Lie group methods for rigid body dynamics and time integration on manifolds *

Elena Celledoni[†] Brynjulf Owren[‡]

December 14, 1999

Abstract

Recently there has been an increasing interest in time integrators for ordinary differential equations which use Lie group actions as a primitive in the design of the methods. These methods are usually phrased in an abstract sense for arbitrary Lie groups and actions. We show here how the methods look when applied to the rigid body equations in particular and indicate how the methods work in general. An important part of the Lie group methods involves the computation of a coordinate map and its derivative. Various options are available, and they vary in cost, accuracy and ability to approximately conserve invariants. We discuss how the computation of these maps can be optimized for the rigid body case, and we provide numerical experiments which give an idea of the performance of Lie group methods compared to other known integration schemes.

AMS Subject Classification: 65L05

Key Words: time integration, geometric integration, numerical integration of ordinary differential equations on manifolds, numerical analysis, Lie algebras, Lie groups.

1 Introduction

The configuration space of a rigid body is $\mathrm{SO}(3,\mathbb{R})$, the orthogonal group of dimension 3. This Lie group is commonly modeled by means of 3×3 orthogonal matrices with unit determinant, usually called rotations. The dynamics of the rigid body can be formulated as a system of differential equations in the 9 elements of the rotation matrix. It is well known that in applying classical approximation methods directly to this system, the inherent dependency between the matrix elements caused by the orthogonality conditions are generally not preserved by the approximations, thus even after one single time step the approximation may no longer be an orthogonal matrix. There is a variety of possible remedies for this problem, there are several classes of orthogonality preserving methods. First, there are classical type integration methods which preserve all quadratic invariants. These include for instance those Runge-Kutta methods for which the stability function $R(z)$ satisfies $R(z)\cdot R(-z)=1$. The methods based on Gauss-Legendre quadrature, like the midpoint rule, are particular examples. It is well known that such methods must be implicit. Secondly, one has the methods that are

*This work was in part sponsored by The Norwegian Research Council under contract no. 111038/410, through the SYNODE project. WWW: `http://www.math.ntnu.no/num/synode/`

[†]Email: `Elena.Celledoni@math.ntnu.no`, WWW: `http://www.math.ntnu.no/~elenac/`

[‡]Email: `Brynjulf.Owren@math.ntnu.no`, WWW: `http://www.math.ntnu.no/~bryn/`

based on projection. One can view these methods as consisting of two substeps, the first step is done by some integration method, which does not necessarily respect the orthogonality constraint, but which approximates the exact solution up to a certain order in the step size Δt. Then the result is projected onto a nearby orthogonal matrix. Typical projectors are obtained through the QR or the polar decomposition methods. The QR method factors a matrix into an orthogonal matrix Q and an upper-triangular method R, and the projection method keeps the orthogonal factor Q. The third class of methods are the intrinsic ones, where the orthogonality is ensured by the format of the integration method, for example one may take the approximation at each time step to be the matrix exponential of some skew-symmetric matrix. This type of methods have been known for some time, see for instance Lewis and Simo [13], Simo and Wong [27] studied orthogonal integration in connection with rigid body dynamics, and Crouch and Grossman gave a more general approach based on rigid frames [6]. McLachlan [17] and Reich [26] both proposed a method for the rigid body based on the exponential mapping combined with splitting.

As an attempt of extending some of the previous ideas to a more general setting, there has been recently a remarkable interest in the literature for more systematic studies regarding integration methods on Lie groups [19], [25], [12]. Assuming that the initial value problem is defined on a Lie group might seem at first glance restrictive. The Lie group setting can however be rephrased successfully in a more effective end natural framework using the language of Lie group actions on manifolds. This framework allows to consider a large variety of significant problems and applications and in the present paper we illustrate this setting considering the example of the rigid body equations. The foundations of this approach and a prototype method of integration on manifolds based on the use of Lie group actions, appeared originally in a paper by Munthe-Kaas [21] and they have been further developed and enriched by various authors, [9], [24].

In this paper we will give an overview of this latter type of integration methods. They are usually being phrased in a rather general framework. Here we will introduce the reader to the general ideas of this approach, but we will apply the methods in particular to the rigid body equations, thereby choosing the Lie group SO(3) as our "case". We will start by presenting the rigid body equations from a generalized viewpoint, mostly by quoting ideas from the recent text by Arnold and Khesin [2] such that several important evolution equations both ordinary and partial differential equations fit within the framework. We always pay close attention to how the Euler equations for the rigid body appear in this framework. Next, we will explain the idea behind the Munthe-Kaas methods, and see in particular how they look when applied to the rigid body equations. The Munthe-Kaas approach can be made more flexible allowing the freedom of choosing a coordinate map, and in Section 4 we will discuss 3 different ones, and their implementation for the Lie algebra $\mathfrak{so}(3)$. Finally, in Section 5 we present some numerical experiments which demonstrate the properties of the presented methods and illustrate how different coordinate maps can affect both qualitative behaviour (e.g. energy preservation) and computational costs.

2 The rigid body equations from a general viewpoint

A point on the rigid body may be described in coordinates relative to the body, say X or relative to an axis system fixed in space, $x(t)$. The transformation between the two coordinate systems, $g(t)$ can thus be modeled as a 3×3 orthogonal matrix, $g(t)$ such that at any time

t, one has $x(t) = g(t) \cdot X$. The velocity is readily computed as

$$\dot{x}(t) = \dot{g}(t) \cdot X = \left(\dot{g} \cdot g^{-1}\right) \cdot x(t) := \omega_s(t) \cdot x(t) \tag{1}$$

The matrix $\omega_s(t)$ is skew-symmetric, as follows easily by differentiating the expression $g \cdot g^T = \mathbb{1}$ with respect to t. Equation (1) is an example of *right trivialization*, the tangent vector at g, $\dot{g} \in T_g\mathrm{SO}(3)$ is represented by ω_s which can be interpreted as a tangent at the identity matrix $\mathbb{1}$, that is, $\omega_s \in T_{\mathbb{1}}\mathrm{SO}(3)$. By furnishing the linear space $T_{\mathbb{1}}\mathrm{SO}(3)$ by the matrix commutator, say $[u, v] = u \cdot v - v \cdot u$ we obtain $\mathfrak{so}(3)$, *the Lie algebra* associated with the Lie group $\mathrm{SO}(3)$; $\mathfrak{so}(3)$ is the 3-dimensional vector space of 3×3 skew-symmetric matrices. We interpret $\omega_s \in \mathfrak{so}(3)$ as the *angular velocity in space coordinates.* It is convenient to express elements in $\mathfrak{so}(3)$ with respect to a basis. In particular, it is usual to make the following identification

$$\omega_s \longleftrightarrow \widehat{\omega}_s, \qquad \begin{bmatrix} 0 & -\omega_{s3} & \omega_{s2} \\ \omega_{s3} & 0 & -\omega_{s1} \\ -\omega_{s2} & \omega_{s1} & 0 \end{bmatrix} \longleftrightarrow \begin{bmatrix} \omega_{s1} \\ \omega_{s2} \\ \omega_{s3} \end{bmatrix}. \tag{2}$$

Thus, in what follows, we put a $\widehat{}$ on elements of $\mathfrak{so}(3)$ when it is useful to signify that they are to be thought of as vectors in $\mathbb{R}^3$.

Alternatively, one can transform the tangent (velocity) vector $\dot{x}(t)$ back to the original body coordinates. Suppressing the time dependence, we obtain another skew-symmetric matrix $\omega_c \in \mathfrak{so}(3)$ from the relation

$$g^{-1} \cdot \dot{x} = \left(g^{-1} \cdot \dot{g}\right) X = \omega_c \cdot X \quad \text{with} \quad \omega_c = g^{-1} \cdot \dot{g}.$$

This is in Lie group terms a *left trivialization* of the tangent vector $\dot{g}$ and we notice that the relation between ω_s and ω_c is $\omega_s = g \cdot \omega_c \cdot g^{-1}$. In the Lie group setting, we write $\omega_s = \mathrm{Ad}_g(\omega_c)$, and the map $g \mapsto \mathrm{Ad}_g$ is called the *adjoint representation* of the Lie group. By using the $\mathbb{R}^3$ representation of $\mathfrak{so}(3)$, one finds that $\mathrm{Ad}_g(u)$ is simply the matrix-vector product $g \cdot \widehat{u}$.

