stepping the numerical method. Thus, suppose that $t_m = mh$ and

$$\begin{aligned} Y_{2n+1} &= \mathrm{e}^{\theta_{2n,1}(t_{2n+1})C_1}\mathrm{e}^{\theta_{2n,2}(t_{2n+1})C_2}\cdots\mathrm{e}^{\theta_{2n,d}(t_{2n+1})C_d}Y_{2n}, \\ Y_{2n+2} &= \mathrm{e}^{\theta_{2n+1,d}(t_{2n+2})C_d}\mathrm{e}^{\theta_{2n+1,d-1}(t_{2n+2})C_{d-1}}\cdots\mathrm{e}^{\theta_{2n+1,1}(t_{2n+2})C_1}Y_{2n+1}, \end{aligned}$$

where $\theta_{m,k}$ are p-degree polynomials. Without loss of generality, we assume that $\theta_{m,k}$ are consistent with order p for $m = 0, 1, \ldots, 2n$. Let

$$X(t) = \mathrm{e}^{\theta_{2n+1,d}(t)C_d}\mathrm{e}^{\theta_{2n+1,d-1}(t)C_{d-1}}\cdots\mathrm{e}^{\theta_{2n+1,1}(t)C_1}Y_{2n+1},$$

hence $Y_{2n+1} = X(t_{2n+1})$ and $Y_{2n+2} = X(t_{2n+2})$. Repeatedly multiplying by inverted exponentials, we obtain

$$Y_{2n+1} = \mathrm{e}^{-\theta_{2n+1,1}(t)C_1}\mathrm{e}^{-\theta_{2n+1,2}(t)C_2}\cdots\mathrm{e}^{-\theta_{2n+1,d}(t)C_d}X(t)$$

Assuming that the $\theta_{2n+1,k}$ are chosen consistently with order p and letting $t = t_{2n}$ we deduce that

$$Y_{2n+1} = \mathrm{e}^{-\theta_{2n+1,1}(t_{2n})C_1}\mathrm{e}^{-\theta_{2n+1,2}(t_{2n})C_1}\cdots\mathrm{e}^{-\theta_{2n+1,d}(t_{2n})C_d}Y_{2n} + \mathcal{O}\big(h^{p+1}\big).$$

In other words, we may take $\theta_{2n+1,k}(t_{2n}) = -\theta_{2n,k}(t_{2n+1})$, $k = 1, 2, \ldots, d$. This, together with the values of $\theta^{(i)}_{2n+1,k}(t_{2n+1})$, $i = 0, 1, \ldots, p-1$, is just right to determine the $\theta_{2n+1,k}$s consistently with order p.

6.5. Canonical coordinates of the second kind II: Admissible bases

The technique of canonical coordinates of the second kind can be enhanced a great deal at the cost of increased mathematical sophistication. The point of departure for our discussion, presently based on the important paper of Owren and Marthinsen (1999*a*), is the equation (6.17). We rewrite it in the form

$$\sum_{k=1}^{d}\theta'_k\mathrm{Ad}_{\,\mathrm{expm}(\theta_1 C_1)}\mathrm{Ad}_{\,\mathrm{expm}(\theta_2 C_2)}\cdots\mathrm{Ad}_{\,\mathrm{expm}(\theta_{k-1}C_{k-1})}C_k = A(t). \tag{6.19}$$

Here $\theta_1, \theta_2, \ldots, \theta_d$ are known and we seek the scalars $\theta'_1, \theta'_2, \ldots, \theta'_d$.

Suppose that we have the means to solve (6.19) for arbitrary inputs $\theta_1, \theta_2, \ldots, \theta_d$ (in (6.17) we needed just $\boldsymbol{\theta} = \mathbf{0}$). This yields a *differential equation*

$$\boldsymbol{\theta}' = \boldsymbol{g}(\boldsymbol{\theta}, A(t)), \quad t \geq 0, \qquad \boldsymbol{\theta}(0) = \mathbf{0}. \tag{6.20}$$

Once the solution of (6.20) is known (or adequately approximated), we can use it

to advance the solution of the Lie-group equation (4.2) through the CCSK representation (6.16). Moreover, the argument extends at once to nonlinear Lie-group equations $Y' = A(t, Y)Y$, where $A : \mathcal{G} \times \mathbb{R}^+ \to \mathfrak{g}$. Again, we represent the solution in the CCSK form (6.16), except that now

$$\boldsymbol{g} = \mathrm{dccsk}_{\boldsymbol{\theta}}^{-1} A = \boldsymbol{g}(\boldsymbol{\theta}, A(t, \boldsymbol{e}^{\theta_1 C_1} \boldsymbol{e}^{\theta_2 C_2} \cdots \boldsymbol{e}^{\theta_d C_d} Y_0))$$

in the *dccskinv equation* (6.20).

Applying a Runge–Kutta method, say, to (6.20) results in time-stepping scheme that is guaranteed to respect Lie group structure. In effect, the only difference between the RK-MK methods of Section 3 and such methods is that, in place of canonical coordinates of the first kind and the dexpinv equation, they utilise canonical coordinates of the second kind and the dccskinv equation. All this motivates a thorough discussion of the problem how to invert an equation of the form (6.19).

Letting $F_k = \mathrm{Ad}_{\,\mathrm{expm}(\theta_k C_k)}$, $v_k = \theta_k'$, $k = 1, 2, \ldots, d$, we commence by writing (6.20) as

$$\sum_{k=1}^{d} v_k F_1 F_2 \cdots F_{k-1} C_k = A.$$

Let $\mathcal{P}_l$ be a projection on the trailing $d - l$ coordinates,

$$\mathcal{P}_l \sum_{k=1}^{d} c_k C_k = \sum_{k=l+1}^{d} c_k C_k,$$

and set $\hat{F}_l = I - \mathcal{P}_l + \mathcal{P}_l A_l$, $l = 1, 2, \ldots, d$. We can easily verify that

$$\hat{F}_l C_k = \begin{cases} F_l C_k, & l < k, \\ C_k, & l \geq k. \end{cases}$$

Owren and Marthinsen (1999*a*) say that $\boldsymbol{C}$ is an *admissible ordered basis (AOB)* if for every $\theta_1, \theta_2, \ldots, \theta_d$ it is true that

$$F_1 F_2 \cdots F_k \mathcal{P}_k = \hat{F}_1 \hat{F}_2 \cdots \hat{F}_k \mathcal{P}_k, \qquad k = 1, 2, \ldots, d-1. \tag{6.21}$$

Provided that $\boldsymbol{C}$ is an AOB, it is simple to prove that our equation can be rewritten in the form

$$\sum_{k=1}^{d} v_k \hat{F}_1 \hat{F}_2 \cdots \hat{F}_{d-1} C_k = \hat{F}_1 \hat{F}_2 \cdots \hat{F}_{d-1} F = A, \tag{6.22}$$

where $F = \sum_{k=1}^{d} v_k C_k$. In the sequel we will see that in a number of important cases AOB implies that each $\hat{F}_k$ can be inverted very cheaply.

At a first glance, the AOB condition (6.21) is exceedingly demanding. Surprisingly, it is often achievable but we need to introduce a little bit more Lie-algebra theory before being in position to describe exactly how. The following brief extract should be ideally supplemented by perusing a Lie-algebra monograph: The book of Varadarajan (1984) is a good place to start.

- A subalgebra $\mathfrak{h}$ of a Lie algebra $\mathfrak{g}$ is an *ideal* if $[\mathfrak{h}, \mathfrak{g}] \subseteq \mathfrak{h}$.
- The Lie algebra $\mathfrak{g}$ is *solvable* if there exists $m \in \mathbb{Z}^+$ such that $\mathfrak{g}^{(m)} = \{0\}$, where $\mathfrak{g}^{(0)} = \mathfrak{g}$ and $\mathfrak{g}^{(i+1)} = [\mathfrak{g}^{(i)}, \mathfrak{g}^{(i)}] \subseteq \mathfrak{g}^{(i)}$.
- The *radical* of $\mathfrak{g}$, denoted by $\operatorname{Rad}\mathfrak{g}$, is the maximal solvable ideal in $\mathfrak{g}$. We say that the Lie algebra is *semisimple* if $\operatorname{Rad}\mathfrak{g} = \{0\}$.
  If definitions have become hazy by now, let us just point out that all specific Lie algebras in this survey (and in known applications within its framework) are semisimple, inclusive of $\mathfrak{sl}(N)$, $\mathfrak{so}(N)$ and $\mathfrak{sp}(N)$. (All these three Lie algebras are, as a matter of fact, *simple:* their only ideals are $\{0\}$ and the algebra itself.)
- An element in $\mathfrak{g}$ is *semisimple* if all the roots of its minimal polynomial are distinct: in a matrix representation it means that the element can be diagonalised.
- A subalgebra is *toral* if all its elements are semisimple. It is easy to see that every toral algebra must be abelian: in a matrix representation we can restate this by saying that the elements of the subalgebra share all eigenvectors, hence they commute.
- Unless $\mathfrak{g}$ is nilpotent, it possesses a nonzero *maximal toral subalgebra.* Such subalgebra, which we denote by $\mathfrak{h}$, is unique up to an isomorphism.
  If $\mathfrak{g}$ is a simple algebra, $\mathfrak{h}$ is also known (subject to an equivalent definition) as a *Cartan subalgebra.*
- Suppose that $\mathfrak{g}$ is a linear space over $\mathbb{C}$. We denote by $\mathfrak{h}^*$ the *dual space* of a maximal toral subalgebra. It consists of all linear functionals $\mathfrak{h} \to \mathbb{C}$. The nonzero functional $\alpha \in \mathfrak{h}^*$ is a *root* if there exists $f \in \mathfrak{g} \setminus \{0\}$ such that

$$[h, f] = \alpha(h) f, \qquad h \in \mathfrak{h}.$$

  In a matrix representation, $\alpha(H)$ is an eigenvalue of the commutator operator generated by $H \in \mathfrak{h}$, while F can be 'translated' into its eigenvector.
- Denote the set of all roots of $\mathfrak{g}$ by Φ. It is possible to prove that $\mathfrak{g}$ can be subjected to the *root-space decomposition*

$$\mathfrak{g} = \mathfrak{h} \oplus \coprod_{\alpha \in \Phi} \mathfrak{g}_\alpha, \tag{6.23}$$

  where $\mathfrak{g}_\alpha = \{f \in \mathfrak{g} : [h, f] = \alpha(h) f,\ h \in \mathfrak{h}\} \neq \{0\}$.
- The decomposition (6.23) motivates the choice of a *Chevalley basis* of the Lie algebra $\mathfrak{g}$: we choose one basis vector for each one-dimensional subspace $\mathfrak{g}_\alpha$, $\alpha \in \Phi$, and combine it with an arbitrary basis of $\mathfrak{h}$.
- There exists an integer $k^* \geq 1$ such that $\operatorname{ad}_h^{k^*+1} = 0$ for every $h \in \mathfrak{g}_\alpha$, $\alpha \in \Phi$.

Many of the above concepts can be illustrated briefly with an example, and we choose $\mathfrak{sl}(N, \mathbb{C})$. It is semisimple (as a matter of fact, we have already mentioned that it is a simple Lie algebra). Using the standard representation of $\mathfrak{sl}(N, \mathbb{C})$ as

matrices of zero trace, we can easily identify a maximal toral subalgebra $\mathfrak{h}$ with diagonal zero-trace matrices. Setting $E_{k,l} = \boldsymbol{e}_k \boldsymbol{e}_l^{\mathrm{T}}$, $k, l = 1, 2, \ldots, N$, we may choose the basis $\{E_{k,k} - E_{k+1,k+1} : k = 1, 2, \ldots, N-1\}$ for $\mathfrak{h}$. Moreover, given

$$\mathfrak{h} \ni H = \sum_{k=1}^{N} h_k E_{k,k}, \qquad \sum_{k=1}^{N} h_k = 0,$$

and letting $h_0 = 0$, we verify easily that

$$[H, E_{r,s}] = (h_r - h_{r-1} - h_s + h_{s-1}) E_{r,s}, \qquad r, s = 1, 2, \ldots, N, \quad r \neq s.$$

Hence we identify the root $\alpha(H) = h_r - h_{r-1} - h_s + h_{s-1}$ and construct our basis by placing there $E_{r,s}$ for every $r \neq s$ and appending to this the above basis of $\mathfrak{h}$. Note that this results in $N^2 - 1$ terms, matching exactly the dimension of $\mathfrak{sl}(N, \mathbb{C})$.

To determine k^* we compute

$$\mathrm{ad}^2_{E_{r,s}} E_{k,l} = \begin{cases} E_{r,s}, & k = s,\ l = r, \\ \boldsymbol{O}, & \text{otherwise}, \end{cases} \qquad r \neq s,\ k \neq l \qquad \Rightarrow \qquad \mathrm{ad}^3_{E_{r,s}} E_{k,l} = \boldsymbol{O}$$

and $\mathrm{ad}^2_{E_{r,s}}(E_{k,k} - E_{k+1,k+1}) = \boldsymbol{O}$. Therefore $\mathrm{ad}^3_{E_{r,s}} = \boldsymbol{O}$, $r \neq s$, and we deduce that $k^* = 2$.[3]

Theorem 6.2 Let $\{\rho_1, \rho_2, \ldots, \rho_{d_*}\}$, where $d_* = d - \dim \mathfrak{h}$, be the set of roots Φ of a semisimple non-nilpotent Lie algebra $\mathfrak{g}$. Suppose that the Chevalley basis $\boldsymbol{C}$ is ordered so that the basis of $\mathfrak{h}$ comes last. Then this basis is AOB if

$$k\rho_i + \rho_j = \rho_m, \quad m < i < j \leq d_*,\ 1 \leq k \leq k^* \quad \Rightarrow \quad \rho_m + \rho_n \notin \Phi \cup \{0\} \tag{6.24}$$

for all $n = m+1, m+2, \ldots, i-1$. Moreover, in that case

$$\hat{F}_k^{-1} = I + \sum_{l=1}^{k^*} (-1)^l \frac{\theta_k^l}{l!} \mathrm{ad}^l_{C_k} \mathcal{P}_k, \qquad k = 1, 2, \ldots, d-1. \tag{6.25}$$

Returning to $\mathfrak{so}(N, \mathbb{C})$, it has been proved in (Owren and Marthinsen 1999*a*) that conditions of Theorem 6.2 are satisfied as long as super-diagonal elements are ordered lexicographically by rows in front of the elements underneath the diagonal, which are ordered lexicographically by columns: thus, the ordered basis is

$$\begin{aligned} &\{E_{k,l} : 1 \leq k < l \leq N\} \cup \{E_{k,l} : 1 \leq l < k \leq N\} \\ &\cup \{E_{k,k} - E_{k+1,k+1} : 1 \leq k \leq N-1\}. \end{aligned}$$

We omit the largely-technical proof. Likewise, it is possible, using Theorem 6.2,

[3] There are easier ways to determine k^* but they require more Lie-algebra theory.

to identify AOB of $\mathfrak{sp}(N,\mathbb{C})$ and $\mathfrak{so}(N,\mathbb{C})$. This, however, is probably of less importance than in the case of $\mathfrak{so}(N,\mathbb{C})$, since the increase in dimension due to the replacement of $\mathbb{R}$ with $\mathbb{C}$ leads to a significant increase in the volume of computations. In the present stage of the development of Lie-group methods it is fair to say, we believe, that the Cayley-transform-based techniques from Section 6.1–2 are the method of choice for quadratic Lie groups, while CCSK should be used with the special linear group.

7. Adjoint methods

In the previous sections we have encountered a number of numerical integrators for Lie groups. Although such methods produce solutions that stay on a given Lie group $\mathcal{G}$ *by design,* it is of interest to study how well such schemes respect other qualitative features of the underlying equations: the retention of a symplectic form, Lie–Poisson structure, conservation of energy, time symmetry, time reversibility *et cetera.* Given the novelty of the proposed schemes, many of the above features and their implications on the 'quality' of the solution are still under investigation (Engø and Faltinsen 1999, Faltinsen n.d.). For this reason, we shall focus here just on *time symmetry* for Lie-group methods, which is at present one of few features that are better understood, deriving adjoint and self-adjoint Lie-group methods.

Before proceeding further, let us recall that the *flow* Φ of the differential equation in $\mathbb{R}^N$

$$\boldsymbol{y}' = \boldsymbol{f}(t,\boldsymbol{y}), \quad t \geq t_0, \qquad \boldsymbol{y}(t_0) = \boldsymbol{y}_0,$$

defined as

$$\Phi(t, t_0, \boldsymbol{y}_0) = \boldsymbol{y}(t)$$

obeys the following conditions:

i. $\Phi(t_0, t_0, \boldsymbol{y}_0) = \boldsymbol{y}_0$,
ii. $\Phi(t+\tau, t_0, \boldsymbol{y}_0) = \Phi(\tau, t, \Phi(t, t_0, \boldsymbol{y}_0))$,

provided that the above function $\boldsymbol{f}$ is Lipschitz with respect to $\boldsymbol{y}$ (Hairer et al. 1993). In particular, the second condition implies that

$$\Phi(-\tau, t+\tau, \Phi(\tau, t, \boldsymbol{y}(t))) = \boldsymbol{y}(t),$$

a condition that in literature is mostly known as *time symmetry* or *self adjointness* of the exact flow Φ.

7.1. Adjoint methods in classical setting

Numerical methods usually approximate the flow Φ by a discrete flow, say Ψ, so that

$$\boldsymbol{y}_{n+1} = \Psi(t_n + h, t_n, \boldsymbol{y}_n), \qquad n \in \mathbb{Z}^+,$$

approximates the exact solution $\boldsymbol{y}(t_n + h)$ to given order p. Although numerical integrators for ODEs always obey the condition $\Psi(t_n, t_n, \boldsymbol{y}_n) = \boldsymbol{y}_n$, they usually fail to satisfy condition ii. However, its weaker variant, *time symmetry,* is easier to impose and numerical methods such that

$$\Psi(-h, t_n + h, \Psi(h, t_n, \boldsymbol{y}_n)) = \boldsymbol{y}_n, \qquad n \in \mathbb{Z}^+,$$

are usually called *self-adjoint* or *time-symmetric* methods. If a method is not self adjoint, its *adjoint* Ψ^* is defined as the map

$$\Psi^*(-h, t_n + h, \Psi(h, t_n, \boldsymbol{y}_n)) = \boldsymbol{y}_n.$$

In shorthand notation, we write Ψ_h for $\Psi(h, t_n, \cdot\,)$, so that Ψ_h^* denotes the adjoint method of Ψ_h and moreover $\Psi_h^* \circ \Psi_h = \mathrm{Id}$, the identity map, or equivalently $\Psi_{-h} = \Psi_h^{-1}$, if and only if the method is self adjoint.

The theory of adjoint and self-adjoint numerical methods for ODEs in $\mathbb{R}^N$ is well established and we refer the reader to (Hairer et al. 1993) for further reading.

One might question why self adjointness of a numerical integration scheme is desirable. It turns out that for numerical integration schemes that are self adjoint it is possible to develop a theory analogous to the KAM theory for Hamiltonian and time-reversible problems (Moser 1973), which usually implies better approximation of the solution and slower accumulation of error over long integration intervals (Estep and Stuart 1995, Reich 1996).

With regards to Lie-group methods and time symmetry, we have already shown in Section 4 that the Magnus expansion truncated by power is time symmetric when applied to linear Lie-group differential equations $Y' = A(t)Y$, provided that the underlying quadrature is based on quadrature nodes in $[0, 1]$ which are symmetric with respect to $\frac{1}{2}$. A similar result applies also to the RK-MK methods, provided that the underlying Runge–Kutta scheme is self adjoint.

However, it can be easily verified by means of numerical experiments that the above-mentioned Lie-group methods are not self adjoint for nonlinear problems. Hence, using self-adjoint methods to solve a Lie-algebra differential equation is not sufficient to derive self-adjoint Lie-group methods!

In this section we shall be discussing a more general procedure, derived by Zanna, Engø and Munthe-Kaas (1999), that allows us to construct self-adjoint Lie-group methods for linear and nonlinear problems alike and for all type of coordinate maps $\phi : \mathfrak{g} \to \mathcal{G}$ that we may use to represent the solution. In lifting the Lie-group equation from $\mathcal{G}$ to $\mathfrak{g}$, we make an implicit choice of a coordinate map, which has to be taken into account in the construction of self-adjoint methods.

7.2. Coordinate maps centred at arbitrary points

Schematically, the first step in the development of Lie-group schemes introduced in Sections 3–6 is the choice of a smooth map, say $\phi : \mathfrak{g} \to \mathcal{G}$, such that $\phi(\boldsymbol{O}) = I$, the

identity of the Lie group G, and $\phi'(\boldsymbol{O}) = \boldsymbol{I}$ (more precisely $\mathrm{T}\phi(\boldsymbol{O}, B) = B$, where $B \in \mathfrak{g}$). In other words, ϕ is a diffeomorphism mapping a neighbourhood of $\boldsymbol{O} \in \mathfrak{g}$ into a neighbourhood of $\boldsymbol{I}$ in $\mathcal{G}$. Thus,

$$\phi(B) = \operatorname{expm}(B), \qquad B \in \mathfrak{g},$$

is an example of such a map, which in (6.14) we have termed *canonical coordinates of the first kind.* Similarly, we might consider *canonical coordinates of the second kind* (6.15), namely

$$\phi(B) = \operatorname{ccsk}(B) = \operatorname{expm}(\beta_1 B_1) \operatorname{expm}(\beta_2 B_2) \cdots \operatorname{expm}(\beta_d B_d), \qquad B \in \mathfrak{g},$$

where $B = \sum_{i=1}^{d} \alpha_i B_i$, α_is being real coefficients, the B_is are basis elements of the algebra $\mathfrak{g}$, and the β_is real functions of $\alpha_1, \alpha_2, \ldots, \alpha_d$. Yet another example of this kind of map is the *Cayley transform* (6.3), which in the current formalism reads

$$\phi(B) = (\boldsymbol{I} - \tfrac{1}{2}B)^{-1}(\boldsymbol{I} + \tfrac{1}{2}B), \qquad B \in \mathfrak{g},$$

that maps $\mathfrak{g}$ into $\mathcal{G}$ whenever $\mathcal{G}$ is a quadratic group and $\mathfrak{g}$ is its corresponding quadratic algebra.

Secondly, the Lie-group differential equation

$$Y' = A(t, Y)Y, \quad t \geq 0, \qquad Y(0) = Y_0 \in \mathcal{G}, \tag{7.1}$$

is *lifted* by means of the inverse of the map $\mathrm{d}\phi$ to an ordinary differential equation in $\mathfrak{g}$. Thus, for coordinates of the first kind, one has $\mathrm{d}\phi^{-1} = \operatorname{dexp}^{-1}$, the *dexpinv equation* (3.2) that we have already encountered time and again in the course of the present article. Similarly, we have derived the expressions dcay^{-1} and $\operatorname{dccsk}^{-1}$ in Section 6.

Finally, the $\mathrm{d}\phi^{-1}$ equation is solved in $\mathfrak{g}$ with either a Runge–Kutta method or with a Magnus or a Cayley-type expansion. Assuming that an approximation $Y_n \in \mathcal{G}$ has been already derived, we typically solve

$$\Theta' = \mathrm{d}\phi_{\Theta}^{-1}(A(t, \phi(\Theta)Y_n)), \qquad \Theta(t_n) = \boldsymbol{O}, \quad t \in [t_n, t_{n+1}],$$

where $t_{n+1} = t_n + h$. The choice of the initial condition $\Theta(t_n) = \boldsymbol{O}$ is equivalent to 'centring' the coordinate map ϕ at Y_n. Instead, coordinates centred at any point $X \in \mathcal{G}$ can be obtained inverting the map $B \in \mathfrak{g} \mapsto \phi(B)X \in \mathcal{G}$. In the general case we write

$$X = \phi(C)^{-1} Y_n,$$

for some $C \in \mathfrak{g}$ to be specified in the sequel. Note that $\phi(C)^{-1}$ is the inverse (in $\mathcal{G}$) of the group element $\phi(C)$. Before proceeding further, we observe that both canonical coordinates of the first kind and the Cayley transform obey the relation

$$\phi(B)^{-1} = \phi(-B),$$

for all $B \in \mathfrak{g}$. For canonical coordinates of the second kind one has instead

$$\operatorname{ccsk}(B)^{-1} = \operatorname{expm}(-\beta_d B_d) \operatorname{expm}(-\beta_{d-1} B_{d-1}) \cdots \operatorname{expm}(-\beta_1 B_1).$$

We seek a solution of (7.1) of the form

$$Y(t) = \phi(\Theta(t))\phi(C)^{-1} Y_n \tag{7.2}$$

for $t \in [t_n, t_{n+1}]$. Differentiating in the usual fashion we obtain a differential equation for $\Theta(t)$

$$\Theta' = \mathrm{d}\phi_\Theta^{-1}(A(t, Y)), \qquad t \in [t_n, t_{n+1}], \tag{7.3}$$

where Y is as in (7.2), in tandem with the initial condition

$$\Theta(t_n) = C. \tag{7.4}$$

Thus, changing the centre of coordinate map does not affect the differential equation obeyed by Θ, just its initial condition.

