

Lawrence Berkeley National Laboratory

LBL Publications

Title

Parallel-In-Time Magnus Integrators

Permalink

<https://escholarship.org/uc/item/8r38q095>

Journal

SIAM Journal on Scientific Computing, 41(5)

ISSN

1064-8275

Authors

Krull, Brandon
Minion, Michael

Publication Date

2019

DOI

10.1137/18M1174854

Peer reviewed

PARALLEL-IN-TIME MAGNUS INTEGRATORS

B. T. KRULL AND M. L. MINION *

Abstract. Magnus integrators are a subset of geometric integration methods for the numerical solution of ordinary differential equations that conserve certain invariants in the numerical solution. This paper explores temporal parallelism of Magnus integrators, particularly in the context of nonlinear problems. The approach combines the concurrent computation of matrix commutators and exponentials within a time step with a pipelined iteration applied to multiple time steps in parallel. The accuracy and efficiency of time parallel Magnus methods up to order six are highlighted through numerical examples and demonstrate that significant parallel speedup is possible compared to serial methods.

Key words. Ordinary differential equations, nonlinear ordinary differential equations, Magnus expansions, isospectral flows, Lax pairs, parallel-in-time

AMS subject classifications. 34L30, 65L05, 65Y05

1. Introduction. The solution of ordinary differential equations (ODEs) is a well established field with applications across the spectrum of scientific disciplines. Numerical methods date back at least to Euler’s work in 1768 [5], and the accuracy, stability, and efficiency of various methods is well studied (see for example Refs. [8, 9]). In more recent years, the study of specialized numerical methods for ODEs that preserve certain mathematical properties of the numerical solution has seen increased interest. Examples include methods for Hamiltonian systems that numerically conserve invariants of the true dynamical system such as the energy or angular momentum. More generally, the true solution of an ODE posed in N -dimensional space may reside for all time on a manifold \mathcal{M} of dimension $d < N$, and the goal is to devise a method for which the numerical solution will also remain on \mathcal{M} . Such methods are referred to in general as *geometric integrators*. The interested reader is encouraged to consult Ref. [7] for a comprehensive introduction to the subject.

As a concrete example, consider the ODE given by

$$(1.1) \quad \frac{d}{dt}Y(t) = F(Y(t)), \quad Y(0) = Y_0,$$

where Y and $F(Y)$ are both $N \times N$ matrices. Depending on the form of F , certain properties of the initial value Y_0 may be preserved for all time, such as the determinant, orthogonality, idempotency, or the spectrum of eigenvalues. In general, standard numerical methods such as linear multistep or Runge-Kutta methods will not produce solutions that conserve such properties [16].

Magnus integrators are a subset of geometric integrators based on the expansion proposed by Wilhelm Magnus in 1954 [14]. The Magnus expansion is closely tied to the concept of Lie groups and algebras due to the presence of matrix commutators, also known as Lie brackets. The discussion concerning solutions to Eq. (1.1) can in fact be generalized to differential equations on Lie groups (see e.g. Ref. [12]), however in this work we strictly use matrix valued solutions. Magnus integrators can also be viewed as a type of exponential integrator (see e.g. Ref. [11] for a review) since they require that the matrix exponential be evaluated. The comprehensive review of the Magnus expansion and applications by Blanes, et. al. [1] provides

*Center for Computational Sciences and Engineering, Lawrence Berkeley National Laboratory, Berkeley, CA, 94720.

43 a description of several physical applications to which the Magnus expansion has
 44 been applied including nuclear, atomic, and molecular dynamics; nuclear magnetic
 45 resonance; quantum field theory and high energy physics; electromagnetism; optics;
 46 geometric control of mechanical systems; and the search for periodic orbits.

47 In general, numerical Magnus integrators are constructed by applying quadrature
 48 rules of suitable order to a truncation of the Magnus expansion. The review [12] presents
 49 various types of Magnus integrators up to sixth-order, and methods of order up to eight
 50 are considered in Ref. [2]. As the order increases, the number of commutator terms
 51 required in the Magnus expansion grows quickly, hence in these papers and others, a
 52 detailed discussion of how to minimize the number of commutators required for a given
 53 order is presented. So called *commutator-free* Magnus integrators (e.g. Refs. [3, 18])
 54 have also been proposed that replace the need to compute matrix commutators with
 55 additional matrix exponentials. This can reduce the total computational cost of
 56 the method depending on the relative cost of computing commutators versus matrix
 57 exponentials. In this paper, an additional avenue for reducing the time to solution for
 58 Magnus integrators is investigated, namely parallelization in time.

59 The study of parallel numerical methods for ODEs dates back at least to Nievergelt
 60 in 1964 [17], and the field has seen an increase in activity in the last 15 years. (See
 61 Ref. [6] for a recent review.) The standard classification of parallel methods for ODEs
 62 includes parallelism across the method, across the problem, and across the time steps
 63 [4]. In this work we demonstrate the utility of parallelization across both the method
 64 and the time steps for Magnus integrators for both linear and nonlinear equations.
 65 Special attention is paid to schemes for solving nonlinear differential equations for
 66 isospectral flow problems, although the methodology that is described can be applied
 67 more generally.

68 The remainder of this paper is organized as follows. Section 2 presents the
 69 mathematical background behind the Magnus expansion and Magnus integrators
 70 followed by some specific Magnus integrators based on Gaussian collocation in section 3.
 71 The parallelization strategies for these integrators is presented in section 4. Numerical
 72 results comparing the efficiency of different parallel methods is presented in section
 73 5. The results demonstrate that significant parallel speedup over serial methods is
 74 possible and show that parallel higher-order methods are superior in terms of accuracy
 75 and time to solution compared to lower-order methods.

76 **2. Mathematical Preliminaries.** In this section, a review of the mathematics
 77 behind the construction of Magnus integrators for both linear and nonlinear problems
 78 is reviewed.

79 **2.1. Matrix Calculus and Differential Equations.** Given a constant matrix
 80 $A \in \mathbb{F}^{N \times N}$, where \mathbb{F} can be either \mathbb{R} or \mathbb{C} , the exponential of A is defined by the power
 81 series

$$82 \quad (2.1) \quad e^A = \sum_{n=0}^{\infty} \frac{A^n}{n!}.$$

83 Given the time-dependent vector $y(t) \in \mathbb{F}^N$, the solution to the differential equation

$$84 \quad (2.2) \quad y'(t) = Ay(t), \quad y(0) = y_0$$

85 is given by

$$86 \quad (2.3) \quad y(t) = e^{At} y_0.$$

87 This is easily shown by differentiating the power series definition (2.1) of the exponential
88 on the right hand side term-by-term to give

$$89 \quad (2.4) \quad \frac{d}{dt}e^{At} = Ae^{At}.$$

90 Now consider the more general case of a time-dependent matrix $A(t)$. Again
91 differentiating definition (2.1) term-by-term and using the product rule yields

$$92 \quad (2.5) \quad \frac{d}{dt}e^{A(t)} = A'(t) + \frac{A(t)A'(t) + A'(t)A(t)}{2!} + \frac{A(t)'A(t)^2 + A(t)A'(t)A(t) + A(t)^2A'(t)}{3!} \dots$$

93 The right hand side of (2.5) can be rearranged to give

$$94 \quad (2.6) \quad \frac{d}{dt}e^{A(t)} = \text{dexp}_{A(t)}(A'(t))e^{A(t)},$$

95 where the operator on the right is defined by

$$96 \quad (2.7) \quad \text{dexp}_{A(t)}(X(t)) = X(t) + \frac{[A(t), X(t)]}{2!} + \frac{[A(t), [A(t), X(t)]]}{3!} \dots,$$

97 and brackets $[\cdot, \cdot]$ denote the matrix commutator $[A, X] = AX - XA$.

98 Now consider the linear system of ODEs

$$99 \quad (2.8) \quad y'(t) = A(t)y(t), \quad y(0) = y_0.$$

100 To find a solution, suppose that it can be written in the form

$$101 \quad (2.9) \quad y(t) = e^{\Omega(t)}y_0$$

102 for some matrix $\Omega(t)$. Using Eq. (2.6), $\Omega(t)$ satisfies

$$103 \quad (2.10) \quad \text{dexp}_{\Omega(t)}(\Omega'(t)) = A(t), \quad \Omega(0) = 0.$$

104 The same formal derivation can be applied to a nonlinear system of equations

$$105 \quad (2.11) \quad y'(t) = A(y(t), t)y(t), \quad y(0) = y_0.$$

106 Again, if the solution to this equation is to take the form of Eq. (2.9), then $\Omega(t)$
107 satisfies

$$108 \quad (2.12) \quad \text{dexp}_{\Omega(t)}(\Omega'(t)) = A(y(t), t), \quad \Omega(0) = 0.$$

109 In the next section, methods for finding $\Omega(t)$ are considered.

110 **2.2. Magnus Expansion.** In 1954, Magnus introduced an explicit expression
111 for the solution of Eq. (2.8), which is reviewed here [14]. The first step is the inversion
112 of the operator $\text{dexp}_{\Omega(t)}$, which gives a differential equation for $\Omega(t)$

$$113 \quad (2.13) \quad \Omega'(t) = \sum_{n=0}^{\infty} \frac{B_n}{n!} \text{ad}_{\Omega(t)}^n(A(t)), \quad \Omega(0) = 0,$$

114 where the B_n are the Bernoulli numbers and

$$115 \quad (2.14) \quad \text{ad}_{\Omega}^k(X) = [\Omega, \text{ad}_{\Omega}^{k-1}X], \quad \text{ad}_{\Omega}^0(X) = X.$$

116 Next, a Picard-type iteration is applied to Eq. (2.13)

$$117 \quad (2.15) \quad \Omega^{k+1}(t) = \int_0^t \sum_{n=0}^{\infty} \frac{B_n}{n!} \text{ad}_{\Omega^k}^n(A(t)).$$