The kinetic energy of a rigid body can be given in terms of the angular velocity in body coordinates as follows

$$E = \frac{1}{2} \widehat{\omega}_c^T \cdot J \cdot \widehat{\omega}_c$$

Here J is the inertia tensor relative to the body, and one can always find a body coordinate system such that J is diagonal and positive. Alternatively one may use space coordinates and write $E = \widehat{\omega}_s^T\, g^T J\, g\, \widehat{\omega}_s^T$. Considering again a more general notation, one can think of the energy as an inner product $\langle \cdot, \cdot \rangle$ on the Lie algebra $\mathfrak{so}(3)$, and one then readily defines the *angular momentum* m_c in body coordinates as an element of the dual space $\mathfrak{so}(3)^*$ through the natural identification of $\mathfrak{so}(3)$ with $\mathfrak{so}(3)^*$ induced by the inner product. Thus for any $u \in \mathfrak{so}(3)$ we set

$$m_c(u) = \langle \omega_c, u \rangle$$

and in particular $E = \frac{1}{2} m_c(\omega_c)$. The inertia tensor J is thus a mapping from $\mathfrak{so}(3)$ to $\mathfrak{so}(3)^*$, and by using the natural bases for $\mathfrak{so}(3)$ and $\mathfrak{so}(3)^*$ induced by the identification (2) we may write $m_c = J \cdot \widehat{\omega}_c$. Since the energy is independent of the coordinate system, we may now compute for the angular momentum in space coordinates

$$E = \frac{1}{2} m_c(\omega_c) = \frac{1}{2} m_s(\omega_s) = \frac{1}{2} m_s(\mathrm{Ad}_g(\omega_c)) = (\mathrm{Ad}_g^* m_s)(\omega_c)$$

thus we have found that

$$m_c = \mathrm{Ad}^*_g m_s \tag{3}$$

where the * denotes the dual operator. Again, by using the above mentioned bases, we can deduce that $\mathrm{Ad}^*_g m_s = g^{-1} \cdot m_s$. The map $g \mapsto \mathrm{Ad}^*_g$ is called the *coadjoint representation* of the Lie group. By applying the principle of least action to the above energy definition, one obtains the differential equations for m_s and m_c due to Euler (see also [1])

1. The quantity m_s is preserved under motion when there are no external forces,

$$\frac{dm_s}{dt} = 0. \tag{4}$$

2. The angular momentum in body coordinates satisfies

$$\frac{dm_c}{dt} = m_c \times \widehat{\omega}_c = m_c \times J^{-1} \cdot m_c = -\omega_c \cdot m_c \tag{5}$$

We copy the commutative diagram in Figure 1 from the text by Arnold and Khesin [2] which is valid in our case for $\mathfrak{g} = \mathfrak{so}(3)$ and $G = \mathrm{SO}(3)$.

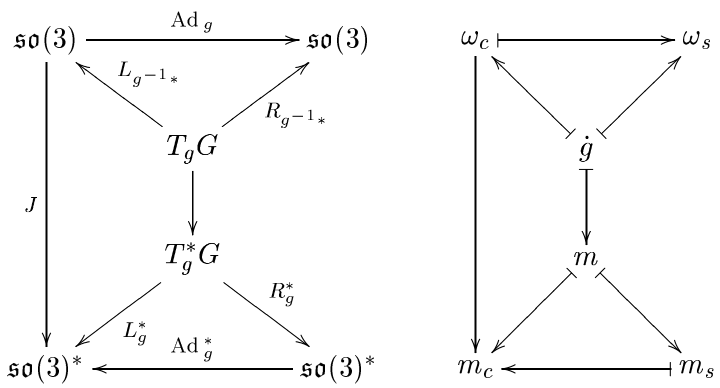

Figure 1: Commutative diagram for the generalized rigid body equations

The operators L_{h*} and R_{h*} are here the *trivialization operators* often denoted the derivative of the left (right) translation map. In the language of matrices, one has $L_{h*}(v) = h \cdot v$ and $R_{h*}(v) = v \cdot h$ for each $v \in \mathfrak{g}$.

Our reasons for carrying along the more abstract language of Lie groups in introducing the rigid body equations, is that we wish to point out that the integration methods discussed here can be applied in a much more general setting than the rigid body. An important observation is that the diagram in Figure 1 makes sense whenever

- The configuration of our physical system can be interpreted as a curve evolving in a Lie group G.
- We have an inner product (physically, an energy) as a left invariant metric on G induced by an inner product on the Lie algebra $\mathfrak{g}$ corresponding to G.

- There is a *principle of least action* that can be used to derive the equations obeyed by the system, this principle says that the configuration $g(t)$ is a geodesic on G. This can be used to generalize the Euler equations (4) and (5).

Equation (4) is of course the same in the general case, whereas in (5) we have made assumptions that hold only for $\mathfrak{so}(3)$. But by using the commutator $[\cdot,\cdot]$ which is defined for any Lie algebra $\mathfrak{g}$, one can, for any $u \in \mathfrak{g}$, define the adjoint operator $\mathrm{ad}_u : \mathfrak{g} \to \mathfrak{g}$ by $\mathrm{ad}_u(v) = [u, v]$. The dual ad_u^* of this operator can be used to define the second Euler equation corresponding to (5) in general as

$$\frac{dm_c}{dt} = \mathrm{ad}_{\omega_c}^*(m_c), \quad \omega_c = J^{-1}m_c$$

For any $m \in \mathfrak{g}^*$ we can define the coadjoint orbits $\mathcal{O}_m^* \subset \mathfrak{g}^*$ as

$$\mathcal{O}_m^* = \{p \in \mathfrak{g}^* : p = \mathrm{Ad}_g^*(m),\ g \in G\}$$

From (4) and (3) we see that the body angular momentum m_c remains in the same coadjoint orbit for all times. For the rigid body, we used the representation $\mathrm{Ad}_g^*(m) = g^{-1}m$ ($m \in \mathbb{R}^3$), and since every g^{-1} is orthogonal, it follows that the coadjoint orbits are spherical shells in $\mathbb{R}^3$. Comparing to the energy expressed in terms of m_c, we have in coordinates $E = \frac{1}{2}{m_c}^T J^{-1} m_c$ whose level surfaces are ellipsoids. The angular momentum of an ellipsoid thus evolves on the intersection of an ellipsoid and a sphere in $\mathbb{R}^3$.

Another example to be considered in the same framework is the Euler hydrodynamic equations of an ideal incompressible fluid on a manifold, in three dimensions it can be written in the form

$$\frac{\partial v}{\partial t} = v \times \mathrm{curl}\, v - \mathrm{grad}\, p$$

where v is the velocity field, satisfying $\mathrm{div}\, v = 0$ and p is the hydrostatic pressure. To go from the rigid body equation to these hydrodynamic equations, one needs to replace the Lie algebra $\mathfrak{so}(3)$ by the infinite dimensional Lie algebra $\mathrm{S\,Vect}\, M$, the space of divergence free vector fields, tangent to the boundary of M and with Lie bracket given by the Lie-Poisson bracket of vector fields (see e.g. [23]). The energy of the fluid is now defined as

$$E = \frac{1}{2} \iint\limits_M v \cdot v\ \mu$$

where μ is a volume form on M. With these definitions, the above diagram still makes sense, and the principle of least action can be used to derive the Euler equations. Of course, in this case it is necessary to introduce a finite dimensional discretization of the Lie algebra in order to implement the methods described in this paper.

