Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Computational Physics 213 (2006) 748–776

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

Efficient integration of large stiff systems of ODEs with exponential propagation iterative (EPI) methods

M. Tokman [*]

Department of Mathematics, University of California, Berkeley, 1091 Evans Hall, Berkeley, CA 94720-3840, United States
School of Natural Sciences, University of California, Merced, CA 95344, United States

Received 31 August 2004; received in revised form 12 August 2005; accepted 29 August 2005
Available online 10 October 2005

Abstract

A new class of exponential propagation techniques which we call exponential propagation iterative (EPI) methods is introduced in this paper. It is demonstrated how for large stiff systems these schemes provide an efficient alternative to standard integrators for computing solutions over long time intervals. The EPI methods are constructed by reformulating the integral form of a solution to a nonlinear autonomous system of ODEs as an expansion in terms of products between special functions of matrices and vectors that can be efficiently approximated using Krylov subspace projections. The methodology for constructing EPI schemes is presented and their performance is illustrated using numerical examples and comparisons with standard explicit and implicit integrators. The history of the exponential propagation type integrators and their connection with EPI schemes are also discussed.
© 2005 Elsevier Inc. All rights reserved.

MSC: 34-xx; 65-xx

Keywords: Exponential propagation iterative methods; EPI schemes; Exponential time differencing schemes; Krylov projections; Fast time integrators; Stiff systems; Numerical methods

1. Introduction

The presence of a wide range of temporal scales in a system of differential equations poses a major difficulty for their integration over long time intervals. Such stiff systems are routinely encountered in scientific applications from plasma modeling to chemical kinetics. The development of numerical techniques which provide computational savings over commonly used algorithms can allow one to solve problems faster and access previously unattainable parameter regimes.

A major difficulty in solving large stiff systems of nonlinear differential equations is choosing an efficient time integration scheme. Typically one has to make a decision whether to use an explicit or an implicit

[*] Tel.: +1 510 381 8295; fax: +1 510 486 6199.
E-mail address: mayya@math.berkeley.edu.

0021-9991/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2005.08.032

method. Explicit schemes require the least amount of computation per time step but the allowable time step is severely restricted by the stability requirements. Implicit schemes have much better stability properties and allow significantly larger time steps compared to explicit integrators. However, this advantage comes at the expense of a significant increase in the amount of computation needed at each time iteration. In particular, a typical choice for solving such problems is a Newton–Krylov implicit integrator. For large-scale stiff systems the cost of this method is dominated by the solutions of large linear systems at each Newton iteration. Experience shows that for general large-scale nonsymmetric problems GMRES is a natural choice for solving the linear systems; however, unless one can exploit the structure of the problem and develop a good preconditioner for these matrices, a Newton–Krylov method can become prohibitively expensive [1]. If it is possible to construct a good preconditioner for a particular problem the Newton–Krylov method becomes a very efficient way to solve the stiff system and is difficult to outperform. But for many problems constructing an effective preconditioner is a non-trivial task and one would like to have an alternative method that would provide savings compared to both explicit schemes and implicit Newton–Krylov integrators and would not require developing a preconditioner. These are the classes of problems where exponential propagation method can become advantageous.

The idea to use exponential time differencing (ETD) to construct an effective integrator for stiff systems has a long history and has been introduced and reintroduced in the literature many times (see Section 3). However, only when it was suggested to combine this idea with the Krylov subspace approximation of functions of matrices has it become viable to use ETD to construct time integrators for large-scale nonlinear stiff systems – the effort to construct such methods is a relatively new development. Due to the novelty of these ideas, limited understanding of their performance, and lack of well tested schemes which compare favorably to standard integrators, these methods have not yet been widely used. In fact, to our knowledge none of the previously developed exponential integrators have been clearly demonstrated to outperform the Newton–Krylov implicit integrators which are methods of choice when a large stiff nonlinear system of ODEs has to be solved.

In this paper, we introduce a new class of exponential propagators which we call exponential propagation iterative (EPI) methods. The methods are based on a key observation: if a Krylov projection is used to approximate a product $f(A)b$ between a function f of a large stiff matrix A and a vector b then the convergence rate of the Krylov iteration will depend on the properties of the function that has to be approximated. Specifically, if a Newton–Krylov implicit integrator is used to solve a large stiff system of ODEs then at each Newton iteration the Krylov projections are used to approximate the product $f(A)b = (I - A)^{-1}b$, where A is the Jacobian matrix multiplied by a time step, I is the identity matrix and b is a vector. So in this case the Krylov projections are used to approximate products of a rational function $f(x) = 1/(1 - x)$ of a matrix and a vector. If one uses currently available exponential integrators then the Krylov projections are used to approximate products between a vector and an exponential of the matrix $\mathrm{e}^{A}b$ or an expression $(\mathrm{e}^{A} - I)A^{-1}b$. To get an accurate approximation to the solution these operations have to be performed several times per time step. We approach the construction of an exponential integrator by expressing the solution in terms of a set of special functions $g_k(x)$ or $\phi_{\gamma k}(x)$, $k = 0, 1, 2, \ldots, \gamma$ (see (28) and (29)). The advantage of an EPI method comes from the fact that the number of Krylov iterations needed to approximate products of these functions of a matrix with a vector decreases as the index k increases, and for each of these function the number of needed Krylov projection steps is in general smaller than the number of such steps required to approximate $f(A)b$ with $f(x) = 1/(1 - x)$ and $f(x) = \mathrm{e}^{x}$. Thus we are able to achieve computational savings compared to an implicit integrator while allowing much larger time steps than explicit schemes.

Below we will (i) describe the ideas behind constructing EPI methods and give an overview of other exponential integrators, (ii) introduce new EPI methods and a methodology for their construction, (iii) discuss the efficient implementation of these techniques, and (iv) based on some test problems provide guidance as to what computational savings one can expect compared to standard explicit and implicit methods. In Section 2, we present the history and the ideas behind exponential propagation methods. A procedure for constructing EPI schemes is given in Section 3. Here we introduce several new methods and discuss how the schemes should be formulated and implemented to be efficient. Finally, in Section 4 several numerical examples are used to illustrate the performance of the schemes; a discussion of their appropriate application is included.

2. History and development of exponential propagation techniques

We begin with a general overview of exponential integrators. Consider an initial-value problem for a large nonlinear autonomous system of ordinary differential equations

$$\begin{aligned} \frac{\mathrm{d}U}{\mathrm{d}t} &= F(U), \\ U(t_0) &= U_0, \end{aligned} \tag{1}$$

where $U(t) = (u_1(t), u_2(t), \ldots, u_N(t))^{\mathrm{T}}$ is the solution of (1) at time t and $F(U) = (f_1(U), f_2(U), \ldots, f_N(U))^{\mathrm{T}}$. If such system comes from a discretization of a partial differential equation, $F(U)$ is the discrete representation of a spatial differential operator and the elements of U contain solution values at each grid point. The solution U of the system at time $t_0 + h$ can be also written in integral form

$$U(t_0 + h) = U(t_0) + \int_{t_0}^{t_0+h} F(U(s))\,\mathrm{d}s. \tag{2}$$

The standard approach to approximating the solution of (1) over a time interval $[t_0, T]$ is to discretize the interval over a set of nodes $t_0 < t_1 < t_2 < \cdots < t_n < \cdots < T$ and to construct a quadrature approximation $\sum_j b_j F(U(s_j))$ to the integral in (2)

$$\int_{t_0}^{t_0+h} F(U(s))\,\mathrm{d}s \approx \sum_j b_j F(U(s_j)). \tag{3}$$

The system is then integrated over each time interval $[t_n, t_n + h_n]$, where h_n is the time step at the nth time iteration.

Typically two major choices have to be made to select a time integrator for (1). First, the quadrature nodes s_j in (3) have to be picked. If the approximations to the solution at previous times are used the resulting scheme is of multistep type. A Gaussian quadrature results in a Runge–Kutta type integrator. Second, it has to be decided whether at the nth time step the solution $U(t_n + h_n)$ will be used as one of the nodes in the quadrature formula. If it is not used the resulting scheme is explicit. If $U(t_n + h_n)$ is one of the quadrature nodes an implicit integrator is obtained and each time step requires solving a nonlinear system of equations.

It is well known (see for instance [2]) that while explicit methods require the least number of computations per time step their stability properties restrict their applicability to solving stiff problems. Roughly speaking, the time step h of an explicit method is bounded by $\Delta t_{\mathrm{stab}} = C/|\lambda_{\max}|$ where C is a constant which is usually not large (e.g., C is 1 for the Euler method and ≈ 2.7853 for the fourth-order explicit Runge–Kutta method) and $\lambda_{\max}$ is the eigenvalue of the Jacobian of (1) with the largest absolute value. In many stiff problems, however, the time step Δt_{acc} determined by accuracy considerations far exceeds Δt_{stab}. Thus the stability restriction forces an unreasonably small time step and consequently a prohibitively large amount of computation.

Implicit methods, on the other hand, have much better stability properties. The time step for some implicit integrators is restricted only by accuracy requirements. However, the amount of computation per time step is drastically increased compared to an explicit method since a large nonlinear system of equations has to be solved at each time iteration. The cost of solving the nonlinear system also grows with increasing stiffness of the problem. Thus the goal of developing more efficient time integrators for stiff systems is to provide alternative methods which have better stability properties than explicit schemes and require fewer arithmetic operations per time step than implicit methods. Below we describe how EPI methods accomplish this task.