118 Collecting terms in Eq. (2.15) yields an infinite series for $\Omega(t)$

$$119 \quad (2.16) \quad \Omega(t) = \Omega^{(1)}(t) + \Omega^{(2)}(t) + \Omega^{(3)}(t) + \dots,$$

120 where

$$121 \quad (2.17) \quad \Omega^{(1)}(t) = \int_0^t d\tau A(\tau)$$

122

$$123 \quad (2.18) \quad \Omega^{(2)}(t) = \frac{1}{2} \int_0^t d\tau_2 \int_0^{\tau_2} d\tau_1 [A(\tau_2), A(\tau_1)]$$

124

$$125 \quad (2.19) \quad \Omega^{(3)}(t) = \frac{1}{6} \int_0^t d\tau_3 \int_0^{\tau_3} d\tau_2 \int_0^{\tau_2} d\tau_1 [A(\tau_3), [A(\tau_2), A(\tau_1)]] + [[A(\tau_3), A(\tau_2)], A(\tau_1)]$$

126

$$127 \quad \Omega^{(4)}(t) = \frac{1}{12} \int_0^t d\tau_4 \int_0^{\tau_4} d\tau_3 \int_0^{\tau_3} d\tau_2 \int_0^{\tau_2} d\tau_1 [[[A(\tau_4), A(\tau_3)], A(\tau_2)], A(\tau_1)]$$

$$128 \quad \quad \quad + [A(\tau_4), [[A(\tau_3), A(\tau_2)], A(\tau_1)]]$$

$$129 \quad \quad \quad + [A(\tau_4), [A(\tau_3), [A(\tau_2), A(\tau_1)]]]$$

$$130 \quad (2.20) \quad \quad \quad + [A(\tau_3), [A(\tau_2), [A(\tau_1), A(\tau_4)]]].$$

131 Each subsequent term in the series contains an additional integration operator, com-
 132 mutators of one higher order, as well as an increasing number of commutator terms.
 133 The reader is referred to the original work of Magnus [14] or the extensive review [1]
 134 for further details. To summarize, the Magnus expansion gives an explicit formula for
 135 the solution of the linear equation given by Eq. (2.8) in the form of the exponential of
 136 a matrix defined by an infinite series given by Eq. (2.16).

137 **2.3. The Magnus Expansion for Nonlinear Problems.** The same formal
 138 procedure used in the last section to construct the solution to the linear problem
 139 Eq. (2.8) can also be applied to the nonlinear system Eq. (2.11). One can still represent
 140 the solution in terms of the exponential of the function $\Omega(t)$, and the only difference
 141 is that in the Magnus expansion terms given above in Eqs. (2.17)-(2.20), the terms
 142 $A(\tau)$ must be replaced with $A(y(\tau), \tau)$, which by the definition of the solution is
 143 $A(e^{\Omega(\tau)}y_0, \tau)$. Although this may appear at first a small change in notation, the
 144 implication is quite important. For the nonlinear problem, the Magnus expansion
 145 does not give an explicit formula for the function $\Omega(t)$ as in the linear case. Instead,
 146 the result is an equation for $\Omega(t)$ involving an infinite expansion of terms containing
 147 integrals of commutators dependent on $\Omega(t)$. The central insight of this paper is that
 148 this equation for $\Omega(t)$ can be solved efficiently by a fixed point iteration that is readily
 149 amenable to parallelization in the time direction.

150 **2.4. Isospectral Flows.** A special type of matrix differential equation for which
 151 the eigenvalues of the solution are independent of time is called *isospectral flow*.
 152 Problems of this form exist in application domains including electronic structure, wave
 153 dynamics, and linear algebra. Isospectral flow is often associated with the concept of
 154 a Lax pair: two matrices or operators $A(t), Y(t)$ dependent on time that satisfy the
 155 Lax equation

$$156 \quad (2.21) \quad Y'(t) = [A(Y, t), Y(t)], \quad Y(0) = Y_0,$$

157 where $Y(t), A(Y, t) \in \mathbb{F}^{N \times N}$.

158 It is straightforward to show that the solution to Eq. (2.21) can be written in the
 159 form of the transformation

$$160 \quad (2.22) \quad Y(t) = (e^{\Omega(t)})Y_0(e^{\Omega(t)})^{-1},$$

161 where $\Omega(t)$ is defined by the Magnus expansion with respect to $A(t)$. Since the form of
 162 $Y(t)$ takes on a similarity transformation, the eigenvalues of $Y(t)$ do not change in time,
 163 hence the term *isospectral*. In the special case where A is Hermitian (or self-adjoint)
 164 and Y is skew-Hermitian (or skew-adjoint), the exponential $e^{\Omega(t)}$ is unitary, which
 165 reduces Eq. (2.22) to

$$166 \quad (2.23) \quad Y(t) = (e^{\Omega(t)})Y_0(e^{\Omega(t)})^\dagger.$$

167 **3. Numerical Methods Based on the Magnus Expansion.** In this section,
 168 the process for constructing numerical methods for differential equations based on the
 169 Magnus expansion is discussed. In general, numerical methods are constructed by
 170 designing appropriate quadrature rules for the Magnus expansion truncated to a given
 171 order. The presentation here is focused on collocation type schemes based on Gaussian
 172 quadrature rules. As proved in Ref. [13], quadrature rules for the terms in the Magnus
 173 expansion based on s Gauss-Legendre quadrature nodes are sufficient for constructing
 174 a method of order $2s$. Here, methods of order two, four, and six are considered
 175 using both Gauss-Legendre and Gauss-Lobatto quadrature nodes. These methods
 176 correspond to quadrature rules applied to one, two, and four terms, respectively, in
 177 Eq. (2.16).

178 Considerable attention in the literature on Magnus integrators is devoted to de-
 179 signing methods requiring the minimum number of function evaluations and matrix
 180 commutators for a given order of accuracy [13, 1, 12]. In the context of time paral-
 181 lelization, the manner in which the cost of commutators and function evaluations are
 182 counted must reflect the fact that much of the work can be done in parallel, and the
 183 minimum parallel cost is not necessarily achieved by a direct parallelization of the
 184 serial method with the fewest number of commutators.

185 Methods for linear equations are discussed first, followed by a discussion of
 186 additional considerations for nonlinear problems in section 3.3.

187 **3.1. Quadrature Rules for the Magnus Expansion.** In this section, the spe-
 188 cific types of quadrature rules used in the numerical methods are described. Quadrature
 189 rules based on Gauss-Lobatto or Gauss-Legendre quadrature rules using either two or
 190 three quadrature nodes are considered here. Table 1 lists the specific nodes used for
 191 each choice as well as the accompanying classical weights. For a method of a given
 192 order, each term in the truncated Magnus expansion must be approximated using the
 193 function values $A_m = A(t_m)$ (or $A_m = A(y(t_m), t_m)$ for nonlinear problems) at the
 194 quadrature nodes t_m corresponding to the quadrature nodes scaled to the time step

Name	Order	Nodes	$q^{(1)}$
Lob-2	2	0, 1	$\frac{1}{2}, \frac{1}{2}$
Lob-3	4	$0, \frac{1}{2}, 1$	$\frac{1}{6}, \frac{4}{6}, \frac{1}{6}$
Leg-3	6	$\frac{1}{2} - \frac{1}{2}\sqrt{\frac{3}{5}}, \frac{1}{2}, \frac{1}{2} + \frac{1}{2}\sqrt{\frac{3}{5}}$	$\frac{5}{18}, \frac{8}{18}, \frac{5}{18}$

TABLE 1

Quadrature nodes and weights for Gauss-Legendre and Gauss-Lobatto rules.

195 interval $[t_n, t_{n+1}]$. For the schemes described below, the same quadrature nodes are
 196 used at each term in the expansion in Eq. (2.16).

197 First consider the approximation to the first term of the expansion $\Omega^{(1)}(t)$ on
 198 the interval $[t_n, t_{n+1}]$ with $\Delta t = t_{n+1} - t_n$. Approximating the integral by Gaussian
 199 quadrature gives

$$200 \quad (3.1) \quad \Omega^{(1)}(t_{n+1}) = \int_{t_n}^{t_{n+1}} A(t) dt \approx \Delta t \sum_{j=1}^M q_j^{(1)} A_j = \Omega_{n+1}^{(1)},$$

201 where M is the number of quadrature nodes. This is classical quadrature, and the
 202 well-known coefficients $q_j^{(1)}$ are given for completeness in Table 1.

203 In order to obtain a fourth-order method, the second term in the Magnus expansion
 204 must be included. The simplest approximation to $\Omega_{n+1}^{(2)}$ sufficient for fourth-order
 205 accuracy requires the calculation of only a single commutator term

$$206 \quad (3.2) \quad \Omega_{n+1}^{(2)-1} = \Delta t^2 q^{(2)-1} [A_1, A_3],$$

207 with $q^{(2)-1} = 1/12$. The method denoted Lob-4-1 (where the 1 denotes one commutator
 208 term) uses this approximation. To compute $\Omega^{(2)}$ to the accuracy required for a
 209 sixth-order method, three nodes can be used and it is necessary to compute three
 210 commutators

$$211 \quad (3.3) \quad \Omega_{n+1}^{(2)-3} = q_1^{(2)-3} [A_1, A_2] + q_2^{(2)-3} [A_1, A_3] + q_3^{(2)-3} [A_2, A_3]$$

212 with the values

$$213 \quad (3.4) \quad q_j^{(2)-3} = [-7.1721913818656e-2, -3.5860956909328e-2, -7.1721913818656e-2].$$

214 Despite the increased computational cost of two additional commutators, in a parallel
 215 implementation all three commutators can be computed simultaneously.