7.3. The adjoint of Lie-group methods

Assume that the Lie-algebra differential equation (7.3), with the initial condition (7.4), is computed with a numerical method Ψ_h, and denote by Ψ_h^* its adjoint in the classical sense of Section 7.1, namely

$$(\Psi_{-h}^* \circ \Psi_h) B = B$$

for all $B \in \mathfrak{g}$. The corresponding Lie-group method is such that

$$\begin{aligned} Y_{n+1} &= \tilde{\Psi}_h Y_n = \phi(\Upsilon_{h,n+1})\phi(C_{h,n})^{-1} Y_n \\ \Upsilon_{h,n+1} &= \Psi(h, t_n, C_{h,n}), \end{aligned}$$

where $C_{h,n}$ is the initial condition of Θ in the interval $[t_n, t_{n+1}]$, and we allow it to depend on the interval of integration and on the step size h. In order to obtain the adjoint of the Lie-group method $\tilde{\Psi}_h$, we need not just to use Ψ_h^* in $\mathfrak{g}$, but also to make sure that the coordinate map employed while stepping forward with the method $\tilde{\Psi}_h$ is the same as when stepping backward with the adjoint method $\tilde{\Psi}_h^*$. Define a pair of methods $\tilde{\Psi}$ and $\tilde{\Psi}^*$ on $\mathcal{G}$ as

$$\begin{aligned} \tilde{\Psi}(t_n + h, t_n, Y_n) &= \phi(\Upsilon_{n,n+1})\phi(C_{h,n})^{-1} Y_n, \\ \tilde{\Psi}^*(t_n + h, t_n, Y_n) &= \phi(\Theta^*_{h,n+1})\phi(C^*_{h,n})^{-1} Y_n, \\ C^*_{-h,n+1} &= \Upsilon_{h,n+1}, \end{aligned} \tag{7.5}$$

where $\Upsilon_{h,n+1} = \Psi(h, t_n, C_{h,n})$ and $\Theta^*_{h,n+1} = \Psi^*(h, t_n, C^*_{h,n})$.

Theorem 7.1. (Zanna et al. (1999)) The method $\tilde{\Psi}^*$ is the Lie-group adjoint of $\tilde{\Psi}$. Moreover, $(\tilde{\Psi}^*)^* = \tilde{\Psi}$.

Proof. With the same notation as above, we have

$$(\tilde{\Psi}^*_{-h} \circ \tilde{\Psi}_h)Y_n = \phi(\Theta^*_{-h,n+2})\phi(C^*_{-h,n+1})^{-1}\phi(\Upsilon_{h,n+1})\phi(C_{h,n})^{-1}Y_n.$$

Because of (7.5), one has $\phi(C_{h,n})^{-1}Y_n = \phi(C^*_{-h,n+1})Y_{n+1}$, from which we deduce that

$$\phi(C^*_{-h,n+1})^{-1}\phi(\Upsilon_{h,n+1}) = \boldsymbol{I}.$$

Furthermore $\Upsilon_{h,n+1}$ and $\Upsilon^*_{-h,n+2}$ are solution of the same differential equation (7.3) whereby the initial condition of Θ^* is the endpoint of Θ and Θ^* is obtained by means of Ψ^*_{-h}, where Ψ^*_h is the adjoint of Ψ_h in the classical sense. Thus $(\tilde{\Psi}^*_{-h} \circ \tilde{\Psi}_h)Y_n = Y_n$ and the assertion follows. □

We have not specified yet what is $C_{h,n}$. In general we let it be a function of the step size of integration h and of the current stage values F_i of a Runge–Kutta method Ψ,

$$C_{h,n} = \vartheta(F_1, F_2, \ldots, F_\nu),$$

ν being the number of the stages of the scheme (cf. Appendix A for notation). Thus,

$$C^*_{-h,n+1} = \vartheta^*(F^*_1, F^*_2, \ldots, F^*_\nu),$$

therefore the functions ϑ and ϑ^* obey the fundamental *adjointness condition*

$$\vartheta^*(F^*_1, F^*_2, \ldots, F^*_\nu) = \vartheta(F_1, F_2, \ldots, F_\nu) + \sum_{i=1}^{\nu} b_i F_i,$$

for the centre of the coordinate map.

The following result, which can be found in (Zanna et al. 1999), characterises self-adjoint Lie-group methods.

Theorem 7.2 Assume that Ψ is self adjoint. Then $\tilde{\Psi}$ is self adjoint provided that

$$\vartheta(F_\nu, F_{\nu-1}, \ldots, F_1) = \vartheta(F_1, F_2, \ldots, F_\nu) + \sum_{i=1}^{\nu} b_i F_i. \tag{7.6}$$

7.4. Geodesic and flow-symmetric coordinate maps

Basically, there are two distinct ways to generate coordinate maps such that (7.6) is obeyed. One way to achieve this goal is to choose ϑ so that the value $Y_{n+1/2} = \phi(C_{h,n})^{-1}Y_n$ is a 'midpoint in space' between Y_n and Y_{n+}, which will generate what we call the *geodesic midpoint method.* An alternative is to choose ϑ so that $Y_{n+1/2} = \phi(C_{h,n})^{-1}Y_n$ is instead a 'midpoint in time', thus generating a *flow midpoint.* The situation is schematically represented in Figure 7.1.

In the first instance we say that the coordinate map is *geodesic symmetric,* while in the second case we say that the coordinate map is *flow symmetric.*

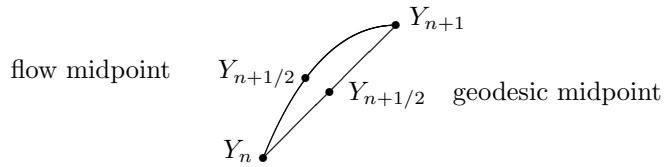


Fig. 7.1. Representation of the geodesic and flow-symmetric midpoint

Geodesic-symmetric coordinate maps are always defined and correspond to the choice

$$C_{h,n} = \vartheta(F_1, F_2, \ldots, F_\nu) = -\tfrac{1}{2} \sum_{i=1}^{\nu} b_i F_i.$$

Flow-symmetric coordinates are instead more naturally defined for methods based on collocation. If $\ell_i(x)$, $i = 1, \ldots \nu$, denote the familiar cardinal polynomials of Lagrangian interpolation, already introduced in Section 5.1, we set

$$w_i = \int_0^{\frac{1}{2}} \ell_i(\tau)\, \mathrm{d}\tau, \qquad i = 1, 2, \ldots, \nu, \tag{7.7}$$

and the choice

$$C_{h,n} = \vartheta(F_1, F_2, \ldots, F_\nu) = -\sum_{i=1}^{\nu} w_i F_i.$$

corresponds to the flow midpoint.

There exist also other choices of functions ϑ that obey (7.6) and it is possible to show that the set of such functions is convex. See (Zanna et al. 1999) for further examples of coordinate maps that yield self-adjoint schemes.

To conclude this section, we illustrate with a numerical experiment the benefits of using the geodesic and flow-symmetric coordinates instead of classical coordinates centred at Y_n. We consider the *Euler equations* for the rigid body,

$$\boldsymbol{y}' = \boldsymbol{y} \times M\boldsymbol{y}, \quad t \geq 0, \qquad \boldsymbol{y}(0) = \boldsymbol{y}_0, \tag{7.8}$$

where $\boldsymbol{y} \in \mathbb{R}^3$ (we assume that $\|\boldsymbol{y}_0\|_2 = 1$), the symbol '$\times$' denotes the classical vector product os $\mathbb{R}^3$ and M is a diagonal matrix, $M = \mathrm{diag}\,(m_1, m_2, m_3)$. This system has the Hamiltonian function $H(\boldsymbol{y}) = \frac{1}{2}(m_1 y_1^2 + m_2 y_2^2 + m_3 y_3^2)$ and obeys $\|\boldsymbol{y}\|_2 = 1$. It can be represented by means of Lie-group action of SO(3) on $\mathbb{R}^3$ by representing the solution $\boldsymbol{y}(t)$ as $\Omega(t)\boldsymbol{y}_n$, $n = 0, 1, 2, \ldots$, with $\Omega \in \mathrm{SO}(3)$, $t \in [t_n, t_{n+1}]$. Hence, in each interval $[t_n, t_{n+1}]$ we solve the differential equation

$$\Omega'(t) = A(\boldsymbol{y}(t))\Omega(t), \quad t \geq t_n, \qquad \Omega(t_n) = I, \tag{7.9}$$

where

$$A(\boldsymbol{y}(t)) = -\begin{bmatrix} 0 & -m_3 y_3 & m_2 y_2 \\ m_3 y_3 & 0 & -m_1 y_1 \\ -m_2 y_2 & m_1 y_1 & 0 \end{bmatrix}.$$

In this numerical experiment, $m_1 = 1$, $m_2 = \frac{1}{3}$ and $m_3 = \frac{1}{5}$ and $h = t_n - t_n = \frac{1}{10}$, while the initial condition is a random 3-vector with unit norm. We remark that such action automatically obeys the homogeneous-space condition $\|\boldsymbol{y}\|_2 = 1$ whenever a Lie-group method is applied to (7.9).

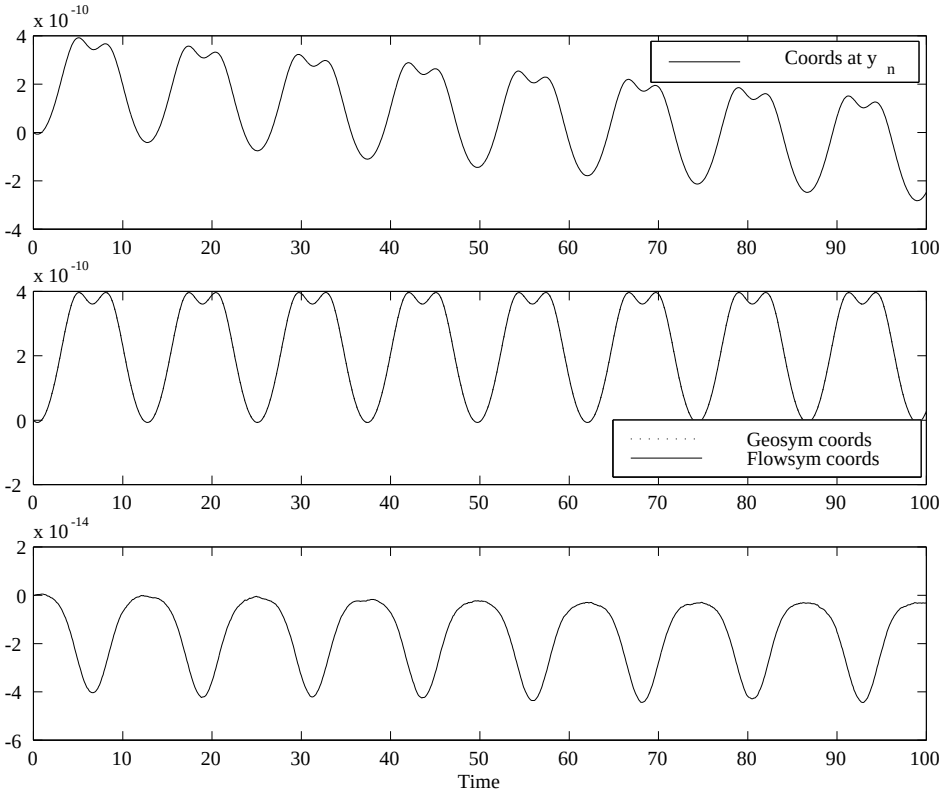


Fig. 7.2. Error in the Hamiltonian versus time for an order-four RK-MK method based on Gaussian nodes. The top plot corresponds to coordinates centred at $\boldsymbol{y}_n$ while the second plot corresponds to geodesic and flow-symmetric coordinates. Although generally the two latter choices would correspond to different error, in this case the errors are very similar. The bottom plot corresponds to the difference between the errors in geodesic and flow-symmetric coordinates.

We compare an RK-MK method of order four based on Gauss–Legendre quadrature, using coordinates centred at $\boldsymbol{y}_n$ with geodesic and flow-symmetric coordinates.

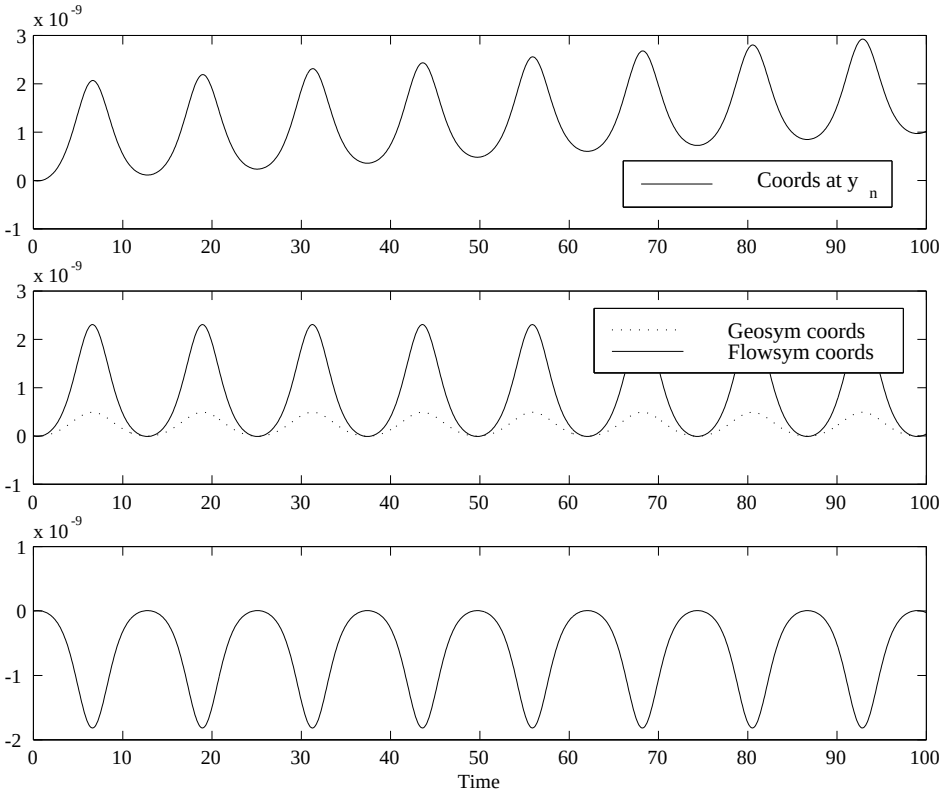


Fig. 7.3. Error in the Hamiltonian versus time for an order-four Magnus method based on Gaussian nodes. The top plot corresponds to coordinates centred at $\boldsymbol{y}_n$ while the second plot corresponds to geodesic and flow-symmetric coordinates. The bottom plot corresponds to the difference between the errors in geodesic and flow-symmetric coordinates.

The error in the Hamiltonian function, evaluated as

$$\mathrm{err}_H = H(\boldsymbol{y}_n) - H(\boldsymbol{y}_0),$$

is displayed in Figure 7.2.

Similar comparison is displayed in Figure 7.3 for a method based on Magnus expansion of order four.

Precise details of numerical schemes used in the above example can be found in Appendix A.

8. Computation of exponentials

8.1. Six dubious ways to compute the exponential of a matrix in a Lie algebra

Most (but by no means all) Lie-group methods require repeated calculation of exponentials or, in a more realistic setting, an approximation of exponentials. In principle, this is a well-tried and familiar task in numerical analysis and can be accomplished in one of several ways: rational approximants (Baker 1975, Iserles and Nørsett 1991), Krylov-subspace methods (Hochbruck and Lubich 1997), Schur decomposition (Golub and Van Loan 1996) and so on. Although such methods have occasionally attracted healthy scepticism (Moler and Van Loan 1978), it is fair to say that they have a distinguished track record across numerical analysis. However, as we have already commented in Section 6, our present task is subject to a crucial restriction: *Our approximant must map the Lie algebra* $\mathfrak{g}$ *to the Lie group* $\mathcal{G}$*!*

Low-dimensional algebras are easy and often we can evaluate the exponential explicitly. In particular, the following two cases are of practical importance.

- Firstly, given $A = \begin{bmatrix} a & b \\ c & -a \end{bmatrix} \in \mathfrak{sl}(2)$, we can easily establish that

$$\mathrm{e}^A = \cosh\omega\, \boldsymbol{I} + \frac{\sinh\omega}{\omega} A, \qquad \text{where} \qquad \omega = \sqrt{a^2 + bc}. \tag{8.1}$$

  This will be of use in Section 11, in our discussion of the application of Magnus expansions to the calculation of Sturm–Liouville spectra.
- Secondly, the exponential of

$$A = \begin{bmatrix} 0 & a & b \\ -a & 0 & c \\ -b & -c & 0 \end{bmatrix} \in \mathfrak{so}(3)$$

  is

$$\boldsymbol{I} + \frac{\sin\sigma}{\sigma} A + \frac{1-\cos\sigma}{\sigma^2} A^2,$$

  where $\sigma = \sqrt{a^2 + b^2 + c^2}$. Given the number of spatial dimensions in our universe, it will come as little surprise that many useful equations, e.g. (7.8), can be formulated in $\mathfrak{so}(3)$. We will return to the above expression, known as the *Rodrigues formula,* in Appendix B.

These, however, are the exceptions.

An exact formula being unavailable, an appealing alternative is to compute the exponential to machine accuracy. This, however, is neither affordable not always reliable. The MATLAB function `expm` computes the exponential by scaling-and-squaring a diagonal Padé approximant: the procedure is very expensive for large dimensions and the outcome often falls short of machine accuracy and is subject to fast error accumulation.

When neither an explicit formula nor computation to machine accuracy are feasible, we must resort to approximation. Any such procedure must conform with two conditions: The outcome lies in the correct Lie group $\mathcal{G}$ and it departs from the exact exponential only to an extent consistent with the order of the Lie-algebra method. A standard means of approximation is to replace e^z with a function $r(z)$, analytic in the neighbourhood of the origin. The action of such function can be extended from $\mathbb{C}$ to $\mathfrak{gl}(N)$, hence to any matrix Lie algebra, by elementary means. Our two desiderata can be now reformulated by requiring that $r(z) = \mathrm{e}^z + \mathcal{O}\big(z^{p+1}\big)$, where $p \geq 1$ is the order of the Lie-algebra time-stepping procedure, and $r(\mathfrak{g}) \subseteq \mathcal{G}$.

As we have already mentioned in Section 6, above conditions might be much too restrictive. As has been proved by Kang and Zai-jiu (1995), the only analytic function that maps $\mathfrak{sl}(N)$ to $\mathrm{SL}(N)$ for every $N \in \mathbb{N}$ and takes zero to identity is $r(z) = \mathrm{e}^{\alpha z}$ for $\alpha \in \mathbb{R}$. Requiring consistency means that we must choose the exact exponential! On the other hand, in a *quadratic* Lie algebra we are faced with an abundance of riches: given an arbitrary odd function f, analytic about the origin, it is true that $\mathrm{e}^{f(\mathfrak{g})} \subseteq \mathcal{G}$ (Celledoni and Iserles 1998). In particular, this is the case with all *diagonal Padé approximants,*

$$r(z) = \frac{p_m(z)}{p_m(-z)}, \qquad \text{where} \qquad p_m(z) = \sum_{k=0}^{m} \binom{m}{k} \frac{(2m-k)!}{(2m)!} z^k, \quad m \in \mathbb{N}.$$

Yet, all this is of lesser utility since, arguably, the method of choice for quadratic Lie algebras rests upon the use of the Cayley transform (i.e., the diagonal Padé approximant with $m = 1$) as an alternative action, thereby avoiding altogether the need to approximate the exponential function!

Yet another option is to evaluate the exponential with a *Krylov-subspace method.* Assuming for simplicity that we wish to approximate $\mathrm{e}^A \boldsymbol{v}$, where $A \in \mathfrak{gl}(N)$ and $\boldsymbol{v} \in \mathbb{R}^N$, such techniques choose the approximant from the space $\mathcal{K}_{N,M} = \mathrm{span}\,\{\boldsymbol{v}, A\boldsymbol{v}, \ldots A^{M-1}\boldsymbol{v}\}$. Surprisingly small values of M produce remarkably good and affordable approximants (Hochbruck and Lubich 1997). Yet, there is absolutely nothing in this approach to guarantee that the outcome resides in the correct Lie group.

The last (and perhaps the most obvious) candidate for our list of alternatives to the matrix exponential is *projection.* For example, to travel from $\mathfrak{sl}(N)$ to $\mathrm{SL}(n)$, we may employ a diagonal Padé approximant. The outcome, $V = r(A)$, say, cannot be expected to reside in the special linear group. However, replacing V with $V/(\det V)^{1/N}$ produces an element in $\mathrm{SL}(n)$. Unfortunately, experience tells that this procedure is prone to instability (cf. Section 11.4).

8.2. Splitting methods

Let $A \in \mathfrak{g}$. In the spirit of (Celledoni and Iserles 1998), we wish to approximate

$$\mathrm{e}^{tA} \approx R(tA) = \mathrm{e}^{tB_1} \mathrm{e}^{tB_2} \cdots \mathrm{e}^{tB_s}, \tag{8.2}$$

where the matrices $\boldsymbol{B} = \{B_1, B_2 \ldots, B_s\}$ are subject to the following requirements.

1 Each B_l resides in $\mathfrak{g}$;

2 It is cheap to evaluate $\operatorname{expm}(tB_l) \in \mathcal{G}$ exactly for each l;

3 It is cheap to multiply exponentials in (8.2);

4 The error is suitably small, $R(tA) = \mathrm{e}^{tA} + \mathcal{O}\big(t^{p+1}\big)$.

We choose the *splitting* $\boldsymbol{B}$ so that it consists of low-rank matrices.

Let K and L be two $N \times r$ matrices, where $r \geq 1$ is small, and assume that $C = B_l = KL^{\mathrm{T}}$. Then

$$\mathrm{e}^{tC} = I + tKD^{-1}(\mathrm{e}^{tD} - I)L^{\mathrm{T}},$$

where $D = L^{\mathrm{T}}K$ (Celledoni and Iserles 1998). Note that D is just $r \times r$ and the cost of evaluating e^{tC} exactly is modest for small values of r.

As an example, let us consider $\mathfrak{g} = \mathfrak{so}(N)$. We let $r = 2$, $s = N - 1$, set $B^{[0]} = A = [\boldsymbol{b}_1^{[0]}, \boldsymbol{b}_2^{[0]}, \ldots, \boldsymbol{b}_N^{[0]}]$, and choose $B_1 = \boldsymbol{b}_1^{[0]}\boldsymbol{e}_1^{\mathrm{T}} - \boldsymbol{e}_1\boldsymbol{b}_1^{[0]^{\mathrm{T}}} \in \mathfrak{so}(N)$, where $\boldsymbol{e}_k \in \mathbb{R}^N$ is the kth unit vector. Letting $B^{[1]} = B^{[0]} - B_1$, we observe that its first row and column vanish. We continue in a manner similar to LU factorisation, letting $B^{[i]} = B^{[i-1]} - B_i$ and

$$B_i = \boldsymbol{b}_i^{[i-1]}\boldsymbol{e}_i^{\mathrm{T}} - \boldsymbol{e}_i\boldsymbol{b}_i^{[i-1]^{\mathrm{T}}} \in \mathfrak{so}(N), \qquad i = 1, 2, \ldots, N-1.$$

We refer to (Celledoni and Iserles 1998) for precise estimation of cost, implementation details and a similar example for $\mathfrak{sl}(N)$, as well as for an example of a splitting, again with $r = 2$, that eliminates two rows and columns of $B \in \mathfrak{so}(N)$ at a time. Although the number of exponentials in (8.2) is generally quite large for low-rank splittings, the underlying linear algebra carries a reasonable price tag.

