Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational and Applied Mathematics 205 (2007) 802–813
JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS
www.elsevier.com/locate/cam

# New numerical integrators based on solvability and splitting☆

Fernando Casas

*Departament de Matemàtiques, Universitat Jaume I, E-12071 Castellón, Spain*

Received 29 June 2005

**Abstract**

When Lie-group integrators such as those based on the Magnus expansion are applied to linear systems of ODEs, it is necessary to evaluate matrix exponentials. This leads to a reduction in their computational efficiency when the dimension of the matrix is very large. For quadratic Lie groups it is possible to approximate the matrix exponential by a rational function and still preserve the Lie-group structure, but this is no longer true in the important case of the special linear group. In this paper we propose a new class of integration algorithms especially designed to avoid this problem. They are based on expressing the solution as a product of upper and lower triangular matrices obtained explicitly in terms of quadratures. We analyse the main features of the procedure and its feasibility as a practical numerical method.


*MSC:* 65L05

*Keywords:* Lie-group methods; Volume-preserving schemes; Splitting

## 1. Introduction

In spite of its apparent simplicity, except for special situations, the general linear matrix differential equation:

$$Y' = A(t)Y, \quad t \geqslant 0, \quad Y(0) = I \tag{1}$$

has to be solved with numerical integration algorithms. Here $A(t)$ is a sufficiently smooth function to ensure (a) the existence and unicity of a solution matrix $Y(t)$ $n \times n$ and (b) the applicability of the discretization methods to be used. As it is well known, (1) governs the evolution of a great variety of physical systems, from stability analysis to quantum mechanics. It is worth noticing that if $A(t)$ belongs to a Lie algebra $\mathfrak{g}$, then the exact solution $Y(t)$ evolves in the corresponding Lie group $\mathcal{G}$. For instance, in Hamiltonian dynamics $\mathcal{G} \equiv \mathrm{Sp}(n)$, whereas in quantum mechanics $\mathcal{G} \equiv \mathrm{SU}(n)$. Since this is a remarkable feature of the exact solution, it seems convenient that the discretization algorithms provide approximate solutions also evolving in $\mathcal{G}$. These so-called *Lie-group methods* [10] constitute a special class of *geometric numerical integrators*: discretization methods for differential equations preserving their known qualitative features, such as invariant quantities and geometric structure. It is widely recognized that this class of algorithms provide

☆ This work has been partially supported by Ministerio de Educación y Ciencia under project MTM2004-00535 (co-financed by the ERDF of the European Union) and Generalitat Valenciana ('GRUPOS03/002').
*E-mail address:* Fernando.Casas@mat.uji.es.

0377-0427/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2006.01.050

numerical approximations which are more accurate and more stable for significant classes of differential equations in the long term evolution [7].

Two possible (and successful) alternatives to get approximate solutions of Eq. (1) lying in $\mathscr{G}$ are the Magnus and Fer expansions. In the Magnus expansion the flow is represented in the form [12,11]

$$Y(t) = \exp(\Omega(t)),$$

where $\Omega$ is obtained as an infinite series, $\Omega(t) = \sum_{k=1}^{\infty} \Omega_k(t)$, whose terms are linear combinations of integrals and nested commutators involving the matrix $A$ at different times. The first terms read explicitly:

$$\Omega_1(t) = \int_0^t A(t_1)\,\mathrm{d}t_1, \quad \Omega_2(t) = \frac{1}{2}\int_0^t \mathrm{d}t_1 \int_0^{t_1} \mathrm{d}t_2[A(t_1), A(t_2)],$$

where $[X, Y] \equiv XY - YX$. The second approach was introduced by Fer [6] and subsequently proposed (as an exercise!) by Bellman [1, p. 204]. Basically, it states that the solution of (1) can be put in the form

$$Y(t) = \mathrm{e}^{P}\mathrm{e}^{P_1}\cdots\mathrm{e}^{P_n}\cdots,$$

where $P = \int_0^t A(s)\,\mathrm{d}s$, and $P_n = \int_0^t Q_n\,\mathrm{d}s$, with

$$Q_n = \mathrm{e}^{-P_{n-1}} Q_{n-1}\mathrm{e}^{P_{n-1}} + \int_0^{-1} \mathrm{e}^{sP_{n-1}} Q_{n-1}\mathrm{e}^{-sP_{n-1}}\,\mathrm{d}s$$

($Q_0 \equiv A$). The infinite product converges if $t$ is sufficiently small [6,2].

Although both expansions originated within a non-numerical context, in recent years several numerical integrators based on them have been proposed. In particular, methods of order 4, 6 and 8 have been constructed requiring the minimum number of commutators and function evaluations. As a result, the new schemes have proved to be highly competitive with other, more conventional integrators with respect to accuracy and computational effort [3,4]. This is so even when one or several matrix exponentials have to be computed (or consistently approximated) at each integration step.

The problem of calculating matrix exponentials has a long tradition in numerical analysis and several procedures are available in the general setting [13]: rational approximants, Krylov-subspace methods, Schur decomposition, etc. For Lie-group methods, however, it is crucial that the approximation used maps the Lie algebra $\mathfrak{g}$ to the Lie group $\mathscr{G}$. This can be achieved, of course, if the exponential is computed to machine accuracy, but the procedure is expensive for large $n$ and the result is subject to fast error accumulation.

Another possibility consists in approximating $\exp(z)$ by a function $R(z)$ differentiable in a neighborhood of $z = 0$ such that $\mathrm{e}^z = R(z) + \mathcal{O}(h^{p+1})$, where $p \geqslant 1$ is the order of the Lie-group method and $R(\mathfrak{g}) \subseteq \mathscr{G}$. This is the case, for instance, of diagonal Padé approximants and the Cayley transform for quadratic Lie groups such as the orthogonal, the symplectic and the Lorentz groups [10].