Construction of a time integrator for the system (1) and (2) can be approached in a different way. Suppose that a Jacobian of the system $\frac{DF}{DU}(U)$ exists and we can expand the right-hand side of (1) as

$$F(U(t)) = F(U_n) + \frac{DF}{DU}(U_n)(U(t) - U_n) + R(U(t)), \tag{4}$$

where $U_n = U(t_n)$ and

$$R(U(t)) = F(U(t)) - F(U_n) - \frac{DF}{DU}(U_n)(U(t) - U_n). \tag{5}$$

Let A_n be the Jacobian

$$A_n = \frac{DF}{DU}(U_n) \tag{6}$$

and let $F_n = F(U_n)$; we can rewrite (1) as

$$\frac{\mathrm{d}U}{\mathrm{d}t}(t) = F_n + A_n(U(t) - U_n) + R(U(t)). \tag{7}$$

The use of an integrating factor e^{-A_nt} in (7) yields

$$\frac{\mathrm{d}}{\mathrm{d}t}(\mathrm{e}^{-A_nt}U(t)) = \mathrm{e}^{-A_nt}(F_n - A_nU_n) + \mathrm{e}^{-A_nt}R(U(t)). \tag{8}$$

Integration of (8) over the time interval $[t_n, t_n + h_n]$ and multiplication by $\mathrm{e}^{A_n(t_n+h_n)}$ leads to its integral form

$$U(t_n + h_n) = U_n + (\mathrm{e}^{A_nh_n} - I)A_n^{-1}F_n + \int_{t_n}^{t_n+h_n} \mathrm{e}^{A_n(t_n+h_n-t)}R(U(t))\,\mathrm{d}t. \tag{9}$$

Eq. (9) is the starting point in developing an exponential propagation iterative scheme. To construct such a scheme we need to (i) formulate an efficient algorithm for computing the second term on the right-hand side of Eq. (9) and the products of matrices $\mathrm{e}^{A_n(t_n+h_n-t)}$ and vectors $R(U(t))$ in the third term and (ii) use a quadrature to approximate the integral in (9). In EPI schemes task (i) is accomplished using Krylov subspace projections and (ii) can be done by either a multistep type or a Runge–Kutta type approach.

The idea of constructing a time integrator using Eq. (9) by formulating a quadrature rule for the integral, or exponential time differencing, have appeared in many publications since the 1960's. Cox has attempted to trace the most notable introductions (and reintroductions) of ETD on his website [3]. Some of the earliest attempts to construct ETD schemes can be found in papers by Certaine [4], Pope [5], Lawson [6] and Nørsett [7]. These papers dealt with task (i) by either considering problems with a diagonal Jacobian matrix or using algorithms like Taylor or Padé expansions to approximate an exponential or another function of a Jacobian. The latter approach is only appropriate for problems for which approximating a function of a matrix does not impose a significant computational overhead. This condition holds trivially for small systems, where any of the algorithms from [8,9] can be used. For large N this can also be true if the stiffness of the problem comes exclusively from the linear terms of $F(U)$. In this case the system (1) is reduced to the initial-value problem

$$\begin{aligned} &\frac{\mathrm{d}U}{\mathrm{d}t} = LU + N(U(t)),\\ &U(t_0) = U_0, \end{aligned} \tag{10}$$

where L is an $N \times N$ constant matrix and $N(U(t))$ is the nonlinear term. The corresponding integral form of this equation is

$$U(t) = \mathrm{e}^{(t-t_0)L}U_0 + \int_{t_0}^{t} \mathrm{e}^{(t-\tau)L}N(U(\tau))\,\mathrm{d}\tau. \tag{11}$$

A particular case of such problems is a linear system of equations with a forcing term, i.e., $N(U(t)) = r(t)$ where $r(t)$ is a known function of t.

Exponential time differencing schemes for problems of type (10) were proposed in a number of papers [10–16]. These methods employed various algorithms to compute an exponential and other functions of the matrix L. Beylkin et al. [12] scale the matrix and compute its exponential by Taylor expansion, then obtain other needed functions of L using a recurrence relation. Kassam and Trefethen [14] use the Cauchy formula

$$f(L) = \frac{1}{2\pi\mathrm{i}} \int_{\Gamma} f(t)(tI - L)^{-1}\,\mathrm{d}t, \tag{12}$$

where the integral is approximated by the trapezoidal rule. Cox and Matthews [13] simply diagonalize the matrix L to compute e^{Lh}. All of these approaches are limited to problems where L is constant and computing a function of L has to be done only once as opposed to every time step.

A more general approach to approximating an exponential or other functions of a large matrix comes from computational linear algebra. Krylov projection methods have been crucial for the development of efficient algorithms to approximate an inverse of a large matrix and find its eigenvalues and eigenvectors [17]. For large ODE systems Krylov methods are used to solve the linear systems that come from Newton method used within implicit integrators [18].

In 1983 Nauts and Wyatt [19] successfully utilized a Krylov projection method for symmetric matrices (i.e., the Lanczos algorithm) to compute the exponentials of discrete Hamiltonian operators for an application in chemical physics. Later the technique was used by Park and Light [20] to exponentially propagate the Schrödinger equation. The idea of approximating general functions of matrices using Krylov subspace projection has been proposed by Van der Vorst [21]. Friesner et al. [22] suggested combining exponential time differencing and Krylov projections to develop a time integrator for general systems of nonlinear ODEs. They proposed a method that used a Chebyshev series approximation to the integrand and an iterative procedure to refine the approximate solution at each time step. Later Gallopoulos and Saad [10] presented their version of an exponential propagation iterative method for linear parabolic equations with a forcing term

$$\frac{\mathrm{d}U}{\mathrm{d}t} = LU + r(t), \tag{13}$$

$$U(t_0) = U_0, \tag{14}$$

and proved some results on the accuracy and stability of these methods. Lawson et al. [11] also considered this class of problems but suggested different quadrature rules for the evaluation of the integral. Since in these methods the computationally expensive Krylov subspace projection had to be performed many times per time step it remained unclear whether these exponential propagation iterative methods offer computational savings compared to standard integrators. These developments were important since they offered a framework which could be used to construct more efficient techniques.

To get an idea of the computational cost of such methods consider possible approximations to the integral in (9). Suppose we choose S nodes to interpolate either the whole integrand $\mathrm{e}^{-A_n t}R(U(t))$ or only the function $R(U(t))$. Then the quadrature formula is either

$$\sum_{i=1}^{S} \mathrm{e}^{-A_n s_i} R(U(s_i)) \quad \text{or} \quad \sum_{i=1}^{S} \phi_i(A_n s_i) R(U(s_i)), \tag{15}$$

where functions $\phi_i(z)$ are the sums of integrals of type $\int_0^1 \mathrm{e}^{zt} t^j \,\mathrm{d}t$ (see the detailed derivation below). Thus to evaluate (9) we need to compute $S+1$ products of a matrix function and a vector. Recall that A_n is a large $N \times N$ matrix and therefore even if the products are evaluated using Krylov subspace projections the whole method becomes computationally expensive. To reduce these computational costs it was proposed in [22] to use Krylov subspace projections with a small *fixed* number of Krylov vectors. However, in [23] it was shown that this formulation of the algorithm could lead to large errors since there was no check whether for a given time step and a fixed number of Krylov vectors approximations to the matrix–vector products were sufficiently accurate. The global residual that was minimized at each time step consisted of the sum of errors incurred by each approximation including every Krylov projection. Cancellations between these errors could occur so that the actual error could still be significant even though the residual was small. Tokman [23] suggested that each component error has to be minimized separately to make the method accurate. However, since this modification makes each Krylov projection more expensive it is unclear how it will affect the efficiency of the whole method and whether the scheme will still be competitive with standard integrators.

A method which addressed the issues discussed above was developed in two excellent papers by Hochbruck et al. [24,25]. The authors used Krylov subspace projections and a Rosenbruck-type methods framework [2, vol. II] to construct exponential propagation iterative methods. They derived analytical error estimates for the convergence of the Krylov projection iteration and general order conditions for exponential Rosenbruck-type methods. They also provided some comparisons of the schemes with explicit and implicit methods. These tests showed the savings the exponential Rosenbruck-type schemes offered compared to explicit algorithms. The question of whether these algorithms can compete with implicit schemes was not directly addressed. However, the results of [24] on the convergence of Krylov iterations suggested that it can be accomplished. To our knowledge the explicit Rosenbruck-type methods represent the first successful attempt to create efficient exponential type schemes for general large systems of type (1).

Finally, we address the question of the application of exponential propagation based methods to large-scale scientific computing problems. As noted above the novelty of exponential propagation based techniques and the lack of understanding of what precisely are the computational savings these schemes provide and what type of problems can benefit from this approach resulted in a very limited use of these techniques. Only a simple second-order exponential propagation method (see (79)) can be found in some applications oriented papers, e.g. [20]. Since this method simply ignores the nonlinear integral in (9) its accuracy is not sufficient to capture the solution of the general nonlinear system (1). To our knowledge none of the other methods have been used and compared to standard techniques in the context of large-scale applications. The only exceptions are the application of the Friesner et al. methods to solve Navier–Stokes equations by Edwards et al. and the numerical integration of resistive magnetohydrodynamics equations to model the solar coronal plasma by Tokman and Bellan [23,26]. In the latter work the fourth-order exponential propagation method of Hochbruck et al. [25] was used, tested and shown to be more efficient than fourth-order explicit Runge–Kutta method [23].