The main disadvantage of low-rank splitting methods is the quality of approximation: in general, we can expect order $p = 1$. The second-order condition is

$$\sum_{k=1}^{s-1} \sum_{l=k+1}^{s} [B_k, B_l] = \boldsymbol{O}$$

and it is easy to verify that it is satisfied when (8.2) is the *Strang splitting:* $s = 2\tilde{s}+1$ and $B_{\tilde{s}+i} = B_{\tilde{s}-i}$, $i = 1, 2, \ldots, \tilde{s}$. In principle, it is easy to convert any low-rank matrix so that it becomes a Strang splitting by first approximating $\frac{1}{2}A$ with a *first-order* splitting (8.2), next approximating the same matrix with the same splitting but with the matrices B_l arranged in reverse order and finally 'aggregating' the middle two terms. The outcome,

$$S(tA) = \mathrm{e}^{tB_1/2}\mathrm{e}^{tB_2/2} \cdots \mathrm{e}^{tB_{s-1}/2}\mathrm{e}^{tB_s}\mathrm{e}^{tB_{s-1}/2} \cdots \mathrm{e}^{tB_2/2}\mathrm{e}^{tB_1/2} \tag{8.3}$$

costs twice as much as (8.2) but it has a crucial advantage: it is not just second-order but also *time symmetric.* This renders (8.3) amenable to the application of the *Yoshida device* (Sanz Serna and Calvo 1994, Yoshida 1990). Thus, the function

$$S(\alpha tA)S((1-2\alpha)tA)S(\alpha tA), \qquad \text{where} \qquad \alpha = \frac{2}{3} + \frac{\sqrt[3]{2}}{3} + \frac{\sqrt[3]{4}}{6}$$

approximates e^{tA} to order four. Similar procedure can we used to increase the order further in increments of two.

A most welcome feature of low-rank splittings is that they can be implemented to take advantage of sparsity. Provided that A is banded, say, all the computations can be confined to the relevant band and sparsity is inherited as we are 'mopping up' rows and columns similarly to the $\mathfrak{so}(N)$ algorithm above.

8.3. Canonical coordinates of the second kind

Our point of departure is similar to the reasoning behind the CCSK representation (6.16). Again, $\boldsymbol{C} = \{C_1, C_2, \dots, C_d\}$ is a basis of the Lie algebra $\mathfrak{g}$ and we seek polynomials $\theta_1, \theta_2, \dots, \theta_d$ so that

$$\mathrm{e}^{\theta_1(t)C_1}\mathrm{e}^{\theta_2(t)C_2}\cdots\mathrm{e}^{\theta_d(t)C_d} = \mathrm{e}^{tA} + \mathcal{O}\big(t^{p+1}\big)\,. \tag{8.4}$$

$\boldsymbol{C}$ being a basis, there exist scalars $a_1, a_2, \dots, a_d$ so that $A = \sum_{k=1}^{d} a_k C_k$. Letting $\theta_k(t) = a_k t$, $k = 1, 2, \dots, d$, gives us a first-order splitting (8.2). With greater generality, we set $\theta_k(0) = 0$, $\theta_k'(0) = a_k$, $k = 1, 2, \dots, d$, to guarantee $p \geq 1$.

To obtain higher-order conditions in (8.4) we proceed like in Section 6.4. Differentiation and further algebra produce, equivalently to (6.17), the equation

$$A = \sum_{k=1}^{d} a_k C_k = \sum_{k=1}^{d} \theta_k'(t)\mathrm{Ad}_{\,\mathrm{expm}[\theta_1(t)C_1]}\cdots\mathrm{Ad}_{\,\mathrm{expm}[\theta_1(t)C_1]}C_k + \mathcal{O}(t^p)\,. \tag{8.5}$$

We go on differentiating (8.5) and letting $t = 0$. This yields order conditions, which need be unscrambled by further algebra, exploiting the *structure constants* (cf. Section 6.4 for the definition) of $\boldsymbol{C}$. The outcome is

$$p \geq 2: \qquad \theta_k''(0) = \sum_{l=1}^{d}\sum_{j=1}^{l-1} a_l c_{l,j}^k a_j, \qquad k = 1, 2, \dots, d$$

$$p \geq 3: \qquad \theta_k'''(0) = 2\sum_{l=1}^{d}\sum_{j=1}^{l-1} c_{l,j}^k[\theta_l''(0)a_j + a_l\theta_j''(0)] + 2\sum_{l=1}^{d}\sum_{j=1}^{l-1}\sum_{i=1}^{j-1}\sum_{m=1}^{d} c_{l.j}^m c_{i,m}^k a_l a_j a_i$$
$$+ \sum_{l=1}^{d}\sum_{j=1}^{d}\sum_{i=1}^{l-1} c_{l,i}^j c_{i,j}^k \beta_l \beta_i^2, \qquad k = 1, 2, \dots, d$$

(Celledoni and Iserles 1999).

The cost of this procedure is, at the first glance, prohibitive. We might just about get away with computing order-two conditions at the cost of $\mathcal{O}\big(d^3\big)$ operations, but a price tag of $\mathcal{O}\big(d^5\big)$ operations for order three, to say nothing of higher orders, is out of the question. This naive impression is misleading, since we are absolutely free to exploit, along the lines of Sections 6.4–5, sparsity in structure constants in a serendipitously-chosen basis $\boldsymbol{C}$.[4] Again, Chevalley bases present the best choice, as well as leading to very easy computation of exponentials in (8.4). For example, choosing the basis $\{E_{k,l} - E_{l,k} \,:\, 1 \leq k < l \leq N\}$ of $\mathfrak{so}(N)$, an order-two approximation is attained by letting

$$\theta_{k,l}(t) = a_{k,l}t + \tfrac{1}{2}\sum_{i=1}^{N} a_{k,i}a_{i,l}t^2, \qquad 1 \leq k < l \leq N,$$

an $\mathcal{O}\big(N^3\big) = \mathcal{O}\big(d^{3/2}\big)$ procedure. In the case of $\mathfrak{sl}(N)$ the approach advocated in (Celledoni and Iserles 1999) is to choose the ordered basis $\boldsymbol{C} = \boldsymbol{C}_1 \cup \boldsymbol{C}_2$, where

$$\boldsymbol{C}_1 = \{E_{k,l} \,:\, k \neq l\}, \quad \boldsymbol{C}_2 = \{E_{k,k} - E_{k+1,k+1} \,:\, k = 1, 2, \ldots, N-1\},$$

where each set is ordered lexicographically. Let

$$A = \sum_{\substack{k,l=1 \\ k \neq l}}^{N} a_{k,l}E_{k,l} + \sum_{k=1}^{N-1} b_k(E_{k,k} - E_{k+1,k+1})$$

and denote the coefficients corresponding to terms in $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ by $\theta_{k,l}$ and η_k respectively. The second-order conditions are

$$\begin{aligned}
\theta_{k,l}(t) \;&=\; a_{k,l}t + \tfrac{1}{2}\left[\sum_{i=1}^{k-1} a_{k,i}a_{i,l} + a_{k,l}(b_{k-1} - b_k + b_l - b_{l-1})\right. \\
&\qquad \left. - \sum_{i=k+1}^{N} a_{k,i}a_{i,l}\right]t^2, \qquad k \neq l, \\
\eta_k(t) \;&=\; b_k t - \tfrac{1}{2}\sum_{i=1}^{k}\sum_{j=k+1}^{N} a_{i,j}a_{j,i}t^2, \qquad k = 1, 2, \ldots, N-1,
\end{aligned}$$

again just $\mathcal{O}\big(N^3\big) = \mathcal{O}\big(d^{3/2}\big)$ operations.

An intriguing aspect of approximants based on CCSK is that they might well be ideally suitable to handle sparsity in the matrix A. Although this issue is by no

[4] Sparsity in structure constants has no connection whatsoever with sparsity (or otherwise) of the matrix A.

means fully understood, there are enough encouraging pointers to justify a brief discussion. One mechanism that exploits sparsity is that the latter can be taken into account in the evaluation of the θ_ks because of the association between terms in a Chevalley basis and the entries of A. For example, the second-order conditions for a *tridiagonal* matrix $A \in \mathfrak{sl}(N)$ reduce to

$$\theta_{k,l}(t) \;=\; \begin{cases} \frac{1}{2}a_{k,k-1}a_{k-1,k-2}t^2, & l = k-2, \\ a_{k,k-1}t - \frac{1}{2}a_{k,k-1}(b_k - 2b_{k-1} + b_{k-2})t^2, & l = k-1, \\ a_{k,k+1}t + \frac{1}{2}a_{k,k+1}(b_{k+1} - 2b_k + b_{k-1})t^2, & l = k+1, \\ -\frac{1}{2}a_{k,k+1}a_{k+1,k+2}t^2, & l = k+2, \\ 0, & |k-l| \neq 1, \end{cases}$$

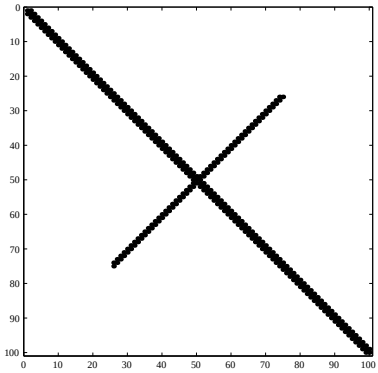
$$\eta_k(t) \;=\; b_k t - \tfrac{1}{2}a_{k,k+1}a_{k+1,k}t^2.$$

This entails just $\mathcal{O}(N)$ operations and, equally importantly, just $\mathcal{O}(N)$ terms survive in the product (8.4). The cost scales with the number of nonzero elements in the matrix, rather than with dimension, a hallmark of a good method for sparse matrices.

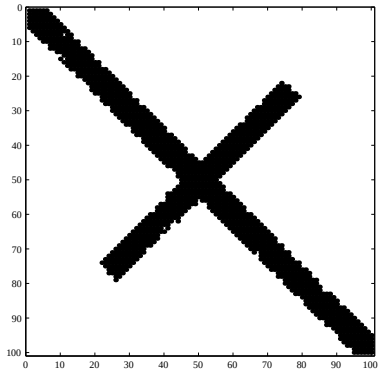
Another mechanism is of a more tentative value, yet we believe that it deserves mentioning. Even if the matrix A is sparse, its exponential is dense. However, it is possible to prove, at least in the case of *banded matrices,* that most of the entries are exceedingly small and that the loci of large elements are predictable (Iserles 1999*a*). As an example, we have averaged 1000 exponentials of 100×100 matrices with the cruciform sparsity pattern displayed in Figure 8.1a and with random elements uniformly distributed in $(-1, 1)$ and normalised so that $\max_{k,l} |a_{k,l}| = 1$. Let W be the average of all the exponentials. The matrix is dense, yet most of the elements of W are tiny! Thus, Figure 8.1b displays the sparsity pattern of W without all its entries that are smaller than 10^{-6} in magnitude. Although the cruciform shape 'swells', most of the matrix consists of zero entries. This observation is affirmed in Figure 8.1c, where we have plotted the matrix $\log_{10} |W|$. The vertical axis tells the magnitude of the entries in terms of significant (decimal) digits. The decay outside the original cruciform shape is evident. Using upper bounds from (Iserles 1999*a*), it is possible to say how fast entries decay for banded A but computer experiments (and, indeed, Figure 8.1) indicate that similar behaviour takes place for more exotic sparsity patterns.

Choosing a Chevalley basis, members of $\boldsymbol{C}$ mostly correspond to elements of A. Given a tolerance $\varepsilon > 0$ and knowing which elements of e^A are bound to be smaller than ε in magnitude, we are free to remove them altogether from the product (8.4). The outcome departs entry-by-entry from the exact exponential by at most ε and, by design, it resides in the Lie group $\mathcal{G}$.

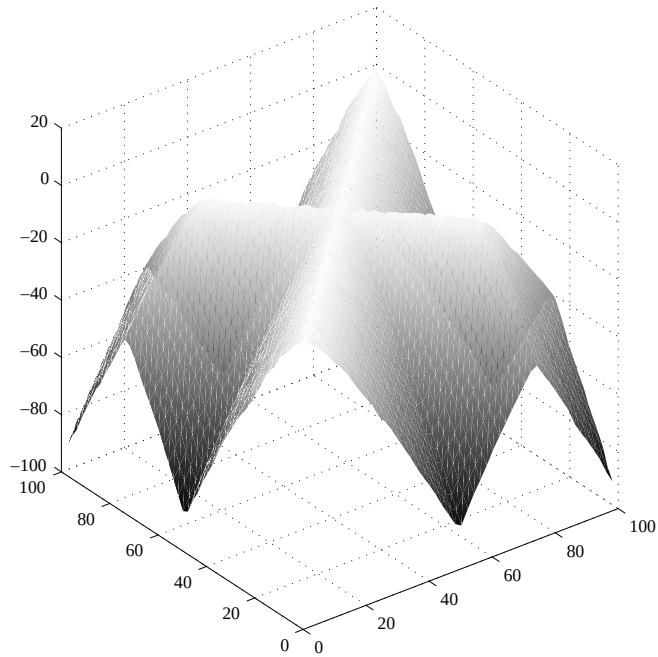
The sparsity pattern of a matrix exponential of a sparse matrix after the excision of small entries is at present unknown. (Banded matrices are an exception.) Moreover, full implications of this phenomenon to the subject matter of this sec-



a. The sparsity pattern in the matrix A



b. Average sparsity pattern in the matrix $|(\mathrm{e}^A)_{k,l}| > 10^{-6}$



c. The 'shape' of the matrix $\log_{10} |(\mathrm{e}^A)_{k,l}|$

Fig. 8.1. How large is the exponential of a sparse matrix?

tion are far from clear. Having said this, our analysis emphasises a very important and welcome feature of methods based upon canonical coordinates of the second kind, applied in conjunction with Chevalley bases: the connection between basis elements and entries of the matrix is a powerful tool, which we can bring to bear with pointwise precision upon the exploitation of sparsity.

9. Stability and backward error analysis

How stable are Lie-group methods? A combined wisdom of half-a-century of computational analysis of ODEs is that numerical algorithms for initial-value problems are to all intents and purposes useless unless they exhibit favourable stability properties. Indeed, much of the narrative of modern numerical analysis of ODEs is the tale of stability, linear and nonlinear alike, culminating in a profound understanding of the subject. We refer the reader to the monograph of Stuart and Humphries (1996) for a comprehensive review of this important subject area.

The word 'stability' has been so far conspicuously absent from our exposition. A partial reason is ignorance: much remains to be done in the realm of stability investigations in a Lie-group setting. Interesting results abound which cannot yet be fitted into general theory. Thus, we can learn from computation that merely projecting a solution into the right manifold often leads to instabilities, while intrinsic Lie-group methods exhibit much more favourable behaviour. Much of the advance in stability theory for classical numerical ODEs was concerned with identifying appropriate *stability models:* broad enough to provide insight about many differential systems of interest, yet sufficiently focused and narrowly defined to be amenable to rigorous analysis. Thus, the linear model, the monotone model and the many more advanced models from (Stuart and Humphries 1996), originating in the theory of nonlinear dynamical systems. This chapter in the narrative of Lie-group methods cannot yet be written, except for the observation that some Lie-group solvers, e.g. RK-MK, Magnus and Fer expansions with exactly-calculated exponentials compute the solution of linear equations with *constant coefficients* exactly: in that case there is no need for stability analysis!

The last few years have seen the emergence of an alternative stability theory, mainly within the context of symplectic integration of Hamiltonian ODEs and discretization of dynamical systems. In addition to asking "how near is the numerical solution to the exact one and how influenced is it by small perturbations?", the new breed of stability researchers also poses a different query: "what is it that our numerical method solves *exactly?* And how far apart is it from the equation that we wish to solve?". This is precisely the question of *backward error analysis* that James Hardy Wilkinson made into the centrepiece of modern numerical linear algebra. Arguably, it is just as relevant in the ODE setting and it has already led to impressive new insights (Benettin and Giorgilli 1994, Hairer 1994, Hairer and Lubich 1997, Neishtadt 1984, Reich 1996).

In this section we report briefly on recent work of Faltinsen (1998), who has generalised backward error analysis to a Lie-group setting.

Discussion of stability is meaningless without the concept of *distance.* Thus, given a Lie group $\mathcal{G}$, we seek $d : \mathcal{G} \times \mathcal{G} \to \mathbb{R}^+$ which is consistent with the standard axioms of a metric in an Euclidean space and, in addition, compatible with the topology of $\mathcal{G}$. A good way to ensure compatibility is to require that d is *left invariant,* i.e. that

$$d(ZX, ZY) = d(X, Y) \qquad X, Y, Z \in \mathcal{G}. \tag{9.1}$$

Ideally, we would have liked d to be *bi-invariant:* satisfying both (9.1) and the condition $d(XZ, YZ) = d(X, Y)$ for all $X, Y, Z \in \mathcal{G}$. This, however, is not always possible. On the other hand, according to the Birkhoff–Kakutani theorem, every Lie group $\mathcal{G}$ admits a left-invariant, *almost right-invariant* metric which, in addition to (9.1), obeys $d(XZ, YZ) \leq \rho(Z)\, d(X, Y)$, where the function ρ is finite.

Assumption 9.1 The metric d is left-invariant and almost right-invariant.

An example is the *geodesic metric* in $\mathrm{O}(N)$ (which, as a matter of fact, is bi-invariant): $d(X, Y) = \|\boldsymbol{\eta}\|_2$, where the eigenvalues of $X^{\mathrm{T}}Y$ are $\sigma_1, \sigma_2, \ldots, \sigma_N$ and $\eta_k = -\mathrm{i}\sigma_k$, $k = 1, 2, \ldots, N$. (Note that the spectrum of elements in $\mathrm{O}(N)$ lives on the complex unit circle, hence the η_ks are real.) Other examples are more complicated to derive explicitly, but this is of little consequence since we do not require d in a closed form.

We are concerned with the solution of the Lie-group equation

$$Y' = A(t, Y)Y, \quad t \geq t_0, \qquad Y(t_0) = Y_0 \in \mathcal{G}, \tag{9.2}$$

where $A : [t_0, \infty) \times \mathcal{G} \to \mathfrak{g}$, by a Lie-group method. The flow corresponding to (9.2) is denoted by $\Psi_{t,t_0,A}$, therefore $Y(t) = \Psi_{t,t_0,A}(Y_0)$.

Assumption 9.2 The matrix function A is real analytic and there exist constants $\alpha, \beta, t^* > 0$ such that

$$\|A(t, \hat{Y})\hat{Y}\|_{\mathrm{F}} \leq \alpha, \qquad t \in [0, t^*], \quad \hat{Y} \in \mathcal{B}_\beta(Y_0),$$

where $\| \cdot \|_{\mathrm{F}}$ is the *Forbenius norm* and $\mathcal{B}_\beta(Y_0) = \{\hat{Y} \in \mathcal{G} : d(Y_0, \hat{Y}) \leq \beta\}$.

We are interested in Lie-group solvers that lift the solution to the corresponding Lie algebra $\mathfrak{g}$, whether once or repeatedely in the course of every time step. All the methods that we have described in this survey: Crouch–Grossman and Runge–Kutta–Munthe-Kaas schemes, Magnus, Fer and Cayley expansions and methods based on canonical coordinates of the second kind fit this framework. An example of a method that is outside the scope of the theory of this section is *projection.* For example, in the case $\mathcal{G} = \mathrm{O}(N)$ we might time-step from Y_n to Y_{n+1}, say with an arbitrary ODE method which produces a new value $\bar{Y}_{n+1}$. We subject $\bar{Y}_{n+1}$ to a *polar decomposition* and retain the orthogonal part as our new Y_{n+1} (Higham

1997). We hasten to acknowledge that this is a perfectly valid procedure, except that it is outside the scope of our present discussion.

The *map* induced by the numerical method will be denoted by $\Phi_{h,t_n,A}$, hence $Y_{n+1} = \Phi_{h,t_n,A}(Y_n)$, $n \in \mathbb{Z}^+$.

Assumption 9.3 The Lie-group method is accurate to order $p \geq 1$, i.e.

$$d(\Psi_{h,t_0,A}(Y_0), \Phi_{h,t_0,A}(Y_0)) = \mathcal{O}\big(h^{p+1}\big)\,.$$

What is it that $\Phi_{h,t_n,A}$ solves exactly? In linear algebra this is precisely the question of backward error analysis. Insofar as ODEs are concerned, however, the situation is slightly more complicated.

Theorem 9.1. (Faltinsen (1998)) Subject to Assumptions 9.1–3, there exists a matrix function $A_h : [t_0, \infty) \times \mathcal{G} \to \mathfrak{g}$ such that

$$\|A_h(t, X) - A(t, X)\|_{\mathrm{F}} = \mathcal{O}(h^p) \tag{9.3}$$

and

$$d(\Psi_{h,t_0,A_h}(Y_0), \Phi_{h,t_0,A}(Y_0)) = \mathcal{O}\Big(\mathrm{e}^{-\gamma/h}\Big)\,, \tag{9.4}$$

where $\gamma > 0$ is a constant.

Note that Ψ_{h,t_0,A_h} evolves on the very same Lie group $\mathcal{G}$ and, because of (9.3), the *modified equation* $Y_h' = A(t, Y_h)Y_h$ 'approximates' the ODE (9.2). The above theorem argues that a single step of the numerical method *departs to an exponentially small extent* from the exact solution of a nearby equation! It is reminiscent of similar results in symplectic integration and its method of proof generalises the work of Reich (1996) to Lie-group setting.

A good approximation across a single step does not tell much. Ideally, we wish to extend the scope of Theorem 9.1 to $[t_0, \infty)$ or, at the very least, to a large number of steps. This, however, requires further conditions. Similarly to backward error analysis for symplectic integration, the verification of such conditions in a nonlinear case might be difficult and require bespoke analysis for different methods. Matters simplify a great deal, though, for linear equations $Y' = A(t)Y$. Suppose that the exact solution is $Y(t) = \operatorname{expm}[\Theta(t)]Y_0$, $t \geq t_0$. Let $\mu \in \mathbb{R}$ be the least real number such that

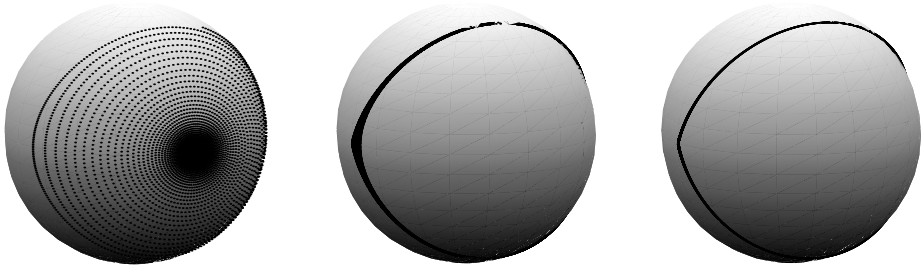
$$\|\mathrm{Ad}_{\operatorname{expm}\Theta} B\|_{\mathrm{F}} \leq c\,\mathrm{e}^{\mu(t-t_0)}\|B\|_{\mathrm{F}}, \qquad t \geq t_0$$

for some $c > 0$ which may depend on Θ. Then (9.4) can be extended to a longer interval. Specifically, it is true that

$$d(\Psi_{t_m,t_0,A_h}(Y_0), \Phi_{mh,t_0,A}(Y_0)) = \mathcal{O}\Big(\mathrm{e}^{-\gamma^*/h}\Big)\,,$$

where $\gamma^* > 0$ and $m \leq M(h)$, where

$$M(h) = \begin{cases} \mathcal{O}(1)\,, & \mu > 0, \\ \mathcal{O}(h^{-p})\,, & \mu = 0, \\ \infty, & \mu < 0. \end{cases}$$



a. Forward Euler b. Runge–Kutta–Heun c. Trapezoidal rule

Fig. 9.1. Rigid body equations (7.8), as solved by three different methods with $h = \frac{1}{10}$, integrating for 100000 steps.

It is possible to prove that $\mathfrak{g} = \mathfrak{so}(N)$ implies that the operator $\mathrm{Ad}_{\exp m\Theta}$ is itself skew symmetric and $\mu = 0$. This implies that the scope of backward error analysis is quite significant at least in this case.

Another interesting phenomenon originally identified in the analysis of Hamiltonian problems is *linear error growth* (Estep and Stuart 1995). Provided that the exact solution of a Hamiltonian differential system is periodic with period $T > 0$ and a pth-order numerical method satisfies convenient requirements (e.g., reversibility or conservation of Hamiltonian energy), it is possible to prove that

$$\|\boldsymbol{y}_{nT/h} - \boldsymbol{y}(nT)\| \leq cnh^p, \tag{9.5}$$

where $c > 0$ (for simplicity we assume that T/h is integer). This is a very important feature of successful long-term integration, since lesser methods typically produce quadratic error growth and the solution is unlikely to remain periodic for long.