There are cases, however, when the approximation of $\exp(z)$ by any such function $R(z)$ does not belong to $\mathscr{G}$ and thus the overall method does not preserve the Lie-group structure. This occurs, in particular, when $\mathscr{G} = \mathrm{SL}(n)$, the special linear group [5], linked in a natural way to systems which preserve volume along their evolution [7].

In this paper we present a procedure to build new numerical integrators for Eq. (1) which do not require the computation of matrix exponentials and nevertheless preserve the structure of the Lie group $\mathrm{SL}(n)$ when $\mathrm{tr}(A) = 0$. The approximate solution is constructed as an infinite product of upper and lower triangular matrices obtained explicitly as solutions of certain linear differential equations in terms of quadratures. The procedure is iterative, but only a few iterations are required to attain methods of extremely high orders. To illustrate the technique, a fourth-order integrator is built and tested on a numerical example. Also a possible generalization to nonlinear systems is considered.

## 2. The procedure

In the linear system $Y' = A(t)Y$, let us denote $Y_0 \equiv Y$, $A_0 \equiv A$, and split the coefficient matrix as

$$A_0(t) = A_{0_+}(t) + A_{0_-}(t),$$

where $A_{0_+} \in \bigtriangledown_n$ is strictly upper-triangular (i.e., all the elements in the main diagonal and below are zero) and $A_{0_-} \in \tilde{\triangle}_n$ is weakly lower-triangular. We represent

$$Y_0(t) = L_0(t)Z(t),$$

where

$$L_0' = A_{0_-}(t)L_0, \quad L_0(0) = I.$$

In consequence, $L_0(t)$ is also lower-triangular and

$$Z' = C_0(t)Z, \quad Z(0) = I \quad \text{where } C_0(t) = L_0^{-1}(t)A_{0_+}(t)L_0(t).$$

Further, we represent

$$Z(t) = U_0(t)Y_1(t), \quad \text{hence } Y_0(t) = L_0(t)U_0(t)Y_1(t),$$

with

$$U_0' = C_{0_+}(t)U_0, \quad U_0(0) = I. \tag{2}$$

Here $C_{0_+} \in \tilde{\bigtriangledown}_n$ is the weakly upper-triangular part of $C_0$, so that $U_0(t)$ is also upper-triangular and can be obtained by solving (2). Now it is easy to show that $Y_1$ satisfies

$$Y_1' = A_1(t)Y_1, \quad Y_1(0) = I,$$

with

$$A_1(t) = U_0^{-1}(t)C_{0_-}(t)U_0(t).$$

This gives a single step of the *solvable cycle*, which can be repeated with $Y_1$ and $A_1$. Thus we split

$$A_1 = A_{1_+} + A_{1_-}, \quad A_{1_+} \in \bigtriangledown_n, \quad A_{1_-} \in \tilde{\triangle}_n$$

and write $Y_1 = L_1U_1Y_2$ with $L_1' = A_{1_-}L_1$, $L_1(0) = I$ and so on. In the end one has the following factorization:

$$Y(t) \equiv Y_0(t) = L_0(t)U_0(t)L_1(t)U_1(t)\cdots L_k(t)U_k(t)Y_{k+1}(t), \quad k = 0, 1, 2, \ldots \tag{3}$$

with

$$\begin{aligned}
&A_k = A_{k_+} + A_{k_-}, \quad A_{k_+} \in \bigtriangledown_n, \quad A_{k_-} \in \tilde{\triangle}_n,\\
&L_k' = A_{k_-}L_k, \quad L_k(0) = I,\\
&C_k \equiv L_k^{-1}A_{k_+}L_k = C_{k_+} + C_{k_-}, \quad C_{k_+} \in \tilde{\bigtriangledown}_n, \quad C_{k_-} \in \triangle_n,\\
&U_k' = C_{k_+}U_k, \quad U_k(0) = I,\\
&A_{k+1} \equiv U_k^{-1}C_{k_-}U_k, \quad Y_{k+1}' = A_{k+1}Y_{k+1}, \quad k = 0, 1, 2, \ldots .
\end{aligned} \tag{4}$$

Usually the factorization is truncated by taking $Y_{k+1} = I$ for a given $k$.

In the sequel we analyse some of the features of this algorithm. To begin with, let us introduce a parameter $\varepsilon > 0$ in the matrix $A$, a common practice in perturbation theory, and consider a Taylor expansion around $t = 0$,

$$A(t) = \varepsilon \sum_{i=0}^{\infty} a_i t^i.$$

When this series is inserted in (4), a straightforward calculation shows that

$$L_0(t) = I + \varepsilon t\alpha_1^{(0)} + \tfrac{1}{2}\varepsilon t^2\alpha_2^{(0)} + \cdots,$$

where $\alpha_j^{(0)}$ is the weakly-triangular part of $a_{j-1}$, $j \geqslant 1$. Then, by computing the product $L_0^{-1} A_{0_+} L_0$ and collecting powers of $\varepsilon$, $t$ we get

$$C_{0_+}(t) = \varepsilon\gamma_1^{(0)} + t(\varepsilon\gamma_2^{(0)} + \varepsilon^2\gamma_3^{(0)}) + \cdots,$$

$$C_{0_-}(t) = t\varepsilon^2\delta_1^{(0)} + t^2(\varepsilon^2\delta_2^{(0)} + \varepsilon^3\delta_3^{(0)}) + \cdots$$

for certain coefficients $\gamma_j^{(0)}$, $\delta_j^{(0)}$ depending on $a_i$, and thus