216 To achieve sixth-order accuracy, the first four terms of the Magnus expansion
 217 must be included. The sixth-order method denoted Leg-6 approximates the $\Omega^{(3)}$ term
 218 using three Gauss-Legendre nodes following the discussion in Ref. [12]. Specifically,

$$219 \quad (3.5) \quad \Omega_{n+1}^{(3)} = \Delta t^3 ([q_{1,1}^{(3)} A_1 + q_{1,2}^{(3)} A_2 + q_{1,3}^{(3)} A_3, [A_1, A_2]] + \\
 220 \quad [q_{2,1}^{(3)} A_1 + q_{2,2}^{(3)} A_2 + q_{2,3}^{(3)} A_3, [A_1, A_3]] + \\
 221 \quad [q_{3,1}^{(3)} A_1 + q_{3,2}^{(3)} A_2 + q_{3,3}^{(3)} A_3, [A_2, A_3]]).$$

223 The values of the coefficients $q_{i,j}^{(3)}$ are the same as those in Ref. [12], and in matrix

224 form are
 225 (3.6)

$$q^{(3)} = \begin{bmatrix} 3.4538506760729e-3 & -5.5849500293944e-3 & -7.1281599059377e-3 \\ 1.6534391534391e-3 & 0.0 & -1.6534391534391e-3 \\ 7.1281599059377e-3 & 5.5849500293945e-3 & -3.4538506760729e-3 \end{bmatrix}.$$

226 Exploiting the linear property of commutators ($[A, X] + [A, Y] = [A, X + Y]$) allows
 227 one to combine terms that share the same inner single commutator and reduce the
 228 number of commutators from nine to three. Note that the single commutator terms,
 229 i.e. $[A_1, A_2]$, are computed during the formation of the $\Omega_{n+1}^{(2)}$ term and need not be
 230 computed again.

231 The fourth term in the Magnus expansion can be approximated using a low-order
 232 quadrature for a sixth-order method. Following the discussion in Ref. [2], the fourth
 233 term is approximated by

$$234 \quad (3.7) \quad \Omega_{n+1}^{(4)} = \Delta t^4 q^{(4)} [B_0, [B_0, [B_0, B_1]]],$$

235 where $q^{(4)} = 1/60$ and

$$236 \quad (3.8) \quad B_i = \Delta t \sum_{j=1}^3 q_j^{(1)} (t_j - 0.5)^i A_j$$

237 with $q_j^{(1)}$ given in the last row of Table 1.

238 In section 5, numerical examples are presented for five different Magnus integrators.
 239 Table 2 lists the specific discretization of each term included for a given method. The
 240 overall order of each method is determined either by the number of terms used in the
 241 expansion, or the order of the quadrature rules. For example, the methods Leg-2, Leg-
 242 4-3, and Leg-6 use the same quadrature nodes, but differ in the number of terms used
 243 in the expansion, while Lob-2 and Leg-2 use different nodes, but are both second-order
 because only one term in the expansion is used.

Name	Order	Nodes	Ω
Lob-2	2	Lob-2	$\Omega^{(1)}$
Leg-2	2	Leg-3	$\Omega^{(1)}$
Lob-4-1	4	Lob-3	$\Omega^{(1)} + \Omega^{(2)-1}$
Leg-4-3	4	Leg-3	$\Omega^{(1)} + \Omega^{(2)-3}$
Leg-6	6	Leg-3	$\Omega^{(1)} + \Omega^{(2)-3} + \Omega^{(3)} + \Omega^{(4)}$

TABLE 2
 Description of the numerical schemes.

244

245 **3.2. The Matrix Exponential and Solution Update.** Once all of the quadra-
 246 ture approximations are applied and the value of Ω_{n+1} is computed, the solution can
 247 be updated by

$$248 \quad (3.9) \quad y_{n+1} = e^{\Omega_{n+1}} y_n.$$

249 There are many approaches to computing the product of a matrix exponential and a
 250 vector of the form $e^A y$ [15], some of which explicitly compute the term e^A and some
 251 which only approximate the product $e^A y$. The choice of method is problem dependent
 252 and does not affect the discussion of time parallelism of the methods. In the numerical
 253 examples presented here, the scaling-and-squaring method from [10] is used to form
 254 the matrix exponential explicitly.

255 **3.3. Considerations for Nonlinear ODEs.** Consider now problems of the
 256 form

$$257 \quad (3.10) \quad y' = A(y, t)y$$

258 where non-linearity is introduced through the y -dependence of A . The terms in the
 259 Magnus expansion approximations introduced above now depend on the solution
 260 through A_m , and cannot simply be evaluated. The numerical solution at each quadra-
 261 ture node t_m will be denoted y_m , and hence $A_m = A(y_m, t_m)$. The values of y_m at
 262 each quadrature node are computed by

$$263 \quad (3.11) \quad y_m = e^{\Omega_m} y_n,$$

264 where Ω_m is an approximation to the Magnus expansion on the interval $[t_n, t_m]$. The
 265 construction of Ω_m is discussed below.

266 A simple fixed-point Picard-type iteration is used to simultaneously solve for the
 267 values Ω_m and y_m . The iterative scheme is initialized by setting $y_m^{k=1} = y_n$ at each
 268 node m , where k denotes the iteration. The solution at each quadrature node is
 269 updated by

$$270 \quad (3.12) \quad y_m^{k+1} = e^{\Omega_m^k} y_n.$$

271 Then Ω_m^{k+1} is computed using values $A(y_m^{k+1}, t_m)$ as described below.

272 To compute Ω_m^k , the process for constructing the quadrature rules in section 3.1
 273 needs to be applied to each quadrature node. Evaluating the values $A(y_m^k, t_m)$ at
 274 each node t_m is straight-forward but needs to be performed each iteration. In all
 275 cases considered here, the same matrix commutators are used for each quadrature
 276 rule, so computing the commutators is also identical to the linear case. The significant
 277 difference is that a quadrature rule for each term in the Magnus expansion must be
 278 computed for each interval $[t_n, t_m]$ rather than just $[t_n, t_{n+1}]$ as in the linear case. The
 279 coefficients for each of the terms are included in Appendix A.

280 Once each Ω_m^k is computed, the matrix exponential can be computed for each
 281 node, and a new solution is obtained at each node by Eq. (3.12). The solution is
 282 considered converged when the maximum absolute value of the residual

$$283 \quad (3.13) \quad R_m^k = e^{\Omega_m^k} y_n - y_m^k = y_m^{k+1} - y_m^k$$

284 is less than a predefined tolerance. In a traditional implementation of this iteration
 285 using Gauss-Legendre nodes, it is not necessary to compute the values Ω_{n+1}^k and y_{n+1}
 286 at the end of the time step during the iterations; however, when pipelining of iterations
 287 is employed as discussed in the next section, y_{n+1} is computed each iteration to update
 288 the initial condition for the next time step.

289 **3.4. Considerations for Isospectral Flows.** Consider now problems of the
 290 form of Eq. (2.21). The procedure for constructing Magnus integrators follows exactly
 291 that laid out in the previous section except that the solution is defined at quadrature
 292 nodes by

$$293 \quad (3.14) \quad Y_m^{k+1} = e^{\Omega_m^k} Y_n e^{-\Omega_m^k},$$

294 and likewise for the computation of Y_{n+1}^{k+1} from Ω_{n+1}^k .

295 **4. Parallelization in Time for Magnus Integrators.** In this section, we
 296 investigate the theoretical computational cost of the Magnus integrators introduced in
 297 the previous section in both serial and parallel settings. We consider both parallelization
 298 across the method and parallelization across the time steps. In the following discussion,
 299 it is assumed that arbitrarily many processors are available for a given problem and
 300 the cost of communication between processors is ignored. It is also assumed that the
 301 matrix exponential is formed explicitly as is done for the the numerical results given
 302 in Section 5.

303 **4.1. The Linear Case.** First consider linear problems where $A(t)$ does not
 304 depend on the solution y . For each of the N time steps, the following tasks must be
 305 performed:

- 306 L1. Evaluate $A_m = A(t_m)$ for each quadrature node t_m
- 307 L2. Compute commutators necessary for each term in the truncated Magnus
 308 expansion
- 309 L3. Apply quadrature rules to compute $\Omega(t_{n+1})$
- 310 L4. Form the exponential of $\Omega(t_{n+1})$
- 311 L5. Compute the solution y_{n+1} from y_n by matrix multiplication

312 Denote by C_A the computational cost of computing $A(t)$ for a given time. Then if M
 313 quadrature nodes are used, L1 has a computational cost of MC_A . Next let n^p denote
 314 the number of commutators required to compute the p th term in the Magnus expansion
 315 and C_C the cost of computing one commutator. Assuming that each commutator
 316 in the $p + 1$ term can be formed with one additional commutator applied to a term
 317 from term p , the total number of commutators to compute is simply $n^1 + \dots + n^P$.
 318 Denoting this sum by N_C , the cost of L2 is $N_C C_C$. Task L3 requires only that a
 319 linear combination of the terms computed in L2 be computed. We can denote this cost
 320 by $N_C C_L$, where C_L is the cost of adding a term in the linear combination. Denote
 321 by C_E the cost of the matrix exponential, and hence the cost of L4 is C_E . Likewise
 322 denote by C_M the cost of multiplying the solution by a matrix which corresponds to
 323 the cost of task L5. Putting these together, the serial cost for N time steps is

$$324 \quad (4.1) \quad \mathbf{C}_S = N(MC_A + N_C(C_C + C_L) + C_E + C_M).$$

325 Now consider the parallelization of the method for the linear problem across the
 326 time steps. In task L1, each function evaluation can be done concurrently, so that the
 327 cost is reduced from MC_A to C_A . For task L2, all the commutators of a given order
 328 can be computed concurrently, so that cost is reduced from $N_C C_C$ to PC_C . Task L3
 329 can be done with cost $\log_2(N_C)C_L$, and the cost of task L4 and L5 remains the same.