3 Integration methods for the rigid body equations

A common feature of most numerical integration methods for ODEs, is that they *respect linear structure* in the following sense. Suppose that the solution of a system of differential equations at time t is denoted $x(t)$, a vector in an m-dimensional linear space V. But suppose it is known that $x(t)$ evolves in a linear subspace $U \subset V$, in other words $\dot{x} = F(t, x)$ with $x(0) \in U$ and $F(t, x) \in U$ for all $t \in \mathbb{R}$ and $x \in U$. Most integration methods obtain new vectors from the ones already computed by making use of two primitive operations:

- Make function evaluations (evaluate $F(t, x)$).

- Form linear combinations of previously obtained vectors.

With this observation, a simple induction argument shows that integration methods respect linear structure, meaning in the above notation that the approximations $x_i \approx x(t_i)$, $i = 0, 1, \dots$, all belong to the subspace U.

We refer to the subgroups of $\mathrm{GL}(n)$, the set of all invertible $n \times n$ matrices, as matricial Lie groups. It is often convenient to model equations on matricial Lie groups (like the rigid body equations) by using $n \times n$ matrices. However, a matrical Lie group rarely forms a linear subspace of $\mathbb{R}^{n\times n}$. Thus, when an integration method is applied in such a setting, there is no guarantee that the numerical approximation will remain on the group even after one single time step. For the rigid body, the Lie group $\mathrm{SO}(3)$ is a submanifold of dimension 3 in $\mathbb{R}^{3\times 3}$.

The main idea of Munthe-Kaas is to introduce new variables such that the equations on the group (or more generally, the manifold) are transformed locally to a linear space. Thus, most integration methods will produce approximations in this linear space, and can be mapped back via the inverse transformation to the group (or manifold).

Generally, we suppose that a Lie group G is acting on a manifold M through a group action $\Lambda : G \times M \to M$. We denote by $\mathfrak{g}$ the Lie algebra corresponding to G, i.e. the tangent space at the identity element, furnished with a Lie bracket. In the case of the rigid body, we have $G = \mathrm{SO}(3)$ the orthogonal group, and $\mathfrak{g} = \mathfrak{so}(3)$ the set of 3×3 skew-symmetric matrices, the bracket being the matrix commutator. Next, one needs to introduce a local (smooth) coordinate map, $\Psi : \mathfrak{g} \to G$ and represent locally the solution $y(t)$ near a point $p \in M$ as a curve $u(t)$ near $0 \in \mathfrak{g}$ through

$$y(t) = \Lambda(\Psi(u(t)), p) := \lambda(u(t), p). \tag{6}$$

For instance, in the rigid body setting, we would need a Ψ which maps skew-symmetric matrices into orthogonal ones, and a typical Λ could be $\Lambda(g, p) = g \cdot p$. Hence, the solution $g(t)$ might be expressed locally by $g(t) = \Psi(u(t)) \cdot p$ near the point (orthogonal matrix) $p \in \mathrm{SO}(3)$.

It is useful, and it causes no loss of generality, to impose the conditions $\Psi(0) = \mathbb{1}$ and $\Psi_0' = \mathbb{1}_{\mathfrak{g}}$. Note that the latter conditon and the smoothness of Ψ ensures that Ψ_u' is invertible if u belongs to some neighborhood of 0 in $\mathfrak{g}$. It is worth while to note that Munthe-Kaas in his initial papers [19, 20, 21] always took Ψ to be the exponential map, defined for matrices as

$$\exp(u) = \sum_{k=0}^{\infty} \frac{u^k}{k!} \tag{7}$$

But several authors [27, 8, 7, 11, 16, 24] have used other coordinate maps in order to obtain methods of less computational cost.

In order to derive the differential equation for u in (6), we use again the rigid body equations as an example. We may then start with (5) which is phrased as an equation on $\mathfrak{so}(3)^*$. Knowing that m_c evolves on the coadjoint orbits, it seems natural to represent the solution $m_c(t)$ by means of the group action

$$m_c(t) = \Lambda(g(t), m_s) = \mathrm{Ad}^*_{g(t)}(m_s) = g^{-1}(t) \cdot m_s, \quad g(t) \in \mathrm{SO}(3), \; m_s \in \mathfrak{so}(3)^*.$$

We differentiate to obtain

$$\dot{m}_c = -g^{-1} \cdot \dot{g} \cdot g^{-1} \cdot m_s$$

keeping in mind that m_s is constant by (4). So far, we have replaced the unknown $m_c(t) \in \mathfrak{so}(3)^*$ by the variable $g(t) \in \mathrm{SO}(3)$. Next, we make another change of variable which is valid only locally, namely, near the point $p \in \mathrm{SO}(3)$ we may use one of the two representations

$$\begin{aligned} g(t) &= \Psi(u(t)) \cdot p, \quad u(t) \in \mathfrak{so}(3) &(8)\\ g(t) &= p \cdot \Psi(\tilde{u}(t)) \quad \tilde{u}(t) \in \mathfrak{so}(3) &(9) \end{aligned}$$

We proceed with the former (8). We need to introduce the *right trivialized derivative* $d\Psi_u$ of Ψ at $u \in \mathfrak{so}(3)$, defined through

$$d\Psi_u(v) \cdot \Psi(u) = \frac{d}{d\tau}\Big|_{\tau=0} \Psi(u + \tau v), \;\; v \in \mathfrak{so}(3)$$

The main motivation for working with $d\Psi_u$ rather than $\Psi'_u(\cdot) = d\Psi_u(\cdot) \cdot \Psi(u)$ is that it simplifies the expressions and also that $d\Psi_u$ maps $\mathfrak{so}(3)$ to $\mathfrak{so}(3)$. So if we use the identification of $\mathfrak{so}(3)$ with $\mathbb{R}^3$ suggested in (2), $d\Psi_u$ takes the form of a 3×3 matrix whose elements depend on u.

For a curve $u(t)$ having tangent $\dot{u}(t)$ we get $\frac{d}{dt}\Psi(u) = d\Psi_u(\dot{u})\Psi(u)$, and we can deduce that

$$\dot{m}_c = -p^{-1}\,\Psi(u)^{-1}\,d\Psi_u(\dot{u})\,m_s = -\omega_c\,p^{-1}\,\Psi(u)^{-1}\,m_s$$

where for the second equality we have used (5). We have in other words two time dependent matrices whose respective actions on the constant vector m_s are the same. Thus it is sufficient to equate the two matrices, this leads to

$$d\Psi_u(\dot{u}) = \mathrm{Ad}_{\,\Psi(u)p}\,\omega_c = \Psi(u)\;p\;\widehat{\omega}_c$$

with a slight abuse of notation. Substituting $\widehat{\omega}_c = J^{-1}m_c = J^{-1}\,p^{-1}\,\Psi(u)^{-1}\,m_s$ we get

$$\dot{u} = d\Psi_u^{-1}\left(\Psi(u)\,p\,J^{-1}\,p^{-1}\,\Psi(u)^{-1}\,m_s\right) \tag{10}$$

In the next section we will give details on how $d\Psi_u^{-1}$ can be explicitly written as a 3×3 matrix for various choices of $\Psi(u)$.

Note that (10) can be written in the form $\dot{u} = d\Psi_u^{-1}(f(\lambda_p(u)))$ where

$$f(g) = g\,J^{-1}\,g^{-1}\,m_s \quad \text{and} \quad \lambda_p(u) = \Psi(u) \cdot p. \tag{11}$$

From the discussion of the previous section, we have that $f(g) = \omega_s$, thus by the very definition $\omega_s = \dot{g} \cdot g^{-1}$ we can think of (10) as originating from the differential equation on the group $\mathrm{SO}(3)$

$$\dot{g} = \omega_s \cdot g = f(g) \cdot g \tag{12}$$

after performing the change of variables (8). As a matter of fact, many authors take (12) as a generic equation to be solved by Lie group integrators [12], [28], [8].