Does (9.5) remain valid in a Lie-group setting? Unfortunately, not always. In a recent paper, though, Engø and Faltinsen (1999) proved that solving a *Lie–Poisson system* with a method that is both a Lie-group action and conserves Hamiltonian energy results in linear error growth. As an example, let us recall the Euler equations for a rigid body (7.8). This is a Lie–Poisson system which conserves the Hamiltonian energy $H(\boldsymbol{y}) = \frac{1}{2}(m_1 y_1^2 + m_2 y_2^2 + m_3 y_3^2)$, as well as evolving on the unit sphere in $\mathbb{R}^3$. Therefore $\boldsymbol{y}(t)$ lives on a circle (an intersection of a sphere and an ellipsoid), a feature shared by energy-conserving methods based on group actions, and is periodic there. We have already seen in Figures 7.2 and 7.3 that self-adjoint methods do well in recovering a periodic solution.

In Figure 9.1 we have integrated the rigid body equations (7.8) with three RK-MK methods: Forward Euler, Runge–Kutta–Heun and the trapezoidal rule, all applied in the Lie algebra. Only the latter method conserves energy (Engø and

Faltinsen 1999). The results have been displayed as point-plots of $\boldsymbol{y}_n$ in three dimensions. First note that all the solution trajectories evolve on the unit sphere: unsurprising, since we are using Lie-group methods, yet beyond the reach of most classical algorithms. Secondly, forward Euler is no respecter of periodicity and its trajectory spirals to a fixed point. Runge–Kutta–Heun is much better, yet more careful examination demonstrates how the error accumulates and periodicity is lost. The solution is qualitatively correct for a while, but long integration leads to false dynamics also in this case. The energy-conserving trapezoidal rule, though, produces a trajectory which to all intents and purposes is periodic.

10. Implementation, error control and *DiffMan*

10.1. Implementation and error control of Lie-group solvers

Practical implementation of Lie-group solvers requires much more than merely to program a numerical method. We must address ourselves to issues like error control and variable-step implementation. As is perhaps natural in a new subject, implementation details have received less attention so far than theoretical issues, a situation that is likely to be remedied in the next few years.

In this section we survey the little that is presently known about implementational issues, commencing with the welcome observation that variable-step procedure does not interfere with the retention of Lie-group structure. This is important, since it is known that another important geometric-integration technique, symplectic solution of Hamiltonian systems, loses many of its most favourable features unless implemented with (essentially) constant step size (Sanz Serna and Calvo 1994).

There are two levels of discretization in Lie-group solvers and each should be monitored in variable-step implementation and contribute to the estimate of local error:

1 The error committed in the evaluation of the coordinate map; and
2 The error incurred in the solution of the Lie-algebraic equation.

Insofar as the coordinate map is concerned, the situation is simple. The Cayley map (6.3) requires an inversion of a matrix. Unless its size is large, this can be accomplished by direct methods, otherwise it requires iteration. In the first case the only source of error is roundoff and this issue is well understood by classical numerical algebra. In the second case the error is determined by the termination criteria and the issue is, again, transparent. The use of techniques based on coordinates of second kind (6.15) generates roundoff error and nothing else, since each individual exponential can be evaluated easily in exact arithmetic. Unfortunately, the situation is different with the most important coordinate map, expm. To date, there are no efficient means to monitor the error of methods from Section 8. The last statement refers not just to approximation methods but also to 'exact' calculation of the exponential: it will be seen in Section 11.5 that the MATLAB function

`expm` is a stumbling block to high-precision long-term integration of some systems.[5] It is entirely conceivable that good techniques to monitor the error in expm should depend on the Lie group in question: intuitively, a compact object like $\mathrm{O}(N)$ is easier to handle than $\mathrm{SL}(N)$, say.

Estimation of the error in Lie algebra presents different challenge, more akin to classical error-control theory for numerical ODEs.

A fair share of quality software for 'classical' ODEs is based on multistep methods. There exist multistep Lie-group solvers (Faltinsen, Marthinsen and Munthe-Kaas 1999), but their implementation requires travelling forwards and backwards between the group and the algebra in a manner which is, arguably, too cumbersome for the use of 'automatic differentiation' techniques of Gear (1971), the cornerstone of most multistep ODE software.

In principle, error control of explicit RK-MK can be accomplished in a straightforward manner, using *embedded Runge–Kutta schemes* (Hairer et al. 1993). The sole difference with the classical framework is that, instead of estimating the error in the original configuration space, we do so in the algebra. An embedded RK scheme has the Butcher tableau

$$\begin{array}{c|cccc} c_1 & a_{1,1} & a_{1,2} & \cdots & a_{1,\nu} \\ c_2 & a_{2,1} & a_{2,2} & \cdots & a_{2,\nu} \\ \vdots & \vdots & \vdots & & \vdots \\ c_\nu & a_{\nu,1} & a_{\nu,2} & \cdots & a_{\nu,\nu} \\ \hline & b_1 & b_2 & \cdots & b_\nu \\ & \bar{b}_1 & \bar{b}_2 & \cdots & \bar{b}_\nu \end{array}$$

and the approximants

$$\begin{aligned} \boldsymbol{y}_{n+1} &= \boldsymbol{y}_n + h \sum_{l=1}^{\nu} b_l \boldsymbol{f}_l \\ \bar{\boldsymbol{y}}_{n+1} &= \boldsymbol{y}_n + h \sum_{l=1}^{\nu} \bar{b}_l \boldsymbol{f}_l \end{aligned}$$

(compare with (3.3)!) are of order p and $\bar{p} \geq p+1$ respectively. The higher-order approximant is used for error control, $\boldsymbol{y}_{n+1} - \boldsymbol{y}(t_{n+1}) \approx \boldsymbol{y}_{n+1} - \bar{\boldsymbol{y}}_{n+1}$. Applying similar trick in the Lie algebra readily yields an estimate of the local error at relatively little extra cost for low-order methods. If the order is high, typically the embedded RK scheme requires increasingly large number of additional stages to evaluate $\bar{\boldsymbol{y}}_{n+1}$ and the cost mounts. This is a problem common to high-order RK methods, also in classical setting.

We have neglected in our discussion of RK-MK one important source of error:

[5] In fairness to MATLAB, in our experience `expm` performed better than alternatives.

in practical applications the dexp^{-1} operator (2.46) is truncated consistently with the order of the method, typically replaced with

$$\mathrm{dexp}_A^{-1}(C,p) = \sum_{j=0}^{p-1} \frac{\mathrm{B}_j}{j!} \mathrm{ad}_A^j C.$$

This carries error which we are forbidden to neglect, yet can estimate easily from the leading term of $\mathrm{dexp}_A^{-1} C - \mathrm{dexp}_A^{-1}(C,p)$,

$$\frac{|\mathrm{B}_q|}{q!} \|\mathrm{ad}_A^q C\|, \qquad \text{where} \qquad q = \begin{cases} 1, & p = 1, \\ 2\lfloor (p+1)/2 \rfloor, & p \geq 2. \end{cases}$$

Much more challenging is error control of Magnus expansions, the subject of (Iserles, Marthinsen and Nørsett 1999). Again, there are two discretization steps: truncation by power (4.14) of the Magnus expansion and the replacement of integrals by quadrature. In the sequel we restrict the discussion to the linear case $Y' = A(t)Y$.

To estimate the error in the leading truncation term, we commence by assuming that the leading terms in the expansion of the matrix A are known,

$$A(t) = C_0 + C_1 t + C_2 t^2 + \cdots .$$

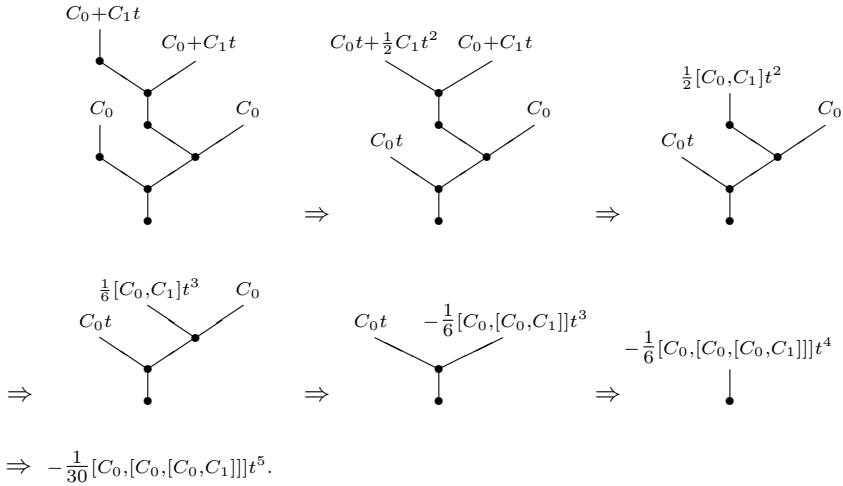
The error term in $\int_0^t \mathcal{C}_\tau(\xi)\mathrm{d}\xi$ can be evaluated from the tree τ in the following manner. We label each leaf of the tree by few leading terms of the expansion and prune the tree according to the original composition rules from Section 4.1:

1 If two leaves share a parent, they are excised and the parent is labelled by the commutator of their labels.

2 If a leaf is the only child of a parent then it is eliminated and its parent is labelled with the integral of its label.

By the end of this procedure we throw away all the terms except for the leading

one. An example will clarify this procedure,



This procedure has been automated in (Iserles et al. 1999), using a `Maple` program.

Often it is easy to find derivatives of A explicitly but in general-purpose software one needs to approximate them by finite differences. The scaled function values $A_1, A_2, \ldots, A_\nu$ are a perfectly good starting point for this elementary calculation. Note that very few derivatives are required, e.g. just C_0 and C_1 for the power-5 tree in the above example, and the approximants need not be very precise. If we are using the adjoint basis, we might just as well use (appropriately scaled) $h^{-l}B_l$ in place of C_{l-1}: the difference between expansions at 0 and $\frac{1}{2}h$ is subsumed into higher-order terms.

The computation of quadrature error is more challenging and the results of (Iserles et al. 1999) are of a more tentative nature. This is hardly surprising, since even the estimation of the error of univariate Gauss–Legendre quadrature is difficult (Davis and Rabinowitz 1984). Indeed, perhaps paradoxically, the univariate quadrature is the most problematic also in our setting. Of course, we may evaluate each integral by two quadratures, e.g. Gauss–Legendre and Gauss–Lobatto, or perhaps consecutive Gauss–Legendre rules, but this doubles the cost. The remedy, proposed in (Iserles et al. 1999), is to use again the derivatives of A: recall that in many instances they are easy to evaluate explicitly.

Straightforward, yet messy, calculation, expanding everything in sight into Taylor series in h, demonstrates that the leading error terms in the first four integrals in

(4.10), using two-point Gauss–Legendre quadrature, are

$$\begin{aligned} E_1 &= -\tfrac{1}{180} C_4 h^5 \\ E_2 &= -(\tfrac{1}{240}[C_0, C_3] + \tfrac{1}{1080}[C_1, C_2])h^5, \\ E_3 &= \tfrac{1}{1080}([C_0, [C_0, C_2]] + [C_1, [C_0, C_1]])h^5, \\ E_4 &= -(\tfrac{1}{270}[C_0, [C_0, C_2]] + \tfrac{1}{720}[C_1, [C_0, C_1]])h^5. \end{aligned}$$

respectively. Note that only E_1 depends on the fourth derivative.

A MATLAB program incorporating variable time-stepping strategy with above error estimators for the fourth-order Magnus method has been applied in (Iserles et al. 1999) to a large number of test problems, in each case behaving predictably and producing error consistent with the specified tolerance. Yet, much work remains to be done in this subject area, not just to estimate quadrature error without the need for higher derivatives, but also to develop the right strategies to the methods of Blanes et al. (1999) from Section 5.4.

10.2. The DiffMan package

DiffMan (Engø et al. 1999) is an object-oriented MATLAB toolbox for solving differential equations on manifolds. The package embodies most of the algorithms discussed in this survey. This software is in the public domain and it can be downloaded from the *DiffMan* home page at

`http://www.ii.uib.no/diffman`.

Some of the numerical examples of Section 11 are distributed with *DiffMan*. Since the package is still in the stage of intensive development, we do not wish to elaborate excessively upon the details of *DiffMan* and of its usage. Instead, we refer the reader to the latest version of the *DiffMan* user manual, which can be found at the above-mentioned homepage.

The basic philosophy behind *DiffMan* is the idea of *Coordinate-free numerics* (Munthe-Kaas and Haveraaen 1996), a research programme devoted to the study of the role of abstract formulations, independent of particular representations, in computational and applied mathematics. We have touched briefly upon this topic in Section 3, emphasising the importance of using abstractions as a tool for organising object-oriented software.

The *DiffMan* package is built upon classes which are modelling continuous mathematical structures. There are currently three main types of classes: *domains, fields* and *flows.* Domains consist of Lie algebras, Lie groups and homogeneous manifolds. Fields are structures built over manifolds. Currently the only fields are vector fields, but more general tensor fields (and possibly the even more general fibre bundles) are likely to be included in the future. Numerical algorithms are collected in the class of flows on manifolds.

The *DiffMan* package displays the importance of integrating symbolic and numerical techniques in the same software. For example, a particular Lie algebra implemented is the *free Lie algebra* class. It is capable of symbolic computations which remain valid in *any* particular finite-dimensional Lie algebra. Hence, numerical algorithms may be developed using this package, and the resulting expressions may be evaluated later, substituting data from a concrete Lie algebra. Abstract concepts play a central role, characteristic not just of pure mathematics but also of modern non-numerical software.

Another important insight that has emerged from this research is the observation that *some* numerical algorithms require detailed knowledge of representations of objects, while others can be completely formulated using general, coordinate-independent operations. This distinction is most easily seen in the area of solving linear algebraic equations. For example, *Gaussian elimination* requires detailed knowledge of matrices in terms of components (there is a major difference between sparse and dense Gaussian elimination!), while other algorithms such as *conjugate gradients* just require access to matrices via their properties as linear operators. Algorithms which can be formulated in terms of general, coordinate-independent operations are much more flexible in their use than algorithms tied to particular representations. The fact that Runge–Kutta methods can indeed be phrased as coordinate-free algorithms has many implications that we are only now beginning to understand.

11. Applications

11.1. The heavy top

Many systems evolving in the physical space can be modelled by motions consisting of rotations, like the rigid body that we have already encountered in (7.8), or translations, or combination thereof, in which case we say that the motion is described by the *special Euclidean group* SE(3).

The *heavy top equations* are an important instance in which we make use of SE(3) action. It differs from the rigid body (7.8) because of the presence of gravity.

The equations of motion can be described in either *space* or *body coordinates.* In this section we briefly discuss both, considering space coordinates first. We denote by $\mathbf{\Pi}_{\mathrm{s}}$ the angular momentum of the heavy top, by $\mathbf{\Omega}_{\mathrm{s}}$ its angular velocity, by I_{s} the inertia tensor and by $\boldsymbol{\lambda}_{\mathrm{s}}$ the unit vector pointing toward the centre of mass of the heavy top. The equations of motion are

$$\begin{aligned} \mathbf{\Pi}'_{\mathrm{s}} &= Mgl\boldsymbol{k} \times \boldsymbol{\lambda}_{\mathrm{s}}, \\ I'_{\mathrm{s}} &= [I_{\mathrm{s}}, \mathbf{\Omega}_{\mathrm{s}}], \\ \boldsymbol{\lambda}'_{\mathrm{s}} &= \mathbf{\Omega}_{\mathrm{s}} \times \boldsymbol{\lambda}_{\mathrm{s}}, \end{aligned}$$

where M is the mass of the top, $g = 9.81$ is the gravitational constant, l is the distance from the origin of the space-coordinate system to the centre of mass of

the heavy top and $\boldsymbol{k}$ is the unit vector along the z-axis. The angular velocity $\boldsymbol{\Omega}_{\rm s}$ is related to the angular momentum $\boldsymbol{\Pi}_{\rm s}$ by the identity

$$\boldsymbol{\Omega}_{\rm s} = I_{\rm s}^{-1}\boldsymbol{\Pi}_{\rm s}.$$

We refer the reader to (Arnold 1989, Goldstein 1980) for background, theory and notation.

It is well known that the axis of the top displays a very special motion composed of a *precession* about the vertical axis z and *nutations,* nodding up and down between two bounding angles θ_1 and θ_2 as displayed in Figure 11.1 for a top with initial angular velocity $\Omega_{\rm b} = [0, 0, 100]^{\rm T}$ about its figure axis, $l = \sqrt{3}/2$, and mass $M = 20/(gl)$. The inertia tensor in body coordinates is $I_{\rm b} = {\rm diag}(1, 1, 1/5)$ and the initial position of the axis of the top is rotated at an angle $\theta = -\pi/10$ with respect to the z-axis.

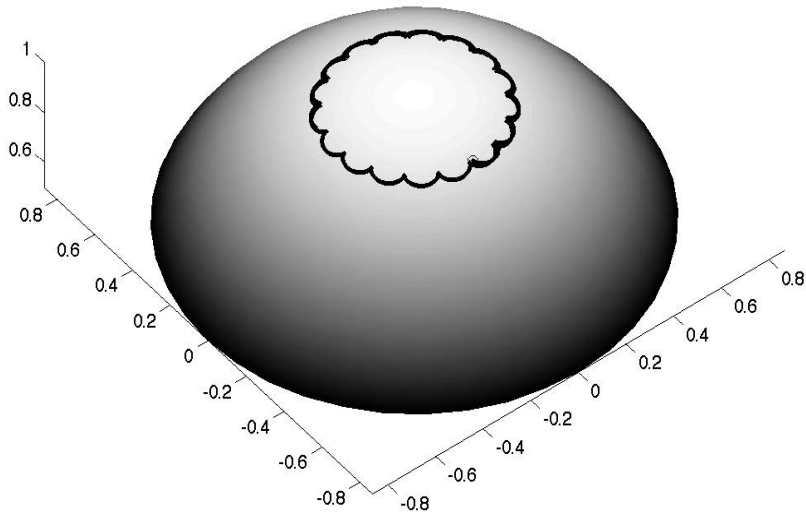


Fig. 11.1. Precession and nutations for $t \in [0, 10]$ of the axis of a heavy top.

The same motions can be described in body coordinates, in terms of two vectors $\boldsymbol{\Pi}_{\rm b}$ and $\boldsymbol{\Gamma}_{\rm b}$ in $\mathbb{R}^3$, the angular momentum and the gravity vector in body coordinates respectively (Marsden and Ratiu 1994). The equations simplify to

$$\begin{aligned} \boldsymbol{\Pi}_{\rm b}' &= \boldsymbol{\Pi}_{\rm b} \times \boldsymbol{\Omega}_{\rm b} + Mgl\boldsymbol{\Gamma}_{\rm b} \times \boldsymbol{\chi}, \\ \boldsymbol{\Gamma}_{\rm b}' &= \boldsymbol{\Gamma}_{\rm b} \times \boldsymbol{\Omega}_{\rm b}, \end{aligned} \tag{11.1}$$

where $\boldsymbol{\chi}$ is the unit vector on the figure axis of the top. Note that in body coordinates the inertia tensor is constant. Also in this case, $\boldsymbol{\Omega}_{\rm b} = I_{\rm b}^{-1}\boldsymbol{\Pi}_{\rm b}$. Although the vector $\boldsymbol{\Gamma}_{\rm b}$ is parallel to the z axis of the space-coordinate system, its motion does not correspond to the motion of the figure axis $\boldsymbol{\lambda}_{\rm s}$ of the heavy top. To reconstruct the motion of the axis we integrate the configuration matrix $R \in {\rm SO}(3)$, so that

$$\boldsymbol{\Omega}_{\rm s} = R\boldsymbol{\Omega}_{\rm b}$$

and in general the same type of transformation of body coordinates to space coordinates holds for all relevant vectors. The configuration matrix R obeys the differential equation

$$R' = -R\widehat{\boldsymbol{\Omega}}_{\rm b}. \tag{11.2}$$

Once the configuration matrix R is known, the position of the axis of the top can be recovered by means of the transformation

$$\boldsymbol{\lambda}_{\rm s} = R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

In our first experiment, the equations (11.1)-(11.2) have been solved with the MATLAB routine `ode45` with variable step size and error control. Since in body coordinates the gravity vector $\boldsymbol{\Gamma}_{\rm b}$ rotates very fast, the MATLAB routine employs a very small step size in the integration interval $[0, 10]$, ranging from $h_{\rm min} \approx 0.5 \times 10^{-6}$ to $h_{\rm max} \approx 0.00214$. The average value is $h_{\rm mean} = 0.0018$ and the standard deviation $\sigma = 1.972 \times 10^{-4}$. The motion of the axis is quite similar to the one observed in Figure 11.1. However, when the integration is performed over longer time, the numerical approximation of $\boldsymbol{\lambda}_{\rm s}$ is not longer on the unit sphere and the amplitude of precessions shrinks (see Figure 11.2).

The routine `ode45` does not preserve much of the geometry of the underlying problem, and many of the conserved quantities of the problem (Casimirs and first integrals, see (Marsden and Ratiu 1994)) are not conserved (see Figure 11.3).

To illustrate the advantages of the methods discussed in this article, we solve numerically the equations of the heavy top (11.1)-(11.2) using the *coadjoint action* of SE(3) on the dual of the algebra $\mathfrak{se}(3)^*$ (Marsden and Ratiu 1994),

$$\Lambda\Big((R, \boldsymbol{\Omega}_{\rm b}), (\boldsymbol{\Pi}_{\rm b}, \boldsymbol{\Gamma}_{\rm b})\Big) = (R\,\boldsymbol{\Pi}_{\rm b} + \boldsymbol{\Omega}_{\rm b} \times R\,\boldsymbol{\Gamma}_{\rm b}, R\,\boldsymbol{\Gamma}_{\rm b}).$$

Using numerically the coadjoint action implies that all the Casimirs, in this case the projection of the angular momentum on the gravity axis, $\boldsymbol{\Pi}_{\rm b} \cdot \boldsymbol{\Gamma}_b$, and the norm of the gravity vector $\|\boldsymbol{\Gamma}_{\rm b}\|$, are automatically preserved to machine accuracy (Engø and Faltinsen 1999, Zanna et al. 1999). In Figure 11.4 we plot the processions and nutations in $[0, 100]$ of the above heavy top, numerically solved with the explicit

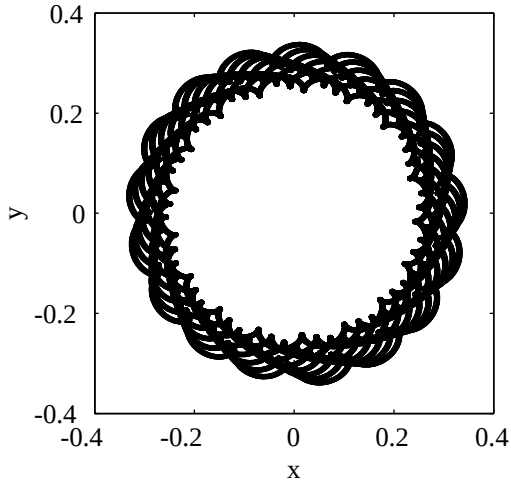


Fig. 11.2. Numerical precession and nutations projected on the (x, y)-plane for $t \in [0, 100]$ of the axis of a heavy top. The equations are solved in body coordinates with the MATLAB routine `ode45`.

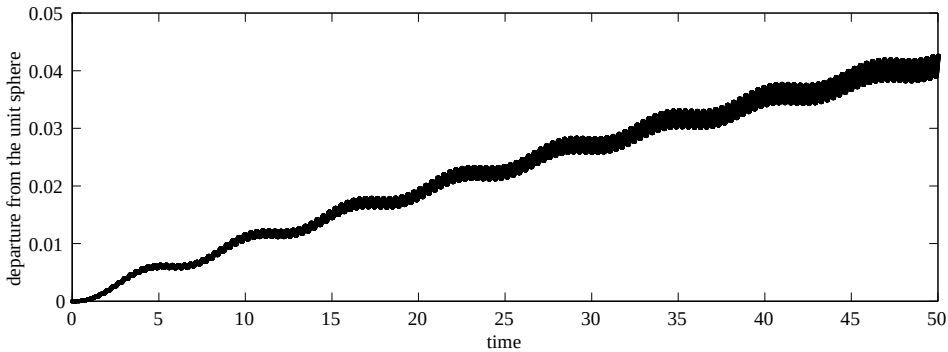


Fig. 11.3. Departure from unity for 'unit vector' along the axis of the top for $t \in [0, 100]$. The equations are solved in body coordinates with the MATLAB routine `ode45`.

fourth-order RK-MK method based on the Butcher tableau

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

(cf. Appendix A), with step size $h = \frac{1}{100}$. We refer to this numerical scheme as `RKMK4`.