330 Next consider the cost when both parallelization across the method and across the
 331 time steps is employed. Given sufficient processors, tasks L1-L4 can all be computed
 332 on all time steps concurrently. Only task L5 must be done serially so that the total
 333 cost using both forms of parallelism becomes

$$334 \quad (4.2) \quad \mathbf{C}_P = C_A + PC_C + \log_2(N_C)C_L + C_E + NC_M.$$

335 Clearly this is a significant reduction in computational cost. If the cost of computing
 336 the commutators and matrix exponential (L2 and L4) dominate the other terms, the
 337 theoretical parallel speedup approaches N .

338 An important point to make about this counting is that the cost per step when
 339 using parallelization across the method depends very little on the number of quadrature
 340 nodes or commutators used in each term since only the cost of task L3 depends on

341 these factors. Hence higher-order methods are only modestly more expensive per
 342 step than lower-order methods and there is less benefit from reducing the number of
 343 commutators required in each term of the Magnus expansion since multiple terms can
 344 be computed in parallel. Furthermore, for a given accuracy, higher-order methods will
 345 typically require fewer time steps (i.e. smaller N).

346 **4.2. The Nonlinear Case.** For nonlinear problems, the theoretical accounting
 347 of cost must be modified somewhat. Since we are using an iterative procedure to
 348 compute $\Omega(t_m)$, the following steps must be done for each iteration in each time step:

- 349 N1. Evaluate $A(y_m^k, t_m)$ for each quadrature node t_m
- 350 N2. Compute commutators necessary for each term in the truncated Magnus
 351 expansion
- 352 N3. Apply quadrature rules to compute Ω_m^k at each quadrature node t_m
- 353 N4. Form exponential of Ω_m^k at each quadrature node t_m .
- 354 N5. Compute y_m^{k+1} at each quadrature node by matrix multiplication by Ω_m^k .

355 The main difference between these tasks and the linear case is that N2-N5 are done
 356 for each quadrature node instead of only once. For simplicity, the serial cost of these
 357 steps will be assumed to be M times that of the linear case. Hence, denoting by K_S
 358 the number of iterations required for each step in a serial implementation, the serial
 359 cost for the nonlinear Magnus method iteration becomes

$$360 \quad (4.3) \quad \mathbf{C}_S = NK_S M(C_A + N_C(C_C + C_L) + C_E + C_M).$$

361 As will be shown below, the number of iterations required for convergence K_S depends
 362 on Δt in a nontrivial way.

363 If we allow parallelization across the method, the tasks above can all be computed
 364 concurrently at each quadrature node, and hence the cost of each iteration for the
 365 nonlinear method is essentially that of one step in the linear case using parallelization
 366 across the method.

$$367 \quad (4.4) \quad \mathbf{C}_I = C_A + PC_C + \log_2(N_C)C_L + C_E + C_M.$$

368 The speedup across the method is bounded by M .

369 Now consider parallelization across the time steps. The simplest way that this can
 370 be accomplished is to pipeline the iterations. We first divide the N time steps into
 371 blocks of size N_P with each step in a block assigned to a group of processors indexed by
 372 n_p . At each time step n_p for each iteration k , the initial condition is assigned the final
 373 value from iteration $k - 1$ of the time step $n_p - 1$. For each block, $N_P - 1$ pipelined
 374 iterations are required before the last processor has a consistent initial condition. After
 375 this initialization step, assume K_P additional iterations are needed for convergence
 376 on every time step in the block. As in the serial case K_P depends in general on the
 377 time step Δt and now also on N_P , increasing as N_P increases and decreasing as Δt
 378 decreases. Furthermore $K_P \geq K_S$.

379 The parallel cost on each block will be $(N_P - 1 + K_P)\mathbf{C}_I$ compared to $(N_P K_S)\mathbf{C}_I$
 380 when no parallelization across the time steps is applied (i.e. $N_P = 1$). The potential
 381 speedup from parallelization across the time steps is then

$$382 \quad (4.5) \quad \mathbf{S} = \frac{N_P K_S}{N_P - 1 + K_P} = \frac{K_S}{1 + (K_P - 1)/N_P}.$$

383 Clearly the speedup is bounded by the number of serial iterations required and the
 384 best speedup will occur when the quantity K_P/N_P remains small as N_P increases.
 385 This ratio will be investigated in the numerical examples.

386 **5. Numerical Examples.** In this section, the performance of the different
 387 Magnus integrators introduced in section 3 is examined in both serial and parallel
 388 settings. First, the performance of the methods is considered as applied to a Toda
 389 lattice problem from the literature. Serial results demonstrating the relative accuracy
 390 and efficiency of the various methods are presented. In section 5.1.3, some preliminary
 391 results on parallelization of the methods are presented. In section 5.2, the parallel
 392 performance of the methods is considered on a problem motivated by real-time time-
 393 dependent density functional theory.

394 **5.1. Test Case 1: The Periodic Toda Lattice.** The numerical methods will
 395 first be evaluated on the test problem of a d -particle periodic Toda lattice [19], a
 396 one-dimensional chain whose dynamics are governed by nonlinear nearest-neighbor
 397 interactions. The equations of motion are a Hamiltonian system for positions q_j and
 398 momenta p_j (assuming unit masses)

$$399 \quad (5.1) \quad \begin{aligned} q'_j &= p_j \\ 400 \quad (5.2) \quad p'_j &= e^{-(q_j - q_{j-1})} - e^{-(q_{j+1} - q_j)}. \end{aligned}$$

402 In order to cast the dynamics in terms of a Lax pair as in Eq. (2.21), one uses the
 403 Flaschka change of variables

$$404 \quad (5.3) \quad \alpha_j = \frac{1}{2} e^{-(q_{j+1} - q_j)/2}$$

$$405 \quad (5.4) \quad \beta_j = \frac{1}{2} p_j,$$

407 which leads to definitions of Y and A
 (5.5)

$$408 \quad Y = \begin{bmatrix} \beta_1 & \alpha_1 & 0 & \dots & \alpha_d \\ \alpha_1 & \beta_2 & \alpha_2 & \ddots & \vdots \\ 0 & \alpha_2 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \alpha_{d-1} \\ \alpha_d & \dots & 0 & \alpha_{d-1} & \beta_d \end{bmatrix}, A(Y) = \begin{bmatrix} 0 & -\alpha_1 & 0 & \dots & \alpha_d \\ \alpha_1 & 0 & -\alpha_2 & \ddots & \vdots \\ 0 & \alpha_2 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -\alpha_{d-1} \\ -\alpha_d & \dots & 0 & \alpha_{d-1} & 0 \end{bmatrix}.$$

409 The numerical example considered here is an 11-particle periodic Toda lattice
 410 taken from Ref. [21] with the initial conditions

$$411 \quad q(0) = [0, \dots, 0]^T$$

$$412 \quad (5.6) \quad p_j(0) = \begin{cases} 4, & 1 \leq j \leq 4, \\ 0, & j \geq 5. \end{cases}$$

414 The periodic Toda lattice is not asymptotically free and has considerably complicated
 415 motion. Figure 1 shows the position and momenta of the 11 particles up to $t_{final} = 10.0$.
 416

417 **5.1.1. Simulation parameters.** The following numerical experiments are all
 418 performed on this 11-particle periodic Toda lattice with initial conditions as in Eq. (5.6)
 419 and a fixed $t_{final} = 10.0$. A Picard iteration tolerance of 10^{-12} is used on the maximum
 420 absolute value of the residual at the end of the timestep. The reference solution is
 421 taken as the Leg-6 method with $\Delta t = 2^{-15} t_{final}$ or 32768 steps. The reported error

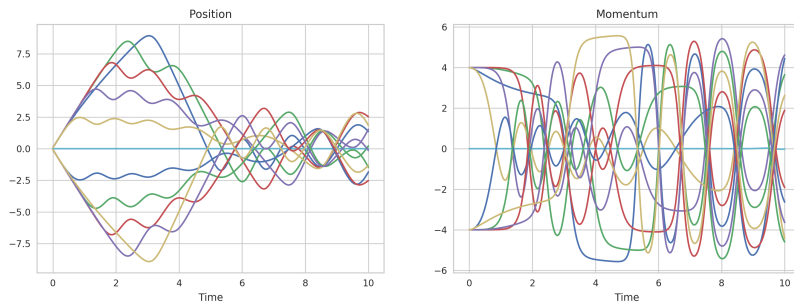


FIG. 1. *Symmetric periodic 11-particle Toda lattice solutions with initial conditions as in Eq. (5.6).*

422 is defined as the matrix 2-norm of the absolute value of the difference between the
 423 solution in question and the reference solution.

424 The methods have been implemented using the *LibPFASST*¹ library. Communi-
 425 cation between pipelined iterations is done in *LibPFASST* using MPI, and the variable
 426 N_P corresponding to the number of parallel steps is also the number of MPI ranks.
 427 For parallelization across the method, OpenMP is used to parallelize the steps in the
 428 nonlinear iteration. All timing results were performed on a single compute node of
 429 NERSC’s Cray XC30 supercomputer, Edison, which contains 24 hardware cores and
 430 64 GB of memory.