$$U_0(t) = I + \varepsilon t\beta_1^{(0)} + t^2(\varepsilon\beta_2^{(0)} + \varepsilon^2\beta_3^{(0)}) + \cdots,$$

with $\beta_1^{(0)} = a_0 - \alpha_1^{(0)}$. Next we compute $A_1 = A_{1_+} + A_{1_-}$ as

$$A_{1_-}(t) = t\varepsilon^2\alpha_1^{(1)} + t^2(\varepsilon^2\alpha_2^{(1)} + \varepsilon^3\alpha_3^{(1)}) + \cdots,$$

$$A_{1_+}(t) = t^2\varepsilon^3\beta_1^{(1)} + t^3(\varepsilon^3\beta_2^{(1)} + \varepsilon^4\beta_3^{(1)}) + \cdots,$$

where, in particular, $\alpha_j^{(1)} = \delta_j^{(0)}$, $j = 1, 2, 3$. Proceeding by induction we have, in general,

$$A_{j_-} = t^{n_j}\varepsilon^{n_j}(\varepsilon\alpha_1^{(j)} + t(\varepsilon\alpha_2^{(j)} + \varepsilon^2\alpha_3^{(j)}) + \mathcal{O}(t^2)),$$

$$A_{j_+} = t^{m_j}\varepsilon^{m_j}(\varepsilon\beta_1^{(j)} + t(\varepsilon\beta_2^{(j)} + \varepsilon^2\beta_3^{(j)}) + \mathcal{O}(t^2)). \tag{5}$$

By inserting (5) again in the algorithm (4) and repeating the above procedure for the $j$th solvable cycle one obtains

$$L_j(t) = I + \frac{1}{n_j+1}(t\varepsilon)^{n_j+1}\alpha_1^{(j)} + \frac{1}{n_j+2}t^{n_j+2}\varepsilon^{n_j}(\varepsilon\alpha_2^{(j)} + \varepsilon^2\alpha_3^{(j)}) + \cdots,$$

$$U_j(t) = I + \frac{1}{m_j+1}(t\varepsilon)^{m_j+1}\beta_1^{(j)} + \frac{1}{m_j+2}t^{m_j+2}\varepsilon^{m_j}(\varepsilon\beta_2^{(j)} + \varepsilon^2\beta_3^{(j)}) + \cdots \tag{6}$$

and expressions of type (5) for $A_{(j+1)_-}(t)$ and $A_{(j+1)_+}(t)$, but now with

$$\begin{aligned} n_{j+1} &= n_j + m_j + 1, \\ m_{j+1} &= n_j + 2m_j + 2, \end{aligned} \qquad j = 1, 2, \ldots . \tag{7}$$

The first values of $n_j$ and $m_j$ are collected in the following table:

| $j$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $n_j$ | 1 | 4 | 12 | 33 | 88 |
| $m_j$ | 2 | 7 | 20 | 54 | 143 |

From the table and expressions (6) it is clear that truncating the factorization (3) at $k=0$ ($Y_1 = I$) provides a first-order approximation in $t$, whereas at $k = 1$ one has

$$Y(t) = L_0(t)U_0(t)L_1(t)U_1(t) + \mathcal{O}(t^5\varepsilon^5). \tag{8}$$

Analogously,

$Y(t) \approx L_0U_0L_1U_1L_2$ provides an approximation of order 7,
$Y(t) \approx L_0U_0L_1U_1L_2U_2$ provides an approximation of order 12,
$Y(t) \approx L_0U_0L_1U_1L_2U_2L_3$ provides an approximation of order 20,
$Y(t) \approx L_0U_0L_1U_1L_2U_2L_3U_3$ provides an approximation of order 33

so that with only four solvable cycles we can construct an approximate solution correct up to order 33 in time. These results also show that algorithm (3)–(4) is particularly suitable for systems of the form $Y' = (B_0 + \varepsilon B_1)Y$ when the

equation $U' = B_0 U$ can be solved exactly. In that case the solution can be factorized as $Y = UV$ with $V' = \varepsilon U^{-1} B_1 U V$ and the previous considerations apply to this last equation.

On the other hand, the behaviour of the procedure with respect to the preservation of the Lie-group structure is described by the following theorem:

**Theorem 1.** *If the coefficient matrix $A(t)$ in $Y' = A(t)Y$ belongs to $\mathfrak{sl}(n)$, the Lie algebra of traceless $n \times n$ matrices, then algorithm* (3)–(4) *provides by construction approximate solutions $Y^{[k]}(t)$ in the corresponding Lie group* $\mathrm{SL}(n)$.

**Proof.** Proceeding by induction, it is sufficient to analyse the $k$th step in the algorithm ($k \geqslant 0$). Let us split $A_k$ into its strictly upper-triangular part $A_{k_+} \in \bigtriangledown_n$ and its weakly lower-triangular part $A_{k_-} \in \tilde{\bigtriangleup}_n$. Then $A_{k_-}$ belongs to (a solvable subalgebra of) $\mathfrak{sl}(n)$, since $A_k$ is traceless. Therefore the solution $L_k(t)$ of the initial value problem $L_k' = A_{k_-} L_k$, $L_k(0) = I$, which can be explicitly obtained, lies in (a solvable subgroup of) $\mathrm{SL}(n)$. Next, observe that $\mathrm{tr}(A_{k_+}) = 0$ and the trace is invariant under a similarity transformation, so that

$$\mathrm{tr}(C_k) = \mathrm{tr}(L_k^{-1} A_{k_+} L_k) = \mathrm{tr}(A_{k_+}) = 0$$

and thus $C_k \in \mathfrak{sl}(n)$. Now this matrix is split as $C_k = C_{k_+} + C_{k_-}$, with $C_{k_+} \in \tilde{\bigtriangledown}_n$ and $C_{k_-} \in \bigtriangleup_n$. By applying the same argument as before, it is clear that the solution $U_k$ of the initial value problem $U_k' = C_{k_+} U_k$, $U_k(0) = I$, also belongs to $\mathrm{SL}(n)$ and $A_{k+1} = U_k^{-1} C_{k_-} U_k \in \mathfrak{sl}(n)$. $\square$

In consequence, the factorization (3) constitutes an example of a volume preserving algorithm for linear problems. Other properties, such as orthogonality, symplecticity, etc. associated with matrix Lie groups different from $\mathrm{SL}(n)$ are preserved only up to the order of approximation of the scheme.