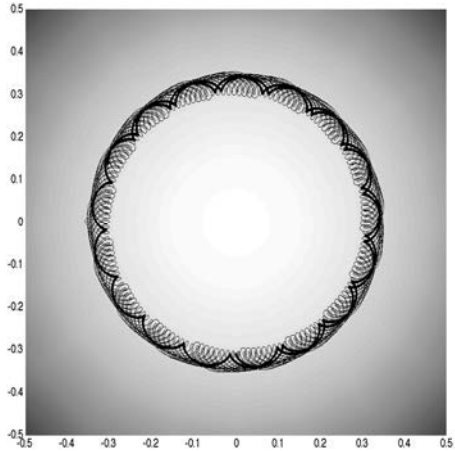


Fig. 11.4. Numerical precession and nutations projected on the (x, y)-plane for $t \in [0, 100]$ of the axis of a heavy top. The equations are solved in body coordinates (employing the coadjoint action) with the Lie-group scheme `RKMK4`.

Comparing Figure 11.4 with Figure 11.2, we observe that the tip of the axis remains on the unit sphere (with five times as large step size as the average one used by `ode45`), as we expected from theory. There is only a very slight variation in the angles θ_1 and θ_2, bounding the nutations. This comes as no surprise: Lie-group type methods perform very well for systems with oscillatory behaviour, as will be further discussed in Section 11.5. Note that this scheme is neither time-reversible nor energy preserving. Using self-adjoint schemes, as those discussed in Section 7, or energy preserving schemes, as those of (Engø and Faltinsen 1999), it is possible to further improve other geometrical features of the numerical solution.

We remark that for this type of actions, solved by Lie-group schemes based on canonical coordinates of the first kind, the integrator requires numerous evaluations of exponentials and dexp^{-1} of 3×3 skew-symmetric matrices, which can be computed exactly and very fast using the formulae described in Appendix B.

11.2. Sturm–Liouville problems

Numerous problems in applied mathematics require the solution of *Sturm–Liouville problems,* finding λ so that

$$\mathcal{L}y = -y'' + q(t)y = \lambda y, \qquad t \in (0, \alpha), \tag{11.3}$$

with the sufficiently smooth *potential* $q \in C^m(0,\alpha)$ and the boundary conditions

$$y(0)\boldsymbol{b}_0 + y(\alpha)\boldsymbol{b}_\alpha = 0, \tag{11.4}$$

where $\operatorname{rank}[\boldsymbol{b}_0, \boldsymbol{b}_\alpha] = 2$. Examples range from fluid flow to Schrödinger spectra to geophysics to NMR imaging and beyond.

It is well known from classical functional analysis that the *spectrum* $\sigma(\mathcal{L})$, i.e. the set of all λ that solve (11.3), is real, countable and accumulates at $+\infty$. Its computation has attracted significant effort since the very dawn of numerical analysis and led to an impressive array of methods and software. We refer the reader to (Pryce 1993) and to references therein.

The problem (11.3) can be always formulated in SL(2),

$$\boldsymbol{y}' = \begin{bmatrix} 0 & 1 \\ q(t)-\lambda & 0 \end{bmatrix} \boldsymbol{y}, \quad t \in (0,\alpha), \qquad \text{where} \qquad \boldsymbol{y} = \begin{bmatrix} y \\ y' \end{bmatrix}, \tag{11.5}$$

and this makes it a 'natural' for Lie-group methods. The main idea is to approximate the fundamental solution of (11.5) by a product of exponentials and impose the boundary conditions (11.4). This technique has been introduced and thoroughly analysed by Moan (1998) and it constitutes a far-fetched generalisation of *Preuss–Fulton methods* (Preuss 1973). The outcome is a scalar nonlinear equation for the spectral parameter λ. For example, suppose that (11.4) simplifies to $y(0) = y(\alpha) = 0$ and our approximation to the fundamental solution of (11.5) is $Y(t) = \operatorname{expm}\begin{bmatrix} a & b \\ c & -a \end{bmatrix}$. It follows from (8.1) that

$$Y(t) = \begin{bmatrix} \cosh\omega + a\frac{\sinh\omega}{\omega} & b\frac{\sinh\omega}{\omega} \\ c\frac{\sinh\omega}{\omega} & \cosh\omega - a\frac{\sinh\omega}{\omega} \end{bmatrix}, \qquad \text{where} \qquad \omega = \sqrt{a^2+bc}.$$

Note that a, b, c, ω are functions of both t and λ. We impose the boundary conditions and the outcome is the equation

$$b(\alpha,\lambda)\frac{\sinh\omega(\alpha,\lambda)}{\omega(\alpha,\lambda)} = 0. \tag{11.6}$$

Suppose for example (and a very trivial example it is!) that Y is the second-order truncation of the Magnus expansion,

$$Y(t) = \operatorname{expm}\begin{bmatrix} 0 & t \\ \int_0^t q(\xi)\mathrm{d}\xi & 0 \end{bmatrix}.$$

Therefore (11.6) becomes

$$\alpha\frac{\sin\sqrt{\alpha^2\lambda - \alpha\int_0^\alpha q(\xi)\mathrm{d}\xi}}{\sqrt{\alpha^2\lambda - \alpha\int_0^\alpha q(\xi)\mathrm{d}\xi}} = 0$$

and the approximate eigenvalues are

$$\lambda_m \approx \frac{m^2\pi^2 + \alpha \int_0^\alpha q(\xi)\mathrm{d}\xi}{\alpha^2}, \qquad m \in \mathbb{N}.$$

Adding the next term to the Magnus expansion yields a somehow better approximation,

$$\lambda_m \approx \frac{m^2\pi^2 + \alpha \int_0^\alpha q(\xi)\mathrm{d}\xi - \frac{1}{4}\left[\int_0^\alpha (2\xi-\alpha)q(\xi)\mathrm{d}\xi\right]^2}{\alpha^2}, \qquad m \in \mathbb{N}.$$

We can continue in this vein, possibly replacing integrals by quadrature, except that the outcome of this naive approach is very poor. Magnus expansions are an excellent means to approximate the fundamental solution *locally* and we can hardly expect a single exponential to approximate the solution well in the entire interval $(0, \alpha)$. The situation is further exacerbated when α is large or when (as is the case in many instances of practical interest) we desire to approximate a large number of eigenvalues. It is possible to show that ω is a polynomial in λ, of a degree that grows with the order of the Magnus expansion. Bearing in mind Theorem 4.1, we can thus hardly expect convergence for large λ. A superior alternative is to partition the interval into small subintervals, where convergence and adequate precision can be assured. The details of this procedure are reported in (Moan 1998).

Let us restrict the discussion for the sake of simplicity to a fourth-order method. Letting

$$q_{n,1} = q((n + \tfrac{1}{2} - \tfrac{\sqrt{3}}{6})h), \qquad q_{n,2} = q((n + \tfrac{1}{2} + \tfrac{\sqrt{3}}{6})h),$$

where $h = \alpha/n^*$, we set

$$\Theta_n(\lambda) = \begin{bmatrix} -\frac{\sqrt{3}}{12}h^2(q_{n,1} - q_{n,2}) & h \\ \frac{1}{2}h(q_{n,1} + q_{n,2}) - h\lambda & \frac{\sqrt{3}}{12}h^2(q_{n,1} - q_{n,2}) \end{bmatrix}, \qquad n = 0, 1, \ldots, n^* - 1.$$

The fourth-order approximant to the solution of (11.5) at α is $Y^\lambda \boldsymbol{y}(0)$, where

$$Y^\lambda = \mathrm{e}^{\Theta_{n^*-1}(\lambda)} \cdots \mathrm{e}^{\Theta_1(\lambda)} \mathrm{e}^{\Theta_0(\lambda)}.$$

To force the boundary conditions $y(0) = y(\alpha) = 0$, say, we seek λ so that $Y^\lambda_{1,2} = 0$. This nonlinear equation can be solved by *Newton–Raphson iteration* which, eigenvalues being simple, converges quadratically near the solution. Moan (1998) recommends a procedure based on the Newton–Raphson method being applied to an augmented equation, thereby simplifying the computation of the derivative.

The outcome is a very efficient method for the computation of Sturm–Liouville problems and it can be easily generalised to more elaborate eigenvalue problems and boundary conditions. However, if the interval in question is long or a large number of eigenvalues is desired, efficiency suffers. This is in particular the case if

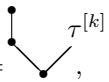
high-order Magnus methods are used, since the degree of elements of Θ_n as polynomials in λ grows. As a matter of fact, the error in a pth-order method grows as $\mathcal{O}\big(h^{p+1}\lambda^{p/2-1}\big)$. To overcome this phenomenon, which occurs also in other methods for the computation of Sturm–Liouville problems, Moan (1998) introduced an interesting device which might be relevant to other applications of Lie-group methods. We observe first that, given matrix values in a self-adjoint basis $B_0, B_1, \ldots, B_{\nu-1}$ (cf. Section 5), it follows at once from (5.7) that only B_0 depends upon λ.

Rather than presenting the construction from (Moan 1998), we introduce a conceptually similar idea of Iserles (n.d.). Our point of departure is a decomposition of the Magnus expansion (4.5) into *streamers:* partial sums of the form

$$\mathcal{H}_{\tau^{[0]}}(t) = \sum_{k=0}^{\infty} \alpha(\tau^{[k]}) \int_0^t \mathcal{C}_{\tau^{[k]}}(\xi)\mathrm{d}\xi,$$

where

$$\tau^{[k+1]} = \quad , \qquad k \in \mathbb{Z}^+. \tag{11.7}$$



Let $\tau = \tau^{[0]}$ be in the form (4.8). It is possible to show that

$$\alpha(\tau^{[k]}) = \frac{\mathrm{B}_{s+k}}{(s+k)!}\hat{\alpha}(\tau), \qquad \text{where} \qquad \hat{\alpha}(\tau) = \prod_{i=1}^{s} \alpha(\tau_i)$$

(cf. (4.9)). Therefore we can write the streamer as

$$\mathcal{H}_{\tau^{[0]}}(t) = \hat{\alpha}(\tau^{[0]}) \int_0^t \sum_{k=0}^{\infty} \frac{\mathrm{B}_{s+k}}{(s+k)!} \mathrm{ad}^k_{\int_0^\xi A(\eta)\mathrm{d}\eta} \mathcal{C}_{\tau^{[0]}}(\xi)\mathrm{d}\xi. \tag{11.8}$$

We say that a tree is *primitive* if it cannot be written in the form (11.7). The set of primitive trees in $\mathbb{F}_m$ is denoted by $\mathbb{F}^{\mathrm{p}}_m$, whereby we might replace the truncated Magnus expansion (4.14) with the *Magnus streamer expansion*

$$\Xi_p(t) = \sum_{m=0}^{p-1} \sum_{\tau\in\mathbb{F}^{\mathrm{p}}} \mathcal{H}_\tau(t). \tag{11.9}$$

Like (4.14), this is a pth-order approximant, except that each tree therein is accompanied by an infinitely-long streamer and there is one less level of truncation in the Magnus expansion (4.5).

Using (11.9) makes sense only if there exists a good method to evaluate the streamer (11.8) without truncating the expansion. This is the case if the Lie algebra is of sufficiently small dimension.

Let $\dim \mathfrak{g} = d$ and let $\boldsymbol{C} = \{C_1, C_2, \ldots, C_d\}$ be a basis of the algebra. Then

there exists a natural map $\boldsymbol{\pi} : \mathfrak{g} \to \mathbb{R}^d$ defined by

$$\boldsymbol{\pi}(B) = \boldsymbol{\theta} \qquad \text{where} \qquad B = \sum_{k=1}^{d} \theta_k C_k.$$

The action of the ad operator is a linear transformation, thus $\boldsymbol{\pi}(\mathrm{ad}_D B) = \mathscr{C}_D \boldsymbol{\pi}(B)$, where $\mathscr{C}_D \in \mathfrak{gl}(d)$ is a *commutator matrix*.[6] It follows from (11.8) and the definition of Bernoulli numbers that

$$\begin{aligned}
\boldsymbol{\pi}(\mathcal{H}_\tau(t)) \;&=\; \hat{\alpha}(\tau) \int_0^t \sum_{k=s}^{\infty} \frac{\mathrm{B}_k}{k!} \mathscr{C}_{D(\xi)}^{k-s} \boldsymbol{\pi}(\mathcal{C}_\tau(\xi)) \mathrm{d}\xi \\
&=\; \hat{\alpha}(\tau) \int_0^t \mathscr{C}_{D(\xi)}^{-s} \left\{ \frac{\mathscr{C}_{D(\xi)}}{\mathrm{e}^{\mathscr{C}_{D(\xi)}} - \boldsymbol{I}} - \sum_{k=0}^{s-1} \frac{\mathrm{B}_k}{k!} \mathscr{C}_{D(\xi)}^{k} \right\} \boldsymbol{\pi}(\mathcal{C}_\tau(\xi)) \mathrm{d}\xi.
\end{aligned}$$

where $D(t) = \int_0^t A(\eta)\mathrm{d}\eta$. Provided that d is small, typically we can obtain the commutator matrix and its exponential explicitly. This, together with quadrature, leads to an explicit formula for the calculation of the streamer that does not require any truncation of the series.

Similar idea, reported in (Moan 1998), ameliorates the deterioration in accuracy for large λ. For example, the error of sixth-order Magnus streamer is $\mathcal{O}\big(h^7\lambda\big)$, hence it behaves like a fourth-order method when $h^2\lambda \approx 1$, when the standard sixth-order Magnus, which carries an error of $\mathcal{O}\big(h^7\lambda^2\big)$, reduces to an order-two method.

11.3. Charged particles in a magnetic field

The motion of charged particles moving in a magnetic field was first studied numerically by Carl Størmer (1907) as a part of his work on explaining the origin of northern light (aurora borealis).

A particle (of unit mass and unit charge) is moving in a magnetic field $\boldsymbol{b}$ according to the equations

$$\begin{aligned}
\boldsymbol{y}'(t) \;&=\; \boldsymbol{v} \\
\boldsymbol{v}'(t) \;&=\; \boldsymbol{b}(t, \boldsymbol{y}) \times \boldsymbol{v}
\end{aligned}$$

where $\boldsymbol{v}$ is the velocity, $\boldsymbol{y}$ is the position and $\boldsymbol{b}$ is the magnetic field. We attempt to obtain a simpler system by assuming that $\boldsymbol{b} = \mathrm{const}$, neglecting the dependence of $\boldsymbol{b}$ on space and time. Written in a matrix form, this yields

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{v} \end{bmatrix}' = \begin{bmatrix} 0 & \boldsymbol{I} \\ 0 & \widehat{\boldsymbol{b}} \end{bmatrix} \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{v} \end{bmatrix}, \tag{11.10}$$

[6] Note that the usual definition of a commutator matrix is as an object in $\mathfrak{gl}(N^2)$ for $\mathfrak{g} \subseteq \mathfrak{gl}(N)$.

where $\widehat{\boldsymbol{b}}$ is a 3×3 skew symmetric matrix given by the hat map ($B.1$) in Appendix B. We compute the bracket of the vector fields in (11.10). Since they are linear, we find that

$$\left[\begin{bmatrix} 0 & \boldsymbol{I} \\ 0 & \widehat{\boldsymbol{b}} \end{bmatrix}, \begin{bmatrix} 0 & \boldsymbol{I} \\ 0 & \widehat{\boldsymbol{d}} \end{bmatrix}\right] = \begin{bmatrix} 0 & \widehat{\boldsymbol{d}} - \widehat{\boldsymbol{b}} \\ 0 & \widehat{\boldsymbol{b}\times\boldsymbol{d}} \end{bmatrix}.$$

Hence, the vector fields do *not* form a Lie algebra. The remedy is to enlarge the family of 'simple' equations into

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{v} \end{bmatrix}' = \begin{bmatrix} 0 & B \\ 0 & \widehat{\boldsymbol{b}} \end{bmatrix} \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{v} \end{bmatrix} = \begin{bmatrix} B\boldsymbol{v} \\ \boldsymbol{b}\times\boldsymbol{v} \end{bmatrix}, \tag{11.11}$$

where B is a general 3×3 matrix. This results in the commutator

$$\left[\begin{bmatrix} 0 & B \\ 0 & \widehat{\boldsymbol{b}} \end{bmatrix}, \begin{bmatrix} 0 & C \\ 0 & \widehat{\boldsymbol{c}} \end{bmatrix}\right] = \begin{bmatrix} 0 & B\widehat{\boldsymbol{c}} - C\widehat{\boldsymbol{b}} \\ 0 & \widehat{\boldsymbol{b}\times\boldsymbol{c}} \end{bmatrix}$$

It is worth remarking on the similarity between this bracket and the brackets in Example 2.8, Example 2.10 and in Section 11.1. These are all special instances of so-called *semidirect product* brackets. Semidirect products are one of the most fundamental ways of constructing Lie algebras and Lie groups from simpler algebras and groups.

The solution of the simplified equation (11.11) is given in terms of initial conditions $\boldsymbol{y}_0$ and $\boldsymbol{v}_0$ as

$$\begin{bmatrix} \boldsymbol{y}(t) \\ \boldsymbol{v}(t) \end{bmatrix} = \operatorname{expm} \begin{bmatrix} 0 & tB \\ 0 & t\widehat{\boldsymbol{b}} \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_0 \\ \boldsymbol{v}_0 \end{bmatrix}.$$

Repeated matrix multiplications yield

$$\operatorname{expm} \begin{bmatrix} 0 & B \\ 0 & \widehat{\boldsymbol{b}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & B\widehat{\boldsymbol{b}}^{-1}(\operatorname{expm}(\widehat{\boldsymbol{b}}) - \boldsymbol{I}) \\ 0 & \operatorname{expm}(\widehat{\boldsymbol{b}}) \end{bmatrix},$$

a formula that can be computed exactly and efficiently using ($B.6$) and ($B.10$).

In a numerical experiment we have taken $\boldsymbol{b}(t, \boldsymbol{y}) = \boldsymbol{b}(\boldsymbol{y})$ as the *magnetic dipole field*

$$\boldsymbol{b}(\boldsymbol{y}) = \frac{3\boldsymbol{e}_{\boldsymbol{y}}^{\mathrm{T}}\boldsymbol{m}\boldsymbol{e}_{\boldsymbol{y}} - \boldsymbol{m}}{\|\boldsymbol{y}\|^3},$$

where $\boldsymbol{m} = [0, 0, 1]^{\mathrm{T}}$ is the magnetic dipole vector. Figure 11.5 displays the orbit of a single particle with initial conditions $\boldsymbol{y}_0 = [0, -\frac{5}{2}, 0]^{\mathrm{T}}$, $\boldsymbol{v}_0 = 12 \cdot 10^{-3}[0, 0, 1]^{\mathrm{T}}$. The time interval is $[0, 10000]$. The particle is moving back and forth in the *van Allen belt* around the Earth. This system is Hamiltonian, the Hamiltonian energy being $H = \|\boldsymbol{v}\|^2$. The preservation of H follows also trivially from the observation that the acceleration is always orthogonal to $\boldsymbol{v}$.

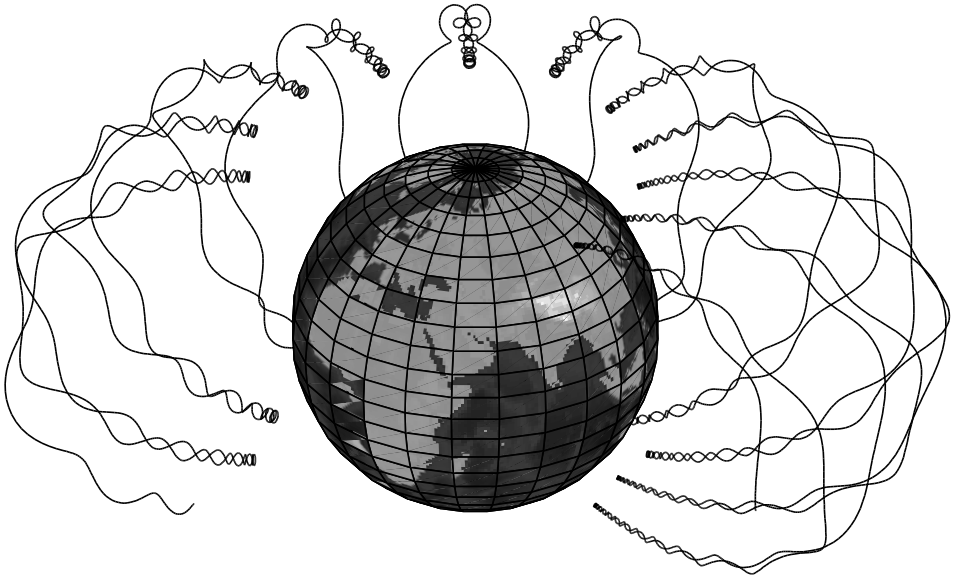


Fig. 11.5. Motion of a charged particle in the Earth's magnetic field.

We wish to compare step sizes in classical integrators and RK-MK-type methods for this problem. The classical integrator used in the study is the `ode45` code of MATLAB, based on DOPRI5(4), an embedded (4,5) pair of Runge–Kutta methods due to Dormand and Prince. The code uses variable step size control. The RK-MK method we use is also based on the DOPRI5(4) method and we have employed the step-size controller of *DiffMan*. The interval of integration ranged from 0 to 500, corresponding to a motion of the particle from the equator up towards the north pole, bouncing back once towards the equator. We varied the tolerance of the step-size controllers and measured the relative error of the answer at the end-point. For each simulation we have counted the number of steps taken. Table 11.1

Table 11.1. *Global error and the number of steps for classical RK and a Lie-group integrator.*

Classical RK		RK-MK	
Error	Number of steps	Error	Number of steps
$7 \cdot 10^{-3}$	836	$5 \cdot 10^{-3}$	104
$2 \cdot 10^{-4}$	2132	$4 \cdot 10^{-4}$	142
$2 \cdot 10^{-6}$	5356	$1 \cdot 10^{-6}$	353

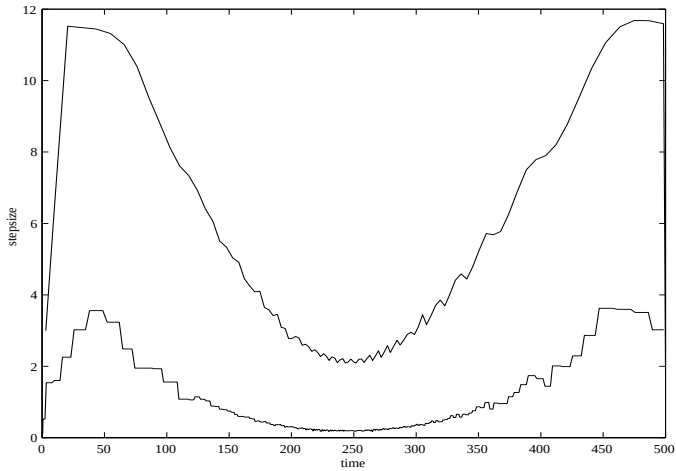


Fig. 11.6. Time-step selection for RK-MK (upper) and classical RK (lower).

summarises the results. We see that the Lie-group method is much more accurate than the classical integrator for the same step size. Figure 11.6 displays the time step selection as a function of time for the two methods for the simulations given in the first line of Table 11.1. The dip in the middle corresponds to the part of the trajectory where the particle is bouncing back. Both methods must reduce the step size in this region, but the classical one does so relatively more than the Lie group method. Note that, whereas the Lie group integrator preserves $\|\boldsymbol{v}\|$ exactly, the `ode45` integrator does so only up to the the order of the method.

We conclude that for this problem, where both commutators and exponentials can be computed quickly, the total cost of a Lie-group integrator is significantly less than that of a classical integrator. The time interval of integration is relatively small in this example: the importance of preserving geometric properties becomes increasingly more crucial for very long integration intervals.

11.4. Toda-lattice equations

Imagine a regular lattice of N particles, all of unit mass, and assume that each particle interacts with just its nearest neighbours, subject to an exponential interaction potential. The outcome is the *Toda lattice,* governed by the Hamiltonian equations of motion

$$\left.\begin{aligned} \frac{\mathrm{d}q_k}{\mathrm{d}t} &= p_k, \\ \frac{\mathrm{d}p_k}{\mathrm{d}t} &= \mathrm{e}^{q_{k-1}-q_k} - \mathrm{e}^{q_k-q_{k+1}}, \end{aligned}\right\} \qquad k = 1, 2, \ldots, N, \tag{11.12}$$

where q_k and p_k are the generalised coordinates and momenta, respectively (Toda 1981), where we have let $q_0 = -\infty$, $q_{N+1} = \infty$ in the linear case and $q_{N+k} = q_k$ when the particles are arranged in a ring (cf. Figure 11.7). Equation (11.12) corresponds to the Hamiltonian potential

$$H(\boldsymbol{p}, \boldsymbol{q}) = \tfrac{1}{2} \sum_{k=1}^{N} p_k^2 + \sum_{k=1}^{N} (\mathrm{e}^{q_k - q_{k+1}} - 1).$$

It is a special case of a *Fermi–Pasta–Ulam (FPU) flow.*

a. Linear configuration b. Ring configuration

Fig. 11.7. Different Toda-lattice configurations with $N = 5$.