431 **5.1.2. Serial Results.** We first perform convergence tests to examine the error
 432 with respect to Δt for different Magnus methods. Figure 2 shows the behavior for
 433 each method summarized in Table 2. Each method displays the proper convergence
 434 rate for a range of Δt . For the second-order methods, note that the error for Leg-2
 435 is significantly smaller than that of Lob-2, which implies the dominant error term
 436 for Lob-2 is due to the quadrature rule as opposed to the truncation of the Magnus
 437 expansion. The Leg-2 method requires more serial work in this case due to the use
 438 of three quadrature nodes instead of two. The difference between the fourth-order
 439 methods is less significant. Leg-4-3 is more accurate than Lob-4-1 (with a higher serial
 440 cost), but since the main difference between the two methods is how the second term
 441 in the Magnus expansion is treated, the difference between the two is smaller than
 442 for the second-order methods. Leg-6 is clearly more accurate than the other methods.
 443 Note that only about nine significant digits of accuracy is attainable for the reference
 444 solution for this problem using double precision due to the sensitivity of the solution
 445 to perturbations.

446 To better demonstrate the relative computational cost of each method, Figure 3
 447 shows the total serial wall-time versus number of steps for the experiment above. As
 448 expected, Lob-2 is the method with the shortest time to solution. Lob-4-1, the simplest
 449 fourth-order method that can be constructed, is actually less expensive than the second-
 450 order Leg-2 scheme despite the fact that Lob-4-1 requires a matrix commutator. For
 451 the small problem size used here, the matrix commutators are relatively inexpensive,
 452 and the fact that matrix exponentials must be computed at three internal quadrature
 453 nodes for Leg-2 more than makes up for the lack of commutator terms. Leg-4-3 and

¹*LibPFASST* is available at <https://github.com/libpfasst/LibPFASST>.

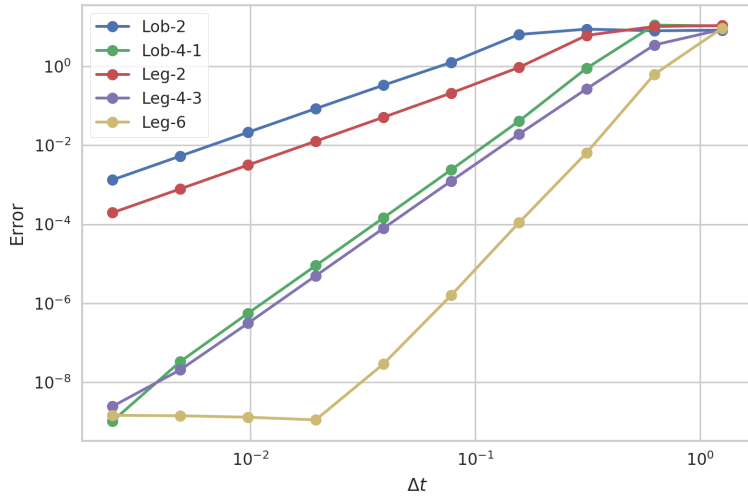


FIG. 2. Error at $t_{final} = 10.0$ versus Δt for the Toda lattice test case.

Leg-6 are unsurprisingly the most expensive in serial.

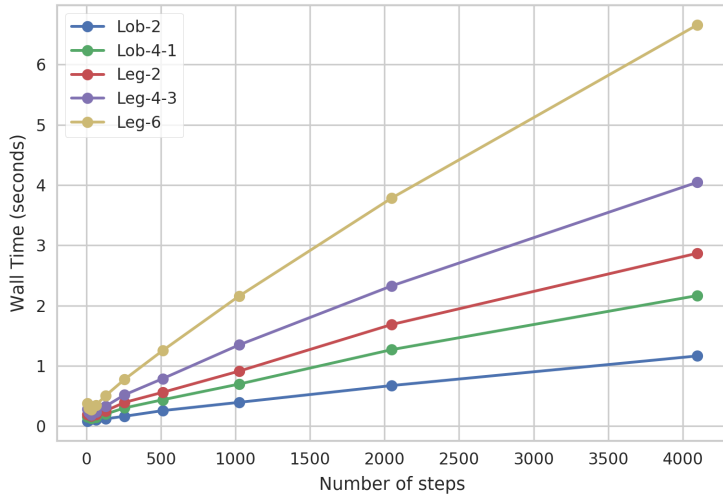


FIG. 3. Total wall-time for the solution for the Toda lattice test case for fixed $t_{final} = 10.0$.

454

455

456

457

458

459

Note that the cost of the methods displayed in Figure 3 does not grow exactly linearly with the number of time steps. This is due to the fact that the number of Picard iterations needed to converge to the tolerance depends on the time step Δt . Figure 4 shows the average number of iterations over all time steps for each method as a function of Δt . Note the higher-order methods require moderately fewer iterations

460 than lower-order methods, and as the time step gets larger, the number of iterations
 461 required for convergence grows rapidly.

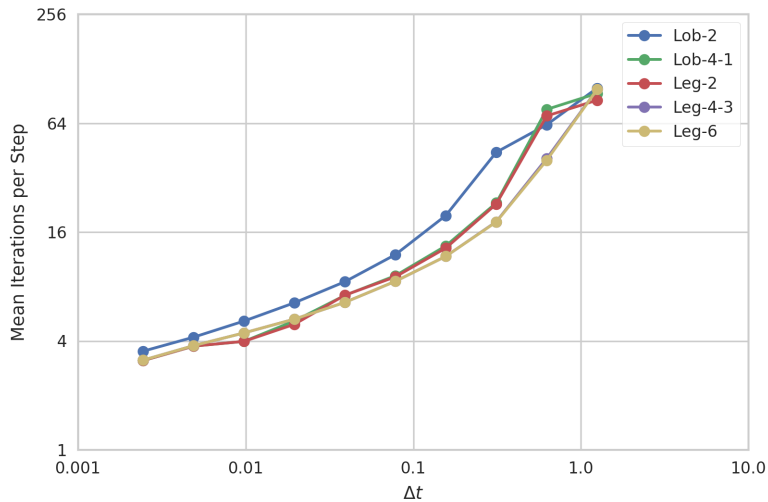


FIG. 4. Average number of iterations as a function of Δt for serial methods on the Toda lattice test case.

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

The three figures provided above demonstrate that it is not necessarily trivial to choose a method and time step that will provide a solution to a given accuracy with the least computational effort. To illustrate this, Figure 5 shows the accuracy versus wall-clock time for the Toda lattice test. Reading from left to right for a given accuracy shows in increasing order, the methods with the fastest time-to-solution. While Lob-2 is by far the cheapest method, it only the fastest method for simulations where the error is $O(1)$. Lob-4-1 is the most efficient for error tolerances to about 10^{-6} , after which Leg-6 becomes the most efficient. For an error of about 10^{-6} , Leg-6 and Lob-4-1 are more than an order of magnitude more efficient than the second-order methods.

5.1.3. Parallel in Time Results. In this section, the relative performance of parallel Magnus integrators is explored. We first consider the speedup due to parallelization over time steps by pipelining the Picard iterations. As discussed in Section 4, the theoretical speedup from pipelining is bounded by the number of serial iterations required for the method and depends on how the total number of parallel iterations required to reach convergence grows as the number of time parallel steps is increased. As in the serial case, the number of iterations required depends on the time step but now depends also on the number of parallel time steps in the pipeline. Figure 6 demonstrates this dependence for the Toda lattice test using method Leg-6 by plotting the average number of iterations required for convergence for different Δt and parallel steps, denoted by N_P . As in the serial case, the number of iterations required decreases with decreasing Δt and here it also increases for fixed Δt as N_P increases.

A second way to display the convergence behavior of the pipelined iteration is to plot the residual after each iteration for each time rank. Figure 7 shows this data for the Leg-6 method using a time step of $\Delta t = 10/128$ in the top panel and $\Delta t = 10/1024$ in the bottom panel for 16 parallel time steps. As discussed in section 4.2, the speedup

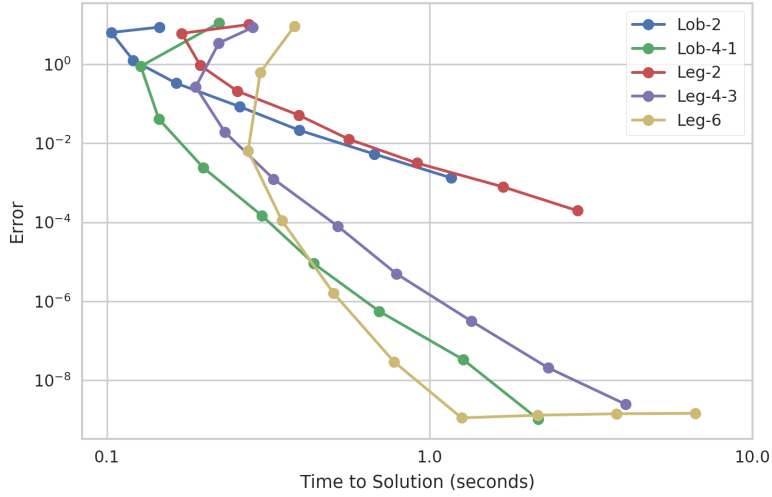


FIG. 5. Error versus time to solution for the serial Toda lattice test case. Each point on a line, read from top to bottom, represents a two-fold increase in the number of steps.

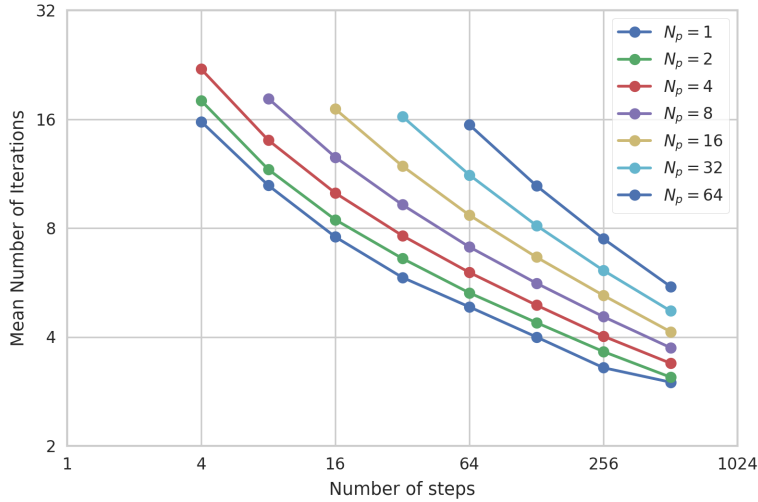


FIG. 6. Average number of pipelined iterations to reach residual of 10^{-12} for the parallel Toda lattice test case using the Leg-6 method.

