NORGES TEKNISK-NATURVITENSKAPELIGE
UNIVERSITET

Multistep Methods Integrating Ordinary Differential Equations on Manifolds

by

Stig Faltinsen, Arne Marthinsen and Hans Z. Munthe-Kaas

PREPRINT
NUMERICS NO. 3/1999

NORWEGIAN UNIVERSITY OF SCIENCE AND
TECHNOLOGY
TRONDHEIM, NORWAY

This report has URL http://www.math.ntnu.no/preprint/numerics/1999/N3-1999.ps
Address: Department of Mathematical Sciences, Norwegian University of Science and Technology,
N-7034 Trondheim, Norway.

Multistep Methods Integrating Ordinary Differential Equations on Manifolds*

Stig Faltinsen† Arne Marthinsen‡ Hans Z. Munthe-Kaas§

February 19, 1999

Abstract

This paper presents a family of generalized multistep methods that evolves the numerical solution of ordinary differential equations on configuration spaces formulated as homogeneous manifolds. Any classical multistep method may be employed as an invariant method, and the order of the invariant method is as high as in the classical setting. We present numerical results that reflect some of the properties of the multistep methods.

AMS Subject Classification: 65L06, 34A50

Key Words: geometric integration, multistep methods, numerical integration of ordinary differential equations on manifolds, numerical analysis, Lie groups, homogeneous spaces

1 Introduction

Classical multistep methods that solve the initial value problem

$$y' = f(t, y), \qquad y(0) = y_0 \in \mathbb{R}^d, \tag{1}$$

are discussed in a number of texts, see e.g. [7, 8, 11]. We write a general k-step method in the form

$$\sum_{i=0}^{k} \alpha_i y_{n+i} = h \sum_{i=0}^{k} \beta_i f(t_{n+i}, y_{n+i}), \qquad n = 0, 1, \dots, \tag{2}$$

where α_j and β_j, $j = 0, 1, \dots, k$, are given constants that are independent of the differential equation to be solved, the stepsize h, and n. By rescaling it may be assumed that $\alpha_k = 1$. The method is explicit if $\beta_k = 0$.

In [9], Iserles showed that methods in the family defined by (2) only can retain linear invariants. In this work we consider a reformulation of the multistep methods in the setting of Lie groups and homogeneous spaces and show that the method respects the configuration space of the problem when implemented in a correct way.

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

†Email: `sf221@damtp.cam.ac.uk`, WWW: `http://www.damtp.cam.ac.uk/user/na/people.html`

‡Email: `Arne.Marthinsen@math.ntnu.no`, WWW: `http://www.math.ntnu.no/~arnema/`

§Email: `Hans.Munthe-Kaas@ii.uib.no`, WWW: `http://www.ii.uib.no/~hans/`

Explicit multistep algorithms based on rigid frames were proposed by Crouch and Grossman [2]. The differential equation on a manifold $\mathcal{M}$ is expressed using smooth vector fields $E_1, \ldots, E_d$ on $\mathcal{M}$:

$$\dot{y} = F(y) = \sum_{i=1}^{d} f_i(y) E_i, \qquad y \in \mathcal{M},$$

where the f_i are real analytic functions on $\mathbb{R} \times \mathcal{M}$. The vector fields constitute a frame on $\mathcal{M}$: at each point $p \in \mathcal{M}$ they span the tangent space $\mathrm{T}\mathcal{M}|_p$. The numerical schemes are defined in terms of vector fields with coefficients frozen relative to the frame vector fields, i.e., $F_p = \sum_{i=1}^{d} f_i(p) E_i$.

The k-step Crouch-Grossman methods may now be written as

$$\begin{aligned}
u^{\ell}_{n+k-1} &= y_{n+k-1}, \\
u^{j}_{n+k-1}\left(h, y_{n+k-1}, \ldots, y_n, u^{j+1}_{n+k-1}\right) &= \exp(h\alpha^j_0 F_{y_{n+k-1}}) \circ \exp(h\alpha^j_1 F_{y_{n+k-2}}) \circ \cdots \circ \\
&\quad\ \exp(h\alpha^j_{k-1} F_{y_n}) \left(u^{j+1}_{n+k-1}\right), \qquad 0 \le j \le \ell - 1, \\
y_{n+k} &= u^0_{n+k-1}\left(h, y_{n+k-1}, \ldots, y_n, u^1_{n+k-1}\right).
\end{aligned}$$

Letting $\ell = 2$, this scheme reads

$$\begin{aligned}
y_{n+k} &= \exp(h\alpha^0_0 F_{y_{n+k-1}}) \circ \exp(h\alpha^0_1 F_{y_{n+k-2}}) \circ \cdots \circ \exp(h\alpha^0_{k-1} F_{y_n}) \circ \\
&\quad\ \exp(h\alpha^1_0 F_{y_{n+k-1}}) \circ \exp(h\alpha^1_1 F_{y_{n+k-2}}) \circ \cdots \circ \exp(h\alpha^1_{k-1} F_{y_n})(y_{n+k-1}),
\end{aligned}$$

and it is clear that if $\alpha_i = \sum_{j=0}^{\ell-1} \alpha^j_i$, $0 \le i \le k-1$, this algorithm reduces to the classical (explicit) Adams method in the Euclidean case:

$$y_{n+k} = y_{n+k-1} + h \sum_{i=0}^{k-1} \alpha_i F(y_{n+k-1-i}).$$

The k-step Crouch-Grossman method evolves the numerical approximation by composing flows of vector fields on $\mathcal{M}$. Computing flows of vector fields are very time consuming operations, and it may be advantageous to consider methods that combine frozen vector fields and compute the flow of the resulting vector field at the end of each step only. This is the approach we follow in this work.

1.1 Background and Notation

The framework for the methods discussed in this paper is given by Lie groups and homogeneous manifolds. We adopt some of the notation from [16] and we let $\mathcal{M}$ be a differentiable manifold and G be a Lie group with Lie algebra denoted by $\mathfrak{g}$.

Let $\Lambda : G \times \mathcal{M} \to \mathcal{M}$ be a (left) Lie group action. We get a (left) Lie algebra action $\lambda : \mathfrak{g} \times \mathcal{M} \to \mathcal{M}$, by $\lambda(v, p) = \Lambda(\exp(v), p)$, where $\exp : \mathfrak{g} \to G$ is the matrix exponential when G is a matrix group. We also define $\lambda_p(v) = \lambda(v, p)$. The mapping induced from λ is denoted by $\lambda_* : \mathfrak{g} \to \mathfrak{X}(\mathcal{M})$, where $\mathfrak{X}(\mathcal{M})$ is the set of all vector fields on $\mathcal{M}$. It is defined as

$$(\lambda_* v)(p) = \left.\frac{\mathrm{d}}{\mathrm{d}t}\right|_{t=0} \lambda(tv, p).$$

As a point of departure we assume that there exists a Lie algebra $\mathfrak{g}$ with a Lie bracket $[\cdot, \cdot]$, a (left) Lie algebra action defined as above, and a function $f : \mathbb{R} \times \mathcal{M} \to \mathfrak{g}$ such that the ordinary differential equation for $y(t) \in \mathcal{M}$ can be written in the form

$$y' = \mathcal{F}_\lambda(t, y) = (\lambda_* f(t, y))(y), \qquad y(0) = p \in \mathcal{M}. \tag{3}$$

Equation (3) is the canonical form of an ordinary differential equation on a manifold. We assume that $y_0 \in \mathcal{M}$ and it follows that $y' \in \mathrm{T}\mathcal{M}_y$, where $\mathrm{T}\mathcal{M}_y$ is the tangent space of $\mathcal{M}$ at $y \in \mathcal{M}$.