3. Construction of the exponential propagation iterative methods

We begin developing an exponential propagation iterative method from the integral form of the solution to (1)

$$U(t_n+h_n) = U_n + (\mathrm{e}^{A_n h_n} - I)A_n^{-1}F_n + \int_{t_n}^{t_n+h_n} \mathrm{e}^{A_n(t_n+h_n-t)}R(U(t))\,\mathrm{d}t. \tag{16}$$

Recall that in this formula

$U_n = U(t_n) \in \mathbb{R}^N$ is the solution of the system (1) at time t_n,
$F_n = F(U_n) \in \mathbb{R}^N$ is the right-hand side of the system (1) evaluated at time t_n,
$A_n = \frac{DF(U_n)}{DU} \in \mathbb{R}^{N\times N}$ and $R(U(t)) = F(U(t)) - F_n - A_n(U(t) - U_n) \in \mathbb{R}^N$

is the nonlinear remainder of the expansion of $F(U)$ around U_n. To construct a time integrator we need to choose a quadrature to approximate the integral in (16). Note that the evaluation of both the second and the third right-hand side terms in (16) requires computation of a product between a function of a large $N\times N$ matrix and a vector in $\mathbb{R}^N$. Thus first we address the question of computing this product using an iterative algorithm.

3.1. Evaluation of $f(A\tau)b$ via Krylov subspace projections

Suppose $A \in \mathbb{R}^{N\times N}$, $b \in \mathbb{R}^N$, and h is a positive constant. Consider the problem of approximating the product $f(A\tau)b$ where $f(z)$ is analytic in a complex domain Γ which includes the eigenvalues of A. In particular, $f(z) = (1-z)^{-1}$, $f(z) = \mathrm{e}^z$, $f(z) = (\mathrm{e}^z - 1)/z$ and some other related functions will be of interest to us. If A is diagonalizable and can be written as $A = E\Lambda E^{-1}$, where Λ is a diagonal matrix with eigenvalues λ_i of A on the main diagonal and E is the matrix with corresponding eigenvectors of A as its columns, then $f(A\tau)$ can be calculated as

$$f(A\tau) = Ef(\Lambda\tau)E^{-1}, \tag{17}$$

where $f(\Lambda\tau)$ is a diagonal matrix with values $f(\lambda_i\tau)$ on the main diagonal. For non-diagonalizable matrices $f(A\tau)$ can be defined using the Taylor expansion of an analytic function $f(z)$. For example, if I is the $N\times N$ identity matrix then

$$(I - A\tau)^{-1}b = (I + A\tau + (A\tau)^2 + (A\tau)^3 + \cdots + (A\tau)^n + \cdots)b, \tag{18}$$

$$e^{A\tau}b = \left(I + A\tau + \frac{(A\tau)^2}{2!} + \frac{(A\tau)^3}{3!} + \cdots + \frac{(A\tau)^n}{n!} + \cdots\right)b, \tag{19}$$

$$\frac{e^{A\tau} - I}{A\tau}b = \left(I + \frac{A\tau}{2!} + \frac{(A\tau)^2}{3!} + \cdots + \frac{(A\tau)^n}{(n+1)!} + \cdots\right)b. \tag{20}$$

To approximate these products we use projections of A and b onto a Krylov subspace

$$K_m(A,b) = \text{span}\{b, Ab, A^2b, \ldots, A^{m-1}b\},$$

i.e., we project A and b onto $K_m(A,b)$ and calculate the product $f(A\tau)b$ in this subspace. This is accomplished as follows. First an orthonormal basis $\{v_1, v_2, \ldots, v_m\}$ in the subspace $K_m(A,b)$ along with the projector P_m onto $K_m(A,b)$ and a Krylov subspace representation H_m of the matrix A are constructed. Then the obtained results are used to evaluate the approximation to $f(A\tau)b$ in $K_m(A,b)$.

The orthonormal basis $\{v_1, v_2, \ldots, v_m\}$ of $K_m(A,b)$ is constructed using an algorithm based on the Gram–Schmidt orthogonalization of the vectors A^kb which was proposed by Arnoldi in 1951 [27]. A numerically preferable version of this algorithm is the Arnoldi modified Gram–Schmidt procedure [17] given below.

Algorithm 1 (*Arnoldi modified Gram–Schmidt algorithm to construct an orthonormal basis of a Krylov subspace* $K_m(A,b)$).

```
INPUT: Matrix $A \in \mathbb{R}^{N\times N}$, vector $b \in \mathbb{R}^N$ and constant $\tau$
OUTPUT: Orthonormal basis $\{v_1,v_2,\ldots,v_m\}$ of $K_m(A,b)$ and an upper Hessenberg matrix $H_m = V_m^T\tau A V_m$, where $V_m = [v_1\ v_2\ \cdots\ v_m] \in \mathbb{R}^{N\times m}$.
1: $v_1 = b/\|b\|_2$
2: for $j = 1,2,\ldots,m$ do
3:    $w_j = \tau A v_j$
4:    for $i = 1,\ldots,j$ do
5:       $h_{ij} = (w_j, v_i)$
6:       $w_j = w_j - h_{ij}\, v_i$
7:    end for
8:    $h_{j+1,j} = \|w_j\|_2$
9:    if $h_{j+1,j} = 0$ then
10:      Stop
11:   else
12:      $v_{j+1} = w_j/h_{j+1,j}$
13:   endif
14: endfor
```

The Arnoldi modified Gram–Schmidt algorithm can be also written in a matrix form

$$\tau A V_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^T, \tag{21}$$

where e_m is the mth unit vector in $\mathbb{R}^m$, $\{v_1, v_2, \ldots, v_m, v_{m+1}\}$ is an orthonormal basis of $K_m(A,b)$, $V_m = [v_1\ v_2\ \cdots\ v_m] \in \mathbb{R}^{N\times m}$, and H_m is an upper Hessenberg matrix which can be calculated using the orthogonality of the vectors v_i by

$$H_m = V_m^T \tau A V_m. \tag{22}$$

Since V_m is the matrix containing vectors of the orthonormal basis of $K_m(A,b)$ as its columns it defines a projector onto the Krylov subspace $P_m = V_m V_m^T$. Thus if we were to project $f(A\tau)$ and b onto the Krylov space we would obtain $P_m\, f(A\tau)$ and $P_m b$ so that the approximation to $f(A\tau)b$ in K_m is

$$f(A\tau)b \approx V_m V_m^T f(A\tau) V_m V_m^T b. \tag{23}$$

Recalling (22) we make another approximation in order to compute the right-hand side of (23)

$$V_m^{\mathrm{T}} f(A\tau) V_m \approx f(H_m). \tag{24}$$

Noting that $v_1 = b/\|b\|_2$ we have $V_m^{\mathrm{T}} b = \|b\|_2 e_1$ and the Krylov subspace approximation is obtained

$$f(A\tau)b \approx \|b\|_2 V_m f(H_m) e_1. \tag{25}$$

As can be seen from a power series definition of the matrix $f(A\tau)$ the accuracy of the approximation (25) depends on m, the number of Krylov vectors constructed, the eigenvalues of A, the magnitude of τ, and the function f that is being approximated. For example, one can expect that fewer Arnoldi iterations will be needed for functions f with a faster converging Taylor expansion. In fact, this is precisely the feature we will exploit in constructing EPI methods. In addition, the error of the approximation (25) is also influenced by the magnitude of b (e.g., if $b = 0$ the approximation immediately exact). Some general error bounds for the approximation (25) can be found in [24,28]. However, these error bounds cannot be used to estimate a priori how many Krylov vectors are needed to achieve specified accuracy for an arbitrary A, b and f. In practice to implement the Krylov projection approximation of $f(A\tau)$ b to within a prescribed tolerance we need a measure of the error computed at every Arnoldi iteration. In general, the method will be efficient if the number of Arnoldi vectors m is small compared to N. If $m \ll N$ then $H_m \in \mathbb{R}^{m\times m}$ is a small matrix and the task – of approximating $f(H_m)$ is computationally inexpensive and can be done using standard techniques. Such algorithms for the case $f(z) = e^z$ are reviewed in [8]. Thus to complete construction of the Krylov approximation to $f(A\tau)b$ we need to specify (i) a stopping criteria for Algorithm 1 and (ii) an algorithm for computing $f(H_m)$.

Task (i) can be accomplished using the residuals computed in the course of the Arnoldi iteration. Saad [29] proposed using

$$\rho_m = \|b\|_2 h_{m+1,m} [f(H_m)]_{m,1} v_{m+1}. \tag{26}$$

Later Hochbruck et al. [25] presented a more convincing derivation of this residual using a Cauchy integral formula. The stopping criteria for the Arnoldi iteration can then be $\|\rho_m\| < \varepsilon$ where ε is a prescribed tolerance given by the accuracy requirements on the solution. From the formulas of Algorithm it should be clear that the value of τ will influence the number of iterations required for convergence. If τ is reduced the residual (26) will also be smaller and since in a general exponential propagation iterative method τ is the time step, the smaller value of τ means less computations per time step. Thus the key to constructing an efficient exponential propagation iterative scheme is keeping the time step small enough so that Krylov projections are cheap and at the same time much larger than the maximum allowed time step for explicit schemes so that the EPI method offers overall savings.