487 achievable from pipelining depends on the ratio K_P/N_P . In this example it is clear
 488 that K_P/N_P is decreasing for large N_P as more pipelined time steps are used.

489 The nontrivial dependence of the parallel iterates on both Δt and N_P makes it
 490 difficult to predict which method with which parameters will minimize the time to
 491 solution for a given accuracy. As in the serial case, it is instructive to consider the
 492 accuracy versus wall-clock for different methods with different number of time-parallel

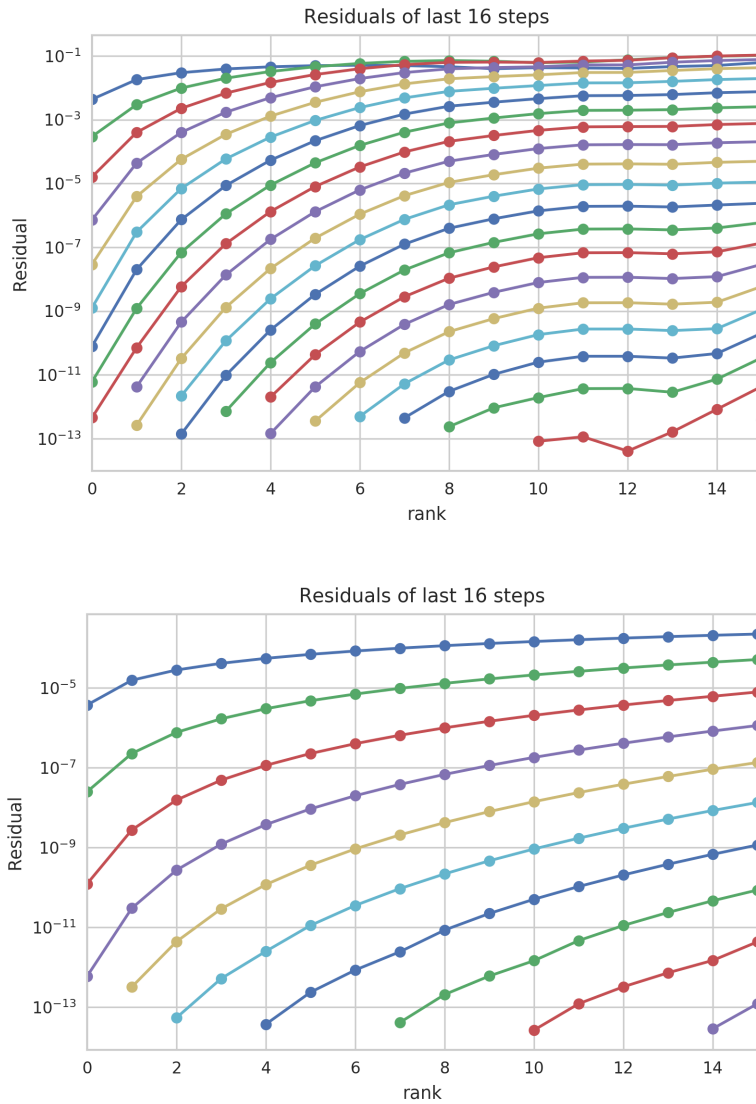


FIG. 7. Residual convergence for each processor in a 16 MPI task simulation using the second-order Legendre method Leg-6 with 128 steps (above) and 1024 steps (below). Every recurrence of a given color is another 6 iterations.

493 steps. Figure 8 displays this information for each of the methods with increasing
 494 number of processors. Across a single method, e.g. Leg-6, there's an optimal value
 495 of parallelization due to the fact that the increased number of iterations required in
 496 the pipeline algorithm starts to cost more than it gains. It is also likely for this test
 497 case that the small problem size implies that communication latency is not negligible.
 498 Nevertheless, it is again clear that higher-order parallel method gives a shorter time
 499 to solution than lower-order alternatives. The second-order methods are only less
 500 expensive than the higher-order methods when no digits of accuracy are computed.

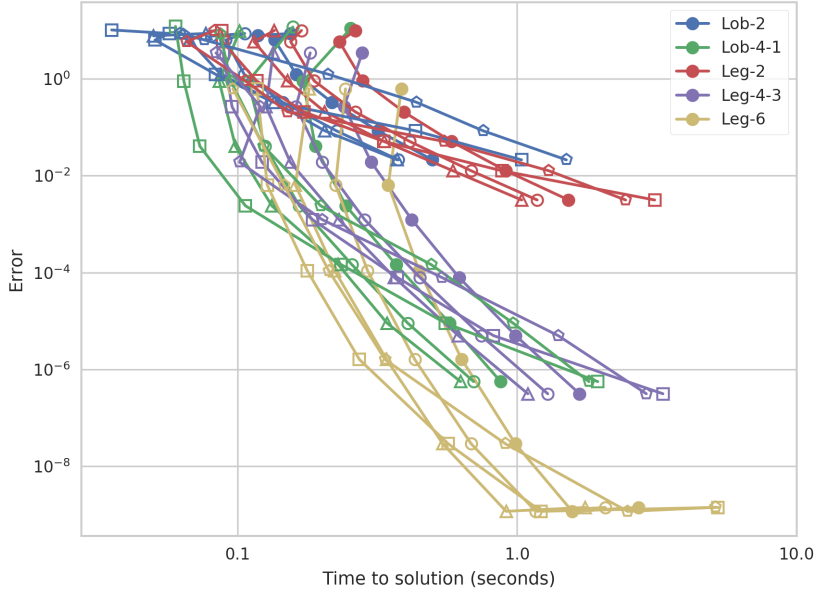


FIG. 8. Error versus time to solution for an 11-particle periodic Toda lattice example. Each color represents a different method and each marker represents a different number of time-parallel steps, $N_p = 1, 2, 4, 8, 16$. Closed circles represent the serial computation, open circles $N_p = 2$, triangles $N_p = 4$, squares $N_p = 8$, and pentagons $N_p = 16$.

501 Finally, we present preliminary results using parallelization across the method and
 502 across the time steps. Figure 9 shows the error versus time to solution for the serial
 503 implementation and time- and method-parallel methods. The particular configuration
 504 of $N_p = 8$ and 3 OpenMP threads uses the entirety of 1 hardware node on NERSC's
 505 Edison Cray XC30 supercomputer with no hyperthreading. The additional parallel
 506 across method provides an additional factor of at best two using simple OpenMP
 507 parallel do loops over nodes. Using both types of parallelism in this context gives
 508 up to a factor of 4.7 in the overall compute time compared to serial methods using
 509 24 total processors. Note that compared to serial second-order methods, the parallel
 510 sixth-order method can compute a solution in less time with more than four orders of
 511 magnitude lower error.

512 **5.2. Test Problem 2: An RT-TDDFT Proxy.** The motivating application
 513 for the parallel Magnus methods introduced here is the simulation of electron dynamics
 514 using real-time time-dependent density functional theory (RT-TDDFT). In this section
 515 we study the parallel performance of the time-parallel Magnus methods on a test
 516 problem with a similar but simpler structure than RT-TDDFT.

517 **5.2.1. Physical Background.** RT-TDDFT describes the time evolution of the
 518 density matrix P through the von Neumann equation of motion

$$519 \quad (5.7) \quad P' = -i[F(P), P],$$

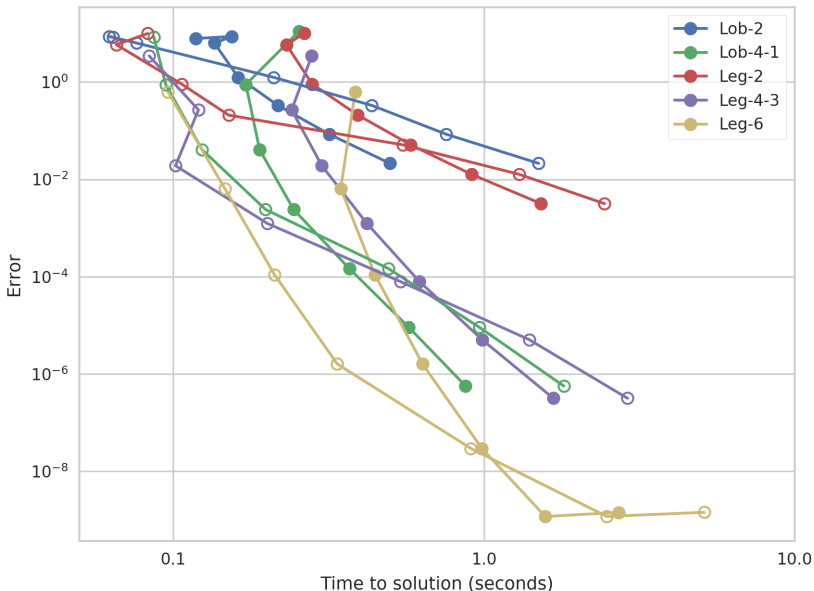


FIG. 9. Error versus time to solution for an 11-particle periodic Toda lattice example. Solid markers indicate serial calculations and open markers indicate time- and method-parallel runs with 8 MPI tasks and 3 OpenMP threads.