To establish rigorously the convergence of the procedure one should require that, in some appropriate norm, $\|L_{k+1} U_{k+1}\| \to 1$ as $k \to \infty$ in (3). In other words, one should look for conditions to be satisfied by the coefficient matrix $A$ such that $A_{k+1} \to 0$ as $k \to \infty$. But

$$\|A_{k+1}\| = \|U_k^{-1} C_{k_-} U_k\| \leqslant \|U_k^{-1}\| \|C_{k_-}\| \|U_k\| \leqslant \kappa(U_k) \|C_k\|,$$

where $\kappa(U)$ denotes the condition number of the matrix $U$. By proceeding similarly from $C_k$, one gets

$$\|A_{k+1}\| \leqslant \kappa(U_k)\kappa(L_k)\|A_k\|$$

and it is not obvious at all how to get useful bounds on $A$ which guarantees convergence from this class of inequalities, so it constitutes an open problem at this stage.

## 3. Construction of numerical integrators

Of course, a prerequisite to build numerical schemes from algorithm (3)–(4) is to compute the matrices $L_k(t)$ and $U_k(t)$, $k \geqslant 0$ by solving the corresponding differential equations.

To simplify matters, let us take in particular $k = 0$ and denote by $a_{ij}(t)$ the elements of $A_0$, $i, j = 1, \ldots, n$, by $L_{ij}(t)$ the elements of $L_0(t)$, $j \leqslant i$, and finally

$$A_{ii}(t) \equiv \int_0^t a_{ii}(s)\, \mathrm{d}s.$$

The lower-triangular matrix $L_0(t)$ can be obtained by integrating in sequence the equation $L_0' = A_{0_-}(t) L_0$, arriving at

$$L_{ij}(t) = \mathrm{e}^{A_{ii}(t)} \left[ \delta_{ij} + (1 - \delta_{ij}) \int_0^t \mathrm{e}^{-A_{ii}(s)} \left( \sum_{k=j}^{i-1} a_{ik}(s) L_{kj}(s) \right) \mathrm{d}s \right], \tag{9}$$

for $i = 1, \ldots, n$ and $j = 1, \ldots, i$. The initial value problem $U_0' = C_{0_+}(t)U_0$, $U_0(0) = I$, can be treated in a similar way. Thus, by denoting

$$C_0(t) = (c_{ij}), \quad U_0(t) = (U_{ij}), \quad C_{ii}(t) = \int_0^t c_{ii}(s)\,\mathrm{d}s, \tag{10}$$

a straightforward calculation shows that

$$U_{ij}(t) = \mathrm{e}^{C_{ii}(t)} \left[ \delta_{ij} + (1-\delta_{ij}) \int_0^t \mathrm{e}^{-C_{ii}(s)} \left( \sum_{k=i+1}^{j} c_{ik}(s) U_{kj}(s) \right) \mathrm{d}s, \right] \tag{11}$$

for $i = 1, \ldots, n$ and $j = i, \ldots, n$. Eqs. (9) and (11) can now be used to calculate the elements of $L_0$ and $U_0$ in terms of multivariate integrals and, by a trivial extension, all $L_k$ and $U_k$. For each value of $i$ we compute first $L_{ii}$ and then, starting with $i = 2$, the elements $L_{ij}$ for $j = 1, \ldots, i-1$ in increasing order. A similar argument applies to $U_0$, but now starting with the last row and proceeding in decreasing order: we evaluate first the diagonal elements $U_{ii}$ and then, starting with $i = n-1$, $U_{ij}$ for $j = i+1, \ldots, n$. Observe that the process is independent and, in fact, it could be amenable for parallel implementation in a computer.

Typically, only in very special circumstances it will be possible to obtain analytical expressions for $L_k$ and $U_k$. Otherwise, the integrals appearing in (9) and (11) have to be approximated by quadratures. To get efficient integrators this has to be done by using as few evaluations of the matrix $A$ per time step as possible.

In the sequel we illustrate the main issues involved in the process by constructing an integrator of order four requiring only two $A$ evaluations per step.

According to (8), for attaining a fourth-order approximation we need two solvable cycles. The first one is fully determined by adopting the following computation scheme:

(1) First we approximate $A_{ii}(h)$, $i = 1, \ldots, n$, up to order 4. This can be done, for instance, with Simpson's rule,

$$\begin{aligned} A_{ii}(h) &= \int_0^h a_{ii}(t)\,\mathrm{d}t = \frac{h}{6}(a_{ii}(0) + 4a_{ii}(h/2) + a_{ii}(h)) + \mathcal{O}(h^5) \\ &\equiv \tilde{A}_{ii}(h) + \mathcal{O}(h^5). \end{aligned} \tag{12}$$

We also need to approximate $A_{ii}(h/2)$, $i = 1, \ldots, n-1$, at least up to order 3 to compute $L_{ij}$. The same function evaluations can be used if we take

$$\begin{aligned} A_{ii}(h/2) &= \frac{h}{24}(5a_{ii}(0) + 8a_{ii}(h/2) - a_{ii}(h)) + \mathcal{O}(h^4) \\ &\equiv \tilde{A}_{ii}(h/2) + \mathcal{O}(h^4). \end{aligned} \tag{13}$$