Note also that most elements of the residual (26) are side products of the Arnoldi iteration but $[f(H_m)]_{m,1}$ is not. Since computing $f(H_m)$ requires $O(m^3)$ operations it might be more efficient not to compute it at each Krylov iteration but to check the residual only at fixed values of m. These values of m can be chosen based on computational cost and on the size of the available memory, since all Krylov vectors $v_1,\ldots,v_m$ must be stored before computing (25). In [25], for instance, it was proposed to evaluate $f(H_m)$ only when $m \in \{1,2,3,4,6,8,11,15,20,27,36,48\}$ to ensure that computing $f(H_m)$ is about as expensive as the calculation of all the previously computed $f(H_j)$. Other strategies can also be employed, e.g., one can simply limit the number of Krylov vectors to the maximum needed for convergence of the slowest Arnoldi iteration during a time step.

If the structure of A guarantees that H_m is diagonalizable then $f(H_m)$ can be computed using formula (17). This happens, for instance, when A is Hermitian and consequently H_m is Hermitian tridiagonal. In general, alternative algorithms have to be used to approximate $f(H_m)$. One of the most popular methods is Padé expansion. When $f(z) = e^z$ or $f(z) = (e^z-1)/z$ this method can be coupled to scaling the matrix as $2^{-k}H_m$ to reduce the number of computations. For a detailed description of this technique as applied to these two functions we refer the reader to [8,9,25]. Evaluation of $f(z) = e^z$ and $f(z) = (e^z-1)/z$ using Padé approximation is implemented in the software package `expokit` [30] which is publicly available at http://www.maths.uq.edu.au/expokit. In Matlab several approximation methods for the exponential function are provided including the routine `expm` which uses scaling and Padé expansion.

It is clear from the definition of a matrix function (e.g., see Eqs. (18)–(20)) that the rate of convergence of a Krylov projection iteration depends on three factors: the spectrum of $A\tau$, the magnitude of b, and the magnitude of the error in approximating $f(z)$ by a polynomial of degree m. One indicator for the latter criteria can be the magnitude of the remainder of the first m terms of a Taylor series for $f(z)$. Thus we can expect that given A and b, the Krylov projection iteration to approximate $f(A)b$ for the functions e^z and $(e^z - 1)/z$ converges faster than the same iteration procedure but with $f(z) = (1-z)^{-1}$. The numerical examples in Section 4 support this conclusion. This feature of Arnoldi iteration is at the base of the computational savings offered by the exponential propagation iterative methods compared to implicit schemes. However, we emphasize that each Krylov subspace projection is a computationally expensive procedure and an EPI method has to be constructed with care so that the number of Krylov projections per time step is minimal and the functions $f(z)$ and vectors b are chosen so that each projection requires the least number of iterations. In the following section we illustrate these points and construct new methods which satisfy this criteria.

3.2. Constructing quadrature-based EPI time integrators

In order to complete the construction of an exponential propagation iterative scheme we have to develop a quadrature rule to approximate the nonlinear integral in (16), i.e., to estimate

$$\int_{t_n}^{t_n+h_n} e^{A_n(t_n+h_n-t)} R(U(t))\,dt. \tag{27}$$

First we have to decide whether to use a polynomial approximation to (i) the function $R(U(t))$ alone or (ii) to the complete integrand $e^{A_n(t_n+h_n-t)}R(U(t))$. If we choose route (i) and construct a multistep type or Runge–Kutta-type scheme the Krylov projection algorithm will have to be used to approximate

$$g_k(A_n h_n)\nabla^k R(U(t_n)) = (-1)^k \int_0^1 e^{A_n h_n(1-s)} \binom{-s}{k} ds \nabla^k R(U(t_n)) \tag{28}$$

or

$$\phi_k(A_n h_n)\Delta^k R(U(t_n)) = \int_0^1 e^{A_n h_n(1-s)} \binom{\gamma s}{k} ds \Delta^k R(U(t_n)), \tag{29}$$

where $k = 0, 1, 2, \ldots, \gamma$ and γ is the number of nodes used in an interpolatory polynomial. The integration variable has been changed from t to s with $t = t_n + sh_n$, $\binom{s}{k} = s(s-1)\cdots(s-k+1)/k!$ is the binomial coefficient, and ∇^k and Δ^k are correspondingly the Newton backward- and forward-difference operators. In case (ii) we have to use Arnoldi iteration to estimate terms of type

$$e^{A_n h_n(1-s_k)} R(U(s_k)), \tag{30}$$

where we used $t = t_n + s_k h_n$ and s_k specifies an interpolation node.

Recalling the factors that influence the convergence rate of an Arnoldi iteration (see Section 3.1) we can argue that using an interpolatory polynomial for $R(U(t))$ is a better choice than expanding the full integrand of (27). First consider the magnitude of vector b in $f(A)b$ which in case (i) is represented by a Newton forward-divided difference of the remainder $R(U(t))$ and in case (ii) is given by the function $R(U(t))$ itself. Since the divided differences are constructed on the nodes within a small time interval we can expect the magnitude of $\Delta^k R(U(t_n))$ to become smaller as k increases. Therefore, from the perspective of the magnitude of b, approach (i) will be preferable over (ii). This conclusion is also supported by considering approximations of functions $g_k(z)$ and $\phi_k(z)$ by polynomials. As indicated above, faster convergence of a Taylor expansion of $f(z)$ corresponds to faster convergence of the Arnoldi iteration to approximate $f(A)b$. As demonstrated in Section 4 with increasing k the functions $g_k(z)$ and $\phi_k(z)$ are better approximated by a polynomial of a fixed degree n. Thus if our method involves Arnoldi iterations applied to estimate matrix–vector products for several such functions, e.g., $\phi_{k1}(z), \phi_{k2}(z), \phi_{k3}(z)$ with $k_1 < k_2 < k_3$ we will expect the amount of computation per time step to be less than for a scheme which uses the same number of Krylov projections but applied to a single function $\phi_{k1}(z)$. Given these considerations supported by our numerical experiments of Section 4 we choose to use the polynomial approximation of $R(U(t))$ to construct quadrature rules.

3.2.1. Multistep type exponential propagation methods

Multistep type exponential propagation schemes can be constructed using quadrature on equally spaced nodes for the integral in (16). Specifically, we discretize the time interval $t_i = t_0 + ih$ and construct an interpolating polynomial approximation to $R(U(t))$ over each interval $[t_n, t_n + h]$ using γ nodes $t_n, t_{n-1}, \ldots, t_{n-(\gamma-1)}$. Let $t = t_n + sh$ with $0 < s < 1$, $R_i = R(U(t_i))$ and ∇^k be a kth Newton backwards difference operator. The nonlinear integral in (16) can be approximated as

$$\int_{t_n}^{t_n+h_n} \mathrm{e}^{A_n(t_n+h-t)} R(U(t))\,\mathrm{d}t \approx h\int_0^1 \mathrm{e}^{A_n h(1-s)}\left(\sum_{k=0}^{\gamma-1}(-1)^k\binom{-s}{k}\nabla^k R_n\right)\mathrm{d}s$$
$$= h\sum_{k=0}^{\gamma-1}(-1)^k\left(\int_0^1 \mathrm{e}^{A_n h(1-s)}\binom{-s}{k}\,\mathrm{d}s\right)\nabla^k R_n. \tag{31}$$

This approximation gives a multistep type exponential propagation scheme of order $\mathrm{O}(h^\gamma)$

$$U(t_n + h_n) = U_n + (\mathrm{e}^{A_n h_n} - I)A_n^{-1}F_n + h\sum_{k=0}^{\gamma-1}(-1)^k\left(\int_0^1 \mathrm{e}^{A_n h(1-s)}\binom{-s}{k}\,\mathrm{d}s\right)\nabla^k R_n, \tag{32}$$

or

$$U(t_n + h_n) = U_n + (\mathrm{e}^{A_n h_n} - I)A_n^{-1}F_n + h\sum_{k=0}^{\gamma-1}(-1)^k g_k(A_n h)\nabla^k R_n, \tag{33}$$

where

$$g_k(z) = \int_0^1 \mathrm{e}^{z(1-s)}\binom{-s}{k}\,\mathrm{d}s. \tag{34}$$

In the context of exponential time differencing schemes for equations of type (10) where $A_n = L$ is time independent these types of methods were suggested in [13]. From the perspective of exponential propagation iterative schemes the efficiency of such methods comes from the observation that for a fixed number m as k increases the error of approximating $g_k(z)$ by a polynomial of degree m decreases. For example, for a third-order method at each time step three Krylov subspace projections must be performed to approximate $(\mathrm{e}^{A_n h_n} - I)A_n^{-1}F_n$, $(\int_0^1 \mathrm{e}^{A_n h(1-s)}\binom{-s}{1}\,\mathrm{d}s)\nabla R_n$ and $(\int_0^1 \mathrm{e}^{A_n h(1-s)}\binom{-s}{2}\,\mathrm{d}s)\nabla^2 R_n$, where we have used $R_n = R(U_n) = F(U_n) - F(U_n) - A_n(U_n - U_n) = 0$. We can expect the number of required Arnoldi iterations to decrease for each subsequent term. To prove this claim rigorously we need to consider projections of the functions $g_k(z)$ onto a space of polynomials of fixed degree and show that the distance becomes smaller as k increases. This analysis is outside the scope of this paper and will be explored in future publications. Here we limit this discussion to a numerical demonstration of this claim in Section 4 and by the following observation.