Letting $\alpha_k = \frac{1}{2}\mathrm{e}^{(q_k - q_{k+1})/2}$, $\beta_k = \frac{1}{2} p_k$, Flaschka (1974) showed that (11.12) can be recast in the *Lax form*

$$Y' = [B(Y), Y], \quad t \geq 0, \qquad Y(0) = Y_0, \tag{11.13}$$

where

$$Y = \begin{bmatrix} \beta_1 & \alpha_1 & 0 & \cdots & \alpha_N \\ \alpha_1 & \beta_2 & \alpha_2 & \ddots & \vdots \\ 0 & \alpha_2 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \beta_{N-1} & \alpha_{N-1} \\ \alpha_N & \cdots & 0 & \alpha_{N-1} & \beta_N \end{bmatrix},$$

with $\alpha_N = 0$ for linear lattice configuration, and $B(Y) = Y_+ - Y_- \in \mathfrak{so}(N)$, where M_+ and M_- denote the upper-triangular and the lower-triangular portions of the matrix M, respectively. We immediately identify (11.13) with the *isospectral flow* (1.1).

A special case of (11.12) that has elicited much interest is three particles in a ring. In that case, the linear transformation

$$\tilde{\boldsymbol{p}} = T\boldsymbol{p}, \quad \tilde{\boldsymbol{q}} = T\boldsymbol{q} \qquad \text{where} \qquad T = \begin{bmatrix} \frac{\sqrt{6}}{6} & -\frac{\sqrt{6}}{3} & \frac{\sqrt{6}}{6} \\ \frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{3}}{3} & \frac{\sqrt{3}}{3} & \frac{\sqrt{3}}{3} \end{bmatrix},$$

in tandem with rescaling and elimination of one of the variables by employing the

linear conservation law $p_1 + p_2 + p_3 \equiv \text{const}$, results in a simplified two-dimensional Hamiltonian function

$$\bar{H}(\bar{\boldsymbol{p}}, \bar{\boldsymbol{q}}) = \tfrac{1}{2}(\bar{p}_1^2 + \bar{p}_2^2) + \tfrac{1}{24}(\mathrm{e}^{2\bar{q}_2 + 2\sqrt{3}\bar{q}_1} + \mathrm{e}^{2\bar{q}_2 - 2\sqrt{3}\bar{q}_1} + \mathrm{e}^{-4\bar{q}_2}) - \tfrac{1}{8} \tag{11.14}$$

in the new variables $\bar{\boldsymbol{p}}, \bar{\boldsymbol{q}} \in \mathbb{R}^2$ (Zanna 1998).[7] The solution now evolves on a compact submanifold of $\mathbb{R}^4$, more specifically on a 2-torus. Therefore, a Poincaré section consists of two closed curves and a good numerical scheme should retain this important property.

The following calculation has been performed in (Zanna 1998) for a variety of methods which follow a set pattern: First we transform the three-particle lattice with the initial conditions $\boldsymbol{p} = [1, 1, 0]^{\mathrm{T}}$, $\boldsymbol{q} = \boldsymbol{0}$, to the Lax form (11.13). We then solve the isospectral form with the constant step size $h = \frac{1}{10}$ for 10^4 steps, transform the variables to $(\bar{\boldsymbol{p}}, \bar{\boldsymbol{q}})$ and sketch the Poincaré section. The following methods have been used:

1 The three-stage, third-order explicit Runge–Kutta method with the Butcher tableau

$$\begin{array}{c|ccc} 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ 1 & -1 & 2 & \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}, \tag{11.15}$$

applied directly to (11.13). Note that this method is not isospectral. The Poincaré section is displayed in Figure 11.8a and we note at once that, instead of periodic trajectories, the motion collapses rapidly to a spurious fixed point.

2 The Runge–Kutta method (11.15) applied to the *orthogonal flow*

$$Q' = B(QY_nQ^{\mathrm{T}})Q, \quad t \geq nh, \qquad Q(nh) = \boldsymbol{I}, \tag{11.16}$$

translating back to (11.13) with $Y_{n+1} = Q_{n+1}Y_nQ_{n+1}^{\mathrm{T}}$. Note that the numerical approximation $Q_{n+1} \approx Q((n+1)h)$, produced by the method (11.15), does not evolve on SO(3). Figure 11.8b demonstrates that the solution again spirals to a fixed point, although less rapidly than in the former case.

3 The former method can be 'orthogonalised' by *projection,* subjecting Q_{n+1} to polar decomposition and discarding the nonorthogonal part. This can be further enhanced by subjecting the intermediate stages to polar decompositions. Relevant Poincaré sections are displayed in Figure 11.9. Evidently, the behaviour improves, yet the solution goes on spiralling to a spurious fixed point.

[7] One reason why this case is interesting is that truncation of $\bar{H}$ to cubic terms results in the famous *Hénon–Heiles Hamiltonian,* which is known to be nonintegrable (Berry 1987, Toda 1981).

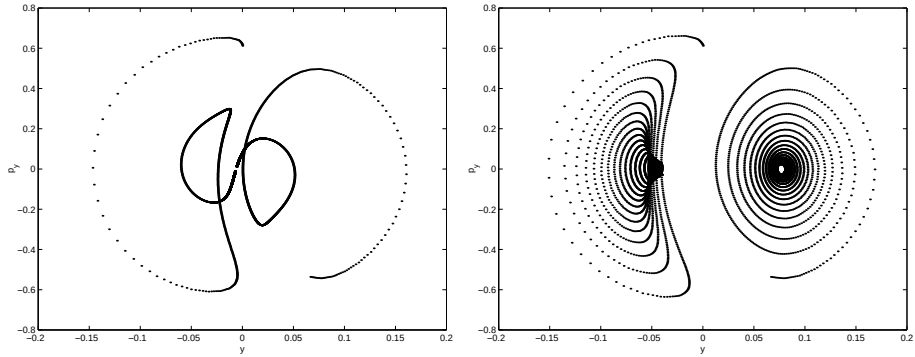


Fig. 11.8. Three-particle lattice solved by the Runge–Kutta method (11.15), applied a. directly to the isospectral flow (11.13) (on the left); and b. to the orthogonal flow (11.16) (on the right).

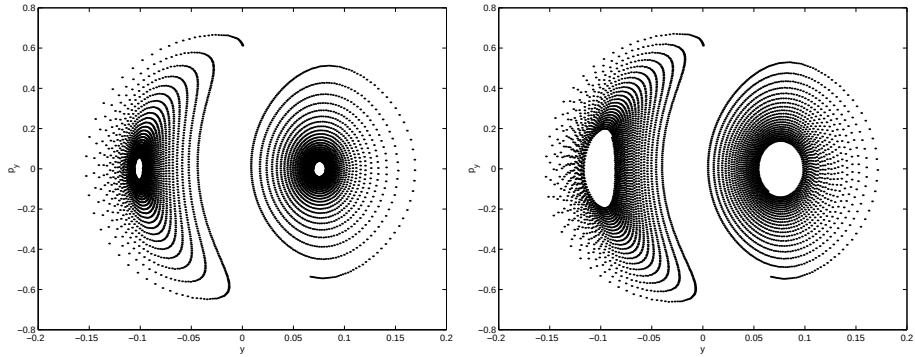


Fig. 11.9. Three-particle lattice solved by the Runge–Kutta method (11.15), applied to the orthogonal flow (11.16) and with polar decomposition. In the plot on the left only the final result is projected, while on the right both the final result and intermediate stages are subjected to this procedure.

4 We solve (11.16) with the RK-MK method, using the Runge–Kutta scheme (11.15). The Poincaré section for this computation, the first with a 'proper' Lie-group method, is displayed in Figure 11.10a. We have not managed to get rid of the unwelcome spurious convergence to a fixed point, yet the trajectory spirals significantly slower.
5 Finally, (11.16) is solved with the fourth-order collocated Magnus method from Section A.3, applied to the orthogonal flow. The outcome is displayed in

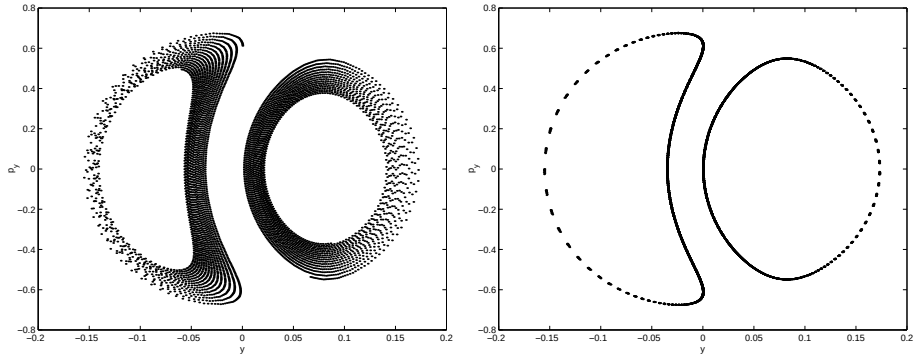


Fig. 11.10. Three-particle lattice solved by Lie-group methods: a. The RK-MK method with the tableau (11.15); and b. fourth-order Magnus collocation from Section A.3.

Figure 11.10b: the Poincaré section consists of the qualitatively-correct two closed curves, a section across a 2-torus!

Figures 11.8–10 display consistent gradual improvement. This is not evident at all from the retention of the Lie-group *per se,* since the qualitative feature under examination, invariant tori, is of a different flavour. Yet, it is clear that conservation of Lie-group structure is advantageous and that purpose-designed Lie-group solvers hold the edge over projection methods.

Note that invariant tori would have been retained by a symplectic method, applied directly to the Hamiltonian equations induced by the potential (11.14). This is hardly a surprise: we have, after all, subjected methods applied in a Lie-group formalism to a 'Hamiltonian test'. It is possible to devise a test according to 'isospectral' ground rules, e.g. by monitoring global error in Lax-equation coordinates. This has been done in (Calvo, Iserles and Zanna 1999), demonstrating a marked advantage of isospectral methods (originating in Lie-group solvers) over symplectic algorithms.

11.5. Highly-oscillatory equations

Highly-oscillatory systems of ODEs feature in many applications and their computation currently absorbs much of the overall effort devoted to numerical analysis of ODEs. It is well known that classical solvers performs poorly, and this has motivated many novel and ingeneous techniques (Petzold, Jay and Jeng 1997).

Past experience indicates that Lie-group methods might be a suitable means to solve highly-oscillatory systems (Iserles and Nørsett 1999, Iserles et al. 1999). We commence from an example that has already featured in (Iserles and Nørsett 1999),

the solution of the *Airy equation*

$$y'' + ty = 0, \quad t \geq 0, \qquad y(0) = 1, \quad y'(0) = 1. \tag{11.17}$$

The exact solution of (11.17) can be represented as a linear combination of *Airy functions*

$$y(t) = \tfrac{1}{2}\Gamma(\tfrac{2}{3})[3^{2/3}\mathrm{Ai}(-t) + 3^{1/6}\mathrm{Bi}(-t)], \qquad t \geq 0.$$

It is easy to prove, e.g. by the WKB technique, that the trajectory is a bounded function that oscillates like $\sin t^{3/2}$: the frequency increases with time. (Cf. Fig. 11.11.) This indicates that long-time integration is difficult and this can be confirmed by endeavouring to solve (11.17) with any popular, general-purpose solver. We have solved the Airy equation with two MATLAB routines, `ode113` and `ode15s`, the first for nonstiff and the latter for stiff problems. Both routines employ variable-order methods in tandem with sophisticated error control. Yet, although we have attempted a wide range of possible error tolerances, including the least that the software would accept (relative tolerance of 2.2×10^{-14}), the pointwise error at $t = 100$ consistently exceeded 9.42×10^{-6} and 5.43×10^{-4} respectively: very poor performance, in particular in the case of the stiff solver `ode15s`.[8]

We have solved the Airy equation (11.17) with Magnus methods of orders 2, 4 and 6 respectively, taken from $(A.9)$, with the constant step-size sequence $h = \frac{1}{10}, \frac{1}{20}, \frac{1}{40}, \frac{1}{80}$. No attempt has been made to monitor the error or to optimize the solution. In particular, no advantage has been taken of the fact that (11.17) has been rendered as an SL(2) equation and, rather than using (8.1), we have computed the exponential with the MATLAB function `expm`. Figure 11.12 displays the outcome

[8] Highly-oscillating ODEs should be never confused with stiff ODEs. Our calculation merely confirms this.

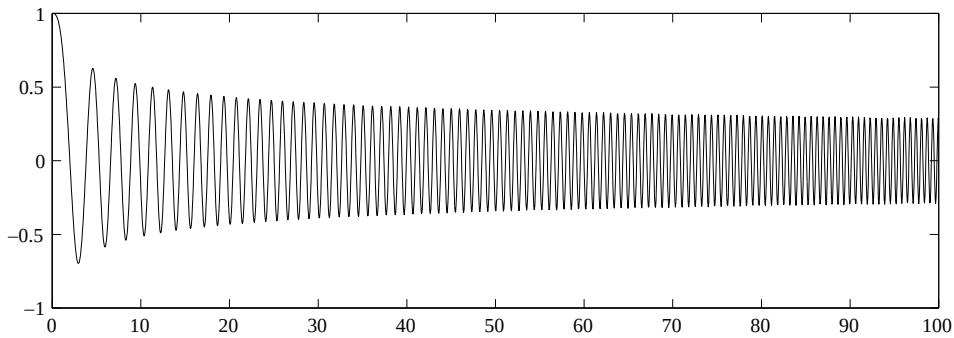


Fig. 11.11. The Airy equation (11.17).

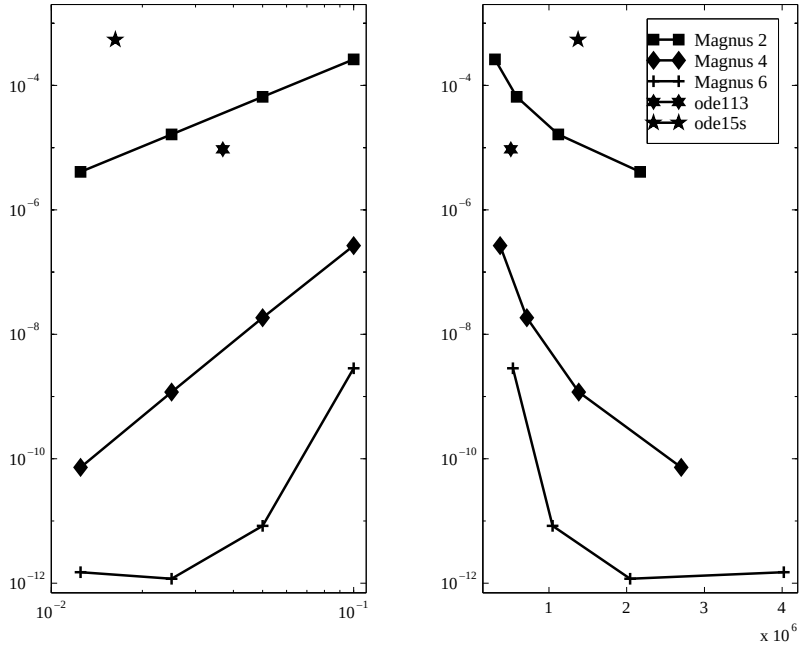


Fig. 11.12. The solution $y(t)$ of the Airy equation (11.17) at $t = 100$.

of our calculations. The plot on the left compares the absolute error at $t = 100$ for different values of h on a doubly-logarithmic scale. For comparison purposes we have included the optimal results for `ode113` and `ode15s`, assigning to them h equal to the average step size. The plot on the right compares the logarithm of the absolute error with the number of flops expanded on the solution: a very imprecise measure, yet a fair reflection of the computational effort.

The main observation is that, unlike the MATLAB routines, Magnus methods perform very well indeed. The deterioration in the accuracy of the sixth-order method for small step size can be explained by the accumulation of small errors in the evaluation of the `expm` function. In principle, we could have replaced it by the exact formula (8.1), except that, while nominally reducing the number of flops, this does not enhance the solution because of inexact computation of hyperbolic functions. In all likelihood, the behaviour in Figure 11.12 is as good as one can expect with IEEE computer arithmetic.

Another example of a highly-oscillatory ODE is

$$t^2y'' - ty' + (1 + t^2)y = 0, \quad t \geq 1, \quad y(1) = J_0(1), \quad y'(1) = J_0(1) - J_1(1), \quad (11.18)$$

whose solution is $y(t) = tJ_0(t)$. Here J_ν is a *Bessel function of a first kind,* whose

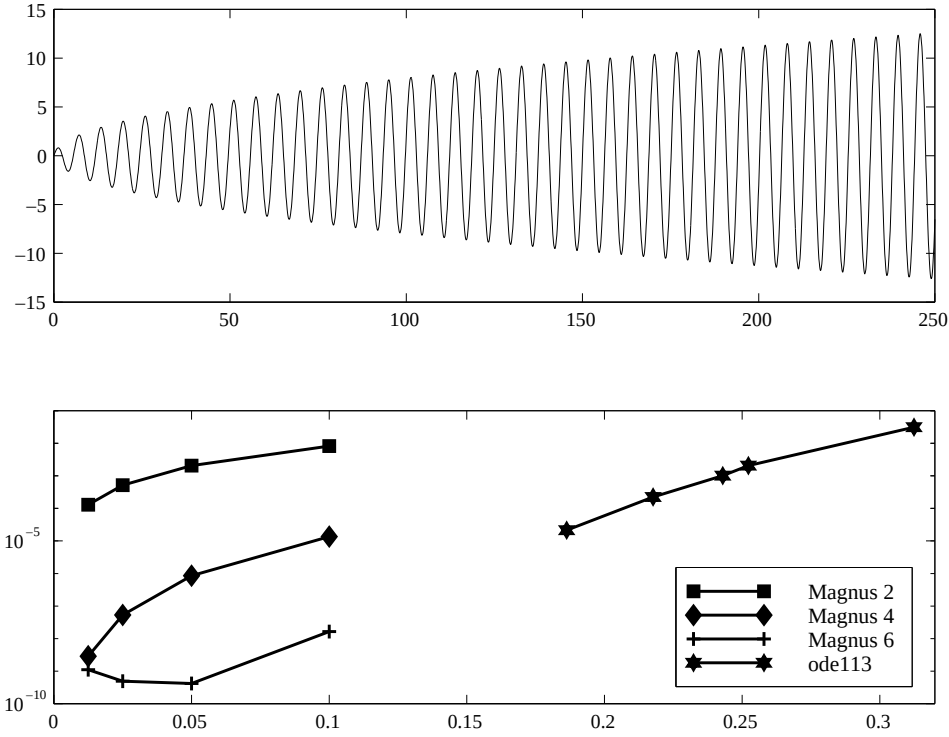


Fig. 11.13. The solution of the Bessel-like equation (11.18).

oscillatory behaviour is well known. Note that we commence integration at $t = 1$, to avoid a regular singularity at the origin. The top plot of Figure 11.13 depicts the exact solution of (11.18). The oscillatory behaviour is evident: as a matter of fact, it is elementary to prove that the solution behaves like $\sqrt{2t/\pi}\sin t$ for $t \gg 1$. The bottom plot displays in a logarithtmic scale the maximal error in the interval $[0, 250]$ in second-order, fourth-order and sixth-order Magnus methods for a range of different step sizes. For comparison purposes, we have also included the same data for the MATLAB routine `ode113` for different tolerances, down to the least permitted by the software. As in the case of Fig. 11.12, we have assigned to each such computation an average step size. Note that the step sizes are significantly larger for MATLAB routines, but so is the error! Even though it uses error control, `ode113` appears incapable of driving pointwise accuracy below quite a large treshold. We note in passing that the constant-step-size Magnus methods were (for the largest step size, $h = \frac{1}{10}$) 244%, 317% and 476% more expensive than `ode113`, but no

attempt has been made to optimize the calculations. As in the case of the Airy equation (11.17), the sixth-order method 'hits the buffers' for small step sizes and it appears that there exists a limit on attainable accuracy, probably caused by accumulation of roundoff error. Yet, this limit is twice the number of significant digints attainable by the general-purpose `ode113`.

The above comparison might appear as unfair to `ode113` which, after all, has not been constructed with highly-oscillatory equations in mind. Such criticism misses the point altogether. The whole purpose of geometric integration is to take advantage of structure, rather than employing general-purpose methods! Moreover, our Magnus expansions have been implemented with constant step sizes. There is little doubt (and much evidence in (Iserles et al. 1999)) that variable-step implementation of Magnus methods, e.g. with the technique from Section 10.1, would have improved the odds drastically in their favour. Of course, there are other bespoke methods for highly-oscillatory systems (Petzold et al. 1997). It is not our claim that Lie-group methods are superior, since this has never been investigated comprehensively. Our aim is more modest, namely to argue that such an investigation might be very interesting.

What is the explanation for the remarkable performance of Magnus methods (and other Lie-group methods that we have not implemented in this section) for a highly-oscillatory problem? Although this issue constitutes an interesting open research problem, our suspicion is that the conservation of SL(2) structure has absolutely nothing to do with it. Classical numerical methods invariably employ the *ansatz* that locally the solution of an ODE behaves like a *polynomial* in t. Unfortunately, polynomials are a very poor means to approximate rapidly-varying and highly-oscillating functions. We either require minute step size or a very high degree of a polynomial – and the latter might lead to ill conditioning. All the Lie-group methods of this paper, however, are based on an entirely different representation of the solution, as an *exponential of a matrix with polynomial entries.* Unlike polynomials, exponentials of matrices with polynomial entries can easily model exponential change, high oscillation and changes in amplitude and frequency. This might well be the mechanism explaining the superior performance of Lie-group methods.

Acknowledgements

The authors wish to thank many of their colleagues and fellow *dex*perts who have read portions of this survey as it was taking shape, in particular Krister Åhlander, Sergio Blanes, Elena Celledoni, Kenth Engø, Stig Faltinsen, Arne Marthinsen, Per Christian Moan, Ranjiva Munasinghe, Brynjulf Owren and Reinout Quispel. Their comments and perceptive remarks have contributed to diminish the number of mistakes, typos and infelicities and to improve the cohesion and the presentation of this paper.

HMK, SPN and AZ wish to acknowledge the support of Norwegian Research

Council through the SYNODE II project (127582/410), while AI gratefully acknowledges the support of the Lars Onsager Fund.

REFERENCES

R. Abraham and J. E. Marsden (1978), *Foundations of Mechanics*, 2nd edn, Addison-Wesley.

M. Abramowitz and I. A. Stegun (1970), *Handbook of Mathematical Functions*, Dover, New York.

V. I. Arnold (1989), *Mathematical Methods of Classical Mechanics*, Vol. 60 of *GTM*, 2nd edn, Springer-Verlag.

G. A. Baker (1975), *Essentials of Padé Approximants*, Academic Press, New York.

G. Benettin and A. Giorgilli (1994), 'On the Hamiltonian interpolation of near-to-the-identity symplectic mappings with application to symplectic integration algorithm', *J. Stat. Phys.* **74**, 1117–1143.

M. V. Berry (1987), Regular and irregular motion, in *Hamiltonian Dynamical Systems* (R. MacKay and J. Meiss, eds), Adam Hilger, Bristol, pp. 27–53.

S. Blanes, F. Casas and J. Ros (1999), Improved high order integrators based on Magnus expansion, Technical Report NA1999/08, DAMTP, University of Cambridge.

S. Blanes, F. Casas, J. A. Oteo and J. Ros (1998), 'Magnus and Fer expansions for matrix differential equations: The convergence problem', *J. Phys A* **31**, 259–268.

N. Bourbaki (1975), *Lie Groups and Lie Algebras*, Addison-Wesley, Reading, MA.

J. C. Butcher (1963), 'Coefficients for the study of Runge–Kutta integration processes', *J. Austral. Math. Soc.* **3**, 185–201.

M. P. Calvo, A. Iserles and A. Zanna (1997), 'Numerical solution of isospectral flows', *Maths Comput.* **66**, 1461–1486.