520 where $F(P)$ is the Fock matrix. The size of the matrices N scales with the number of
 521 electrons times the number of basis functions used to represent the electron density.
 522 The evaluation of $F(P)$, the construction of the Fock matrix, requires for each entry in
 523 P the approximation of one- and two-electron integrals which formally requires $O(N^4)$
 524 work. Furthermore, the time scales on which electron dynamics occur are typically
 525 on the order of tens or hundreds of atto-seconds, hence even simulations of moderate
 526 sized molecules at the femto-second range are extremely computationally expensive.
 527 In the RT-TDDFT simulation suite in the NWChem code [20], Eq. (5.7) is integrated
 528 using a serial second-order Magnus integrator, and part of the motivation of this paper
 529 is to improve the efficiency of this choice.

530 **5.2.2. The Proxy Problem.** To give an example of the performance of the
 531 parallel Magnus methods without requiring the considerable infrastructure necessary
 532 to compute the true Fock matrix, we consider a test problem inspired by a simplified
 533 one-dimensional RT-TDDFT problem. Here P is again an $N \times N$ complex matrix,
 534 and

$$535 \quad (5.8) \quad F(P)_{i,j} = \sum_{i=1}^N \sum_{j=1}^i \left(Z f_1(i, j) + \sum_{m=1}^N \sum_{n=1}^N E(f_2(i, j, m, n) - \frac{1}{2} f_2(i, n, m, j)) \right)$$

536 The functions f_1 and f_2 correspond to the one- and two-electron integrals in RT-
 537 DTDDFT, with the f_1 term corresponding to the single electron Hamiltonian and the
 538 two f_2 terms corresponding to the Coulomb and exchange operators respectively. The
 539 simplified forms here are derived by considering single Gaussian basis functions in one

540 dimension with single point approximations of the integrals. Specifically

$$541 \quad (5.9) \quad f_1(i, j) = -\frac{P_{i,j}\bar{P}_{j,i}}{r^2} \exp\left(-\frac{(x(i) - x(j))^2}{2}\right)$$

542 with $r = (x(i) + x(j))/2$. Similarly
 (5.10)

$$543 \quad f_2(i, j, m, n) = \frac{P_{m,n}\bar{P}_{n,m}P_{i,j}\bar{P}_{j,i}}{r^2} \exp\left(-\frac{(x(i) - x(j))^2}{2}\right) \exp\left(-\frac{(x(m) - x(n))^2}{2}\right)$$

544 with $r = ((x(i) + x(j))/2 - (x(m) + x(n))/2)$. Note that the computation of $F(P)$
 545 in the problem is formally $O(N^4)$ whereas the matrix commutator terms are $O(N^3)$.
 546 The matrix exponential when computed by power series is also $O(PN^3)$ where P
 547 is the number of terms in expansion. Hence in this application (as in full RT-TDDFT
 548 simulations), the computation of $F(P)$ is the dominant cost for even moderate N .
 549 In the numerical tests, the spatial locations are uniformly spaced on $[-1, 1]$ so that
 550 $x(i+1) - x(i) = 2/(N-1)$. The values for Z and E are 0.5 and 0.05 respectively.

551 **5.2.3. Parallel in Time Results.** For the test problem described above, we run
 552 the parallel Magnus algorithms Lob-2, Lob-4-1, and Leg-6 on a 20 particle RT-TDDFT
 553 proxy problem as described above, using a final time of the run $T = 0.05$. For each
 554 choice of method, runs using 48, 96, 144, and 192 time steps are compared for different
 555 numbers of parallel processors $N_p = 1, 2, 4, 8, 16$ and 24. A uniform time step is used
 556 in all cases, which is justified since the character of the dynamics does not change in
 557 time. As above, computations are run on 1 hardware node on NERSC's Edison Cray
 558 XC30 consisting of 24 cores. For these comparisons, only the parallelism across time
 559 steps is used. Unlike the results in the previous section, the convergence tolerances
 560 are adjusted by case to avoid unnecessary iterations. In practice this means that
 561 the tolerances are larger for less accurate simulations since additional SDC iterations
 562 will reduce the residual but not the numerical error determined by the underlying
 563 quadrature rule. The tolerances here were chosen using knowledge of the error in the
 564 simulations and are shown in table 3 . Dynamically choosing the optimal tolerance for
 serial or parallel SDC methods is an open problem.

	48	96	144	196
Lob-2	$1e-4$	$1e-4$	$2e-5$	$1e-5$
Lob-4-1	$1e-5$	$1e-6$	$5e-7$	$1e-7$
Leg-6	$1e-6$	$1e-8$	$1e-9$	$1e-10$

TABLE 3

Residual tolerances for the RT-TDDFT proxy problem for different methods and number of steps.

565
 566 Fig. 10 compares the accuracy versus total computational time for each variation
 567 and demonstrates the attractiveness of both higher-order methods and time parallelism.
 568 The reported error in each case is the maximum absolute error along the diagonal of
 569 P (which in RT-TDDFT are the relevant quantities). It is clear from the timings that
 570 significant parallel speedup is attainable for methods of all order. As expected, the
 571 smaller the error tolerance, the larger the possible parallel speedup since there are
 572 in general more iterations to amortize over processors. Note that serial second-order
 573 methods are more expensive than the fastest fourth-order methods. In particular,
 574 the fastest serial fourth-order method runs in less time than the fastest serial second-
 575 order method because fewer iterations are needed for convergence (despite the smaller

576 residual tolerance). Parallel sixth-order methods run faster than serial second-order
 577 methods while producing smaller errors by several orders of magnitude. In general,
 the optimal choice of method and parallelization depends on the error tolerance.

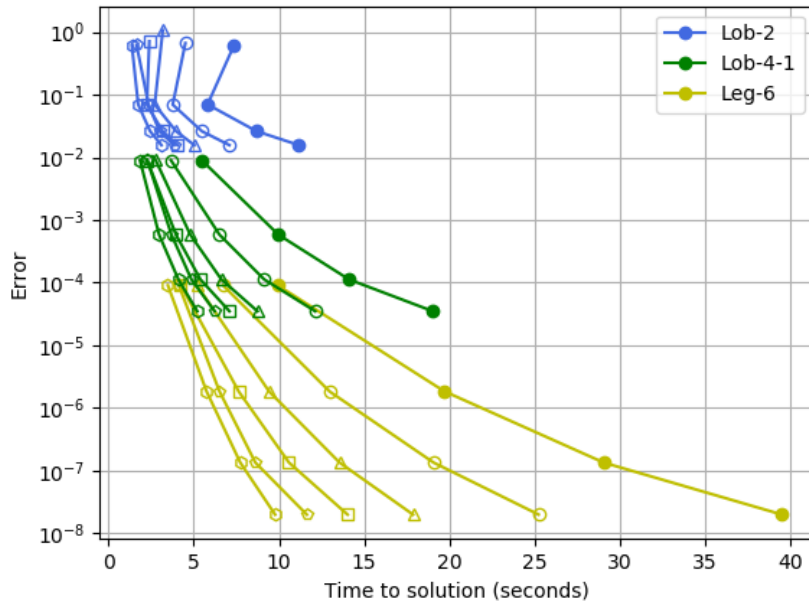


FIG. 10. Error versus time to solution for a 20-particle RT-TDDFT proxy example. Each color represents a different method and each marker represents a different number of time-parallel steps, $N_p = 1, 2, 4, 8, 16, 24$. Closed circles represent the serial computation, open circles $N_p = 2$, triangles $N_p = 4$, squares $N_p = 8$, pentagons $N_p = 16$, hexagons $N_p = 24$.

578

579 **6. Discussion.** Much of the initial work on Magnus integrators is focused on
 580 linear problems where only a single evaluation of the Magnus expansion is required
 581 for each time step. In contrast, this paper explores the accuracy and cost of Magnus
 582 integrators applied to nonlinear problems. Nonlinear Magnus methods require solving
 583 an implicit equation involving the Magnus expansion to obtain the solution in each
 584 timestep, and the methods proposed here use a simple fixed point iteration for this
 585 solution that can be readily parallelized in time.

586 One conclusion presented here is that in both the linear and nonlinear case,
 587 straight-forward parallelization across the method is possible, leading to higher-order
 588 methods with only marginally higher computational cost than lower-order methods.
 589 This is complimentary to previous results in the literature where considerable effort is
 590 placed on reducing the complexity of each of the terms in regards to the number of
 591 commutators required.

592 The second level of parallelism described in this work, namely parallelization
 593 across the time steps through pipelined iterations, can further decrease the overall
 594 time-to-solution for nonlinear problems. There is a non-trivial relationship between
 595 the number of pipelined time steps, the time step size, and the number of iterations
 596 required for convergence of the iteration, hence it is not easy to predict *a priori* which
 597 choice of parameters will lead to the shortest wall clock time given a desired level

598 of accuracy. Nevertheless, for the test cases considered here, in most situations, the
 599 parallel sixth-order methods require the least computation time and are far superior
 600 to serial second-order methods.

601 It is important to note that the results here are preliminary and are performed on
 602 two test cases of moderate size. In general, the possible parallel speedup attainable
 603 for a given problem will depend on the sensitivity of the problem to perturbations,
 604 the relative cost of the operations such as computing commutators or the matrix
 605 exponential, and the ratio of computation to communication costs. In future work,
 606 the authors will use this parallel methodology to investigate real-time electronic
 607 dynamics, where the calculation of the right-hand side values is more expensive than
 608 both commutators and matrix exponentials as in the second numerical example. This
 609 paper addresses only the use of time-parallelism to reduce the wall clock time of serial
 610 Magnus methods, and not important issues like the best way to compute the matrix
 611 exponential, the optimal choice of time step, nor the dynamic stopping criteria for
 612 iterations. Such issues are present in both serial and parallel implementations of
 613 Magnus methods and are most likely problem dependent.