It is proven in [16] that the solution of (3) is given, for sufficiently small t, as $y(t) = \lambda(u(t), p)$, with $y(0) = p$, where $u(t) \in \mathfrak{g}$ satisfies the differential equation

$$u' = \widetilde{f}(u) = \mathrm{dexp}_u^{-1}(f(t, \lambda(u, p))), \qquad u(0) = 0 \in \mathfrak{g}. \tag{4}$$

This results follows by proving the commutativity of the following diagram:

$$\begin{tikzcd} \mathrm{T}\mathfrak{g} \arrow[r, "\lambda_p'"] & \mathrm{T}\mathcal{M} \\ \mathfrak{g} \arrow[u, "\widetilde{f}"] \arrow[r, "\lambda_p"] & \mathcal{M} \arrow[u, "\mathcal{F}"'] \end{tikzcd} \tag{5}$$

Here, λ_p' denotes the tangent mapping of λ_p. Note that

$$\Lambda\big(\exp(u_1), \Lambda(\exp(u_2), p)\big) = \Lambda\big(\exp(u_1) \cdot \exp(u_2), p\big) = \Lambda\big(\exp(\mathcal{B}(u_1, u_2)), p\big),$$

and hence

$$\lambda\big(u_1, \lambda(u_2, p)\big) = \lambda\big(\mathcal{B}(u_1, u_2), p\big), \tag{6}$$

where $\mathcal{B}$ is the Baker-Campbell-Hausdorff formula [20].

2 The Multistep Algorithm

Consider the problem of integrating (3). The basic idea is to look at the 'pulled back' form (4) on the linear space $\mathfrak{g}$ and to solve it numerically there using classical approaches.

By a slight abuse of language, we can consider λ_p to be a *local* coordinate chart taking a neighborhood of $0 \in \mathfrak{g}$ to a neighborhood of $p \in \mathcal{M}$. The matrix dexp_u represents the Jacobian of the coordinate chart (right trivialized). Since $\mathrm{dexp}_0 = I$, the chart is perfectly conditioned near $p \in \mathcal{M}$. As we move away from p it will in general become worse conditioned, and the numerical properties of the coordinates will degrade. It is therefore necessary to center the coordinates at a point p near the point on $\mathcal{M}$ we want to advance. At a step where we want to compute y_{n+1}, we may choose $p = y_n$, the nearest known point. Thus the center of the coordinate system changes at each step. If a Runge–Kutta method is used to solve (4), changing the center of the coordinate system poses no computational problems since the method does not use any information from a previous step. If instead a multistep method is used, we must transform old information to the new coordinate system in each step. We will see how this is done.

Let t_i be equidistant time points, and let $y_i \approx y(t_i)$. At step n of our algorithm we will compute y_{n+k} using a coordinate chart centered at $p = y_{n+k-1}$. Let $\omega_i^{(n)} \in \mathfrak{g}$ be the points corresponding to $y_{n+i} \in G$ at step n, i.e.

$$\lambda_{y_{n+k-1}}(\omega_i^{(n)}) = y_{n+i} \qquad \text{for } i = 0, \ldots, k+1.$$

If $f_i = f(t_i, y_i)$ and $\omega_i^{(n)}$ are known for $i = 0, \ldots, k-1$, then y_{n+k} can be found by the following multistep algorithm:

$$\widetilde{f}_i^{(n)} = \mathrm{dexp}^{-1}_{\omega_i^{(n)}}(f_{n+i})$$

$$\sum_{i=0}^{k} \alpha_i \omega_i^{(n)} = h \sum_{i=0}^{k} \beta_i \widetilde{f}_i^{(n)} \tag{7}$$

$$y_{n+k} = \lambda_{y_{n+k-1}}(\omega_k^{(n)}). \tag{8}$$

To continue, we need to find the corresponding ω-points at the next step $n+1$ by solving the equation $\lambda_{y_{n+k-1}}(\omega_i^{(n)}) = \lambda_{y_{n+k}}(\omega_{i-1}^{(n+1)})$ for $\omega_{i-1}^{(n+1)}$. Since $y_{n+k} = \lambda_{y_{n+k-1}}(\omega_k^{(n)})$, we get

$$\lambda\big(\omega_i^{(n)}, y_{n+k-1}\big) = \lambda\big(\omega_{i-1}^{(n+1)}, y_{n+k}\big) = \lambda\big(\omega_{i-1}^{(n+1)}, \lambda\big(\omega_{k+1}^{(n)}, y_{n+k-1}\big)\big).$$

Using (6) this yields $\omega_i^{(n)} = \mathcal{B}\big(\omega_{i-1}^{(n+1)}, \omega_k^{(n)}\big)$. Thus the transformation of ω becomes

$$\omega_{i-1}^{(n+1)} = \mathcal{B}(\omega_i^{(n)}, -\omega_k^{(n)}). \tag{9}$$

This yields the following multistep algorithm:

Algorithm 2.1. *Let the coefficients α_i and β_i, $i = 0, \ldots, k$, define a classical k-step method. Assume that y_{n-1+i}, $\omega_i^{(n-1)}$ and $f_{n-1+i} = f(t_{n-1+i}, y_{n-1+i})$ are known for $i = 1, \ldots, k$. Then the following k-step Lie group method advances the solution of (3) from time t_{n+k-1} to t_{n+k}.*

$$\begin{aligned}
\omega_i^{(n)} &= \mathcal{B}\big(\omega_{i+1}^{(n-1)}, -\omega_k^{(n-1)}\big), \qquad i = 0, \ldots, k-2 \\
\omega_{k-1}^{(n)} &= 0 \\
\widetilde{f}_i^{(n)} &= \operatorname{dexp}^{-1}_{\omega_i^{(n)}}\big(f_{n+i}\big), \qquad i = 0, \ldots, k \\
\sum_{i=0}^{k} \alpha_i \omega_i^{(n)} &= h \sum_{i=0}^{k} \beta_i \widetilde{f}_i^{(n)} \\
y_{n+k} &= \lambda_{y_{n+k-1}}\big(\omega_k^{(n)}\big).
\end{aligned}$$

This defines an equation system for the unknowns y_{n+k}, $\omega_k^{(n)}$ and $f_{n+k} = f(t_{n+k}, y_{n+k})$. If $\beta_k = 0$, then the system can be solved by explicit substitutions.