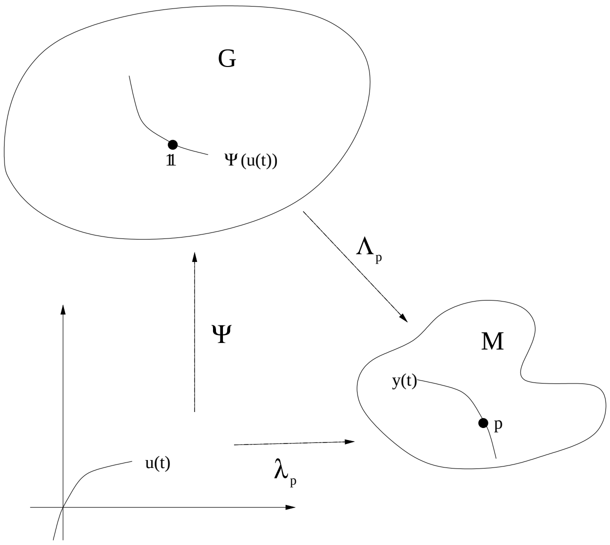

Figure 2: The connection between $\mathfrak{g}$, G and M

The Lie group integrators produce approximations $g_1, g_2, \ldots$ in SO(3), setting in each time step $p = g_n$ and assigning the initial value $u(t_n) = 0$ in (10). Then one step of a standard integration procedure is applied with time step Δt to this initial value problem, yielding an approximation v to $u(t_{n+1})$. We finally set $g_{n+1} = \Psi(v) \cdot g_n = \Psi(v) \cdot p$.

In [24] we have derived the differential equation for u for an arbitrary Lie group action, and an arbitrary coordinate map. This is an elementary modification of a result by Munthe-Kaas in [21]. A crucial assumption is that the vector field F of the ODE on M can be expressed at each point as a transformed tangent on the Lie algebra $\mathfrak{g}$ by means of the group action. Or, in precise terms, for each $y \in M$ there must be a $v_y \in \mathfrak{g}$ such that

$$F(y) = \left.\frac{d}{dt}\right|_{t=0} \Lambda(\Psi(tv_y), y) := \lambda_*(v_y)(y) \tag{13}$$

So the generalization of the function f defined in (10) for the rigid body equations is precisely $f(y) = v_y$.

An illustration of the connection between $G, \mathfrak{g}$ and M in general is given in Figure 2. For a fixed point $p \in M$, one works locally with the maps $\Lambda_p : \mathfrak{g} \to M$ and $\lambda_p : \mathfrak{g} \to M$. The presentation of the ODE in the form (13) is ensured as long as λ_p is locally onto a neighborhood of $p \in \mathcal{M}$. Thus, locally around a point $p \in M$ we transform the solution curve $y(t)$ on M, via a curve through the identity element on G to a curve $u(t)$ through $0 \in \mathfrak{g}$. We may now solve the resulting local initial value problem

$$\dot{u} = d\Psi_u^{-1}\big((f \circ \lambda_p)(u)\big) := \tilde{f}(u), \quad u(0) = 0.$$

by any numerical integration method, and then the result is transformed back to M via the map λ_p. We note now that in addition to the problem specific function f, there are the two maps Ψ and $d\Psi_u^{-1}$ which appear as a part of the differential equations to be solved. Many of the recently developed Lie group methods differ in the choice of Ψ and in the next section we will discuss some of the most used coordinate maps, their generic definition, and also how

Ψ and $d\Psi_u^{-1}$ can be computed efficiently for the case $\mathfrak{so}(3)$ corresponding to the rigid body equations.

Vi conclude this Section by presenting in algorithmic form, one step of a Lie group method based on the classical 4th order Runge-Kutta method, see e.g. [10, p. 138]. This particular version works for any matricial differential equation of the form $\dot{g} = f(g) \cdot g$. For the rigid body equations the algorithm is applied with the function f given in (11)

Given g_n
$k_1 = f(g_n)$ $u_2 = \frac{\Delta t}{2} k_1, \quad k_2 = d\Psi_{u_2}^{-1}\left(f(\Psi(u_2) \cdot g_n)\right)$ $u_3 = \frac{\Delta t}{2} k_2, \quad k_3 = d\Psi_{u_3}^{-1}\left(f(\Psi(u_3) \cdot g_n)\right)$ $u_4 = \Delta t\, k_3, \quad k_4 = d\Psi_{u_4}^{-1}\left(f(\Psi(u_4) \cdot g_n)\right)$
$v = \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4)$
$g_{n+1} = \Psi(v) \cdot g_n$

Table 1: The algorithm for taking one step with a Lie group method

4 A variety of coordinate maps.

4.1 The exponential map

For any Lie group G with corresponding Lie algebra $\mathfrak{g}$ the exponential map is defined as $\exp : \mathfrak{g} \to G$. Its definition can be given independently of the group representation, through flows of (left) right invariant vector fields. However, in most applications that we know, a matrix representation is chosen for $\mathfrak{g}$, and then the formula (7) can be taken as the definition. Unfortunately, the exponential function cannot be expressed exactly for matrices in a finite number of operations with elementary functions. There are a variety of approximate methods, see [18] for a summary. In order to avoid defeating the purpose of the integration methods we discuss here, it is of importance that the approximation we take, belong to G (within machine accuracy). For instance, Padé approximants of high order can be used, but the computational cost will in a typical floating point environment be of the order around 25–30 n^3 for an $n \times n$ matrix. However for the special case $\mathfrak{so}(3)$ the Rodrigues formula has been known for a long time

$$\exp(u) = \mathbb{1} + \frac{\sin\alpha}{\alpha} u + \frac{1 - \cos\alpha}{\alpha^2} u^2, \quad \alpha = \sqrt{u_1^2 + u_2^2 + u_3^2}, \quad u \in \mathfrak{so}(3) \tag{14}$$

Here $\widehat{u} = (u_1, u_2, u_3)$ are the parameters of the skew-symmetric matrix u, as in (2).

As far as we know, a generic expression for $d\exp_u^{-1}$ was first found by Baker [3], it was given as an series expansion in the operator ad_u.

$$d\exp_u^{-1} = \sum_{k=0}^{\infty} \frac{B_k}{k!} \mathrm{ad}_u^k \tag{15}$$

where the powers of ad_u are obtained by composition, $\mathrm{ad}_u^k = \mathrm{ad}_u \circ \mathrm{ad}_u^{k-1}$, $\mathrm{ad}_u^0 = \mathbb{1}_{\mathfrak{g}}$. The B_k's are the Bernoulli numbers, defined through the function $h(z) = \dfrac{z}{\exp(z)-1} = \sum_k \dfrac{B_k}{k!} z^k$. One easily checks that $B_{2k+1} = 0$ whenever $k \geq 1$. The formula (15) may seem to be of little use, but there are two reasons why this is not so.

1. Since the iterated commutators $\mathrm{ad}_u^k(v) \in \mathfrak{g}$ for every $k \geq 0$, each term and thereby the truncated series after q terms, by Munthe-Kaas denoted $\mathtt{dexpinv}(u, v, q)$ is an element of the Lie algebra $\mathfrak{g}$. The map Ψ is therefore well defined on elements $w = \mathtt{dexpinv}(u, v, q)$.

2. In formulas like the Runge-Kutta method, one finds that the u to be inserted in $d\exp_u^{-1}(v)$ always satisfies $u = \mathcal{O}(h)$ where h is the time step. Thus $\mathrm{ad}_u^k = \mathcal{O}(h^k)$ and one may discard terms from the series of higher order than the integration method itself.