(2) The computation of $L_0$ proceeds as follows. First we determine the main diagonal elements:

$$\begin{aligned} &L_{ii}(h) = \exp(\tilde{A}_{ii}(h)) + \mathcal{O}(h^5), \quad i = 1, \ldots, n, \\ &L_{ii}(h/2) = \exp(\tilde{A}_{ii}(h/2)) + \mathcal{O}(h^4), \quad i = 1, \ldots, n-1 \end{aligned} \tag{14}$$

and then the corresponding approximations to $L_{ij}(h)$ and $L_{ij}(h/2)$, $j < i$. From (9),

$$L_{ij}(h) = \mathrm{e}^{A_{ii}(h)} \int_0^h F_{ij}(t)\,\mathrm{d}t \quad \text{with } F_{ij}(t) \equiv \mathrm{e}^{-A_{ii}(t)} \sum_{k=j}^{i-1} a_{ik}(t) L_{kj}(t),$$

so that

$$\begin{aligned} &L_{ij}(h) = \mathrm{e}^{\tilde{A}_{ii}(h)} \frac{h}{6}(a_{ij}(0) + 4F_{ij}(h/2) + F_{ij}(h)) + \mathcal{O}(h^5), \\ &L_{ij}(h/2) = \mathrm{e}^{\tilde{A}_{ii}(h/2)} \frac{h}{24}(5a_{ij}(0) + 8F_{ij}(h/2) - F_{ij}(h)) + \mathcal{O}(h^4), \end{aligned} \tag{15}$$

where $F_{ij}(h/2)$ and $F_{ij}(h)$ have to be determined up to order $h^3$.

(3) Now the matrix $C_0$ can be approximated at the quadrature nodes as

$$C_0(0) = A_{0_+}(0),$$
$$C_0(h/2) = L_0^{-1}(h/2)A_{0_+}(h/2)L_0(h/2) \quad \text{with error } \mathcal{O}(h^4),$$
$$C_0(h) = L_0^{-1}(h)A_{0_+}(h)L_0(h) \quad \text{with error } \mathcal{O}(h^5).$$

(4) Just as in Step (1), we approximate now $C_{ii}(h)$ and $C_{ii}(h/2)$ in (10) up to order 4 and 3, respectively, by

$$\tilde{C}_{ii}(h) = \frac{h}{6}(c_{ii}(0) + 4c_{ii}(h/2) + c_{ii}(h)),$$
$$\tilde{C}_{ii}(h/2) = \frac{h}{24}(5c_{ii}(0) + 8c_{ii}(h/2) - c_{ii}(h)).$$

(5) Next, the integrals in (11) are replaced by quadratures in a similar way as $L_{ij}$ in Step (2) (but now starting with the $n$th row). For the diagonal elements we take

$$U_{ii}(h) = \mathrm{e}^{\tilde{C}_{ii}(h)} + \mathcal{O}(h^5), \quad U_{ii}(h/2) = \mathrm{e}^{\tilde{C}_{ii}(h/2)} + \mathcal{O}(h^4),$$

whereas for the remaining ones we have

$$U_{ij}(h) = \mathrm{e}^{C_{ii}(h)} \int_0^h G_{ij}(t)\,\mathrm{d}t \quad \text{with } G_{ij}(t) \equiv \mathrm{e}^{-C_{ii}(t)} \sum_{k=i+1}^{j} c_{ik}(t)U_{kj}(t),$$

and thus

$$U_{ij}(h) = \mathrm{e}^{\tilde{C}_{ii}(h)} \frac{h}{6}\,(c_{ij}(0) + 4G_{ij}(h/2) + G_{ij}(h)) + \mathcal{O}(h^5),$$
$$U_{ij}(h/2) = \mathrm{e}^{\tilde{C}_{ii}(h/2)} \frac{h}{24}\,(5c_{ij}(0) + 8G_{ij}(h/2) - G_{ij}(h)) + \mathcal{O}(h^4). \tag{16}$$

For the second cycle we determine the matrix $A_1$ at the quadrature nodes,

$$A_1(0) = C_{0_-}(0),$$
$$A_1(h/2) = U_0^{-1}(h/2)C_{0_-}(h/2)U_0(h/2) \quad \text{with error } \mathcal{O}(h^4),$$
$$A_1(h) = U_0^{-1}(h)C_{0_-}(h)U_0(h) \quad \text{with error } \mathcal{O}(h^5)$$

and Steps (1)–(5) can be repeated again with $A_1$. Finally, the product $Y_{n+1} = L_0U_0L_1U_1Y_n$ is computed to approximate $Y(t_{n+1} = t_n + h)$. In this way we have shown explicitly that it is possible to construct a method of order 4 with only two $A$ evaluations (three for the first step).

To assess the amount of computational work of the previous algorithm, it is useful to estimate the number of operations involved in each step. As is evident, the main contributions come from Steps (2), (3) and (5) of each solvable cycle, as well as the formation of the matrix $A_1$ and the computation of $Y_{n+1}$.

The calculation of $L_{ij}(h/2)$ requires a total of $2(i-j)+5$ operations, whereas $2(i-j)+4$ are needed for $L_{ij}(h)$. Thus the total number of operations involved in the computation of $L_0(h/2)$ and $L_0(h)$ is

$$\sum_{i=2}^{n}\sum_{j=1}^{i-1}(4i - 4j + 9) = \frac{1}{6}\,n(n-1)(4n+31),$$

i.e., for moderately large values of $n$, about $(2/3)n^3 + (9/2)n^2$ operations (counting the exponential as just one operation). A similar estimate for $U_0$ allows to conclude that the number is now $(2/3)n^3 + 13n^2$, whereas for $L_1$ and $U_1$ one requires $(2/3)n^3 + (15/2)n^2$ and $(2/3)n^3 + 10n^2$ operations, respectively.