The main idea is given in the above algorithm, but we still need to find an efficient way to compute the initial values. It is natural to use a one-step method of Munthe-Kaas or generalized Magnus type (see e.g. [16, 5]) of sufficient order to start the procedure, i.e. to compute $\big(y_i, f(t_i, y_i)\big)$, $i = 1, \ldots, k-1$. But in addition, we store the elements $v_i \in \mathfrak{g}$ defined by $y_{i+1} = \lambda\big(v_i, y_i\big)$, $i = 0, \ldots, k-2$. Recall that if $p = \lambda(v, q)$ then $q = \lambda(-v, p)$.

How to start the procedure? Obviously,

$$\omega_{k-1}^{(0)} = 0 \qquad \text{and} \qquad \omega_{k-2}^{(0)} = -v_{k-2}. \tag{10}$$

Next,

$$\begin{aligned}
y_{k-3} &= \Lambda\big(\exp(\omega_{k-3}^{(0)}), y_{k-1}\big) \\
&= \Lambda\big(\exp(\omega_{k-3}^{(0)}), \Lambda\big(\exp(-\omega_{k-2}^{(0)}), y_{k-2}\big)\big) \\
&= \Lambda\big(\exp(\omega_{k-3}^{(0)}) \cdot \exp(-\omega_{k-2}^{(0)}), y_{k-2}\big).
\end{aligned} \tag{11}$$

But $y_{k-3} = \Lambda\big(\exp(-v_{k-3}), y_{k-2}\big)$. By comparing this expression to (11) we see that $\omega_{k-3}^{(0)} = \mathcal{B}(-v_{k-3}, \omega_{k-2}^{(0)})$. Similar calculations yield in general

$$\omega_{k-i}^{(0)} = \mathcal{B}(-v_{k-i}, \omega_{k-i+1}^{(0)}), \qquad i = 2, \ldots, k. \tag{12}$$

After having started the multistep algorithm with a one-step method and the above described algorithm, we still have to express the available information in canonical coordinates of first kind at each step in Algorithm 2.1.

As will be shown in Theorem 2.2, we do not need to compute $\mathcal{B}$ or dexp_u^{-1} exactly. If the order of the multistep method is p, then it suffices to compute

$$\omega_i^{(n+1)} = \mathrm{bch}\big(\omega_{i+1}^{(n)}, \omega_{k-2}^{(n+1)}, p\big) = \mathcal{B}\big(\omega_{i+1}^{(n)}, \omega_{k-2}^{(n+1)}\big) + \mathcal{O}(h^{p+1}),$$

where h is the stepsize, $\mathrm{bch}(u, v, q)$ approximates $\mathcal{B}(u, v)$ to order p, and

$$\widetilde{f}_i^{(n)} = \mathrm{dexpinv}\big(\omega_i^{(n)}, f_{n+i}, p\big) = \mathrm{dexp}^{-1}_{\omega_i^{(n)}}(f_{n+i}) + \mathcal{O}(h^{p+1}),$$

where $\mathrm{dexpinv}(u, v, q)$ is defined and described in [16].

To define the order of a numerical method, we use the local diffeomorphism λ_p to pull back the equation from $\mathcal{M}$ to $\mathfrak{g}$. Since $\mathfrak{g}$ is a vector space, the classical definition of order applies here [11]. The numerical method on $\mathcal{M}$ has order p if the corresponding method on $\mathfrak{g}$ has order p in the classical sense.

Theorem 2.2. *If $(y_{n+i}, f(t_{n+i}, y_{n+i})) \in \mathcal{M} \times \mathfrak{g}$, $i = 0, \ldots, k-1$, then Algorithm 2.1 generates an element $y_{n+k} \in \mathcal{M}$. If the classical multistep method defined by the coefficients α_i and β_i, $i = 0, \ldots, k$, is of order q, then the order of approximation of y_{n+k} to $y(t_{n+k})$ is q.*

Proof. We observe that $\omega_i^{(n)}$ and $\widetilde{f}_i^{(n)}$, $i = 0, \ldots, k-1$, as well as $\widetilde{f}_k^{(n)}$ are elements of $\mathfrak{g}$. Solution of (7) yields an element $\omega_k^{(n)} \in \mathfrak{g}$. The first part of the theorem now follows, since $\lambda_y : \mathfrak{g} \to \mathcal{M}$ for all $y \in \mathcal{M}$.

We use a classical multistep method of order q to integrate (4). After observing that the Baker-Campbell-Hausdorff formula, $\mathcal{B}$, introduces an $\mathcal{O}(h^{q+1})$ modification of $\omega_i^{(n)}$, $i = 0, \ldots, k-2$, and that $\mathrm{dexp}^{-1}_{\omega_i^{(n)}}$ introduces an $\mathcal{O}(h^{q+1})$ modification of f_{n+i}, $i = 0, \ldots, k-2$, the second part of the theorem follows by noting that we compute the pullback vector field $\widetilde{f}$ in (4) correct to order q (see also [16]). $\square$

2.1 Implicit Methods

When implementing implicit methods, i.e. when $\beta_k \neq 0$, we have to iterate to find the solution of (7), since the correction of f_{n+k} depends on $\omega_k^{(n)}$:

$$\widetilde{f}_k^{(n)} = \mathrm{dexp}^{-1}_{\omega_k^{(n)}}(f_{n+k}).$$

We have

$$\sum_{i=0}^{k-1} \alpha_i \omega_i^{(n)} + \omega_k^{(n)} = h \sum_{i=0}^{k-1} \beta_i \widetilde{f}_i^{(n)} + \beta_k \widetilde{f}_k^{(n)},$$

since we assume that $\alpha_k = 1$. As in the classical situation we can use a predictor-corrector method and let the explicit predictor generate starting values $\omega_{n+k}^{[0]}$ to the following standard iteration, where we have suppressed information about current time-step:

$$\omega_k^{[\ell+1]} = h\beta_k \mathrm{dexp}^{-1}_{\omega_k^{[\ell]}}\big(f_k^{[\ell]}\big) + \mathcal{R}, \tag{13}$$

where

$$\mathcal{R} = h \sum_{i=0}^{k-1} \beta_i \widetilde{f}_i^{(n)} - \sum_{i=0}^{k-1} \alpha_i \omega_i^{(n)}$$

is known.

Theorem 2.3. *Assume that the Lie algebra $\mathfrak{g}$ is a Banach space and that $\widetilde{f}$ is Lipschitz with constant L. Then the iteration (13) converges provided that $h\beta_k L < 1$.*

Proof. Let $\|\cdot\|_{\mathfrak{g}}$ be a norm on $\mathfrak{g}$. Consider $\widehat{\omega}_k^{[\ell]}$ also defined by iteration (13) with initial condition $\widehat{\omega}_k^{[0]} \neq \omega_k^{[0]}$. Note that $\mathcal{R}$ remains the same for all iterations. We then get that

$$\|\omega_k^{[\ell+1]} - \widehat{\omega}_k^{[\ell+1]}\| = h\beta_k\|\widetilde{f}(\omega_k^{[\ell]}) - \widetilde{f}(\widehat{\omega}_k^{[\ell]})\| \leq h\beta_k L\|\omega_k^{[\ell]} - \widehat{\omega}_k^{[\ell]}\|.$$

Since $h\beta_k L < 1$, this is a contraction and there exists a unique fixed-point of iteration (13) in the complete space $\mathfrak{g}$. $\square$

Note: One might argue that it is more natural to impose a Lipschitz condition on the original vector field $\mathcal{F}$. This is done in [5], where it is also shown that this implies a Lipschitz condition on $\widetilde{f}$.