Munthe-Kaas and Owren [22] have found ways to make the number of commutators even smaller than what comes from the crude discussion above by applying techniques from the theory of *free Lie algebras.*

Coming back to $\mathfrak{so}(3)$ once again, it is possible to derive an exact expression for the map $d\exp_u^{-1}$, one obtains the formula

$$d\exp_u^{-1} = \mathbb{1} - \frac{1}{2}u + \frac{1 - \frac{\alpha}{2}\cot\frac{\alpha}{2}}{\alpha^2} u^2 \tag{16}$$

Here we have represented $d\exp_u^{-1}$ with respect the basis of (2) for $\mathfrak{so}(3)$, $\mathbb{1}$ is simply the 3×3 identity matrix, whereas $u \in \mathfrak{so}(3)$ is to be understood as a skew-symmetric matrix.

4.2 Canonical coordinates of the second kind

The exponential map is sometimes called *canonical coordinates of the first kind* for the Lie group G in question. If $e_1, \ldots, e_d$ is a basis of a d dimensional Lie algebra $\mathfrak{g}$, one can think of the coordinates $v_1, \ldots, v_d$ as describing the point $\exp(v_1 e_1 + \cdots + v_d e_d)$ on the corresponding Lie group G. Similarly, one can define *canonical coordinates of the second kind* with respect to a basis as the map

$$\mathtt{ccsk} : \sum_{i=1}^{d} v_i e_i \mapsto \exp(v_1 e_1)\exp(v_2 e_2)\cdots\exp(v_d e_d)$$

This map is a local diffeomorphism in a neighborhood of $0 \in \mathfrak{g}$. At first glance, it may seem peculiar from a computational point of view to replace *one* exponential by several of them. However, in a matrix representation, one may for instance choose basis elements where typically 1 or 2 elements are nonzero, thus the matrix exponential is given explicitly, typically as a low-rank update of the identity matrix. Furthermore, these elementary exponentials can be multiplied together through a series of computationally inexpensive updates.

It is quite straightforward to derive the following somewhat complicated expression for $d\mathtt{ccsk}_u^{-1}$, where $u = \sum_i u_i e_i$

$$d\mathtt{ccsk}_u\left(\sum_{i=1}^{d} v_i e_i\right) = \sum_{i=1}^{d} v_i B_i(e_i)$$

where

$$B_1 = \mathbb{1}, \quad B_i = \mathrm{Ad}_{e^{u_1 e_1}} \cdots \mathrm{Ad}_{e^{u_{i-1} e_{i-1}}}, \; i \geq 2$$

Here the operator $\mathrm{Ad}_{e^{u_k e_k}}$ is the same as the one appearing in Section 2, for matrix Lie algebras it is defined as conjugation, i.e. $\mathrm{Ad}_{e^{u_k e_k}}(v) = e^{u_k e_k} \cdot v \cdot e^{-u_k e_k}$

Clearly one also needs to invert the operator $d\mathtt{ccsk}_u$ and this may seem like a difficult and computationally expensive task. However, in [24] explicit expressions for $d\mathtt{ccsk}_u^{-1}$ are derived for several Lie algebras with particular ordered bases. In many cases, it leads to less expensive methods than those based on the exponential map.

We return now to the rigid body equations, and we use as a basis for $\mathfrak{so}(3)$ the matrices which correspond to the canonical unit vectors in the $\widehat{\ }$ representation.

$$e_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, \quad e_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}, \quad e_3 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{17}$$

The exponential of these matrices corresponds to rotation around the principal axes in the coordinate system, for instance

$$e^{u_1 e_1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos u_1 & -\sin u_1 \\ 0 & \sin u_1 & \cos u_1 \end{bmatrix}$$

We may express $d\mathtt{ccsk}_u^{-1} : \mathfrak{so}(3) \to \mathfrak{so}(3)$ as a 3×3 matrix with respect to this basis

$$d\mathtt{ccsk}_u^{-1} = \begin{bmatrix} 1 & \sin u_1 \tan u_2 & -\cos u_1 \tan u_2 \\ 0 & \cos u_1 & \sin u_1 \\ 0 & -\sin u_1 \sec u_2 & \cos u_1 \sec u_2 \end{bmatrix} \tag{18}$$

4.3 The Cayley map

There is an important class of Lie groups which one can characterize through their matrix representation as follows. Let $M \in \mathrm{GL}(n)$ be an arbitrary invertible $n \times n$ matrix, and set

$$G_M = \{g \in \mathrm{GL}(n) : g^T M g = M\}$$

One can easily check that G_M is a Lie group, and in fact, one has the important examples $G_{\mathbb{1}} = \mathrm{SO}(n)$ as well as the symplectic group in even dimension, $n = 2m$ and

$$M = \begin{bmatrix} 0 & \mathbb{1} \\ -\mathbb{1} & 0 \end{bmatrix}$$

The Lie algebra $\mathfrak{g}_J$ which corresponds to G_J is the linear space of matrices

$$\mathfrak{g}_M = \{u \in \mathfrak{gl}(n) : u^T M + M u = 0\}$$

furnished with the matrix commutator as Lie bracket. Clearly, $\mathfrak{g}_{\mathbb{1}} = \mathfrak{so}(n)$ the skew-symmetric matrices. It was proved by Celledoni and Iserles [5] that any map $r(z)$, analytic at 0 and such that $r(z) \cdot r(-z) = 1$ will map $\mathfrak{g}_M$ into G_M, or in other words, $u \in \mathfrak{g}_M \Rightarrow r(u) \in G_M$. A particular example of such a function is the Cayley transform

$$\mathtt{cay}(u) = \left(\mathbb{1} - \frac{u}{2}\right)^{-1} \cdot \left(\mathbb{1} + \frac{u}{2}\right)$$

[1] has the property that it maps $\mathfrak{g}_M$ into G_M for any $M \in \mathrm{GL}(n)$. For the orthogonal case $G_{\mathbb{1}} = \mathrm{SO}(n)$, the Cayley transform has been used by many authors for devising integration methods, see for instance [15, 14, 8, 7, 11]. We also need the map $d\mathtt{cay}_u^{-1}$ which is easily derived as

$$d\mathtt{cay}_u^{-1}(v) = \left(\mathbb{1} - \frac{u}{2}\right) v \left(\mathbb{1} + \frac{u}{2}\right) = \mathbb{1} - \frac{1}{2}[u, v] - \frac{1}{4} u \cdot v \cdot u$$

In the case $\mathfrak{so}(3)$ we find that

$$\mathtt{cay}(u) = \mathbb{1} + \frac{1}{1 + \frac{1}{4}\alpha^2} \left(u + \frac{1}{2}u^2\right), \quad \alpha = \|\widehat{u}\|_2$$

and the matrix of $d\mathtt{cay}_u^{-1}$ with respect to the basis (17)

$$d\mathtt{cay}_u^{-1} = \left(1 + \frac{1}{4}\alpha^2\right) \mathbb{1} - \frac{1}{2}u + \frac{1}{4}u^2.$$

4.4 Computational cost

We summarize the cost of computing the various coordinate maps in the table below for the Lie algebra $\mathfrak{so}(3)$, using the expressions derived above for the three coordinate maps exp, `ccsk` and `cay`.