Consider the remainder of the first m terms of a Taylor expansion of $g_k(z)$ around $z = 0$ given by

$$G_m^k = \left(\int_0^1 (1-s)^{m+1}\mathrm{e}^{\xi(z)(1-s)}\frac{s(s+1)\cdots(s+(k-1))}{k!}\right)\frac{z}{(m+1)!}. \tag{35}$$

If we denote

$$h(z,s) = (1-s)^{m+1}\mathrm{e}^{\xi(z)(1-s)}\frac{s(s+1)\cdots(s+(k-1))}{k!}, \tag{36}$$

the remainder for the next function $g_{k+1}(z)$ can be written as

$$G_m^{k+1} = \left(\int_0^1 h(z,s)\frac{s+k}{k+1}\,\mathrm{d}s\right)\frac{z}{(m+1)!}. \tag{37}$$

Since $h(z,s) \geqslant 0$ and $0 < (s+k)/(k+1) < 1$ for any real or complex number z and $0 \leqslant s \leqslant 1$ we have

$$|G_m^{k+1}| \leqslant |G_m^k|. \tag{38}$$

Thus we can expect the approximation to $g_{k+1}(A_nh)b$ in an m-dimensional Krylov subspace K_m to be better than the approximation of $g_k(A_nh)b$ in the same subspace. Noting also that for a multistep type EPI scheme we need to approximate $g_{k+1}(A_nh)b_2$ and $g_k(A_nh)b_1$ where b_1 and b_2 are Newton divided differences of increasing order, we conclude that estimating $g_{k+1}(A_nh)b_2$ will require fewer Arnoldi iterations than approximating $g_k(A_nh)b_1$.

By examining Taylor expansions of the exact solution and the approximation scheme built using quadrature on the nodes t_n, t_{n-1} we were also able to construct an additional third-order method which does not follow from formula (32). This method is preferable to a third-order scheme that can be constructed by straightforward application of formula (32) since it requires fewer Krylov projections. This two-step third-order method which we label EPI3 is given by

$$U_{n+1} = U_n + g_0(A_nh)hF_n + \frac{2}{3}g_1(A_nh)hR_{n-1}, \tag{39}$$

where

$$\begin{aligned} g_0(z) &= \frac{e^z - 1}{z}, \\ g_1(z) &= \frac{e^z - (1+z)}{z^2}, \\ A_n &= \frac{DF}{DU}(U_n), \\ R_{n-1} &= F(U_{n-1}) - F(U_n) - A_n(U_{n-1} - U_n). \end{aligned} \tag{40}$$

We do not know if other higher-order methods of this type exist and plan to investigate it in the future.

While multistep type schemes are very easy to derive and program using them to design an adaptive step method is costly. The problem, of course, lies in the necessity to re-compute solutions at several previous time iterations if the time step size h is changed. Typically, Runge–Kutta-type schemes yield a cheaper way to build an adaptive time step method. In the next subsection we present a way to derive Runge–Kutta exponential propagation methods and introduce several new schemes of this type.

3.2.2. *Runge–Kutta-type exponential propagation methods*

To construct a Runge–Kutta-type EPI scheme we begin by approximating $R(U(t))$ over the interval $[t_n, t_n + h]$ by an interpolating polynomial defined on γ equally spaced nodes $t_n, t_n + \frac{h}{\gamma}, t_n + \frac{2h}{\gamma}, \ldots, t_n + \frac{(\gamma-1)h}{\gamma}$

$$R(U(t)) = R(U(t_n + sh)) \approx R_n + \sum_{k=1}^{\gamma-1} \frac{(t-t_n)\cdots(t - t_{n+\frac{k-1}{\gamma}})}{k!(\frac{h}{\gamma})^k}\Delta^k R_n = R_n + \sum_{k=1}^{\gamma-1}\binom{\gamma s}{k}\Delta^k R_n, \tag{41}$$

where $0 \leqslant s \leqslant 1$ and $R_n = R(U(t_n))$. Combining this formula with Eq. (16) we obtain

$$U(t_n + h) = U_n + h\frac{e^{A_nh} - I}{A_nh}F_n + h\sum_{k=0}^{\gamma-1}\left(\int_0^1 e^{A_nh(1-s)}\binom{\gamma s}{k}\,ds\right)\Delta^k R_n. \tag{42}$$

To complete the construction of a method we need to approximate the unknown vectors $R(U(t_n + k/\gamma))$. This is achieved by reusing the formula (42) on a smaller number of nodes, i.e., to approximate $R(U(t_n + (\gamma-1)/\gamma))$ we construct quadrature on the nodes $t_n, t_n + h/\gamma, \ldots, t_n + (\gamma-1)/\gamma$, to get $R(U(t_n + (\gamma-2)/\gamma))$ we use nodes $t_n, t_n + h/\gamma, \ldots, t_n + (\gamma-2)/\gamma$, etc. The quadrature weights in these formulas should be computed to ensure the method is of a required order. This procedure is clarified in the following paragraphs where we construct Runge–Kutta EPI methods of order two, three, and four and give general order conditions for the coefficients of those schemes.

Denoting

$$\phi_{\gamma k}(z) = \int_0^1 e^{z(1-s)}\binom{\gamma s}{k}\,ds, \tag{43}$$

we can write a two-stage Runge–Kutta-type exponential propagation scheme as

$$
\begin{aligned}
r_1 &= U_n + a_{11}\phi_{\gamma 0}\left(A_n\frac{h}{\gamma}\right)\frac{h}{\gamma}F_n,\\
U_{n+1} &= U_n + \phi_{\gamma 0}(A_n h)hF_n + b_1\phi_{\gamma 1}(A_n h)hR(r_1).
\end{aligned}
\tag{44}
$$

Using Taylor expansions of the numerical and exact solutions we find that the method is second order for any coefficients a_{11}, b_1, γ. Clearly the computationally cheapest scheme has $a_{11} = b_1 = 0$, i.e., it is the second-order EPI2 scheme

$$
U_{n+1} = U_n + \frac{e^{A_n h} - I}{A_n}F_n. \tag{45}
$$

To obtain a third-order method the following conditions must be satisfied by the coefficients γ, a_{11}, b_1

$$
\begin{aligned}
&\gamma = 2,\\
&\frac{3a_{11}^2 b}{\gamma^2} = 1.
\end{aligned}
\tag{46}
$$

If (46) hold, the functions $\phi_{\gamma k}(z)$ used in (44) are

$$
\begin{aligned}
\phi_{20}(z) &= \int_0^1 e^{z(1-s)}\binom{2s}{0}\,ds = \frac{e^z - 1}{z},\\
\phi_{21}(z) &= \int_0^1 e^{z(1-s)}\binom{2s}{1}\,ds = 2\frac{e^z - (1+z)}{z^2}.
\end{aligned}
\tag{47}
$$

Thus an example of a third-order method is a scheme with $\gamma = 2$, $a_{11} = 2$, and $b = 1/3$ which we label EPIRK3. In the future we will investigate what coefficients yield the smallest error constant.

The general formulas for the third- and fourth-order methods can be developed starting from the following formulation:

$$
\begin{aligned}
r_1 &= U_n + a_{11}\phi_{\gamma 0}\left(A_n\frac{h}{\gamma}\right)\frac{h}{\gamma}F_n,\\
r_2 &= U_n + a_{21}\phi_{\gamma 0}\left(A_n\frac{2h}{\gamma}\right)\frac{2h}{\gamma}F_n + a_{22}\phi_{\gamma 1}\left(A_n\frac{2h}{\gamma}\right)\frac{2h}{\gamma}R(r_1),\\
U_{n+1} &= U_n + \phi_{\gamma 0}(A_n h)hF_n + b_1\phi_{\gamma 1}(A_n h)hR(r_1) + b_2\phi_{\gamma 2}(A_n h)h(-2R(r_1) + R(r_2)),
\end{aligned}
\tag{48}
$$

where we used $R_n = R(U_n) = 0$ and the definition (43). Once again we interpret γ as the number of nodes in the interpolatory polynomial and set it to $\gamma = 3$. Then the functions $\phi_{\gamma k}(z)$ used in (48) are

$$
\begin{aligned}
\phi_{30}(z) &= \frac{e^z - 1}{z},\\
\phi_{31}(z) &= 3\frac{e^z - (1+z)}{z^2},\\
\phi_{32}(z) &= \frac{3}{2}\frac{e^z(6-z) - (6 + 5z + 2z^2)}{z^3}.
\end{aligned}
\tag{49}
$$

Expanding the numerical and exact solutions in Taylor series with the help of the symbolic computation software *Mathematica* we obtain the order conditions for the coefficients in the method. Specifically, we find that the coefficients of all third-order methods of type (48) must satisfy

$$
a_{11}^2 b_1 - a_{11}^2 b_2 + 2a_{21}^2 b_2 = 2. \tag{50}
$$

In order for the method to be of order 4, in addition to condition (50), the coefficients must also satisfy

$$
\begin{aligned}
&4a_{11}^2 b_1 - 3a_{11}^2 b_2 + 10a_{21}^2 b_2 = 12,\\
&2a_{11}^3 b_1 - 2a_{11}^3 b_2 + 8a_{21}^3 b_2 = 9.
\end{aligned}
\tag{51}
$$

First we observe that the order conditions are independent of the coefficient a_{22}. Therefore, to decrease the amount of required computations per time step we can set $a_{22} = 0$. Now to simplify the conditions and find particular methods of order four we denote $c_1 = a_{11}^2 b_1$, $c_2 = a_{11}^2 b_2$, and $c_3 = a_{21}^2 b_2$. Then the system of Eqs. (50) and (51) can be written as a linear system for c_1, c_2, c_3

$$\begin{aligned} &c_1 - c_2 + 2c_3 = 2, \\ &4c_1 - 3c_2 - 10c_3 = 12, \\ &2a_{11}c_1 - 2a_{11}c_2 + 8a_{21}c_3 = 9. \end{aligned} \tag{52}$$