2.2 Variable Stepsize and Error Estimation

The main reason for wanting variable stepsize in numerical integrators is to be able to control, in some sense, the error in the approximations by means of stepsize changes. We also need this device if we start the multistep procedure with a one-step method and use interpolation to provide the necessary, intermediate values.

Let Ψ be a classical, convergent multistep method of order p. Assume that $\omega_0^{(n)}, \omega_1^{(n)}, \ldots, \omega_{k-1}^{(n)}$ are error free. Then, by classical theory,

$$\omega(t_{n+k}) - \omega_k^{(n)} = ch^{p+1}\omega^{(p+1)}(t_{n+k}) + \mathcal{O}(h^{p+2}),$$

as $h \to 0$, where c is the local error constant of method Ψ. Let further $\widetilde{\Psi}$ be another classical, convergent multistep method of same order but with local error constant $\widetilde{c}$. Then,

$$\omega(t_{n+k}) - \omega_k^{(n)} \approx \frac{c}{c - \widetilde{c}}\big(\widetilde{\omega}_k^{(n)} - \omega_k^{(n)}\big). \tag{14}$$

The Milne device (14) makes perfectly sense as an error estimator in the generalized setting considered in this paper if $\mathcal{M}$ is a Lie group, since Faltinsen [5] has shown that

$$d\big(\lambda_p(u), \lambda_p(v)\big) \leq C\|u - v\|,$$

where $d : G \times G \to \mathbb{R}$ is the distance function on G, $\|\cdot\|$ is a norm on $\mathfrak{g}$, $\lambda : \mathfrak{g} \times G \to G$ and G is a Lie group.

By applying interpolation as suggested by Marthinsen [13], techniques involving Nordsieck vectors are applicable. We will, however, not describe these approaches in the present paper.

2.3 Computation of the Baker–Campbell–Hausdorff formula

The Baker–Campbell–Hausdorff formula, $\mathcal{B} : \mathfrak{g}\times\mathfrak{g} \to \mathfrak{g}$, is given as

$$\mathcal{B}(u, v) = \exp(u)\exp(v). \tag{15}$$

It is well known that this function can be expressed in terms of commutators [20]. We have seen that the transfer of information from one coordinate representation to another involves $\mathcal{B}$, and we will here discuss optimization of this computation.

We use the *DiffMan* [4] free Lie algebra module to compute $\mathcal{B}$. The algorithm used is given in [17]. For order 6, the *DiffMan* `bch` program yields

```
>> format rat; z = bch(6)
   z = [1] + [2] + 1/2*[1,2] + 1/12*[1,[1,2]] - 1/12*[2,[1,2]]
      - 1/24*[2,[1,[1,2]]] - 1/720*[1,[1,[1,[1,2]]]]
      - 1/180*[2,[1,[1,[1,2]]]] + 1/180*[2,[2,[1,[1,2]]]]
      + 1/720*[2,[2,[2,[1,2]]]] - 1/120*[[1,2],[1,[1,2]]]
      - 1/360*[[1,2],[2,[1,2]]] + 1/1440*[2,[1,[1,[1,[1,2]]]]]
      + 1/360*[2,[2,[1,[1,[1,2]]]]] + 1/1440*[2,[2,[2,[1,[1,2]]]]]
      + 1/240*[[1,2],[2,[1,[1,2]]]] + 1/720*[[1,2],[2,[2,[1,2]]]]
      - 1/240*[[1,[1,2]],[2,[1,2]]]
```

Here z is a symbolic expression that can be used to compute $\mathcal{B}$ to order 6 on two elements u and v. In *DiffMan* this is done by using the evaluation function

```
>> w = eval(z,u,v)
```

The main problem with $\mathcal{B}$ is that the number of commutators grows quickly with the order, see [17] for exact formulas counting commutators. Several authors [10, 18] have discussed ways of reducing the number of commutators by doing a change of variables. We will here introduce a simple trick that is not found in these sources, and which yields a reduction of commutators which is similar to the reductions obtained by other means in [10, 18]. For some orders q, our approach is a little better than other reported approaches, for other orders it is a little worse. Our trick is based on exploiting the symmetry

$$\mathcal{B}(-v,-u) = -\mathcal{B}(u,v),$$

which follows from $(\exp(u)\exp(v))^{-1} = \exp(-v)\exp(-u)$. The symbolic expression for z is based on using `[1]` and `[2]` as symbols representing u and v. We might instead change basis and let $(u+v)/2 \leftrightarrow$ `[1]` and $(u-v)/2 \leftrightarrow$ `[2]`. In this basis the symmetry $(u,v) \mapsto (-v,-u)$ corresponds to (`[1]`, `[2]`) $\mapsto$ ($-$`[1]`, `[2]`), hence the basis maps to itself, with a sign change. If $\mathcal{B}$ changes sign under this transformation, we can conclude: *In the new basis the $\mathcal{B}$ formula has nonzero coefficients only in terms where* `[1]` *occurs an odd number of times.* This reduces the number of commutators approximately to the half. Exact formulas counting the number of commutators of this type can be found in [6] (formulated in the language of colored beads in a necklage). Below is the result of the change of basis.

```
>> e1 = basis(z,1)
   e1 = [1]
>> e2 = basis(z,2)
   e2 = [2]
>> znew = eval(z,e1+e2,e1-e2)
   znew =
   2*[1] - [1,2] - 1/3*[2,[1,2]] + 1/12*[1,[1,[1,2]]] - 1/12*[2,[2,[1,2]]]
   + 7/180*[2,[1,[1,[1,2]]]] - 1/60*[2,[2,[2,[1,2]]]]
   - 1/90*[[1,2],[1,[1,2]]] - 1/120*[1,[1,[1,[1,[1,2]]]]]
   + 1/90*[2,[2,[1,[1,[1,2]]]]] - 1/360*[2,[2,[2,[2,[1,2]]]]]
   + 1/360*[[1,2],[2,[1,[1,2]]]] + 1/90*[[1,[1,2]],[2,[1,2]]]
```

Evaluation of $w = \mathcal{B}(u,v)$ is now done as

```
>> w = eval(znew,(u+v)/2,(u-v)/2);
```

An alternative approach to compute $\mathcal{B}$ numerically is simply

$$\mathcal{B}(u,v) = \log(\exp(u)\exp(v)).$$

2.4 Computational Cost and Complexity

From a computational point of view, Algorithm 2.1 looks expensive, since we have to transform the data to a common coordinate system at each step. It turns out that we, for an explicit k-step method, may use as much as $k-2$ computations of $\mathcal{B}$, $k-1$ computations of dexp^{-1}, one exponentiation and one matrix multiplication per step, in addition to one function evaluation and a number of linear combinations. However, in many situations the cost is less.