To accomplish Step (3) we need to compute $L_0^{-1}$ and several matrix products. Since $L_0$ is triangular, its inverse only requires $(1/3)n(n^2+2)$ operations, whereas the calculation of $C_0$ can be arranged in such a way that $(5/3)n^3 + (1/3)n$ operations are needed. The same considerations apply also to $A_1$.

Finally, taking into account that the product $L_0U_0$ involve $(2/3)n^3 + (1/3)n$ operations, the computation of $L_0U_0L_1U_1Y_n$ can be done with just $(16/3)n^3 - 2n^2$. Adding up all these quantities, one concludes that the total number of operations involved in the fourth-order integration algorithm proposed here is approximately $20n^3 + 33n^2$, ignoring terms of size $\mathcal{O}(n)$. For comparison, a fourth-order integrator based on the Magnus expansion implemented in Matlab requires $\approx 30 - 35n^3$ floating point operations (if the matrix exponential is evaluated up to machine accuracy).

At this point some additional remarks are in order. Firstly, other quadrature rules could also be used (i.e., Gauss–Legendre), but then there are not enough nodes, in general, to approximate all the multivariate integrals required to achieve a given order. In that case it would be necessary to use also matrix evaluations from the previous and next time steps. This is not the case with Newton–Cotes (NC) rules (although the error introduced might be important when high orders are considered). Also for highly oscillatory problems other especially tailored quadratures could be used [9].

Secondly, by following a similar strategy as for the fourth-order method, higher order schemes can be designed. For instance,

$$Y^{[6]} = L_0U_0L_1U_1L_2 \tag{17}$$

provides a sixth-order integrator if the integrals are replaced by a five point NC quadrature rule (requiring four $A$ evaluations per step). If a seven point NC rule is used instead, the order of (17) is risen to 7. Moreover, just by completing the third solvable cycle, $Y^{[12]} = Y^{[6]}U_2$, the procedure leads to a 12th-order scheme (with a NC quadrature rule of 11 points).

Finally, the local extrapolation technique is trivial to implement in this setting, so that the resulting scheme may incorporate an automatic step size selection device. For instance, if we choose

$$Y_1 \equiv L_0U_0L_1, \quad \hat{Y}_1 \equiv L_0U_0L_1U_1 = Y_1U_1,$$

then

$$\|\hat{Y}_1 - Y_1\| = \|Y_1U_1 - Y_1\| = \|Y_1(U_1 - I)\|$$

can be used for the purpose of step size selection when the integration is continued with the fourth-order approximation $\hat{Y}_1$ [8, p. 167].

Next, in order to illustrate the main features of the algorithm proposed and the practical efficiency of the fourth-order integrator previously constructed, we consider a particular linear system (1) evolving in $\mathrm{SL}(n)$ and obtain the corresponding efficiency diagram. Specifically, the elements of the proposed (symmetric) matrix $A(t)$ are

$$A_{ij} = \sin[t(i^2 - j^2)], \quad 1 \leqslant i \leqslant j \leqslant n \tag{18}$$

and $A_{ji} = A_{ij}$, with $n = 10$. Observe that $A_{ii} = 0$ and thus $\mathrm{tr}(A) = 0$.

The integration is carried out with constant time step $h$ in the interval $t \in [0, 10]$ and the approximate solutions are compared with the exact one at the final time $t_f = 10$. There the corresponding error is determined for different values of $h$ by computing the Frobenius norm of the difference. The error is then represented as a function of the computational effort measured in terms of the number of floating point operations required. For comparison we also include the result obtained with the following fourth-order integrator based on the Magnus expansion [11]:

$$\begin{aligned} &A_0 = hA(t_n), \quad A_1 = hA(t_n + h/2), \quad A_2 = hA(t_n + h), \\ &\Omega^{[4]}(h) = \tfrac{1}{6}(A_0 + 4A_1 + A_2) - \tfrac{1}{12}[A_0, A_2], \\ &Y_{n+1} = \exp(\Omega^{[4]}(h))Y_n. \end{aligned} \tag{19}$$

The computation is done in Matlab 5.3 and the function `expm` is used to evaluate the matrix exponential $\exp(\Omega^{[4]}(h))$. The number of operations is measured with the in-built function `flops` and also the previous estimate for the new method. The corresponding efficiency diagrams are plotted (in a log-log scale) in Fig. 1. Lines marked with + are obtained by the new integrator proposed here when (a) the number of operations is determined directly by `flops` (broken line) and (b) by our estimate of $20n^3 + 33n^2$ operations per step (solid line), whereas the solid line with circles corresponds to method (19).