Solving this system for c_1, c_2, c_3 we obtain

$$\begin{aligned} c_1 &= a_{11}^2 b_1 = \frac{9 + 2a_{11} - 12a_{21}}{a_{11} - 2a_{21}}, \\ c_2 &= a_{11}^2 b_2 = \frac{9 + 4a_{11} - 16a_{21}}{2(a_{11} - 2a_{21})}, \\ c_3 &= a_{21}^2 b_2 = \frac{-9 + 4a_{11}}{4(a_11 - 2a_{21})}.\ cr \end{aligned} \tag{53}$$

A compatibility condition can be derived from (53) by noticing that we must have $b_2 = c_2/a_{11}^2 = c_3/a_{21}^2$. Substituting the expressions in (53) into this identity and simplifying the resulting expression we obtain

$$a_{11}^2(-9 + 4a_{11}) - 2(9 + 4a_{11})a_{21}^2 + 32a_{21}^3 = 0. \tag{54}$$

This is a cubic equation with respect to either a_{11} or a_{21} so its roots can be computed exactly. However, it is convenient to have coefficients of a method as rational numbers and to find pairs of rational numbers satisfying (54) is a more difficult task. We used *Mathematica* to search for rational numbers that obey (54) and found the following pairs $(a_{11}, a_{21}) = (9/4, 9/8), (11/16, 55/64), (27/28, 27/28), (27/76, 27/38)$. Once a_{11} and a_{21} have been determined, the construction of a fourth-order Runge–Kutta exponential propagation method is completed by computing the coefficients b_1, b_2 from

$$\begin{aligned} b_1 &= \frac{9 + 2a_{11} - 12a_{21}}{a_{11}^2(a_{11} - 2a_{21})}, \\ b_2 &= \frac{-9 + 4a_{11}}{4a_{21}^2(a_{11} - 2a_{21})}. \end{aligned} \tag{55}$$

Note that this procedure also yields a convenient way to construct an adaptive time step scheme since to form a third-order method b_1, b_2 can be picked to satisfy only condition (50), i.e.,

$$b_2 = \frac{a_{11}^2 b_1 - 2}{a_{11}^2 - 2a_{21}^2}. \tag{56}$$

Table 1 lists the coefficients for several methods of type (48). The third- and fourth-order methods can be embedded to create an adaptive time stepping scheme, in particular, methods EPIRK3A and EPIRK4A can be efficiently used as embedded methods.

Table 1
Coefficients for the third- and fourth-order EPIRK methods of type (48) with $\gamma = 3$

Method's label	4A	4B	4C	4D	3A	3B
Order	4	4	4	4	3	3
a_{11}	$\frac{9}{4}$	$\frac{11}{16}$	$\frac{27}{28}$	$\frac{27}{76}$	$\frac{9}{4}$	$\frac{11}{16}$
a_{21}	$\frac{9}{8}$	$\frac{55}{64}$	$\frac{27}{28}$	$\frac{27}{38}$	$\frac{9}{8}$	$\frac{55}{64}$
a_{22}	0	0	0	0	0	0
b_1	$\frac{160}{243}$	$\frac{-512}{3993}$	$\frac{1568}{2187}$	$\frac{-57760}{6561}$	$\frac{32}{81}$	$\frac{512}{121}$
b_2	$\frac{128}{243}$	$\frac{8192}{3993}$	$\frac{3136}{2187}$	$\frac{23104}{6561}$	0	0

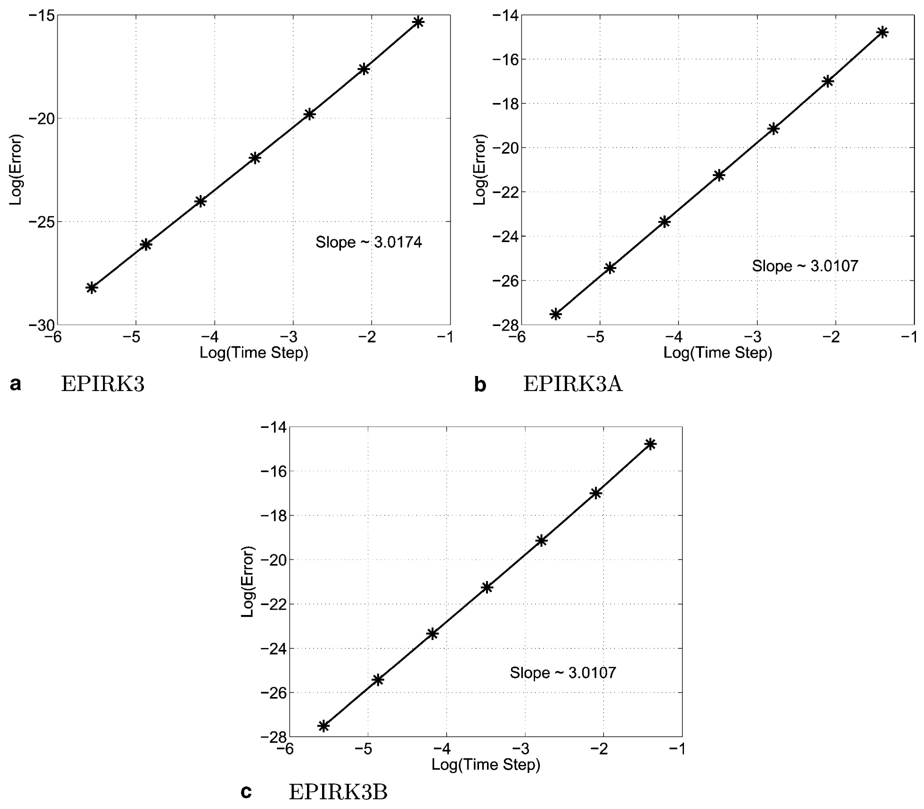


Fig. 1. The graphs show that methods EPIRK3, EPIRK3A, EPIRK3B are indeed third-order exponential propagation Runge–Kutta schemes by plotting the logarithm of the error $\log(\epsilon(h))$ of an approximation to the solution of $y' = -(y+1)(y+3)$ at time $t = 5$ vs. logarithm of a time step $\log(h)$ used for the time integration. Since if the order of a method is p then we expect $\epsilon(h) \approx Ch^p$ so that the slope of the line $\log(\epsilon) = p\log(h) + \log(C)$ in the graphs gives an approximation to the order of the method.

In Figs. 1 and 2 we confirm the order of the methods using a simple one-dimensional autonomous ODE $y' = -(y+1)(y+3)$ on the time interval $t \in [0.1, 5]$ with the exact solution $y(t) = -3 + 2/(1 + e^{-2t})$.

The procedure outlined above can also be used to construct higher-order exponential Runge–Kutta methods. However, it would probably be more efficient to adapt the theory of Butcher's trees [31,32] to derive the general structure of the order conditions for these schemes. We plan to investigate this option in our future work.

3.3. Some properties and implementation of EPI methods

First we comment on the stability of the exponential propagation methods. Note that any exponential propagation method of either multistep or Runge–Kutta type is trivially A-stable. Recall that the linear part of the integrated system of ODEs is computed explicitly in the method as an exponential of the Jacobian. Thus if the products of the Jacobian exponential and vectors are computed exactly the methods are exact for linear systems of ODEs. A-stability trivially follows from the exactness of the methods for linear systems. The nonlinear stability of the methods is a more difficult question and will be the subject of future investigations. In this paper, we demonstrate the performance of the methods on numerical examples of Section 4.

Another issue that has to be discussed is the efficiency of the implementation of the EPI techniques. Since using Arnoldi iterations to compute terms of type $\phi_{\gamma k}(A_n kh/\gamma)b$ can be expensive one has to be careful in implementing EPI schemes in a way that would still yield a computationally efficient method. Consider, for instance, methods of type (48). Note that since the Arnoldi algorithm is scale-invariant (i.e., if for a matrix

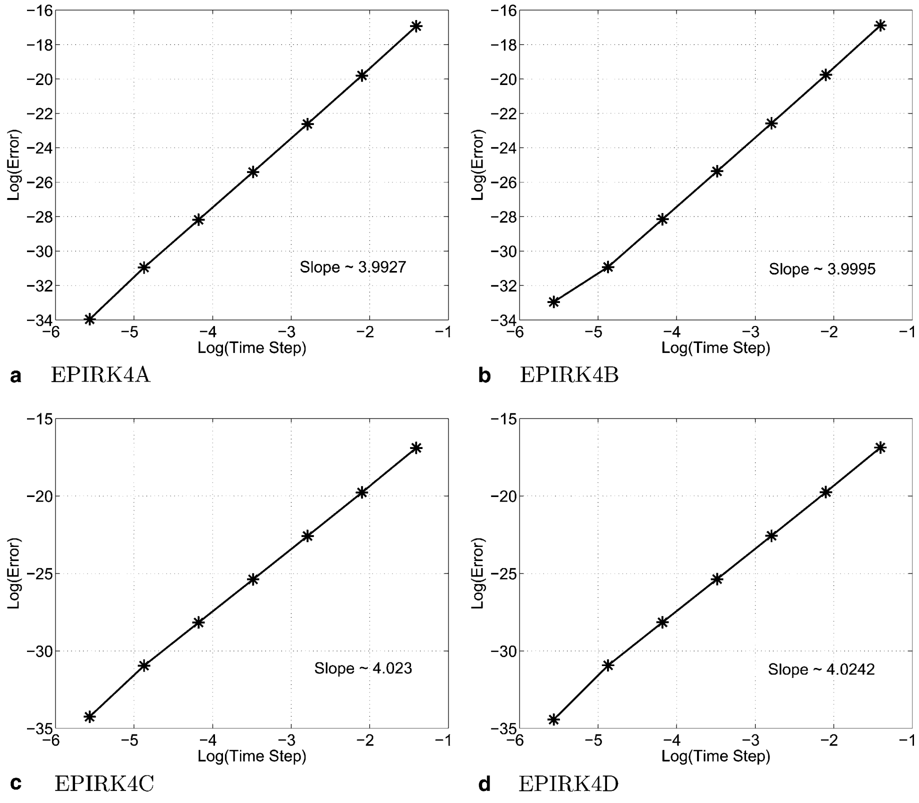


Fig. 2. The graphs show that methods EPIRK4A, EPIRK4B, EPIRK4C and EPIRK4D are indeed fourth-order exponential propagation Runge–Kutta schemes by plotting the logarithm of the error $\log(\epsilon(h))$ of an approximation to the solution of $y' = -(y+1)(y+3)$ at time $t = 5$ vs. the logarithm of the time step $\log(h)$ used for the time integration. Since if the order of a method is p, we expect $\epsilon(h) \approx Ch^p$ so that the slope of the line $\log(\epsilon) = p\log(h) + \log(C)$ in the graphs gives an approximation to the order of the method.