M. P. Calvo, A. Iserles and A. Zanna (1999), 'Conservative methods for the Toda lattice equations', *IMA J. Num. Anal.* **19**, 509–523.

E. Celledoni and A. Iserles (1998), Approximating the exponential from a Lie algebra to a Lie group, Technical Report 1998/NA3, DAMTP, University of Cambridge. To appear in *Maths Comput.*

E. Celledoni and A. Iserles (1999), Methods for the approximation of the matrix exponential in a Lie-algebraic setting, Technical Report 1999/NA3, DAMTP, University of Cambridge. To appear in *IMA J. Num. Anal.*

M. T. Chu (1993), On a differential equation $\frac{dX}{dt} = [X, k(X)]$ where k is a Toeplitz annihilator, Technical report, North Carolina State Univ.

M. T. Chu (1998), 'Inverse eigenvalue problems', *SIAM Rev.* **40**, 1–39.

R. Cools (1997), 'Constructing cubature formulas: the science behind the art', *Acta Numerica* **6**, 1–54.

G. Cooper (1987), 'Stability of Runge–Kutta methods for trajectory problems', *IMA J. Num. Anal.* **7**, 1–13.

P. E. Crouch and R. Grossman (1993), 'Numerical integration of ordinary differential equations on manifolds', *J. Nonlinear Sci.* **3**, 1–33.

P. J. Davis and P. Rabinowitz (1984), *Methods of Numerical Integration*, 2nd edn, Academic Press, Orlando, FL.

P. Deift, T. Nanda and C. Tomei (1983), 'Ordinary differential equations and the symmetric eigenvalue problem', *SIAM J. Num. Anal.* **20**, 1–22.

L. Dieci, R. D. Russell and E. S. van Vleck (1994), 'Unitary integrators and applications to continuous orthonormalization techniques', *SIAM J. Num. Anal.* **31**, 261–281.

K. Engø (1998), On the construction of geometric integrators in the RKMK class, Technical Report No. 158, Dept Comp. Sc., University of Bergen.

K. Engø and S. Faltinsen (1999), Numerical integration of Lie-Poisson systems while preserving coadjoint orbits and energy, Technical Report No. 179, Dept Comp. Sc., University of Bergen.

K. Engø, A. Marthinsen and H. Z. Munthe-Kaas (1999), DiffMan — an object oriented MATLAB toolbox for solving differential equations on manifolds, Technical Report No. 164, Dept Comp. Sc., University of Bergen.

D. J. Estep and A. M. Stuart (1995), 'The Rate of Error Growth in Hamiltonian-Conserving Integrators', *Z. Angew. Math. Phys.* **46**, 407–418.

S. Faltinsen (1998), Backward error analysis for Lie-group methods, Technical Report 1998/NA12, DAMTP, University of Cambridge.

S. Faltinsen (n.d.), Can Lie-group methods be symplectic?, (to appear).

S. Faltinsen, A. Marthinsen and H. Z. Munthe-Kaas (1999), Multistep methods integrating ordinary differential equations on manifolds, Technical Report Numerics No. 3/1999, The Norwegian University of Science and Technology, Trondheim.

F. Fer (1958), 'Résolution del l'equation matricielle $\dot{U} = pU$ par produit infini d'exponentielles matricielles', *Bull. Classe des Sci. Acad. Royal Belg.* **44**, 818–829.

H. Flaschka (1974), 'The Toda lattice i', *Phys. Rev. B* **9**, 1924–1925.

C. W. Gear (1971), *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, N.J.

H. Goldstein (1980), *Classical Mechanics*, Addison-Wesley, Reading, MA.

G. H. Golub and C. F. Van Loan (1996), *Matrix Computations*, 3rd edn, Johns Hopkins University Press, Baltimore.

V. Guillemin and A. Pollack (1974), *Differential Topology*, Prentice–Hall, Englewood Cliffs.

E. Hairer (1994), Backward analysis of numerical integrators and symplectic methods, in *Scientific Computation and Differential Equations* (K. Burrage, C. Baker, P. van der Houwen, Z. Jackiewicz and P. Sharp, eds), Vol. 1 of *Annals of Numer. Math.*, J.C. Baltzer, Amsterdam, pp. 107–132. Proceedings of the SCADE'93 conference, Auckland, New-Zealand, January 1993.

E. Hairer and C. Lubich (1997), 'The life-span of backward error analysis for numerical integrator', *Numer. Math.* **76**, 441–462.

E. Hairer, S. P. Nørsett and G. Wanner (1993), *Solving Ordinary Differential Equations I: Nonstiff Problems*, 2nd revised edn, Springer Verlag, Berlin.

F. Harary (1969), *Graph Theory*, Addison-Wesley, Reading MA.

F. Hausdorff (1906), 'Die symbolische Exponentialformel in der Gruppentheorie', *Berichte der Sächsischen Akademie der Wißenschaften (Math. Phys. Klasse* **58**, 19–48.

D. J. Higham (1997), 'Time-stepping and preserving orthonormality', *BIT* **37**, 24–36.

M. Hochbruck and C. Lubich (1997), 'On Krylov subspace approximations to the matrix exponential operator', *SIAM J. Num. Anal.* **34**, 1911–1925.

A. Iserles (1984), 'Solving linear ordinary differential equations by exponentials of iterated commutators', *Numerische Mathematik* **45**, 183–199.

A. Iserles (1997), Multistep methods on manifolds, Technical Report 1997/NA13, DAMTP, University of Cambridge.

A. Iserles (1999*a*), How large is the exponential of a bounded matrix?, Technical Report 1999/NA1, DAMTP, University of Cambridge.

A. Iserles (1999*b*), On Cayley-transform methods for the discretization of Lie-group equations, Technical Report 1999/NA4, DAMTP, University of Cambridge.

A. Iserles (n.d.), Fast computation of Magnus series, (to appear).

A. Iserles and S. P. Nørsett (1991), *Order Stars*, Chapman and Hall, London.

A. Iserles and S. P. Nørsett (1999), 'On the solution of linear differential equations in Lie groups', *Phil. Trans Royal Society A* **357**, 983–1020.

A. Iserles and A. Zanna (2000), 'On the dimension of certain graded Lie algebras arising in geometric integration of differential equations', *LMS J. Comput. & Maths* **3**, 44–75.

A. Iserles, A. Marthinsen and S. P. Nørsett (1999), 'On the implementation of the method of Magnus series for linear differential equations', *BIT* **39**, 281–304.

A. Iserles, S. P. Nørsett and A. F. Rasmussen (1998), Time-symmetry and high-order Magnus methods, Technical Report 1998/NA06, DAMTP, University of Cambridge.

G. Julia (1918), 'Memoire sur l'iteration des fonctions rationalles', *J. Math.* **8**, 47–245.

F. Kang and S. Zai-jiu (1995), 'Volume-preserving algorithmms for source-free dynamical systems', *Numerische Mathematik.* **71**, 451–463.

H. J. Landau (1994), 'The inverse eigenvalue problem for real symmetric Toeplitz matrices', *J. Amer. Math. Soc.* **7**, 749–767.

Y. Liu (1998), Projected Runge–Kutta methods for differential equations on matrix Lie groups, Technical Report 1998/NA1, DAMTP, University of Cambridge.

W. Magnus (1954), 'On the exponential solution of differential equations for a linear operator', *Comm. Pure and Appl. Math* **VII**, 649–673.

J. E. Marsden and T. S. Ratiu (1994), *Introduction to Mechanics and Symmetry*, Springer-Verlag, New York.

R. I. McLachlan, G. R. W. Quispel and N. Robidoux (1998), 'A unified approach to Hamiltonian systems, Poisson systems, gradient systems, and systems with Lyapunov functions and/or first integrals', *Phys. Rev. Letts* **81**, 2399–2403.

P. C. Moan (1998), Efficient approximation of Sturm–Liouville problems using Lie-group methods, Technical Report 1998/NA11, DAMTP, University of Cambridge.

P. C. Moan (n.d.), On the convergence of Magnus and Cayley expansions, (to appear).

C. Moler and C. F. Van Loan (1978), 'Nineteen dubious ways to compute the exponential of a matrix', *SIAM Rev.* **20**, 801–836.

J. Moser (1973), *Stable and Random Motion in Dynamical Systems*, Princeton University Press.

H. Munthe-Kaas (1995), 'Lie-Butcher Theory for Runge-Kutta Methods', *BIT* **35**, 572–587.

H. Munthe-Kaas (1998), 'Runge–Kutta methods on Lie groups', *BIT* **38**, 92–111.

H. Munthe-Kaas (1999), 'High order Runge–Kutta methods on manifolds', *Applied Numerical Mathematics* **29**, 115–127.

H. Munthe-Kaas and M. Haveraaen (1996), 'Coordinate Free Numerics – Closing the gap between 'Pure' and 'Applied' mathematics?', *Proceedings of ICIAM–95, Zeitschrift fuer Angewandte Mathematik und Mechanik (ZAMM).*

H. Munthe-Kaas and E. Lodden (n.d.), Lie group integrators for parabolic PDEs, (to appear).

H. Munthe-Kaas and B. Owren (1999), 'Computations in a free Lie algebra', *Phil. Trans Royal Society A* **357**, 957–982.

H. Munthe-Kaas and A. Zanna (1997), Numerical integration of differential equations on homogeneous manifolds, in *Foundations of Computational Mathematics* (F. Cucker and M. Shub, eds), Springer Verlag, pp. 305–315.

A. I. Neishtadt (1984), 'The separation of motions in systems with rapidly rotating phase', *J. Appl. Math. Mech.* **48**, 133–139.

S. P. Nørsett and G. Wanner (1981), 'Perturbed collocation and Runge–Kutta methods', *Numer. Math.* **38**, 193–208.

P. J. Olver (1995), *Equivalence, Invariants, and Symmetry*, Cambridge University Press.

B. Owren and A. Marthinsen (1999*a*), Integration methods based on canonical coordinates of the second kind, Technical Report Numerics No. 5/1999, Norwegian University of Science and Technology, Trondheim.

B. Owren and A. Marthinsen (1999*b*), 'Runge–Kutta methods adapted to manifolds and based on rigid frames', *BIT* **39**, 116–142.

B. Owren and B. Welfert (1996), The Newton iteration on Lie groups, Technical Report Numerics No. 3/1996, Norwegian University of Science and Technology, Trondheim.

L. R. Petzold, L. O. Jay and Y. Jeng (1997), 'Numerical solution of highly oscillatory ordinary differential equations', *Acta Numerica* **6**, 437–483.

S. Preuss (1973), 'Solving linear boundary value problems by approximating the coefficients', *Maths Comp.* **27**, 551–561.

J. D. Pryce (1993), *Numerical Solution of Sturm–Liouville Problems*, Oxford University Press, New York.

S. Reich (1996), Backward error analysis for numerical integrators, Technical Report SC 96-21, Konrad-Zuse Zentrum für Informationstechnik Berlin.

J. M. Sanz Serna and M. P. Calvo (1994), *Numerical Hamiltonian Problems*, Chapman & Hall.

C. Størmer (1907), 'Sur les trajectoires des corpuscules électrisés', *Arch. Sci. Phys. Nat., Genève* **24**, 5–18, 113–158, 221–247.

A. M. Stuart and A. R. Humphries (1996), *Dynamical Systems and Numerical Analysis*, Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, Cambridge.

M. Toda (1981), *Theory of Nonlinear Lattices*, Springer Verlag, Berlin.

W. F. Trench (1997), 'Numerical solution of the inverse eigenvalue problem for real symmetric Toepliz matrices', *SIAM J. Sci. Comput.* **18**, 1722–1736.

V. S. Varadarajan (1984), *Lie Groups, Lie Algebras, and Their Representations*, GTM 102, Springer-Verlag.

J. Wei and E. Norman (1964), 'On global representations of the solutions of linear differential equations as a product of exponentials', *Proc. Amer. Math. Soc.* **15**, 327–334.

H. Yoshida (1990), 'Construction of higher order symplectic integrators', *Phys. Letts A* **150**, 262–268.

N. J. Zabusky and M. D. Kruskal (1965), 'Interaction of solitons in a colisionless plasma and the recurrences of initial states', *Phys. Rev. Letts* **15**, 240–243.

A. Zanna (1996), The method of iterated commutators for ordinary differential equations on Lie groups, Technical Report 1996/NA12, DAMTP, University of Cambridge.

A. Zanna (1998), On the Numerical Solution of Isospectral Flows, PhD thesis, University of Cambridge, England.

A. Zanna (1999), 'Collocation and relaxed collocation for the Fer and the Magnus expansions', *SIAM J. Num. Anal.* **36**, 1145–1182.

A. Zanna and H. Munthe-Kaas (1997), Iterated commutators, Lie's reduction method and ordinary differential equations on matrix Lie groups, in *Foundation of Computational Mathematics* (F. Cucker and M. Shub, eds), Springer-Verlag, pp. 434–441.

A. Zanna, K. Engø and H. Z. Munthe-Kaas (1999), Adjoint and selfadjoint Lie-group methods, Technical Report NA1999/02, DAMTP, University of Cambridge.

A. List of methods

In this appendix we list few of the methods that have been described in the survey. No attempt has been expanded to explain the methods, beyond references to the relevant earlier material.

A.1. RK-MK methods

All classical RK methods can be translated into Lie-group methods. Assume that the Butcher tableau

$$\begin{array}{c|cccc} c_1 & a_{1,1} & a_{1,2} & \cdots & a_{1,\nu} \\ c_2 & a_{2,1} & a_{2,2} & \cdots & a_{2,\nu} \\ \vdots & \vdots & \vdots & & \vdots \\ c_\nu & a_{\nu,1} & a_{\nu,2} & \cdots & a_{\nu,\nu} \\ \hline & b_1 & b_2 & \cdots & b_\nu \end{array}$$

defines a Runge–Kutta method of order p in the classical sense (Hairer et al. 1993) and that ϕ is a map from $\mathfrak{g}$ to $\mathcal{G}$, for instance $\phi = \exp$, the exponential mapping, or $\phi = \mathrm{cay}$, the Cayley mapping for quadratic Lie groups. Then the corresponding order-p RK-MK algorithm for the Lie-group equation $Y' = A(t, Y)Y$ is obtained as

$$\left.\begin{aligned} \Theta_k &= \sum_{l=1}^{\nu} a_{k,l} F_l, \\ F_k &= \mathrm{d}\phi^{-1}(\Theta_k, A_k, p), \\ A_k &= hA(t_n + c_k h, \phi(F_k)Y_n), \end{aligned}\right\} \quad k = 1, \ldots, \nu, \qquad \begin{aligned} \Theta &= \sum_{l=1}^{\nu} b_l F_l, \\ Y_{n+1} &= \phi(\Theta)Y_n, \end{aligned} \tag{A.1}$$

for $n \in \mathbb{N}$, and it is explicit provided that the underlying RK scheme is explicit. The function $\mathrm{d}\phi^{-1}(B, C, p)$ is a truncation of $\mathrm{d}\phi_B^{-1}(C)$ to order $p - 1$, which is usually sufficient for a method of order p, given that the error is subsumed in the $\mathcal{O}\big(h^{p+1}\big)$ term. In some instances, like when $\mathfrak{g} = \mathfrak{so}(3)$, the function $\mathrm{d}\phi_B^{-1}(C)$ can be evaluated exactly (see Appendix B).

Some popular schemes of the type (*A*.1) are Lie-group versions of

- forward Euler,

$$Y_{n+1} \;=\; \phi\big(hA(t_n, Y_n)\big)Y_n;$$

- the implicit midpoint rule,

$$\begin{aligned} F_1 &= \tfrac{1}{2}hA\big(t_n + \tfrac{1}{2}h, \phi(F_1)Y_n\big), \\ \Theta &= F_1, \\ Y_{n+1} &= \phi(\Theta)Y_n; \end{aligned}$$

- the trapezoidal rule,

$$\begin{aligned} F_1 &= hA\big(t_n, Y_n\big), \\ F_2 &= hA\big(t_n + h, \phi(\tfrac{1}{2}(F_1 + F_2))Y_n\big), \\ \Theta &= \tfrac{1}{2}(F_1 + F_2), \\ Y_{n+1} &= \phi(\Theta)Y_n; \end{aligned}$$

- Heun's method,

$$\begin{aligned} F_1 &= hA\big(t_n, Y_n\big), \\ F_2 &= hA\big(t_n + \tfrac{1}{2}h, \phi(\tfrac{1}{2}F_1)Y_n\big), \\ \Theta &= F_2, \\ Y_{n+1} &= \phi(\Theta)Y_n. \end{aligned}$$

The scheme (*A*.1) employs coordinates centred at Y_n. Schemes with coordinates centred at an arbitrary point can be obtained in a similar manner,

$$\left.\begin{aligned} \Theta_k &= C_{h,n} + \sum_{j=1}^{\nu} a_{k,l}F_l \\ F_k &= \mathrm{d}\phi^{-1}(\Theta_k, A_k, p), \\ A_k &= hA(t_n + c_k h, \phi(F_k)\phi(C_{h,n})^{-1}Y_n), \end{aligned}\right\} \quad k = 1, \ldots, \nu,$$

$$\begin{aligned} C_{h,n} &= \vartheta(F_1, \ldots, F_\nu), \\ \Theta &= C_{h,n} + \sum_{l=1}^{\nu} b_l F_l, \\ Y_{n+1} &= \phi(\Theta)\phi(C_{h,n})^{-1}Y_n \end{aligned}$$

for $n = 0, 1, 2, \ldots$. In particular, for *geodesic symmetric coordinates* we have

$$\left.\begin{aligned} \Theta_k &= \sum_{j=1}^{\nu} \big(a_{i,j} - \tfrac{1}{2} b_l\big) F_l, \\ F_k &= \mathrm{d}\phi^{-1}(\Theta_k, A_k, p), \\ A_k &= hA(t_n + c_k h, \phi(F_k)\phi(C_{h,n})^{-1} Y_n), \end{aligned}\right\} \quad k = 1, \ldots, \nu,$$
$$\begin{aligned} C_{h,n} &= -\tfrac{1}{2} \sum_{l=1}^{\nu} b_l F_l, \\ \Theta &= \tfrac{1}{2} \sum_{l=1}^{\nu} b_l F_l \quad (= -C_{h,n}), \\ Y_{n+1} &= \phi(\Theta)\phi(C_{h,n})^{-1} Y_n, \end{aligned} \tag{A.2}$$

while *flow-symmetric coordinates* yield

$$\left.\begin{aligned} \Theta_k &= \sum_{j=1}^{\nu} (a_{i,j} - w_l) F_l \\ F_k &= \mathrm{d}\phi^{-1}(\Theta_k, A_k, p), \\ A_k &= hA(t_n + c_k h, \phi(F_k)\phi(C_{h,n})^{-1} Y_n), \end{aligned}\right\} \quad k = 1, 2, \ldots, \nu,$$
$$\begin{aligned} C_{h,n} &= -\sum_{l=1}^{\nu} w_l F_l, \\ \Theta &= \sum_{l=1}^{\nu} (b_l - w_l) F_l, \\ Y_{n+1} &= \phi(\Theta)\phi(C_{h,n})^{-1} Y_n, \end{aligned} \tag{A.3}$$

where the weights w_ls are obtained integrating the Lagrangian cardinal polynomials, as in (7.7).

Herewith a number of important examples, originating in familiar classical RK methods.

- Order-four Gauss–Legendre scheme,

$$\begin{array}{c|cc} \frac{1}{2} - \frac{\sqrt{3}}{6} & \frac{1}{4} - \frac{\sqrt{3}}{6} & \frac{1}{4} \\ \frac{1}{2} + \frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4} + \frac{\sqrt{3}}{6} \\ \hline \boldsymbol{b}^{\mathrm{T}} & \frac{1}{2} & \frac{1}{2} \\ \boldsymbol{w}^{\mathrm{T}} & \frac{1}{4} + \frac{\sqrt{3}}{8} & \frac{1}{4} - \frac{\sqrt{3}}{8} \end{array} \tag{A.4}$$

- Order-six Gauss–Legendre scheme,

$$\begin{array}{c|ccc}
\frac{1}{2}-\frac{\sqrt{15}}{10} & \frac{5}{36} & \frac{2}{9}-\frac{\sqrt{15}}{15} & \frac{5}{36}-\frac{\sqrt{15}}{30} \\
\frac{1}{2} & \frac{5}{36}+\frac{\sqrt{15}}{24} & \frac{2}{9} & \frac{5}{36}-\frac{\sqrt{15}}{24} \\
\frac{1}{2}+\frac{\sqrt{15}}{10} & \frac{5}{36}+\frac{\sqrt{15}}{30} & \frac{2}{9}+\frac{\sqrt{15}}{15} & \frac{5}{36} \\
\hline
\boldsymbol{b}^{\mathrm{T}} & \frac{5}{18} & \frac{4}{9} & \frac{5}{18} \\
\boldsymbol{w}^{\mathrm{T}} & \frac{5}{36}+\frac{\sqrt{15}}{24} & \frac{2}{9} & -\frac{1}{24}
\end{array} \tag{A.5}$$

- Order-four Gauss–Lobatto scheme,

$$\begin{array}{c|ccc}
0 & 0 & 0 & 0 \\
\frac{1}{2} & \frac{5}{24} & \frac{1}{3} & -\frac{1}{24} \\
1 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\
\hline
\boldsymbol{b}^{\mathrm{T}} & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\
\boldsymbol{w}^{\mathrm{T}} & \frac{5}{24} & \frac{1}{3} & -\frac{1}{24}
\end{array}. \tag{A.6}$$

Note that both in the case of the Gauss–Legendre of order six and Gauss–Lobatto of order four it is true that $C_{h,n} = -F_2$, hence $C_{h,n}$ need not be explicitly evaluated in (*A*.3).

Algorithms (*A*.1–3) are practical for methods of order $p \leq 4$, since for higher-order methods the computation of $\mathrm{d}\phi^{-1}(A, B, p)$ requires the evaluation of a large number of commutators. For higher-order schemes it is recommended instead to use schemes based on graded Lie algebras, along the lines of Section 5.3.

Graded Lie algebras can be also effectively used to optimise existing RK-MK methods. The scheme

$$\begin{aligned}
A_1 &= hA(t_n, Y_n), \\
B_1 &= A_1, \\
A_2 &= hA\big(t_n + c_2 h,\ \mathrm{expm}(\tfrac{1}{2}B_1)Y_n\big), \\
B_2 &= A_2 - A_1 \\
A_3 &= hA\big(t_n + c_3 h,\ \mathrm{expm}(\tfrac{1}{2}B_1 + \tfrac{1}{2}B_2 - \tfrac{1}{8}[B_1, B_2])Y_n\big), \\
B_3 &= A_3 - A_2, \\
A_4 &= hA\big(t_n + c_4 h,\ \mathrm{expm}(B_1 + B_2 + B_3)Y_n\big), \\
B_4 &= A_4 - 2A_2 + A_1, \\
\Theta &= B_1 + B_2 + \tfrac{1}{3}B_3 + \tfrac{1}{6}B_4 - \tfrac{1}{6}[B_1, B_2] - \tfrac{1}{12}[B_1, B_4] \\
Y_{n+1} &= \mathrm{expm}(\Theta)Y_n,
\end{aligned}$$

based on the Runge–Kutta tableau

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array} \tag{A.7}$$

requires only two commutators instead of six if implemented using (*A*.1).

A.2. Higher-order methods for linear equations using graded Lie algebras

In this subsection we present methods for linear equations $Y' = A(t)Y$, whereby the number of commutators is reduced using graded algebras and a further technique due to Blanes et al. (1999) and described briefly in Section 5.4. We do not distinguish between RK-MK and Magnus-type methods because, subject to these reductions, the two methods produce very similar results.

We consider collocation-type schemes. Denote by

$$\mathrm{VDM}(\boldsymbol{d}) = (d_i^{j-1})_{i,j=1}^{\nu}$$

the Vandermonde matrix generated by the vector $\boldsymbol{d}$. In particular,

- set $V = \mathrm{VDM}(\boldsymbol{c})$, $\boldsymbol{c}^{\mathrm{T}} = (c_1, c_2, \dots c_\nu)$ for non-symmetric collocation schemes;
- for symmetric collocation schemes, we take full advantage of symmetry, setting $V = \mathrm{VDM}(\boldsymbol{c} - \frac{1}{2})$.