The general form of the backward differentiation formulae (BDF) for (1) is

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = h\beta_k f_{n+k}.$$

Since these methods are implicit, iterative solution (e.g. of the form (13)) must take place. However, no function values enter $\mathcal{R}$, and we save the dexp^{-1} corrections. The general form of Adams methods is

$$y_{n+k} - y_{n+k-1} = h\sum_{j=0}^{k} \beta_j f_{n+j}.$$

These methods are the most expensive ones in the setting of Algorithm 2.1. We have to correct all the function values using dexp^{-1}, and we therefore need to compute all the ω even though only $\omega_{k-1}^{(n)}$ and $\omega_k^{(n)}$ enter equation (7) explicitly.

Example 2.4. *Consider the following explicit, third order Adams-Bashforth method ($k=3$):*

$$\alpha = [1,-1,0,0] \qquad \textit{and} \qquad \beta = \left[0, \tfrac{23}{12}, -\tfrac{16}{12}, \tfrac{5}{12}\right].$$

Assume that the startup procedure is completed. The data available from the previous step ($n+1$) is then $\omega_3^{(n+1)}$, $\omega_2^{(n+1)}$, $\omega_1^{(n+1)}$, f_{n+1} and f_n. The cost of taking one new step is as follows:

Evaluation: *we have to evaluate the function f at time t_{n+1}.*

Transformation: *we immediately have that*

$$\omega_2^{(n+2)} = 0 \qquad \textit{and} \qquad \omega_1^{(n+2)} = -\omega_3^{(n+1)}.$$

Furthermore, (9) gives $\omega_0^{(n+2)} = \mathcal{B}\big(\omega_1^{(n+1)}, \omega_1^{(n+2)}\big)$.

Correction: *we have to correct f_{n+2}, f_{n+1} and f_n using* dexp^{-1}*:*

$$\widetilde{f}_2^{(n)} = f_{n+2}, \qquad \widetilde{f}_1^{(n)} = \mathrm{dexp}^{-1}_{\omega_1^{(n+2)}}\big(f_{n+1}\big) \qquad \textit{and} \qquad \widetilde{f}_0^{(n)} = \mathrm{dexp}^{-1}_{\omega_0^{(n+2)}}\big(f_n\big).$$

Step: *computation of $\omega_3^{(n+2)}$ from (7) consists of linear combinations only, while advancing the solution according to (8) costs one matrix exponential and one matrix multiplication when λ is of the form $\lambda_p(v) = \exp(v)\cdot p$.*

We compare these numbers to the cost of the third order (one-step) Crouch-Grossman method defined by the scheme (22) in [19] and Kutta's third order method implemented in the setting of Munthe-Kaas [16]. The cost of the Crouch-Grossman method is roughly 3 *function evaluations,* 6 *exponentiations and* 6 *matrix multiplications. The Munthe-Kaas method costs* 3 *function evaluations,* 3 *algebra actions (giving* 3 *exponentiations and* 3 *matrix multiplications),* 2 dexp^{-1} *as well as linear combinations of matrices.*

One matrix multiplication costs roughly $2n^3$ flops, exponentiation costs $25n^3$ flops, dexp^{-1} *of order 3 costs $8n^3$ flops and $\mathcal{B}$ of order 3 costs roughly $8n^3$ flops. In total, the Munthe-Kaas method implemented with Kutta's third order coefficients costs roughly $97n^3$ flops plus 3 function evaluations, the third order Crouch-Grossman method costs $162n^3$ flops plus 3 function evaluations while Algorithm 2.1 with Adams-Bashforth's third order coefficients costs $51n^3$ flops plus one function evaluation.*

Note, however, that it is the efficiency (global error as a function of floating point operations) that is of most importance and interest. The numerical experiments in Section 4 show the relative efficiency of some methods.

3 General Linear Methods

Several authors have discussed integration of (3) by generalized Runge-Kutta methods, see e.g. [16, 15, 5]. This paper is devoted to analysis of the use of linear multistep methods to integrate (3). We will, however, point out that integration using general linear methods readily follows through a combination of these techniques.

Assume that $y_{n+k-1-i}$, $i = 0, \dots, k-1$, are available and express, as before, the data in a coordinate system around y_{n+k-1}. This gives us $\omega_i^{(n)}$, $i = 0, \dots, k-1$, which we collect in the vector $\omega^{(n)}$. The general linear methods can now be adapted to homogeneous spaces as follows:

$$\begin{aligned}
\widetilde{v}^{(n)} &= \widetilde{A}\omega^{(n)} + h\widetilde{B}F^{(n)},\\
\widetilde{\omega}^{(n+1)} &= A\omega^{(n)} + hBF^{(n)},\\
y_{n+1} &= \lambda\big(\widetilde{\omega}_1^{(n+1)}, y_n\big).
\end{aligned}$$

Here, $A \in \mathbb{R}^{k\times k}$, $B \in \mathbb{R}^{k\times s}$, $\widetilde{A} \in \mathbb{R}^{s\times k}$, and $\widetilde{B} \in \mathbb{R}^{s\times s}$ are coefficients describing the method, while $F^{(n)}$ is the s-vector with elements $F_i^{(n)} = \mathrm{dexp}^{-1}_{\widetilde{v}_i^{(n)}}\Big(f\big(t_n + c_i h, \lambda(\widetilde{v}_i^{(n)}, y_n)\big)\Big)$, $i = 0, \dots, s-1$. We see that by putting $k = 1$, we recover the Munthe-Kaas methods. By letting $s = k$, $\widetilde{B} = 0$, and $\widetilde{A} = I$, we recover the linear multistep methods.

The starting values $\omega_i^{(0)}$, $i = 0, \dots, k-1$, can be computed as in (12). When a step is completed, the values in $\omega^{(n+1)}$ must be computed from $\omega^{(n)}$. We have immediately that $\omega_0^{(n+1)} = 0$ and $\omega_1^{(n+1)} = -\widetilde{\omega}_1^{(n+1)}$, while the rest of $\omega^{(n+1)}$ is computed essentially as in (9):

$$\omega_i^{(n+1)} = \mathcal{B}\big(\omega_{i-1}^{(n)}, \omega_1^{(n+1)}\big), \qquad i = 2, \dots, k-1.$$

As an example, consider the 2-step, 2-stage method of order 3 by Butcher [7, p. 444] ($k = 2$, $s = 2$):

$$A = \begin{pmatrix} \frac{16}{11} & -\frac{5}{11} \\ 1 & 0 \end{pmatrix}, \qquad B = \begin{pmatrix} \frac{104}{99} & -\frac{50}{99} \\ 0 & 0 \end{pmatrix}, \qquad \widetilde{A} = \begin{pmatrix} \frac{3}{2} & -\frac{1}{2} \\ \frac{3}{2} & -\frac{1}{2} \end{pmatrix}, \qquad \widetilde{B} = \begin{pmatrix} 0 & 0 \\ -\frac{9}{10} & 0 \end{pmatrix}.$$