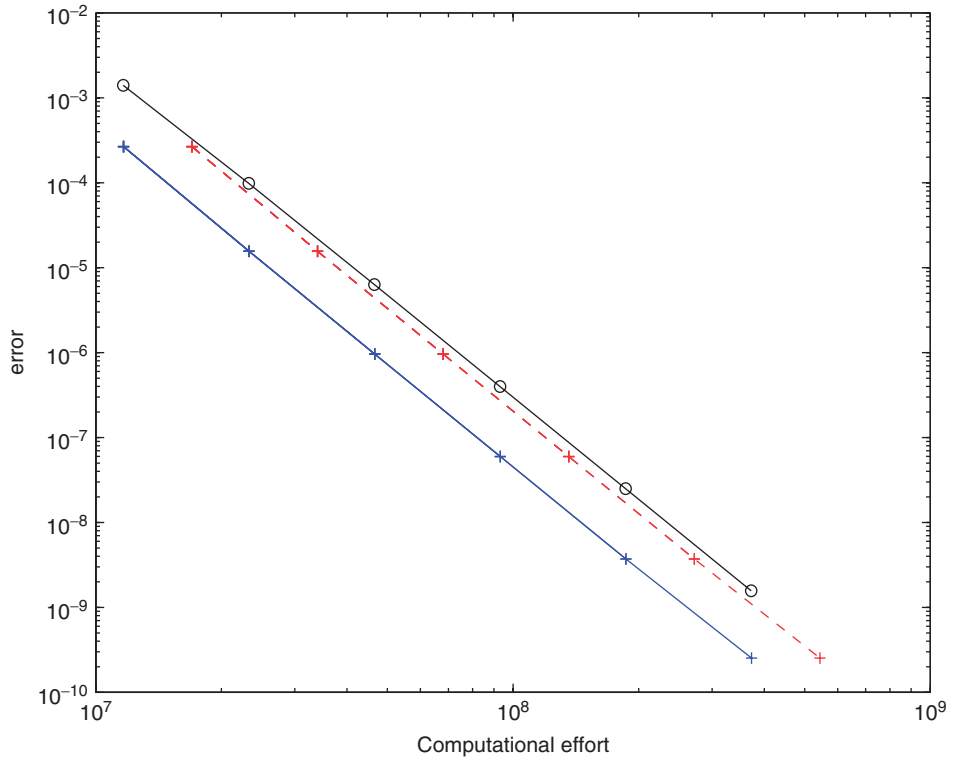


Fig. 1. Efficiency diagram (in logarithmic scale) obtained with the fourth-order integrator based on Magnus (solid line with circles) and the new algorithm (3) based on solvability and splitting when the computational effort is measured with `flops` (broken line with +) and with the estimate $20n^3 + 33n^2$ (solid line with +).

Observe that, even with a direct implementation of the procedure in Matlab (i.e., generic matrices are considered) the new scheme is slightly more efficient than the fourth-order scheme based on Magnus (which corresponds in practice to an estimate of $23n^3$ operations per step for this particular example). The improvement is more noticeable when the structure of triangular matrices is fully exploited to evaluate the matrix inversions and products involved in the algorithm.

## 4. Generalization to nonlinear systems

As a matter of fact, the procedure developed in Section 2 can be generalized to the nonlinear equation

$$Y' = A(t, Y)Y, \quad Y(0) = Y_0 \in \mathrm{GL}(n) \tag{20}$$

in such a way that it possesses essentially the same features and also reproduces the linear case. For simplicity we take $Y_0 = I$ and analyse only the first cycle. The starting point is to build a matrix function $A_-$ in a solvable Lie algebra such that the system

$$L_0' = A_-(t, L_0)L_0, \quad L_0(0) = I \tag{21}$$

can be integrated exactly. In other words, up to an isomorphism, $A_-$ will be lower-triangular, with the following pattern of dependence upon the elements of $L_0$. Let us denote the rows of $L_0$ by

$$\mathbf{l}_1^\top, \mathbf{l}_2^\top, \ldots, \mathbf{l}_n^\top,$$

and use a similar notation for the rows of $A_-$ ($\mathbf{a}_i^\top$) and $I$ ($\mathbf{i}_i^\top$). In addition, let us introduce the matrices

$$C_m = \begin{bmatrix} \mathbf{l}_1^\top \\ \vdots \\ \mathbf{l}_{m-1}^\top \\ \mathbf{i}_m^\top \\ \vdots \\ \mathbf{i}_n^\top \end{bmatrix}, \quad m = 1, 2, \ldots, n. \tag{22}$$

Observe that $C_1 = Y(0) = I$. Now we let

$$\mathbf{a}_1^\top(t, L_0) = [a_{11}(t, C_1), 0, \ldots, 0]$$

and thus $\mathbf{l}_1^{\top\prime} = a_{11}(t, C_1)\mathbf{l}_1^\top$, which is solvable by quadrature. Next we choose

$$\mathbf{a}_2^\top(t, L_0) = [a_{21}(t, C_2), a_{22}(t, C_2), 0, \ldots, 0],$$

rendering

$$\mathbf{l}_2^{\top\prime} = a_{21}(t, C_2)\mathbf{l}_1^\top + a_{22}(t, C_2)\mathbf{l}_2^\top$$

solvable. In general, we let

$$\mathbf{a}_m^\top = [a_{m1}(t, C_m), a_{m2}(t, C_m), \ldots, a_{mm}(t, C_m), 0, \ldots, 0], \quad m = 1, 2, \ldots, n. \tag{23}$$

In this way $A_-$ is lower-triangular and Eq. (21) is solvable by quadrature, thus obtaining explicitly $L_0(t)$. Next we set

$$Y(t) = L_0(t)Z(t), \tag{24}$$

where it follows that

$$Z' = L_0^{-1}[A(t, L_0Z) - A_-(t, L_0)]L_0Z, \quad Z(0) = I. \tag{25}$$

From the previous construction, the matrix elements of $A(t, L_0Z) - A_-(t, L_0)$ are

$$\begin{aligned} &a_{mj}(t, L_0Z) - a_{mj}(t, C_m), \quad 1 \leqslant j \leqslant m, \\ &a_{mj}(t, L_0Z), \quad m + 1 \leqslant j \leqslant n \end{aligned} \tag{26}$$

and $m = 1, \ldots, n$. Eq. (25) can be written as

$$Z' = C_0(t, Z)Z, \quad Z(0) = I \quad \text{where } C_0(t, Z) \equiv L_0^{-1}(A - A_-)L_0$$

and a similar procedure can be applied to solve it. Thus we represent $Z(t) = U_0(t)Y_1(t)$, where

$$U_0' = C_{0_+}(t)U_0, \quad U_0(0) = I. \tag{27}$$

As before, $C_{0_+}$ and $U_0$ are constructed simultaneously with the help of the matrices

$$D_m = \begin{bmatrix} \mathbf{i}_1^\top \\ \vdots \\ \mathbf{i}_{n-(m-1)}^\top \\ \mathbf{u}_{n-m+2}^\top \\ \vdots \\ \mathbf{u}_n^\top \end{bmatrix}, \quad m = 1, 2, \ldots, n.$$