A we have $H = V^T A V$ then for the matrix σA we have $\sigma H = V^T(\sigma A)V)$ all of the terms $\phi_{\gamma 0}(A_n h/\gamma)(h/\gamma)F_n$, $\phi_{\gamma 0}(A_n 2h/\gamma)(2h/\gamma)F_n$, $\phi_{\gamma 0}(A_n h)hF_n$ can be computed using only one Arnoldi iteration. Since the residual of the Krylov projection depends on the factor that multiplies both A and b the last of these terms $\phi_{\gamma 0}(A_n h)hF_n$ should be computed first to make sure that the residuals for all three products are within the required tolerance. Based on these considerations we can also conclude that once the Krylov projection of $\phi_{\gamma 1}(A_n h)hR(r_1)$ is performed the same Krylov basis can be used to compute $\phi_{\gamma 2}(A_n h)h(-2R(r_1))$. However, the residual also scales with the norm of b in $\phi_{\gamma k}(A)b$. Since the vector $-2R(r_1) + R(r_2)$ is a Newton forward-difference it can be much smaller than $R(r_2)$. Thus Arnoldi iteration to estimate $\phi_{\gamma 2}(A_n h)h(-2R(r_1) + R(r_2))$ could be much cheaper than using the previously computed Krylov basis to calculate $\phi_{\gamma 2}(A_n h)h\,(-2R(r_1))$ along with executing another Arnoldi iteration to calculate $\phi_{\gamma 2}(A_n h)hR(r_2)$. These issues have to be judged based on the application, and systematic procedures should be developed to adaptively decide the course of action, e.g., we can cheaply compute the norm of $-2R(r_1) + R(r_2)$ and decide what would be a more efficient way to compute the solution at the next time step.

Let us address the question of the efficiency of EPI methods compared to explicit schemes. To decide whether using an EPI method will be beneficial for a particular problem one has to evaluate whether the added computational cost per time step will be compensated by the savings provided by a larger time step. Surely, if for a particular problem the stability bound time step Δt_{stab} is nearly equal to the size of the time step Δt_{acc} given by accuracy requirements, the explicit methods must be used rather than either implicit or exponential propagation schemes. Now suppose $\Delta t_{\mathrm{acc}} \gg \Delta t_{\mathrm{stab}}$, then we can estimate at what value of the ratio $\Delta t_{\mathrm{acc}}/\Delta t_{\mathrm{stab}}$ EPI schemes become more efficient than an explicit method. We derive an estimate of this ratio for an explicit scheme and an EPI method both of order p. Assume that an explicit method requires N_{RHS} operations to evaluate the right-hand side $F(U)$, then the number of operations needed by the explicit method per time step can generally be approximated by pN_{RHS}. Thus the total number of operations to integrate the system up to some time T with the explicit method is approximately $C_{\mathrm{EXPL}} = pN_{\mathrm{RHS}}T/\Delta t_{\mathrm{stab}}$. The computational cost of each time step of an EPI method will be dominated by Arnoldi iterations. Computation of the orthonormal basis of a Krylov subspace of dimension m requires $2m(N_{\mathrm{J}} + mN)$ operations, where N_{J} is the number of operations it takes to evaluate Jacobian-vector products and N is dimensionality of the original system of ODEs. Once the basis is computed the evaluation of functions of $m \times m$ matrix H requires $\mathrm{O}(m^3)$ operations. As noted above even if several Arnoldi iterations are needed by the straightforward formulation of a method, careful implementation can allow us to reuse the Krylov basis and significantly reduce the amount of computation. However, here we consider the worst case scenario where we need to perform $p - 1$ Krylov projections for a method of order p. Then the total number of operations needed to integrate a system up to time T by the EPI scheme can be approximated as

$$C_{\mathrm{EPI}} = (pN_{\mathrm{RHS}} + (p-1)(2m(N_J + mN) + Cm^3))\frac{T}{\Delta t_{\mathrm{acc}}}, \tag{57}$$

where C is a constant. In order for the EPI method to be more efficient than the explicit scheme we need $C_{\mathrm{EXPL}} \gtrapprox C_{\mathrm{EPI}}$ which is equivalent to

$$\frac{\Delta t_{\mathrm{acc}}}{\Delta t_{\mathrm{stab}}} \gtrapprox 1 + \frac{(p-1)(2m(N_J + mN) + Cm^3)}{pN_{\mathrm{RHS}}}. \tag{58}$$

While this, of course, is a very rough estimate the procedure we outlined can be used as guidance for evaluating whether an EPI scheme can be used more efficiently than an explicit method.

Finally, note that exponential propagation methods are as parallelizable as the implicit methods with Newton–Krylov solvers, i.e., if efficient parallel codes can be developed to evaluate the right-hand side operator of the ODE system $F(U)$ and products between a Jacobian matrix and a vector, then the whole method can be efficiently used on a parallel computer. Just like the Newton–Krylov methods, the EPI techniques are matrix free schemes where the Jacobian matrix does not have to be computed and stored explicitly. Instead a function which evaluates the Jacobian-vector products must be implemented. Therefore, the extensive experience of parallelizing implicit methods with Newton–Krylov solvers can be easily used to parallelize an EPI method.

4. Numerical examples

To test the EPI methods and compare them with explicit and implicit schemes we consider the following three problems commonly used to test numerical methods for stiff systems. Since we are only interested in these problems from the perspective of testing the performance of our numerical methods we will discuss neither the applications associated with these differential equations nor the full spectrum of the behavior exhibited by their solutions. In fact, where possible we will set initial and boundary conditions and the parameters to the values given in previously published numerical tests [2].

All calculations presented in this section were done using Matlab programs on a dual 3 GHz-processor Pentium PC with 2Gb memory. When an inverse or an exponential of a small matrix H_m had to be evaluated we used the Matlab functions `inv` or `expm` which implements scaling and squaring algorithm with a Padé approximation. Note that a general implementation of these methods will require more careful treatment of the computations involving H_m. In particular, it has to be checked whether matrix H_m is close to a singular matrix and in case it is the functions $\phi_{\gamma k}(H_m)$ might have to be computed using Taylor expansions. We have not encountered such cases in practice. However, one can argue that since the Taylor series for functions $\phi_{\gamma k}$ converge rapidly and the matrix H_m is small, such computation should not be much more expensive than the non-singular case.

We investigate the performance of the EPI methods using the following example problems.

BRUSS – these are the Brusselator equations [33] which model multimolecular reactions using the laws of chemical kinetics:

$$\begin{aligned}\frac{\partial u}{\partial t} &= 1 + uv^2 - 4u + \alpha\frac{\partial^2 u}{\partial x^2},\\ \frac{\partial v}{\partial t} &= 3u - u^2 v + \alpha\frac{\partial^2 v}{\partial x^2}.\end{aligned} \tag{59}$$

Following [2] we choose $0 \leqslant x \leqslant 1$ with initial and boundary conditions

$$\begin{aligned}&u(0,t) = u(1,t) = 1, \quad v(0,t) = v(1,t) = 3,\\ &u(x,0) = 1 + \sin(2\pi x), \quad v(x,0) = 3.\end{aligned}$$

We discretize the diffusive terms in (59) using second-order centered finite-differences on the spatial grid $x_i = i/(N+1)$ with the node spacing $\Delta x = 1/(N+1)$ and obtain a system of $2N$ ODEs

$$\begin{aligned}\frac{du_i}{dt} &= 1 + u_i^2 v_i - 4u_i + \frac{\alpha}{(\Delta x)^2}(u_{i-1} - 2u_i + u_{i+1}),\\ \frac{dv_i}{dt} &= 3u_i - u_i^2 v_i + \frac{\alpha}{(\Delta x)^2}(v_{i-1} - 2v_i + v_{i+1}), \quad i = 1,\ldots,N,\end{aligned} \tag{60}$$

with $u_0(t) = u_{N+1}(t) = 1$, $v_0(t) = v_{N+1}(t) = 3$ and initial values

$$u_i(0) = 1 + \sin(2\pi x_i), \quad v_i(0) = 3, \quad i = 1,\ldots,N.$$

The Jacobian of this system is a $2N \times 2N$ matrix

$$J_{\text{BRUSS}} = \begin{pmatrix}\operatorname{diag}(2u_i v_i - 4) & \operatorname{diag}(u_i^2)\\ \operatorname{diag}(3 - 2u_i v_i) & \operatorname{diag}(-u_i^2)\end{pmatrix} + \frac{\alpha}{(\Delta x)^2}\begin{pmatrix}K & 0\\ 0 & K\end{pmatrix}, \tag{61}$$

where

$$K = \begin{pmatrix}-2 & 1 & & & \\ 1 & -2 & 1 & & \\ & 1 & \ddots & \ddots & \\ & & \ddots & -2 & 1\\ & & & 1 & -2\end{pmatrix}. \tag{62}$$