This leads to

$$\begin{aligned}
\widetilde{v}_1^{(n)} &= \tfrac{3}{2}\omega_1^{(n)} - \tfrac{1}{2}\omega_2^{(n)}\\
\widetilde{v}_2^{(n)} &= \tfrac{3}{2}\omega_1^{(n)} - \tfrac{1}{2}\omega_2^{(n)} - \tfrac{9}{10}hf\big(t_n + \tfrac{1}{2}h, \widetilde{v}_1^{(n)}\big)\\
\widetilde{\omega}_1^{(n+1)} &= \tfrac{16}{11}\omega_1^{(n)} - \tfrac{5}{11}\omega_2^{(n)} + h\Big(\tfrac{104}{99}f\big(t_n + \tfrac{1}{2}h, \widetilde{v}_1^{(n)}\big) - \tfrac{50}{99}f\big(t_n - \tfrac{2}{5}h, \widetilde{v}_2^{(n)}\big)\Big)\\
\widetilde{\omega}_2^{(n+1)} &= \omega_1^{(n)}\\
y_{n+1} &= \lambda\big(\widetilde{\omega}_1^{(n+1)}, y_n\big).
\end{aligned}$$

In this case, $\omega_1^{(n)} = 0$ and $\omega_2^{(n)} = -\widetilde{\omega}_1^{(n)}$.

4 Numerical Results

We want to apply Algorithm 2.1, with different classical multistep methods, to some test problems. The methods are classically defined by

AM3:	Adams-Moulton, order 3:	$y_{n+3} - y_{n+2} = h\left[\frac{9}{24}f_{n+3} + \frac{19}{24}f_{n+2} - \frac{5}{24}f_{n+1} + \frac{1}{24}f_n\right]$,
N4:	Nyström, order 4:	$y_{n+4} - y_{n+2} = h\left[\frac{8}{3}f_{n+3} - \frac{5}{3}f_{n+2} + \frac{4}{3}f_{n+1} - \frac{1}{3}f_n\right]$,
BDF3:	BDF, order 3:	$y_{n+3} - \frac{18}{11}y_{n+2} + \frac{9}{11}y_{n+1} - \frac{2}{11}y_n = \frac{6}{11}hf_{n+3}$,

where $f_{n+i} = f(t_{n+i}, y_{n+i})$. In addition we have used the classical fourth order Runge-Kutta method (**RK4**) and the fourth order Gauss-Legendre method (**GL4**) described by the following Butcher tableaus:

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{2}{6} & \frac{2}{6} & \frac{1}{6} \end{array} \qquad \begin{array}{c|cc} \frac{3-\sqrt{3}}{6} & \frac{1}{4} & \frac{3-2\sqrt{3}}{12} \\ \frac{3+\sqrt{3}}{6} & \frac{3+2\sqrt{3}}{12} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

The **MK4** method is the Munthe-Kaas method [16] implemented with the **RK4** coefficients. All simulations are done with MATLAB, and the flops-counts presented have been computed with the MATLAB `flops` function.

An orthogonal problem. As an illustration of the algorithms, we first choose an example by Zanna [21]. Let the manifold $\mathcal{M} = G$ be a matrix Lie group with Lie algebra $(\mathfrak{g}, [\cdot,\cdot])$. The action of the Lie algebra on G is given by $\lambda : \mathfrak{g} \times G \to G$, where $\lambda(v,p) = \exp(v) \cdot p$. Now (3) reduces to

$$y' = f(t,y) \cdot y \qquad \text{with} \qquad y(0) \in G. \tag{16}$$

Let the right hand side of (16) be defined by $f : \mathrm{SO}(n) \to \mathfrak{so}(n)$ with (in MATLAB notation)

```
f(y)=diag(diag(y,1),1)-diag(diag(y,1),-1);
```

and we solve this problem with initial values (a random, orthogonal, 5×5 matrix with determinant $+1$)

```
rand('seed',0); [y0,r]=qr(rand(5));
```

Since the initial value is a matrix in SO(5), the solution will evolve on this Lie group. We integrate from $t_0 = 0$ to $t_{\text{end}} = 3$. Some results from the simulations are shown in Figure 1. The constant stepsize used in the simulations was $h = \frac{1}{10}$. The lower left figure shows the efficiency of the codes when applied to this problem. We measure efficiency as number of flops needed to obtain a certain global error. We see that, although the multistep methods respect the underlying manifold, the global error is relatively high compared to the error generated by the other methods.

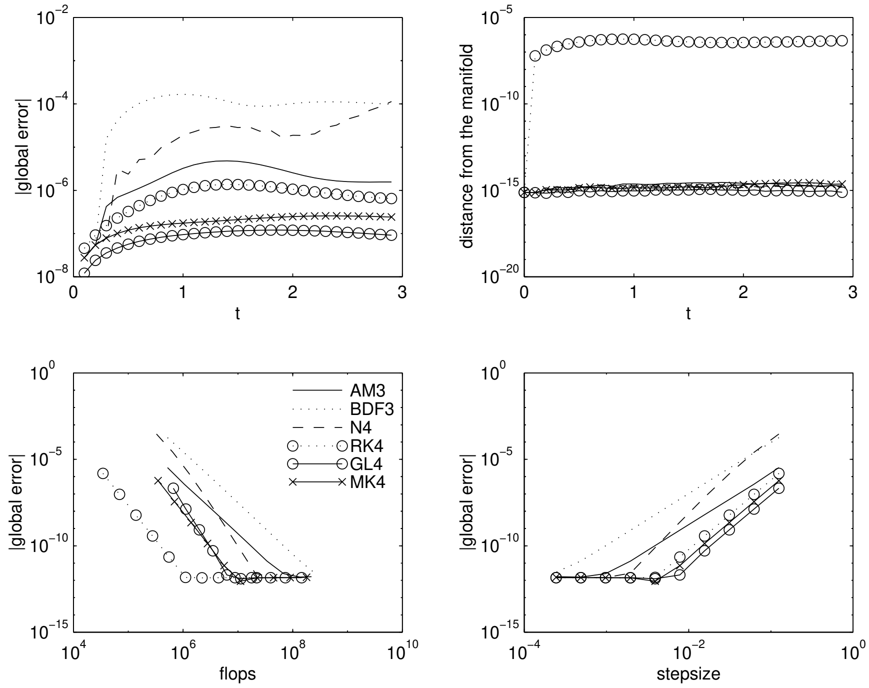

Figure 1: The 2-norm of the global error, departure from orthogonality, efficiency and global error versus stepsize for the orthogonal problem by Zanna. The legend in the lower left figure is valid for all the figures.

An isospectral problem: the Toda lattice equations. The general form of an isospectral flow, due to Lax [12], is the differential equation

$$L'(t) = [B(t, L(t)), L(t)] \qquad \text{with} \qquad L(0) = L_0, \tag{17}$$

where $B(t, L(t))$, $L(t)$ and L_0 are $d \times d$ matrices and $[\cdot, \cdot]$ is the matrix commutator.