Here $\mathbf{u}_1^\top, \ldots, \mathbf{u}_n^\top$ denote the rows of $U_0$ and the elements of the matrix $C_{0_+}$ are taken as

$$\mathbf{c}_m^\top(t, U_0) = [0, 0, \ldots, c_{mm}(t, D_{n-m+1}), c_{m,m+1}(t, D_{n-m+1}), \ldots, c_{mn}(t, D_{n-m+1})],$$

$m = 1, 2, \ldots, n$, where $c_{ij}$ are the elements of $C_0$. Observe that $C_{0_+}$ is an upper-triangular matrix, so that (27) is solved by quadrature. The equation satisfied by $Y_1$ is finally

$$Y_1' = U_0^{-1}(C_0 - C_{0_+})U_0 Y_1 \equiv A_1(t, Y_1)Y_1, \quad Y_1(0) = I \tag{28}$$

and the process can be repeated again. Notice that this procedure reduces to that presented in Section 2 when (20) is a linear equation.

## 5. Final comments

We have presented a new family of integration schemes for the linear differential equation (1) which provide approximate solutions as a product of upper and lower triangular matrices written in terms of multivariate integrals. They do not require the computation of matrix exponentials and nevertheless the approximations obtained evolve in $\mathrm{SL}(n)$ when the trace of the coefficient matrix $A(t)$ vanishes. In this sense they can be considered as a new class of volume-preserving integrators for any dimension $n$.

Although their implementation is not trivial, especially for high orders, it should be stressed that these methods only need to replace the multivariate integrals appearing in the algorithm by conveniently chosen quadrature rules. In this work we have considered Newton–Cotes rules, but other options are perfectly valid.

The new schemes are particularly appropriate to carry out numerical integrations in the Lie group $\mathrm{SL}(n)$ when the dimension $n$ is very large. In that case they are likely to be more efficient than other classes of Lie-group methods such as those based on Magnus and Fer expansions. Also when the matrix $A(t)$ contains a perturbation parameter $\varepsilon$ the approximate solutions obtained with a few solvable cycles differ from the exact one only in very high powers of $\varepsilon$.

A possible generalization to nonlinear equations of the form $Y' = A(t, Y)Y$ has been also proposed. The treatment is certainly more involved, but it embraces in a natural way the linear case and might constitute a novel approach for the numerical integration of nonlinear problems in $\mathrm{SL}(n)$.

The algorithm exposed here can equally be applied to autonomous linear systems. In that situation it might constitute a novel procedure for approximating the matrix exponential $\mathrm{e}^{At}$.

## Acknowledgements

This work has been done at the Numerical Analysis Group of the University of Cambridge on sabbatical leave from the Universitat Jaume I. Financial support from the Ministerio de Educación y Ciencia and the Generalitat Valenciana (through their mobility grants programs) are gratefully acknowledged. The author wishes to thank Arieh Iserles for suggesting to him the study of this problem and for his insightful remarks, and the University of Cambridge for its hospitality.

## References

[1] R. Bellman, Introduction to Matrix Analysis, McGraw-Hill, New York, 1960.
[2] S. Blanes, F. Casas, J.A. Oteo, J. Ros, Magnus and Fer expansions for matrix differential equations: the convergence problem, J. Phys. A: Math. Gen. 31 (1998) 259–268.
[3] S. Blanes, F. Casas, J. Ros, Improved high order integrators based on the Magnus expansion, BIT 40 (2000) 434–450.
[4] S. Blanes, F. Casas, J. Ros, High order optimized geometric integrators for linear differential equations, BIT 42 (2002) 262–284.
[5] K. Feng, Z. Shang, Volume-preserving algorithms for source-free dynamical systems, Numer. Math. 71 (1995) 451–463.
[6] F. Fer, Résolution de l'équation matricielle $U' = pU$ par produit infini d'exponentielles matricielles, Bull. Classe Sci. Acad. Roy. Bel. 44 (1958) 818–829.
[7] E. Hairer, C. Lubich, G. Wanner, Geometric Numerical Integration. Structure-Preserving Algorithms for Ordinary Differential Equations, Springer Series in Computational Mathematics, vol. 31, Springer, Berlin, 2002
[8] E. Hairer, S.P. Nørsett, G. Wanner, Solving Ordinary Differential Equations I, second ed., Springer Series in Computational Mathematics, vol. 8, Springer, Berlin 1993.
[9] A. Iserles, On the global error of discretization methods for highly-oscillatory ordinary differential equations, BIT 42 (2002) 561–599.

[10] A. Iserles, H.Z. Munthe-Kaas, S.P. Nørsett, A. Zanna, Lie group methods, Acta Numer. 9 (2000) 215–365.
[11] A. Iserles, S.P. Nørsett, On the solution of linear differential equations in Lie groups, Philos. Trans. Roy. Soc. London Ser. A 357 (1999) 983–1019.
[12] W. Magnus, On the exponential solution of differential equations for a linear operator, Comm. Pure Appl. Math. 7 (1954) 649–673.
[13] C. Moler, C. Van Loan, Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later, SIAM Rev. 45 (2003) 3–49.