	$\Psi(u)$			$d\Psi_u^{-1}$			Both		
	exp	`ccsk`	`cay`	exp	`ccsk`	`cay`	exp	`ccsk`	`cay`
add	15	4	15	15	-	9	25	4	24
mult	17	14	20	18	6	9	29	20	23
$\sqrt{\cdot}$	1	-	-	1	-	-	1	-	-
trig	2	6	-	1	4	-	3	6	-
total	35	24	35	35	10	18	58	30	47

Table 2: Computational cost for the various coordinate maps

The rows of the table give the respective number of additions, multiplications, square roots, trigonometric functions, and the result of adding all these together. The multicolumn $\Psi(u)$ shows the cost of computing the coordinate map alone, $d\Psi_u^{-1}$ correspondingly the matrix representing its inverse differential alone. Note that in the last multicolumn "Both" is not the sum of the previous two. The reason for this is that one can take advantage of the fact that $\Psi(u)$ and $d\Psi_u^{-1}$ make use of common partial expressions. The `ccsk` map looks cheaper in total cost than the others, but one should keep in mind that it includes 6 trigonometric

[1]Historically, the Cayley transform has been defined without the factor $\frac{1}{2}$, i.e. $\mathtt{cay}(z) = (1+z)/(1-z)$, but we rather prefer the Padé $(1, 1)$ form above in order that $d\mathtt{cay}_0 = \mathbb{1}_g$

functions which are each equivalent to several additions or multiplications. For convenience of implementation, we always compute the matrix of $d\Psi_u^{-1}$ rather than directly the result $d\Psi_u^{-1}(v)$ as a vector in $\mathbb{R}^3$. So one has to take into account the 6 additions and 9 multiplications resulting from computing the matrix-vector product. This added cost is significant compared to the numbers in Table 2. For instance in the case of `ccsk`, we see from (18) that the sparsity of the first column may be used to lower this cost to 6 multiplications and 4 additions.

5 Numerical experiments

In this section we compare the performance of various Lie group methods implemented with different coordinate maps with other methods. For the comparison we considered four additional methods:

- the fourth order MacLachlan-Reich symplectic momentum preserving method (MR);
- the second order Simo-Wong implicit energy-momentum preserving method based on the use of the Cayley map (S);
- the fourth order Lie group method developed by Buss based on the use of the exponential map (B4);
- the classical fourth order Runge-Kutta method (RK) [10, p. 138].

We divide the set of experiments in two parts. We first consider the behaviour of the methods in terms of energy preservation and then the short time behaviour of the methods considering accuracy versus cost.

5.1 Short description of the comparison methods

The first method we consider for comparison was developed independently by McLachlan and Reich [17], [26]. It is a symplectic and momentum preserving method based on the use of splittings. The skew-symmetric matrix ω_c in

$$\frac{dm_c}{dt} = -\omega_c \cdot m_c, \qquad \omega_c = J^{-1} m_c,$$

is split in terms of the basis (17) of $\mathfrak{so}(3)$ as follows

$$\omega_c = \omega_{c1}\, e_1 + \omega_{c2}\, e_2 + \omega_{c3}\, e_3.$$

Each of the elementary flows $\exp(t\,\omega_{ci}\, e_i)$, $i = 1, 2, 3$ has an easily computable explicit expression and is obtained exactly. The symmetric composition *a' la* Strang splitting and Yoshida, of the three elementary flows computed with time $\alpha_i t$ scaled with suitable parameters α_i, gives second order, fourth order, and higher order approximations respectively. We remark here that different orderings of the elementary flows affect in a problem-dependent manner the behaviour of the method; we implemented the method with the ordering 123, where the indexes refer to the basis elements of $\mathfrak{so}(3)$.

The cost per step of the order 2 method is 6 trigonometric functions for the computation of the elementary flows and 4 matrix-matrix multiplications, which, exploiting the sparsity of the elementary flows, cost 18 flops each, in total 72 multiplications and additions and 6 trigonometric functions; in the fourth order method the elementary flows to be computed are 10, and the matrix-matrix multiplications are 12, in total 216 plus 20 trigonometric functions.

The second method we have considered is energy and momentum preserving, it is an implicit second order method developed by Simo and Wong in [27]. In this case the rotation and the body angular momentum at the $n+1$-th step of the method are given by

$$\begin{aligned} g_{n+1} &= g_n \mathtt{cay}(\zeta) \\ m_{c,n+1} &= \mathtt{cay}(-\zeta) m_{c,n} \end{aligned}$$

where $\zeta \in \mathfrak{so}(3)$ is computed in the Lie algebra in such a way that the energy preservation is fulfilled. More precisely this is achieved when

$$\zeta = k(\bar{\zeta})\bar{\zeta}, \qquad \bar{\zeta} = h/2 J^{-1}(m_{c,n+1} + m_{c,n}),$$

where $k : \mathbb{R}^3 \to \mathbb{R}$ is an arbitrary function and in the algorithm it is fixed to be

$$k(\bar{\zeta}) = \frac{\tan(1/2\bar{\zeta})}{1/2\|\zeta\|}.$$

The method has been shown to be energy-momentum preserving and symplectic for a particular different choice of the function k, [13]. The method is implicit in $m_{c,n+1}$ and there are various equivalent ways to implement the Newton iteration. In our implementation of the method, instead of considering directly the equation $m_{c,n+1} = \mathtt{cay}(-\zeta)m_{c,n}$ we applied the Newton iteration on the equivalent equation

$$\left(\mathbb{1} - \frac{\zeta(m_{c,n+1})}{2}\right) m_{c,n} - \left(\mathbb{1} + \frac{\zeta(m_{c,n+1})}{2}\right) m_{c,n+1} = 0.$$

Another more general approach for performing the Newton iteration on Lie groups that could lead to a lower computational cost, can be found in [25]. The cost per one Newton iteration in our implementation is about 160 flops and 1 trigonometric function, while the update of the rotation g_{n+1} requires the application of the Cayley map (18 flops), a matrix-matrix multiplication (54 flops), additional 17 flops and a trigonometric function, in total 249 flops and 2 trigonometric functions if we count a single Newton iteration per time step.

The third method is due to Buss and it has recently appeared in [4]. This method is similar to those described in this paper. The rotation g_n is updated by using the exponential map in the following way,

$$g_{n+1} = \exp(\bar{\omega}_s) g_n$$

and here

$$\bar{\omega}_s = \omega_{s,n} + h/2\omega_{s,n}' + h^2/6\omega_{s,n}'' + h^2/12[\omega_{s,n}', \omega_{s,n}] + h^3/24\omega_{s,n}''' + h^3/24[\omega_{s,n}'', \omega_{s,n}].$$

The derivatives of $\omega_{s,n}$ can be computed explicitly in terms of commutators of $\omega_{s,n}$ itself, m_s and the lower order derivatives of $\omega_{s,n}$ and can be derived by differentiating the relationship obtained from (11)

$$m_s = g\, J\, g^{-1}\, \omega_s,$$

[4]. The method requires 10 commutators, in the form of cross products they cost 9 flops each, one exponential map and one matrix-matrix multiplication, plus some extra flops due

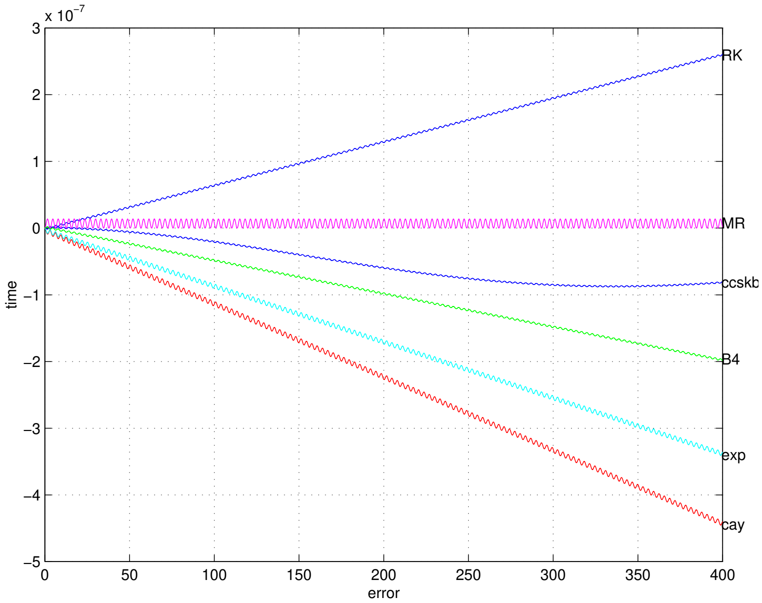

Figure 3: Energy plots with asymmetric inertia tensor

to additions of 3-dimensional vectors and some matrix-vector multiplications that in total amount in our implementation to 192 flops.