Then,

$$\left.\begin{aligned} A_k &= hA(t_N + c_k h), \\ B_k &= \sum_{l=1}^{\nu} (V^{-1})_{k,l} A_l, \end{aligned}\right\} \quad k = 1, 2, \dots, \nu,$$
$$\Theta = \mathrm{d}\varphi^{-1}(B_1, B_2, \dots, B_\nu), \tag{A.8}$$
$$Y_{n+1} = \phi(\Theta) Y_n,$$

where $\mathrm{d}\varphi^{-1}$ is an order-p truncation to the $\mathrm{d}\phi^{-1}$-equation in the graded basis $B_1, \dots, B_\nu$. For symmetric collocation schemes, the function $\mathrm{d}\varphi^{-1}$ in (*A*.8) depends on terms of odd grade only. Specifically, for $\phi = \mathrm{expm}$,

- For order-six Gauss–Legendre scheme

$$\begin{aligned} \mathrm{d}\varphi^{-1}(B_1, B_2, B_3) \;=\; & B_1 + \tfrac{1}{12} B_3 - \tfrac{1}{12}[B_1, B_2] + \tfrac{1}{240}[B_2, B_3] \\ & + \tfrac{1}{360}[B_1, [B_1, B_3]] - \tfrac{1}{240}[B_2, [B_1, B_2]] \\ & + \tfrac{1}{720}[B_1, [B_1, [B_1, B_2]]] \end{aligned} \tag{A.9}$$

(for order two and order four it suffices to consider the first one and two terms only, respectively, and the nodes of the corresponding quadrature), which can be evaluated with just four commutators using the technique of Blanes et al. (1999),

$$\begin{aligned} \mathrm{d}\varphi^{-1}(B_1, B_2, B_3) &\approx C_1 + C_2 + C_3, \\ C_1 &= B_1 + \tfrac{1}{12} B_3 \\ C_2 &= -\tfrac{1}{12}[B_1 + \tfrac{1}{20} B_3, B_2], \\ C_3 &= [B_1, [B_1, \tfrac{1}{360} B_3 - \tfrac{1}{60} C_2]] - \tfrac{1}{20}[B_2, C_2], \end{aligned}$$

producing an order-six time-symmetric truncation of $\mathrm{d}\varphi^{-1}(B_1, B_2, B_3)$.

- For order-four Gauss–Lobatto scheme (Ehle $\mathrm{III_A}$),

$$\mathrm{d}\varphi^{-1}(B_1, B_2, B_3) = B_1 + \tfrac{1}{12} B_3 - \tfrac{1}{12}[B_1, B_2].$$

- For order-six Gauss–Lobatto scheme, with nodes $c_1 = 0$, $c_2 = \frac{1}{2} - \frac{\sqrt{5}}{10}$, $c_3 = \frac{1}{2} + \frac{\sqrt{5}}{10}$ and $c_4 = 1$,

$$\begin{aligned} \mathrm{d}\varphi^{-1}(B_1, B_2, B_3, B_4) = {} & B_1 + \tfrac{1}{12} B_3 - \tfrac{1}{12}[B_1, B_2] - \tfrac{1}{8}[B_1, B_4] + \tfrac{1}{240}[B_2, B_3] \\ & + \tfrac{1}{360}[B_1, [B_1, B_3]] - \tfrac{1}{240}[B_2, [B_1, B_2]] \\ & + \tfrac{1}{720}[B_1, [B_1, [B_1, B_2]]]. \end{aligned}$$

which can be evaluated to correct order as

$$\begin{aligned} \mathrm{d}\varphi^{-1}(B_1, B_2, B_3, B_4) &\approx C_1 + C_2 + C_3, \\ C_1 &= B_1 + \tfrac{1}{12} B_3 \\ C_2 &= -\tfrac{1}{12}[B_1 + \tfrac{1}{20} B_3, B_2 + \tfrac{3}{2} B_4], \\ C_3 &= [B_1, [B_1, \tfrac{1}{360} B_3 - \tfrac{1}{60} C_2]] - \tfrac{1}{20}[B_2, C_2], \end{aligned}$$

with just four commutators.

It may strike the reader that, after the transformation in the self-adjoint basis $\{B_i\}$, the expression of Θ has always the same type of expansion, regardless of the choice of collocation nodes. This is no surprise, the information about the nodes being hidden in the basis elements B_i. Changing into the self-adjoint basis implies that we integrate a Taylor-type expansion, in combination with a truncation of dexp^{-1}, which is independent of the nodes, depending only on the order of the method.

A.3. Methods using the Magnus expansion

Magnus-type methods are well suited to collocation-type techniques. This results in implicit schemes for nonlinear Lie-group equations.

- Order-four Gauss–Legendre:

$$\begin{aligned}
F_1 &= \tfrac{1}{4}A_1 + (\tfrac{1}{4} - \tfrac{\sqrt{3}}{6})A_2 + (\tfrac{5}{144} - \tfrac{\sqrt{3}}{48})[A_1, A_2],\\
A_1 &= hA\big(t_n + c_1 h,\ \mathrm{expm}(F_1)Y_n\big),\\
F_2 &= (\tfrac{1}{4} + \tfrac{\sqrt{3}}{6})A_1 + \tfrac{1}{4}A_2 - (\tfrac{5}{144} + \tfrac{\sqrt{3}}{48})[A_1, A_2],\\
A_2 &= hA\big(t_n + c_2 h,\ \mathrm{expm}(F_2)Y_n\big)\\
\Theta &= \tfrac{1}{2}(A_1 + A_2) - \tfrac{\sqrt{3}}{12}[A_1, A_2],\\
Y_{n+1} &= \mathrm{expm}(\Theta)Y_n,
\end{aligned}$$

  where $c_i = \frac{1}{2} \pm \frac{\sqrt{3}}{6}$, $i = 1, 2$.
- Order-four Gauss–Lobatto:

$$\begin{aligned}
F_1 &= \boldsymbol{O},\\
A_1 &= hA(t_n, Y_n),\\
F_2 &= \tfrac{5}{24}A_1 + \tfrac{1}{3}A_2 - \tfrac{1}{24}A_3 - (\tfrac{11}{480}[A_1, A_2] + \tfrac{5}{1152}[A_1, A_3] + \tfrac{1}{144}[A_2, A_3]),\\
A_2 &= hA\big(t_n + \tfrac{1}{2}h,\ \mathrm{expm}(F_2)Y_n\big),\\
F_3 &= \tfrac{1}{6}A_1 + \tfrac{2}{3}A_2 + \tfrac{1}{6}A_3 - (\tfrac{1}{15}[A_1, A_2] + \tfrac{1}{60}[A_1, A_3] + \tfrac{1}{15}[A_2, A_3]),\\
A_3 &= hA\big(t_n + h,\ \mathrm{expm}(F_3)Y_n\big),\\
\Theta &= \tfrac{1}{6}A_1 + \tfrac{2}{3}A_2 + \tfrac{1}{6}A_3 - (\tfrac{1}{15}[A_1, A_2] + \tfrac{1}{60}[A_1, A_3] + \tfrac{1}{15}[A_2, A_3])\\
Y_{n+1} &= \mathrm{expm}(\Theta)Y_n.
\end{aligned}$$

Relaxing the collocation conditions, it is possible to obtain explicit methods.

- An explicit order-three scheme:

$$\begin{aligned}
A_1 &= hA(t_n, Y_n),\\
A_2 &= hA\big(t_n + \frac{1}{2}h,\ \mathrm{expm}(A_1)Y_n\big),\\
A_3 &= hA\big(t_n + h,\ \mathrm{expm}(-A_1 + 2A_2)Y_n\big),\\
\Theta &= \tfrac{1}{6}A_1 + \tfrac{2}{3}A_2 + \tfrac{1}{6}A_3 - [A_1 - A_3, \tfrac{1}{15}A_2 + \tfrac{1}{60}A_3],\\
Y_{n+1} &= \mathrm{expm}(\Theta)Y_n
\end{aligned}$$

A.4. Magnus-type methods with geodesic/flow coordinates

Magnus-type methods introduced in Appendix A.3 employ coordinates at Y_n, hence they cease to be self adjoint for nonlinear problems. Below we describe their self-adjoint modification.

- Order-four Gauss–Legendre method based on geodesic coordinates

$$\begin{aligned}
F_1 &= \tfrac{1}{4}A_1 + \left(\tfrac{1}{4} - \tfrac{\sqrt{3}}{6}\right) A_2 + \left(\tfrac{5}{144} - \tfrac{\sqrt{3}}{24}\right)[A_1, A_2],\\
A_1 &= hA\Big(t_n + c_1 h,\ \operatorname{expm}(F_1 - C_{h,n})\operatorname{expm}(C_{h,n})Y_n\Big),\\
F_2 &= \left(\tfrac{1}{4} + \tfrac{\sqrt{3}}{6}\right) A_1 + \tfrac{1}{4}A_2 - \left(\tfrac{5}{144} + \tfrac{\sqrt{3}}{24}\right)[A_1, A_2],\\
A_2 &= hA\Big(t_n + c_2 h,\ \operatorname{expm}(F_2 - C_{h,n})\operatorname{expm}(C_{h,n})Y_n\Big),\\
C_{h,n} &= \tfrac{1}{4}(A_1 + A_2) - \tfrac{\sqrt{3}}{24}[A_1, A_2],\\
\Theta &= 2C_{h,n},\\
Y_{n+1} &= \operatorname{expm}(\Theta)Y_n.
\end{aligned}$$

- Order-four Gauss–Legendre methods with flow coordinates

$$\begin{aligned}
F_1 &= \tfrac{1}{4}A_1 + \left(\tfrac{1}{4} - \tfrac{\sqrt{3}}{6}\right) A_2 + \left(\tfrac{1}{288} - \tfrac{\sqrt{3}}{96}\right)[A_1, A_2],\\
A_1 &= hA\Big(t_n + c_1 h,\ \operatorname{expm}(F_1 - C_{h,n})\operatorname{expm}(C_{h,n})Y_n\Big),\\
F_2 &= \left(\tfrac{1}{4} + \tfrac{\sqrt{3}}{6}\right) A_1 + \tfrac{1}{4}A_2 - \left(\tfrac{1}{288} + \tfrac{\sqrt{3}}{96}\right)[A_1, A_2],\\
A_2 &= hA\Big(t_n + c_2 h,\ \operatorname{expm}(F_2 - C_{h,n})\operatorname{expm}(C_{h,n})Y_n\Big),\\
C_{h,n} &= \left(\tfrac{1}{4} + \tfrac{\sqrt{3}}{8}\right) A_1 + \left(\tfrac{1}{4} - \tfrac{\sqrt{3}}{8}\right) A_2 - \tfrac{\sqrt{3}}{96}[A_1, A_2],\\
\Theta &= \tfrac{1}{2}(A_1 + A_2) - \tfrac{\sqrt{3}}{48}[A_1, A_2],\\
Y_{n+1} &= \operatorname{expm}(\Theta - C_{h,n})\operatorname{expm}(C_{h,n})Y_n.
\end{aligned}$$

Although Magnus-type methods with geodesic/flow coordinates exist for every order p, devising such schemes for nonlinear problems and order greater than four is hard. For this reason we shall restrict our attention to order-four methods. We let $c_1, c_2, \ldots, c_\nu$ be the collocation nodes.

- Collocation order-four Magnus method with geodesic coordinates:

$$\left.\begin{aligned}
F_k &= \sum_{j=1}^{\nu} a_{i,j}A_l + \tfrac{1}{2}\sum_{l,j=1}^{\nu}(a_{k;l,j} + \tfrac{1}{2}b_l a_{k,l})[A_l, A_j],\\
A_k &= hA(t_n + c_k h,\ \operatorname{expm}(F_k - C_{h,n})\operatorname{expm}(C_{h,n})Y_n),
\end{aligned}\right\} \quad k = 1, \ldots, \nu,$$

$$\begin{aligned}
C_{h,n} &= \tfrac{1}{2}\sum_{l=1}^{\nu} b_l A_l + \tfrac{1}{4}\sum_{l,j=1}^{\nu} b_{l,j}[A_l, A_j],\\
\Theta &= 2C_{h,n},\\
Y_{n+1} &= \operatorname{expm}(\Theta)Y_n.
\end{aligned}$$

- Collocation order-four Magnus method with flow coordinates:

$$\left.\begin{aligned} F_k &= \sum_{l=1}^{\nu} a_{k,l} A_l + \tfrac{1}{2} \sum_{l,j=1}^{\nu} (a_{k;l,j} + w_l a_{k,l})[A_l, A_j], \\ A_k &= hA(t_n + c_k h,\ \mathrm{expm}(F_k - C_{h,n})\,\mathrm{expm}(C_{h,n}) Y_n), \end{aligned}\right\} \quad k = 1, \ldots, \nu,$$

$$\begin{aligned} C_{h,n} &= \sum_{l=1}^{\nu} w_l A_l + \tfrac{1}{2} \sum_{l,j=1}^{\nu} w_{l,j} [A_l, A_j], \\ \Theta &= \sum_{l=1}^{\nu} b_l A_l + \tfrac{1}{2} \sum_{l,j=1}^{\nu} (b_{l,j} + w_l b_j)[A_l, A_j], \\ Y_{n+1} &= \mathrm{expm}(\Theta - C_{h,n})\,\mathrm{expm}(C_{h,n}) Y_n. \end{aligned}$$

  In both cases the $a_{k;l,j}$s are evaluated according to (5.15) and the $b_{l,j}$s and $w_{l,j}$s are evaluated from (5.15) for $\theta = 1$ and $\theta = \frac{1}{2}$ respectively. The weights w_l are given by (7.7).

To conclude, it should be noted that all Lie-group methods for nonlinear problems require a number of exponential evaluations in the internal stages. This is a consequence of the fact that the funtion $A(t, Z)$ may fail to be an element of $\mathfrak{g}$ for arguments $Z \notin \mathcal{G}$. If $A(t, Z) \in \mathfrak{g}$ for all matrices Z then exponentiations (or, with greater generality, evaluations of the map ϕ) in the internal stages may be disregarded, at the cost of a minor increase of local truncation error. If $A(t, Z) \in \mathfrak{g}$ only when $Z \in \mathcal{G}$, however, disregarding the evaluation of the map ϕ in the internal stages would compromise the assurance that the numerical approximation Y_{n+1} resides in $\mathcal{G}$. However, as observed by Liu (1998), in some cases it is possible to devise simplified versions of the methods, whereby $A(t, Z)$ is projected on $\mathfrak{g}$ according to need. Specifically, setting $A = A(t, Z)$, we note that

$$\boldsymbol{P}(A) = \tfrac{1}{2}(A - A^{\mathrm{T}})$$

is a projector onto $\mathfrak{so}(N)$, the algebra of skew-symmetric matrices,

$$\boldsymbol{P}(A) = A - \delta \boldsymbol{I}, \qquad \delta = \tfrac{1}{N} \mathrm{tr} A,$$

is a projector onto $\mathfrak{sl}(N)$, the algebra of matrices with zero trace, and

$$\boldsymbol{P}(A) = \tfrac{1}{2}(A + J A^{\mathrm{T}} J), \qquad J = \begin{bmatrix} \boldsymbol{O} & \boldsymbol{I} \\ -\boldsymbol{I} & \boldsymbol{O} \end{bmatrix},$$

is a projector onto $\mathfrak{sp}(N)$, the algebra of symplectic matrices. Using these projectors in the internal stages of Lie-group methods may significantly reduce the number of evaluations of the map ϕ.

B. Fast computation of 3D rotations

Rotations in three dimensions are ubiquitous in computational mechanics, hence it is important to have fast algorithms for their computation. The Lie algebra $\mathfrak{so}(3)$ can be realized either as the set of all skew-symmetric 3×3 matrices with the matrix commutator as the bracket, or as the Euclidean space $\mathbb{R}^3$ with the vector product as the bracket. As we will see, some formulae are most easily expressed by representing $\mathfrak{so}(3)$ as 3-vectors, while other formulae appear more naturally in terms of skew-symmetric 3×3 matrices. It is convenient to switch back and forth between these forms. The Lie algebra isomorphism between these two representations is given by the *hat map*,

$$\widehat{\boldsymbol{x}} = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix} \tag{B.1}$$

mapping $\boldsymbol{x} \in \mathbb{R}^3$ into a 3×3 skew-symmetric matrix $\widehat{\boldsymbol{x}}$ such that $\widehat{\boldsymbol{x}}\boldsymbol{y} = \boldsymbol{x}\times\boldsymbol{y}$. (Note that, for clarity sake, we have abandoned our convention of denoting matrices with upper-case letters.) In particular, the identities

$$\begin{aligned}
[\widehat{\boldsymbol{x}}, \widehat{\boldsymbol{y}}] &= \widehat{\boldsymbol{x}}\widehat{\boldsymbol{y}} - \widehat{\boldsymbol{y}}\widehat{\boldsymbol{x}} = \widehat{\boldsymbol{x}\times\boldsymbol{y}}, && \text{(B.2)}\\
\widehat{\boldsymbol{x}}\widehat{\boldsymbol{y}}\widehat{\boldsymbol{x}} &= -(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{y})\widehat{\boldsymbol{x}}, && \text{(B.3)}\\
\widehat{\boldsymbol{x}}^2\widehat{\boldsymbol{y}} + \widehat{\boldsymbol{y}}\widehat{\boldsymbol{x}}^2 &= -(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x})\widehat{\boldsymbol{y}} - (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{y})\widehat{\boldsymbol{x}}, && \text{(B.4)}\\
\widehat{\boldsymbol{x}}^2\widehat{\boldsymbol{y}}^2 - \widehat{\boldsymbol{y}}^2\widehat{\boldsymbol{x}}^2 &= -(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{y})\widehat{\boldsymbol{x}\times\boldsymbol{y}} && \text{(B.5)}
\end{aligned}$$

are obeyed. For future convenience, we let

$$\theta = \|\boldsymbol{x}\| = (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x})^{1/2}, \qquad \vartheta = \frac{\theta}{2}.$$

Note that since $\widehat{\boldsymbol{x}}^3 = -\theta^2\widehat{\boldsymbol{x}}$, we deduce that for any real analytic function $f(z)$ we can easily obtain real functions $c_0(z)$, $c_1(z)$ and $c_2(z)$ such that

$$f(\widehat{\boldsymbol{x}}) = c_0(\theta)\boldsymbol{I} + c_1(\theta)\widehat{\boldsymbol{x}} + c_2(\theta)\widehat{\boldsymbol{x}}^2.$$

Interesting examples include

$$\begin{aligned}
\operatorname{expm}(\widehat{\boldsymbol{x}}) &= \boldsymbol{I} + \frac{\sin\theta}{\theta}\widehat{\boldsymbol{x}} + \frac{1}{2}\frac{\sin^2\vartheta}{\vartheta^2}\widehat{\boldsymbol{x}}^2, && \text{(B.6)}\\
(\boldsymbol{I} - \widehat{\boldsymbol{x}})^{-1} &= \boldsymbol{I} + \frac{1}{1+\theta^2}(\widehat{\boldsymbol{x}} + \widehat{\boldsymbol{x}}^2), && \text{(B.7)}\\
\operatorname{cay}(\widehat{\boldsymbol{x}}) &= \left(\boldsymbol{I} + \tfrac{1}{2}\widehat{\boldsymbol{x}}\right)\left(\boldsymbol{I} - \tfrac{1}{2}\widehat{\boldsymbol{x}}\right)^{-1} && \text{(B.8)}\\
&= \boldsymbol{I} + c\widehat{\boldsymbol{x}} + \frac{c}{2}\widehat{\boldsymbol{x}}^2,
\end{aligned}$$

where

$$c \equiv c(\theta) = \frac{4}{4+\theta^2}. \tag{B.9}$$

The first of those, $(B.6)$, is well known in literature as the *Rodrigues formula* for the exponential mapping (Marsden and Ratiu 1994).

In many instances we need to compute repeatedly expressions in the general form $f(\mathrm{ad}_{\boldsymbol{x}})(\boldsymbol{y})$ for some function $f(x)$. These expressions in $\mathfrak{so}(3)$ can be computed fast in a very similar manner. Since $\mathrm{ad}_{\boldsymbol{x}}(\boldsymbol{y}) = \widehat{\boldsymbol{x}}\boldsymbol{y}$, it is true that

$$f(\mathrm{ad}_{\boldsymbol{x}})(\boldsymbol{y}) = f(\widehat{\boldsymbol{x}})\boldsymbol{y},$$

and we may use the same technique as above to simplify $f(\widehat{\boldsymbol{x}})$.

$$\mathrm{dexp}_{\boldsymbol{x}} = \frac{\exp\widehat{\boldsymbol{x}} - \boldsymbol{I}}{\widehat{\boldsymbol{x}}} = \boldsymbol{I} + \frac{\sin^2\vartheta}{2\vartheta^2}\widehat{\boldsymbol{x}} + \frac{\theta - \sin\theta}{\theta^3}\widehat{\boldsymbol{x}}^2 \tag{B.10}$$

$$\mathrm{dexp}^{-1}_{\boldsymbol{x}} = \frac{\widehat{\boldsymbol{x}}}{\exp\widehat{\boldsymbol{x}} - \boldsymbol{I}} = \boldsymbol{I} - \frac{1}{2}\widehat{\boldsymbol{x}} - \frac{\theta\cot\vartheta - 2}{2\theta^2}\widehat{\boldsymbol{x}}^2 \tag{B.11}$$

$$\mathrm{dcay}_{\boldsymbol{x}} = \left(\boldsymbol{I} - \tfrac{1}{2}\widehat{\boldsymbol{x}}\right)^{-1}\left(\boldsymbol{I} + \tfrac{1}{2}\widehat{\boldsymbol{x}}\right)^{-1} = c\left(\boldsymbol{I} + \tfrac{1}{2}\widehat{\boldsymbol{x}}\right) \tag{B.12}$$

$$\mathrm{dcay}^{-1}_{\boldsymbol{x}} = c\left(\boldsymbol{I} - \tfrac{1}{2}\widehat{\boldsymbol{x}} + \tfrac{1}{4}\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}\right), \tag{B.13}$$

with c as in $(B.9)$. These formulae are based on the representation of $\mathfrak{so}(3)$ as 3-vectors, so e.g. $(B.12)$ should read $\mathrm{dcay}_{\boldsymbol{x}}(\boldsymbol{y}) = c(\boldsymbol{y} + \frac{1}{2}\widehat{\boldsymbol{x}}\boldsymbol{y}) = c(\boldsymbol{y} + \frac{1}{2}\boldsymbol{x}\times\boldsymbol{y})$.

If $U = \mathrm{expm}(\widehat{\boldsymbol{x}})$ is an orthogonal matrix, the matrix $\widehat{\boldsymbol{x}}$ can be obtained by means of the matrix logarithm as

$$\widehat{\boldsymbol{x}} = \mathrm{logm}(U) = \frac{\sin^{-1}\|\boldsymbol{y}\|}{\|\boldsymbol{y}\|}\widehat{\boldsymbol{y}}, \qquad \widehat{\boldsymbol{y}} = \tfrac{1}{2}(U - U^{\mathrm{T}}). \tag{B.14}$$

Similarly, we may invert the Cayley map as

$$\widehat{\boldsymbol{x}} = \mathrm{cay}^{-1}(U) = 2\left(\frac{1 - \sqrt{1 - \|\boldsymbol{y}\|^2}}{\|\boldsymbol{y}\|^2}\right)\widehat{\boldsymbol{y}}, \qquad \widehat{\boldsymbol{y}} = \tfrac{1}{2}(U - U^{\mathrm{T}}). \tag{B.15}$$

Also canonical coordinates of the second kind (6.15) can be evaluated explicitly. Letting

$$C_1 = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad C_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}, \qquad C_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}$$

be a Chevalley basis, we have

$$\mathrm{expm}(\widehat{\boldsymbol{x}}) = \mathrm{expm}(\alpha_1 C_1)\,\mathrm{expm}(\alpha_2 C_2)\,\mathrm{expm}(\alpha_3 C_3),$$

where

$$\begin{aligned} \alpha_1 &= -\tan^{-1}\frac{x_3\theta^{-1}\sin\theta + x_1x_2\theta^{-2}(1-\cos\theta)}{\cos\theta + x_1^2\theta^{-2}(1-\cos\theta)}, \\ \alpha_2 &= -\sin^{-1}[x_2\theta^{-1}\sin\theta + x_1x_3\theta^{-2}(1-\cos\theta)], \\ \alpha_3 &= -\tan^{-1}\frac{x_1\theta^{-1}\sin\theta + x_2x_3\theta^{-2}(1-\cos\theta)}{\cos\theta + x_3^2\theta^{-2}(1-\cos\theta)}. \end{aligned} \tag{B.16}$$

Different ordering of C_1, C_2, C_3 leads to similar formulae. Note that, to avoid loss of significant digits for small θ, it is convenient to implement $1 - \cos\theta$, a recurring factor in $(B.16)$, using an angle-doubling formulae. The outcome,

$$1 - \cos\theta = 2\sin^2\vartheta,$$

is more stable when numerical methods are implemented with small step size.