The Toda lattice equations models the motion of a finite number of particles on the line with exponential interactions between the nearest neighbors (see e.g. [1, 22]). The Hamiltonian function in the non-periodic case with d particles is given as

$$H(p, q) = \frac{1}{2} \sum_{k=1}^{d} p_k^2 + \sum_{k=1}^{d-1} \left\{ \exp\left(2(q_k - q_{k+1})\right) - 1 \right\},$$

where q_ℓ is the position of the ℓth particle and p_ℓ is its momentum. With $q_0 = q_{d+1} = 0$ the equations of motion are

$$\begin{array}{rcl} q_k' &=& p_k, \\ p_k' &=& 2\left(\exp\left(2(q_{k-1} - q_k)\right) - \exp\left(2(q_k - q_{k+1})\right)\right), \end{array} \qquad k = 1, \ldots, d.$$

This system can be written in the form (17) by introducing the variables α_ℓ and β_ℓ given by

$$\begin{array}{rcll} \beta_k &=& -p_k, & 1 \le k \le d, \\ \alpha_k &=& \exp(q_k - q_{k+1}), & 1 \le k \le d-1, \end{array}$$

and letting the matrices $L(t)$ and $B(t, L(t))$ have the following structure:

$$L(t) = \begin{bmatrix} \beta_1 & \alpha_1 & 0 & \cdots & 0 \\ \alpha_1 & \beta_2 & \alpha_2 & \ddots & \vdots \\ 0 & \alpha_2 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \alpha_{d-1} \\ 0 & \cdots & 0 & \alpha_{d-1} & \beta_d \end{bmatrix} \quad \text{and} \quad B(t, L(t)) = \begin{bmatrix} 0 & \alpha_1 & 0 & \cdots & 0 \\ -\alpha_1 & 0 & \alpha_2 & \ddots & \vdots \\ 0 & -\alpha_2 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \alpha_{d-1} \\ 0 & \cdots & 0 & \alpha_{d-1} & 0 \end{bmatrix}.$$

Let $\mathcal{M} \subset \mathrm{GL}(d)$. Let $G = \mathrm{SO}(d)$ with Lie algebra $\mathfrak{g} = \mathfrak{so}(d)$. By letting $\lambda(v, p) = \exp(v) \cdot p \cdot \exp(-v)$ and the matrices L and B be defined as above, (3) reduces to the isospectral equation

$$y' = B\big(t, L(t)\big) \cdot L(t) - L(t) \cdot B\big(t, L(t)\big), \qquad L(0) = L_0,$$

and the multistep method will exactly preserve isospectrality. Numerical results verifying this are shown in Figure 2. The simulations used $d = 3$,

$$L_0 = \begin{bmatrix} -1 & 1 & 0 \\ 1 & 0.5 & 1 \\ 0 & 1 & 0.5 \end{bmatrix}$$

and the integration interval was $[0, 1]$. We see that the multistep methods, by construction, preserve the eigenvalues of the solution matrix to machine accuracy. The classical methods generate substantial drift in the eigenvalues. Figure 3 shows global error as a function of time and the relative efficiency of the codes. Again, the efficiency is computed as the 2-norm of the global error as a function of number of flops used by the code.

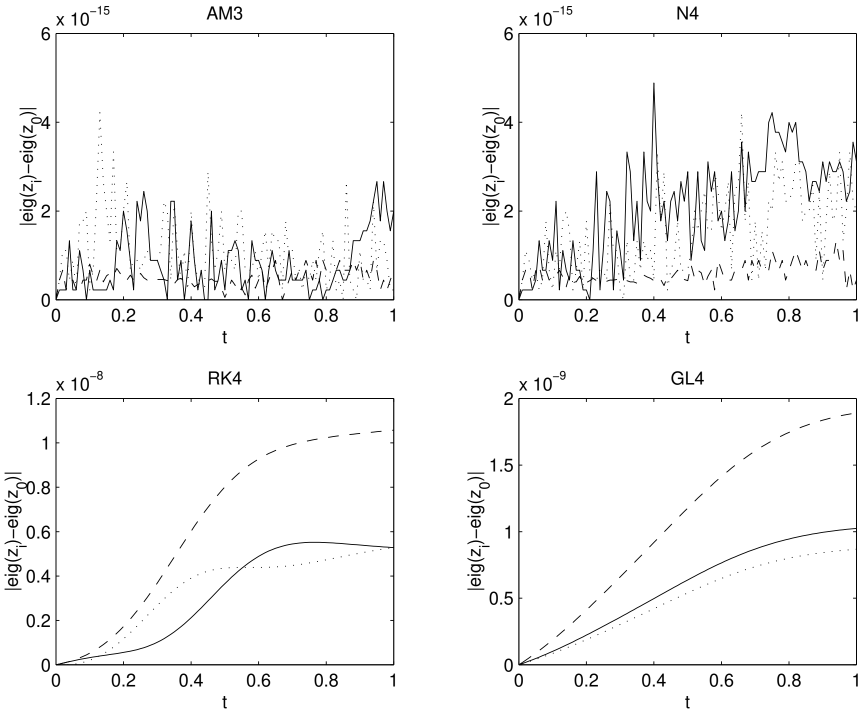

Figure 2: Drift in the eigenvalues of the solution matrix when integrating the Toda flow problem. The upper figures show the results from **AM3** and **N4**, while the lower figures show the results from **RK4** and **GL4**.

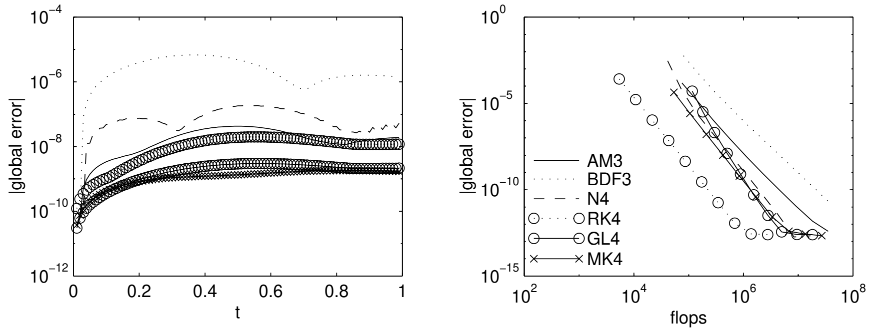

Figure 3: The left figure shows the 2-norm of the global error as a function of time while the right figure shows the efficiency of the codes when integrating the Toda flow problem. The legend is valid for both figures.

The spinning top problem. We now solve the spinning top problem, which evolves on TSO(3). The equations are described in [14, 3], but we include here a short description for completeness.