For more details on the three previous algorithms we refer the reader to the original articles.

The fourth and last comparison method is the classical fourth order Runge-Kutta method, see [10, p. 138].

Note that throughout the experiments we distinguish between three different implementations of the `ccsk` method: we indicate with `ccskf` the method implemented with ordering of the basis elements 123, with `ccskb` the method implemented with the ordering 321 and simply with `ccsk` the method that performs one step with `ccskf` and one with `ccskb`.

5.2 Energy experiments

The first two test examples are the same as in [13]. In the first example it is considered an asymmetric inertia tensor $J = \mathrm{diag}(1, 2, 3)$ and the initial angular velocity is fixed to be $\omega_{s,0} = [0.2, 0, 1]$. The integration is carried out on the interval $[0, 400]$ with step-size $\Delta t = 0.05$ for a total number of steps $N = 8000$. In figure 3 we plot the values of the energy error for the various methods.

The second experiment is analogous to the previous one, but we considered a symmetric inertia tensor $J = \mathrm{diag}(1, 1, 2)$, the initial angular velocity is the same as in the previous case. The integration is carried out on the interval $[0, 400]$ with $\Delta t = 0.05$. The energy errors are plotted in figure 4.

In these two first experiments we report only the results of the `ccsk` method implemented with ordering 321 (`ccskb`). In the third experiment the integration is carried out on the interval $[0, 400]$ with step-size $\Delta t = 0.0004$ for a total number of steps $N = 10^5$. In this example the inertia tensor is $J = \mathrm{diag}(1, 4, 18)$ with spatial angular momentum $m_s = [0, 2.75, 2.75]$.

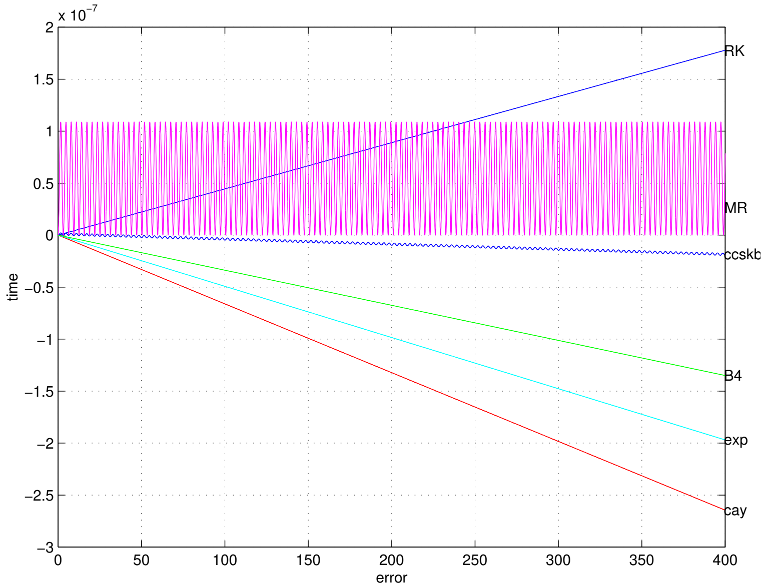

Figure 4: Energy plots with symmetric inertia tensor

	Energy error	Mean value
MR	1.0330e-05	8.5929e-04
exp	0.0300	0.0150
ccsk	0.2064	0.1026
ccskb	0.0089	0.0046
ccskf	0.4204	0.2094
cay	0.3339	0.1671
B4	0.1450	0.0726
RK4	0.5863	0.2958

Table 3: Energy error for the methods

The energy in this example is much bigger then in the previous cases. In Table 3 we report in the first column the values of the error in the energy at $t = 400$ and in the second column we report the mean value of the energy error.

The results given by the symplectic method are invariably better than the those we obtain with Lie group methods and the classical Runge-Kutta method. We can not claim numerical evidence that orthogonal integration, e.g. preservation of momentum, helps in getting better energy features, since most of the Lie group methods perform in our experiments only marginally better then the classical Runge-Kutta method. However we can observe that some of the methods present a clearly better energy behaviour than others (in these experiments ccskb for example). In the third experiment we reported the energy behaviour for the three versions of the ccsk method: while the ccskf performs almost as bad as the classical Runge-Kutta method and the ccsk behaves as the average of the Lie group techniques, the method ccskb is clearly the best Lie group method in terms of energy preservation. This suggests that there is something to gain in choosing the correct coordinate map also in terms of energy

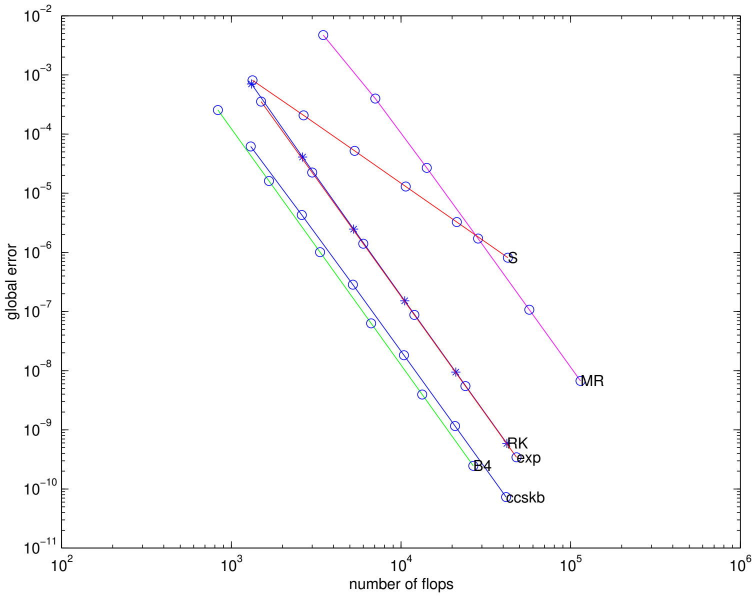

Figure 5: Computational costs. Symmetric inertia tensor

behaviour.

5.3 Accuracy tests

In this second part of the experiments we consider the behaviour of the methods in short time intervals. We compare computational cost to accuracy. The interval of integration is $[0, 1]$, we considered the symmetric inertia tensors $J = \text{diag}(1, 1, 2)$ and $J = \text{diag}(1, 4, 18)$ in the two experiments, the initial values are the same as in the energy experiments. We counted the total number of flops required from each method for performing a fixed number of steps N, for different increasing values of N ($N = 2^p$ and $p = 1, \ldots, 6$) and corresponding decreasing values of the global error. In figure 5, we plot number of flops versus global error for the first experiment.

The Lie group methods and the RungeKutta 4th order method perform clearly better then the energy preserving method of Simo-Wong and the syplectic McLachlan-Reich algorithm, that have optimal energy behaviour. It is fair to recall here that the method of Simo-Wong is a 2nd order method whereas all the others are 4th order methods. It is remarkable that some Lie group methods perform, in our experiments, even better than the explicit Runge-Kutta method that is not momentum-preserving.

In Table 4 we report the count of flops required for obtaining a prefixed accuracy ε, that in our case we fixed to be 10^{-3} and 10^{-6}. The first two columns of the table report the number of flops for the experiment with the symmetric inertia tensor $J = \text{diag}(1, 1, 2)$ and the third and the fourth column refer instead to the experiment with inertia tensor $J = \text{diag}(1, 4, 18)$.