614 **7. Acknowledgments.** The work here was supported by the U.S. Department of
 615 Energy, Office of Science, Office of Advanced Scientific Computing Research, Applied
 616 Mathematics program under contract number DE-AC02005CH11231. Part of the
 617 simulations were performed using resources of the National Energy Research Scientific
 618 Computing Center (NERSC), a DOE Office of Science User Facility supported by the
 619 Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-
 620 05CH11231.

621 **Appendix A. Quadrature Rules for Intermediate Nodes.** The necessary
 622 weights for implementing each of the methods Lob-4-1, Leg-2, Leg-4-3, and Leg-6 for
 623 nonlinear problems are presented here. For Lob-2, no additional rules are necessary.

624 For the computation of the single integrals in the first Magnus term, the necessary
 625 rules take the form

$$626 \quad (\text{A.1}) \quad \Omega_m^{(1)} = \Delta t \sum_{j=1}^M q_{m,j}^{(1)} A_j,$$

627 and the coefficients $q_{m,j}^{(1)}$ correspond to the classical collocation schemes (see e.g. [9])
 628 For Lob-3, only one additional rule is needed at the midpoint t_2 ,

$$629 \quad (\text{A.2}) \quad q_{2,j}^{(1)} = \left[\frac{5}{24}, \frac{1}{3}, -\frac{1}{24} \right].$$

630 For Leg-3, rules for each node are required, and the $q_{m,j}^{(1)}$ are given by

$$631 \quad (\text{A.3}) \quad q_{m,j}^{(1)} = \begin{bmatrix} \frac{5}{36} & \frac{2}{9} - \frac{\sqrt{15}}{15} & \frac{5}{36} - \frac{\sqrt{15}}{30} \\ \frac{5}{36} + \frac{\sqrt{15}}{24} & \frac{2}{9} & \frac{5}{36} - \frac{\sqrt{15}}{24} \\ \frac{5}{36} + \frac{\sqrt{15}}{30} & \frac{2}{9} + \frac{\sqrt{15}}{15} & \frac{5}{36} \end{bmatrix}.$$

632 For the second term $\Omega^{(2)}$ there are two versions described in the linear case by
 633 Eqs. (3.2) and (3.3). In the first case, only one additional coefficient is needed, namely
 634 $q_2^{(2)-1} = 1/48$. In the second case, we have

$$635 \quad (\text{A.4}) \quad \Omega_m^{(2)-3} = q_{m,1}^{(2)-3} [A_1, A_2] + q_{m,2}^{(2)-3} [A_1, A_3] + q_{m,3}^{(2)-3} [A_2, A_3],$$

636 with the values of $q_{m,j}^{(2)-3}$ given by
 (A.5)

$$637 \quad q_{m,j}^{(2)-3} = \begin{bmatrix} -7.0825623244174e-4 & -3.5291589565775e-2 & -7.8891497044705e-2 \\ 2.0142743933468e-4 & 4.4826196136660e-3 & -1.8131905893999e-2 \\ -2.6081558162830e-6 & -5.6936734355286e-4 & -3.5152700676886e-2 \end{bmatrix}$$

638 The third term takes the form

$$639 \quad \Omega_m^{(3)} = [q_{m,1,1}^{(3)}A_1 + q_{m,1,2}^{(3)}A_2 + q_{m,1,3}^{(3)}A_3, [A_1, A_2]] +$$

$$640 \quad [q_{m,2,1}^{(3)}A_1 + q_{m,2,2}^{(3)}A_2 + q_{m,2,3}^{(3)}A_3, [A_1, A_3]] +$$

$$641 \quad (A.6) \quad [q_{m,3,1}^{(3)}A_1 + q_{m,3,2}^{(3)}A_2 + q_{m,3,3}^{(3)}A_3, [A_2, A_3]].$$

643 with
 (A.7)

$$644 \quad q_{1,i,j}^{(3)} = \begin{bmatrix} 1.4667828928181e-6 & -2.5468454487434e-6 & 7.1885579589404e-7 \\ -3.0653702506833e-7 & 6.9623363228690e-7 & -1.9684558120029e-7 \\ -2.2622163607144e-8 & -2.7279719400850e-9 & 8.5484354192049e-10 \end{bmatrix}$$

(A.8)

$$645 \quad q_{2,i,j}^{(3)} = \begin{bmatrix} 1.0401143365317e-3 & -1.7143302808715e-3 & 1.9808827525182e-4 \\ -6.9105495969459e-5 & 2.9054016014502e-4 & -3.4658846939476e-5 \\ 9.2451884893203e-5 & 1.2595057164957e-5 & -2.4709074423914e-6 \end{bmatrix}$$

(A.9)

$$646 \quad q_{3,i,j}^{(3)} = \begin{bmatrix} 4.1482959753609e-3 & -6.3874218931689e-3 & -3.5942319108173e-3 \\ 9.9737811032708e-4 & 1.2415302375576e-4 & -3.8059754231607e-4 \\ 3.7183849345731e-3 & 1.6935142950568e-3 & -1.0604085845381e-3 \end{bmatrix}.$$

647 Finally, for the fourth term, $\Omega_m^{(4)}$ is computed as in Eqs. (3.7) and (3.8), with
 648 $q^{(4)} = 1/60$ and $q_j^{(1)}$ in (3.8) replaced with $q_{m,j}^{(1)}$ from Eq. (A.3).

649

REFERENCES

- 650 [1] S. BLANES, F. CASAS, J. A. OTEO, AND J. ROS, *The Magnus expansion and some of its*
651 *applications*, Physics Reports, 470 (2009), pp. 151–238.
- 652 [2] S. BLANES, F. CASAS, AND J. ROS, *Improved High Order Integrators Based On The Magnus*
653 *Expansion*, Bit, 40 (2000), pp. 434–450.
- 654 [3] S. BLANES AND P. C. MOAN, *Fourth- and sixth-order commutator-free Magnus integrators*
655 *for linear and non-linear dynamical systems*, Applied Numerical Mathematics, 56 (2006),
656 pp. 1519–1537.
- 657 [4] K. BURRAGE, *Parallel methods for ODEs*, Advances in Computational Mathematics, 7 (1997),
658 pp. 1–3.
- 659 [5] L. EULER, *Institutiones calculi integralis*, no. v. 2, Acad. Imper. scientiarum, 1768.
- 660 [6] M. J. GANDER, *50 years of time parallel time integration*, in Multiple Shooting and Time
661 Domain Decomposition Methods, T. Carraro, M. Geiger, S. Körkel, and R. Rannacher, eds.,
662 Cham, 2015, Springer International Publishing, pp. 69–113.
- 663 [7] E. HAIRER, C. LUBICH, AND G. WANNER, *Geometric numerical integration : structure-preserving*
664 *algorithms for ordinary differential equations*, Springer, 2006.
- 665 [8] E. HAIRER, S. P. NØRSETT, AND G. WANNER, *Solving ordinary differential equations I. nonstiff*
666 *problems*, Mathematics and Computers in Simulation, 29 (1987), p. 447.
- 667 [9] E. HAIRER AND G. WANNER, *Solving Ordinary Differential Equations II : Stiff and Differential-*
668 *Algebraic Problems*, Springer Berlin Heidelberg, 1991.
- 669 [10] N. J. HIGHAM, *The Scaling and Squaring Method for the Matrix Exponential Revisited*, SIAM
670 Journal on Matrix Analysis and Applications, 26 (2005), pp. 1179–1193.
- 671 [11] M. HOCHBRUCK AND A. OSTERMANN, *Exponential integrators*, Acta Numerica, 19 (2010),
672 pp. 209–286.
- 673 [12] A. ISERLES, H. Z. MUNTHE-KAAS, S. P. NØRSETT, AND A. ZANNA, *Lie-group methods*, Acta
674 Numerica, (2000), pp. 215–365.
- 675 [13] A. ISERLES AND S. P. NØRSETT, *On the solution of linear differential equations in Lie groups*,
676 Philosophical Transactions: Mathematical, Physical and Engineering Sciences, 357 (1999),
677 pp. 983–1019.
- 678 [14] W. MAGNUS, *On the exponential solution of differential equations for a linear operator*, Com-
679 munications on pure and applied . . . , VII (1954), pp. 649–673.
- 680 [15] C. MOLER AND C. VAN LOAN, *Nineteen Dubious Ways to Compute the Exponential of a Matrix,*
681 *Twenty-Five Years Later*, SIAM Review, 45 (2003), pp. 3–49.
- 682 [16] H. MUNTHE-KAAS AND B. OWREN, *Computations in a free Lie algebra*, Philosophical Transac-
683 tions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 357 (1999),
684 pp. 957–981.
- 685 [17] J. NIEVERGELT, *Parallel methods for integrating ordinary differential equations*, Communications
686 of the ACM, (1964), pp. 731–733.
- 687 [18] M. THALHAMMER, *A fourth-order commutator-free exponential integrator for nonautonomous*
688 *differential equations*, SIAM J. Numer. Anal., 44 (2006), pp. 851–864 (elect).
- 689 [19] M. TODA, *Vibration of a chain with nonlinear interaction*, Journal of the Physical Society of
690 Japan, 22 (1967), pp. 431–436.
- 691 [20] M. VALIEV, E. BYLASKA, N. GOVIND, K. KOWALSKI, T. STRAATSMA, D. W. H.J.J. VAN DAM,
692 J. NIEPLOCHA, E. APRA, T. WINDUS, AND W. DE JONG, *NWChem: a comprehensive*
693 *and scalable open-source solution for large scale molecular simulations*, Comput. Phys.
694 Commun., (2010), pp. 1477–1489.
- 695 [21] A. ZANNA, *Numerical solution of isospectral flows*, PhD thesis, University of Cambridge, 1998.