We model the spinning top on the Lie group $G = \mathrm{TSO}(3) \simeq \mathrm{SO}(3) \times \mathfrak{so}(3)$, and an element in the group is denoted by the pair $\big(B(t), \omega(t)\big)$. The Lie algebra of G is denoted by $\mathfrak{g}$. The differential equation describing the motion of the top is given by $y' = F(y)y$, where $y \in G$ and $F : G \to \mathfrak{g}$. The function F is given by the mapping

$$\big(B(t), \omega(t)\big) \mapsto \big(B'(t), \omega'(t)\big) = \big(\omega, [\omega, \mathrm{I}^{-1}]\mathrm{L} + \mathrm{I}^{-1}\mathrm{M}\big).$$

Here, $\mathrm{I} = B(t)\mathrm{I}_0 B^{-1}(t)$ denotes the inertia tensor, $\mathrm{L} = B(t)\mathrm{I}_0 B^{-1}(t)\omega(t)$ denotes the angular momentum, and $\mathrm{M} = mc \times f$ denotes the torque. m is the mass of the top and f is the vector of gravity.

The following initial values are used throughout the simulations:

$$B_0 = \begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & \cos(\phi) & \sin(\phi) \\ 0.0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \quad \text{and} \quad \omega_0 = \begin{bmatrix} 0.0 & -1.0 & 0.0 \\ 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{bmatrix},$$

with $\phi = \pi/16$. The centroid vector, the vector of gravity and the interia tensor of the top are taken to be

$$c = \begin{bmatrix} 0 \\ 0 \\ \sqrt{3}/2 \end{bmatrix}, \quad f = \begin{bmatrix} 0 \\ 0 \\ -9.81 \end{bmatrix} \quad \text{and} \quad \mathrm{I}_0 = \frac{1}{8}\begin{bmatrix} 7 & 0 & 0 \\ 0 & 7 & 0 \\ 0 & 0 & 2 \end{bmatrix},$$

respectively.

Figure 4 shows some of the results from the simulations. The conclusions are similar to the ones from the orthogonal and the isospectral problem. The global error as a function of time for the multistep methods are slightly larger than the error from **MK4** and the classical methods. Also, the global error versus flops are less good for the multistep methods than for the classical methods. However, by construction they respect the configuration space of the problem.

5 Concluding Remarks

The area of geometric integration, and in particular integration methods on Lie groups and homogeneous spaces, has received much attention the last few years. Most of the development has

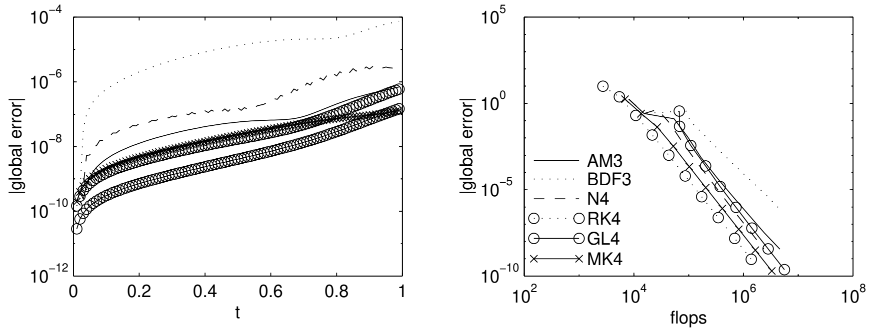

Figure 4: The spinning top. The left figure shows the global error as a function of time while the right hand figure shows the efficiency of the codes, measured as global error versus number of floating point operations. The legend is valid for both figures.

been related to generalization of Runge-Kutta and other one-step methods. In this work we have analyzed multistep methods in the setting of homogeneous manifolds. Both general and Lie type problems are discussed, and a number of numerical experiments are included.

Theoretically, the multistep methods are less expensive than the one-step Munthe-Kaas and Crouch-Grossman methods. The numerical results, however, indicate that the one-step methods are more efficient than the multistep methods. This depends on the particular multistep-method chosen. This picture may change when the evaluation of the governing equations (right hand side of the equations) is costly. Explicit linear multistep methods require only one function evaluation at each step, as opposed to s evaluations for an s-stage Runge-Kutta method.

The geometric integration methods are generally more expensive than the classical methods. However, the efficiency, measured as global error versus number of floating point operations, are in many cases better for the geometric integration methods. These methods are constructed to respect the configuration space of the problem, and numerical experiments show that they tend to produce less global error as a function of floating point operations. In addition, to obtain qualitatively good results from long-time integration it is of vital importance that the configuration space of the problem is respected by the numerical solution.

Acknowledgment. The authors are grateful to Syvert P. Nørsett for valuable comments and suggestions throughout the work with this paper.

References

[1] M. P. Calvo, A. Iserles, and A. Zanna. Numerical solution of isospectral flows. *Math. Comp.*, 66(220):1461–1486, 1997.

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

[3] K. Engø and A. Marthinsen. Modeling and solution of some mechanical problems on Lie groups. *Multibody System Dynamics*, 2:71–88, 1998.

[4] K. Engø, A. Marthinsen, and H. Z. Munthe-Kaas. Diffman — an object oriented MATLAB toolbox for solving differential equations on manifolds. Technical Report No. 164, Department of Informatics, University of Bergen, 1999.

[5] S. Faltinsen. Lie Group Methods and Lyapunov Stability. Master's thesis, Department of Mathematical Sciences, NTNU, Norway, 1998.

[6] E. N. Gilbert and J. Riordan. Symmetry types of periodic sequences. *Illinois J. Math.*, 5:657–665, 1961.

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

[8] A. Iserles. *A First Course in the Numerical Analysis of Differential Equations*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 1996.

[9] A. Iserles. Multistep methods on manifolds. Technical Report 1997/NA13, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, England, 1997.

[10] M. Kolsrud. Maximal reductions in the Baker–Hausdorff formula. *J. of Math. Phys.*, 34(1):270–285, 1993.

[11] J. D. Lambert. *Numerical Methods for Ordinary Differential Systems*. John Wiley & Sons, 1991.

[12] P. D. Lax. Integrals of Nonlinear Equations of Evolution and Solitary Waves. *Comm. Pure Appl. Math.*, 21:467–490, 1968.

[13] A. Marthinsen. Interpolation in Lie groups and homogeneous spaces. Technical Report 139, Department of Mathematics, Arizona State University, Tempe, Arizona, 1998.

[14] A. Marthinsen, H. Munthe-Kaas, and B. Owren. Simulation of ordinary differential equations on manifolds — some numerical experiments and verifications. *Modeling, Identification and Control*, 18:75–88, 1997.

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

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

[17] H. Munthe-Kaas and B. Owren. Computations in a free Lie algebra. Technical Report No. 148, Department of Informatics, University of Bergen, Norway, 1998. To appear in Philosophical Transactions of the Royal Society.

[18] J. A. Oteo. The Baker–Campbell–Hausdorff formula and nested commutator identities. *J. of Math. Phys.*, 32(2):419–424, 1991.

[19] B. Owren and A. Marthinsen. Runge–Kutta methods adapted to manifolds and based on rigid frames. *BIT*, 39(1):116–142, 1999.

[20] V. S. Varadarajan. *Lie Groups, Lie Algebras, and Their Representations*. GTM 102. Springer-Verlag, 1984.

[21] A. Zanna. The method of iterated commutators for ordinary differential equations on Lie groups. Technical Report 1996/NA12, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, England, 1996.

[22] A. Zanna. Lie-group methods for isospectral flows. Technical Report 1997/NA02, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, England, 1997.