The methods B4 and `ccskb` (ordering 321) are those that obtain better values of the global error in the first experiment and imply a lower number of flops, the `ccsk` presents however a better behaviour in the energy test (see figure 4). The method B4 seams to maintain its good features also in the second experiment trading them with a not too good energy behaviour,

	$J = \mathrm{diag}(1,1,2)$		$J = \mathrm{diag}(1,4,18)$	
	$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-6}$	$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-6}$
MR	$10^{3.7350}$	$10^{4.5078}$	$10^{5.3546}$	$10^{6.0545}$
S	$10^{3.0821}$	$10^{4.5864}$	$10^{6.3032}$	$10^{7.4649}$
exp	$10^{3.0635}$	$10^{3.8140}$	$10^{4.9248}$	$10^{5.6144}$
`ccsk`	$10^{3.0764}$	$10^{3.8278}$	$10^{5.1211}$	$10^{5.8551}$
`ccskb`	$10^{2.8129}$	$10^{3.5740}$	$10^{5.0725}$	$10^{5.8319}$
`ccskf`	$10^{3.1603}$	$10^{3.9074}$	$10^{5.2132}$	$10^{5.9964}$
`cay`	$10^{3.0539}$	$10^{3.8052}$	$10^{5.0890}$	$10^{5.7877}$
B4	$10^{2.7731}$	$10^{3.5237}$	$10^{4.7440}$	$10^{5.4732}$
RK4	$10^{3.0773}$	$10^{3.8207}$	$10^{5.2167}$	$10^{5.9541}$

Table 4: Number of flops vs global error for the methods

as shown by the values in Table 3 of the previous Section.

6 Conclusion

We have presented a selection of the recently developed Lie group methods and studied in particular their implementation for the rigid body equations. More generally, we have seen that on one hand, the rigid body equation is a prototype example of a evolutionary system that can be phrased by means of Lie group actions, and that a mathematical framework exists for treating such systems in a unified manner. Efficient approximation with good qualitative properties can be obtained by choosing the action in the right way. Secondly, we have seen that the Lie group methods can be applied, in particular to such generalized rigid body equations, but they have a much broader applicability than that. The performance of the Lie group methods depends strongly on which coordinate map is used. We have presented 3 of those, 2 which work for any Lie group and 1 that works for the particular class which includes the orthogonal and symplectic groups.

The numerical experiments show that these explicit Lie group integrators are competitive for integration over short and medium time intervals, but that the long term behaviour is better with methods which are either symplectic or conserve other quantities. It is not clear whether Lie group methods as presented here can be symplectic, but recently Zanna et al. [29] have constructed time reversible methods of this kind. They are implicit, and we have not yet found a sufficiently efficient implementation to make them competitive to the ones we presented here in the numerical experiments.

There are still many open problems to be addressed, both in determining what is a good action for a given problem, and in finding methods with good long time behaviour.

Acknowledgments

The authors are grateful to Arieh Iserles and Stig Faltinsen for many fruitful discussions during their stay at NTNU in the autumn 1999.

References

[1] V. I. Arnold. *Mathematical Methods of Classical Mechanics.* Springer-Verlag, GTM 60, Second edition, 1989.

[2] V. I. Arnold and B. A. Khesin. *Topological Methods in Hydrodynamics.* Number 125 in Applied Mathematical Sciences. Springer-Verlag, 1998.

[3] H. F. Baker. Alternants and continuous groups. *Proc. London Math. Soc.*, 3:24–47, 1905.

[4] S. Buss. Accurate and efficient simulation of rigid body rotations. Technical report, Department of Mathematics, University of California, San Diego, 1999.

[5] E. Celledoni and A. Iserles. Approximating the matrix exponential from a Lie algebra to a Lie group. Technical Report 1998/NA03, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, England, 1998. To appear in Math. Comp.

[6] P. E. Crouch and R. Grossman. Numerical integration of ordinary differential equations on manifolds. *J. Nonlinear Sci.*, 3:1–33, 1993.

[7] F. Diele, L. Lopez, and R. Peluso. The Cayley transform in the numerical solution of unitary differential systems. *Adv. Comput. Math.*, 8(4):317–334, 1998.

[8] F. Diele, L. Lopez, and T. Politi. One step semi-explicit methods based on the Cayley transform for solving isospectral flows. *J. Comput. Appl. Math*, 89:219–223, 1998.

[9] S. Faltinsen, A. Marthinsen, and H. Z. Munthe-Kaas. Multistep methods integrating ordinary differential equations on manifolds. Technical Report Numerics No. 3/1999, The Norwegian University of Science and Technology, Trondheim, Norway, 1999.

[10] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I, Nonstiff Problems.* Springer-Verlag, Second revised edition, 1993.

[11] A. Iserles. On Cayley-transform methods for the discretization of Lie-group equations. Technical Report 1999/NA4, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, England, 1999.

[12] A. Iserles and S. P. Nørsett. On the solution of linear differential equations in Lie groups. In C. J. Budd and A. Iserles, editors, *Geometric Integration: Numerical Solution of Differential Equations on Manifolds*, volume 357 of *Philosophical Transactions of the Royal Society A*, pages 983–1020. London Mathematical Society, 1999.

[13] D. Lewis and J. C. Simo. Conserving algorithms for the dynamics of Hamiltonian systems of Lie groups. *J. Nonlinear Sci.*, 4:253–299, 1994.

[14] D. Lewis and J. C. Simo. Conserving algorithms for the n dimensional rigid body. *Fields Inst. Com.*, 10, 1995.

[15] D. Lewis and J. C. Simo. Conserving Algorithms for the n Dimensional Rigid Body. In J. E. Mardsen, G. W. Patrick, and W. F. Shadwick, editors, *Integration Algorithms and Classical Mechanics*, pages 121–140. American Mathematical Society, 1996.

[16] A. Marthinsen and B. Owren. Quadrature methods based on the Cayley transform. Technical Report Numerics No. 1/1999, The Norwegian University of Science and Technology, Trondheim, Norway, 1999.

[17] R. I. McLachlan. Explicit Lie-Poisson integration and the Euler equations. *Physical Review Letters*, 71:3043–3046, 1993.

[18] C. B. Moler and C. F. Van Loan. Nineteen dubious ways to compute the exponential of a matrix. *SIAM Review*, 20:801–836, 1979.

[19] H. Munthe-Kaas. Lie–Butcher theory for Runge–Kutta methods. *BIT*, 35(4):572–587, 1995.

[20] H. Munthe-Kaas. Runge–Kutta methods on Lie groups. *BIT*, 38(1):92–111, 1998.

[21] H. Munthe-Kaas. High order Runge–Kutta methods on manifolds. *Appl. Numer. Math.*, 29:115–127, 1999.

[22] H. Munthe-Kaas and B. Owren. Computations in a free Lie algebra. In C. J. Budd and A. Iserles, editors, *Geometric Integration: Numerical Solution of Differential Equations on Manifolds*, volume 357 of *Philosophical Transactions of the Royal Society A*, pages 957–982. London Mathematical Society, 1999.

[23] P. J. Olver. *Applications of Lie Groups to Differential Equations.* GTM 107. Springer-Verlag, Second edition, 1993.

[24] B. Owren and A. Marthinsen. Integration methods based on canonical coordinates of the second kind. Technical Report Numerics No. 5/1999, The Norwegian University of Science and Technology, Trondheim, Norway, 1999. To appear in Numer. Math.

[25] B. Owren and B. Welfert. The Newton iteration on Lie groups. Technical Report Numerics No. 3/1996, Norwegian University of Science and Technology, Trondheim, Norway, 1996. To appear in BIT.

[26] S. Reich. Symplectic integrators for systems of rigid bodies. *Fields Inst. Com.*, 1995. To appear.

[27] J. C. Simo and K. K. Wong. Unconditionally stable algorithms for rigid body dynamics that exactly preserve energy and momentum. *Internat. J. Numer. Methods Engrg.*, 31:19–52, 1991.

[28] A. Zanna. Collocation and relaxed collocation for the Fer and the Magnus expansions. *SIAM J. Numer. Anal.*, 36(4):1145–1182, 1999.

[29] A. Zanna, K. Engø, and H. Z. Munthe-Kaas. Adjoint and selfadjoint Lie-group methods. Technical Report 1999/NA02, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, England, 1999.