BURGERS – This is the Burgers equation

$$u_t + uu_x = \nu u_{xx} \tag{63}$$

which is discretized in space on a grid of N points to produce the following system of ODEs

$$\frac{du_i}{dt} = -\frac{u_{i+1}^2 - u_{i-1}^2}{4\Delta x} + \frac{\nu}{(\Delta x)^2}(u_{i+1} - 2u_i + u_{i+1}), \quad i = 1,\ldots,N \tag{64}$$

over the spatial domain $0 \leqslant x \leqslant 1$ with initial and boundary values

$$u_0(t) = u_{N+1}(t) = 0, \quad u_i(0) = (\sin(3\pi x_i))^2(1 - x_i)^{3/2}, \quad x_i = i/(N+1). \tag{65}$$

The Jacobian matrix of this system is

$$J_{\text{BURGERS}} = \frac{1}{2\Delta x}\begin{pmatrix}0 & -u_2 & & & & \\ u_1 & 0 & -u_3 & & & \\ & u_2 & 0 & -u_4 & & \\ & & \ddots & \ddots & \ddots & \\ & & & u_{N-2} & 0 & -u_N\\ & & & & u_{N-1} & 0\end{pmatrix} + \frac{\nu}{(\Delta x)^2}K, \tag{66}$$

where K is the same $N \times N$ matrix as in (62).

CUSP – As explained in [2] this system is the combination of threshold-nerve-impuls mechanism of Fitz-Hugh and Nagumo nerve conduction equation [34,35], the cusp catastrophe "with smooth return" [36], and the Van der Pol oscillator

$$\begin{aligned}
\frac{\partial y}{\partial t} &= -\frac{1}{\varepsilon}(y^3 + ay + b) + \sigma\frac{\partial^2 y}{\partial x^2},\\
\frac{\partial a}{\partial t} &= b + 0.07v + \sigma\frac{\partial^2 a}{\partial x^2},\\
\frac{\partial b}{\partial t} &= (1 - a^2)b - a - 0.4y + 0.035v + \sigma\frac{\partial^2 b}{\partial x^2},
\end{aligned} \tag{67}$$

where

$$v = \frac{u}{u + 0.1}, \quad u = (y - 0.7)(y - 1.3). \tag{68}$$

These equations are considered on the domain $0 \leqslant x \leqslant 1$ and discretized on a grid of N points $x_i = i/N$ with spacing $\Delta x = 1/N$. Periodic boundary conditions are imposed on y, a, b and the initial conditions are set to

$$y_i(0) = 0, \quad a_i(0) = -2\cos(2\pi x_i), \quad b_i(0) = 2\sin(2\pi x_i), \quad i = 1, \ldots, N.$$

In discrete form these equations constitute a system of $3N$ ODEs

$$\begin{aligned}
\frac{\mathrm{d}y_i}{\mathrm{d}t} &= -\frac{1}{\varepsilon}(y_i^3 + a_i y_i + b_i) + \frac{\sigma}{(\Delta x)^2}(y_{i-1} - 2y_i + y_{i+1}),\\
\frac{\mathrm{d}a_i}{\mathrm{d}t} &= b_i + 0.07v_i + \frac{\sigma}{(\Delta x)^2}(a_{i-1} - 2a_i + a_{i+1}),\\
\frac{\mathrm{d}b_i}{\mathrm{d}t} &= (1 - a_i^2)b_i - a_i - 0.4y_i + 0.035v_i + \frac{\sigma}{(\Delta x)^2}(b_{i-1} - 2b_i + b_{i+1})
\end{aligned} \tag{69}$$

for $i = 1, \ldots, N$ with

$$v_i = \frac{u_i}{u_i + 0.1}, \quad u_i = (y_i - 0.7)(y_i - 1.3), \tag{70}$$

and

$$\begin{aligned}
&y_0 = y_N, \quad a_0 = a_N, \quad b_0 = b_N,\\
&y_{N+1} = y_1, \quad a_{N+1} = a_1, \quad b_{N+1} = b_1.
\end{aligned}$$

The parameters in the problem are chosen so that the stiffness comes from both the spatial discretization of the diffusive terms as well as the small factor ε which multiplies the nonlinear term of the right-hand side in the equation for y, i.e., $\varepsilon = 10^{-4}$ and $\sigma = 1/144$. The Jacobian matrix of this system (69) is

$$J_{\text{CUSP}} = \begin{pmatrix}
\text{diag}((-1/\varepsilon)(3y_i^2 + a_i)) & \text{diag}((-1/\varepsilon)y_i) & \text{diag}(-1/\varepsilon)\\
\text{diag}\left(0.014\frac{y_i - 1}{(y_i^2 - 2y_i + 1.01)^2}\right) & \text{diag}(0) & \text{diag}(1)\\
\text{diag}\left(-0.4 + 0.007\frac{y_i - 1}{(y_i^2 - 2y_i + 1.01)^2}\right) & \text{diag}(-1 - 2b_i a_i) & \text{diag}(1 - a_i^2)
\end{pmatrix} + \frac{\sigma}{(\Delta x)^2}\begin{pmatrix} K_p & 0 & 0\\ 0 & K_p & 0\\ 0 & 0 & K_p \end{pmatrix}, \tag{71}$$

where K_p is the matrix resulting from a second-order centered finite difference discretization of the diffusive term given periodic boundary conditions, i.e.,

$$K_p = \begin{pmatrix} -2 & 1 & 0 & \dots & 0 & 1 & 0 & 0 \\ 1 & -2 & 1 & \dots & \dots & \dots & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & & & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & & & \vdots \\ \vdots & & & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & & & 1 & -2 & 1 \\ 0 & 0 & 1 & 0 & \dots & 0 & 1 & -2 \end{pmatrix}. \tag{72}$$

To clarify the advantages of the EPI methods over standard integrators we break the tests into two parts each designed to show a particular aspect of the methods' performance. The goal of the first set of tests is to demonstrate that using Arnoldi iteration to approximate an exponential of the Jacobian matrix as well as functions of type $g_k(z)$ and $\phi_{\gamma k}(z)$ defined in Section 3 is a more efficient procedure than using the same Arnoldi algorithm to invert the matrix $(I - hA_n)$. To see how this relates to the performance of an EPI scheme compared to implicit methods consider a simple example of an initial value problem for a N-dimensional linear system of ODEs:

$$\begin{aligned} &\frac{dU}{dt} = AU, \\ &U(t_0) = U_0, \end{aligned} \tag{73}$$

where $A \in \mathbb{R}^{N\times N}$, $U_0, U(t) \in \mathbb{R}^N$. The simplest and computationally cheapest implicit integrator for this problem is the backward Euler method:

$$U_{n+1} = U_n + hAU_{n+1}, \tag{74}$$

that can also be written as

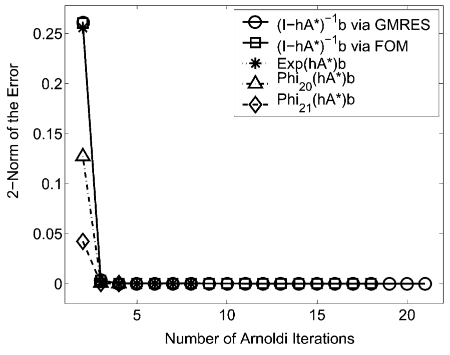
$$U_{n+1} = (I - hA)^{-1}U_n, \tag{75}$$

where I is the $N \times N$ identity matrix and h is the time step. The exponential propagation integrator for problem (73) is

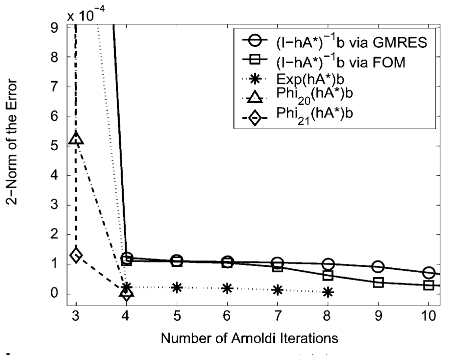
$$U_{n+1} = e^{hA}U_n, \tag{76}$$

which is the exact solution of the system. If N is large and A is general, so no efficient solver optimized for inverting $(I - hA)$ is available, in both of the solvers (75) and (76) an iterative Krylov projection based method must be used. However, in (75) Arnoldi iteration has to be performed in the context of methods like FOM or GMRES to estimate $(I - hA)^{-1}U_n$ while in (76) the same Arnoldi iteration is used to approximate $e^{hA}U_n$. Since both methods are A-stable the time step is not restricted by the stability requirement and the method with less computations per time step is more efficient. Similarly, for nonlinear systems where $A = A(U(t_n)) = A_n$ is time dependent the implicit method involves Newton iteration within which a product of vectors and an inverse of $(I - hA_n)$ has to be computed. For an EPI scheme products of vectors with e^{hA_n}, $g_k(hA_n)$, or $\phi_k(hA_n)$ have to be calculated.

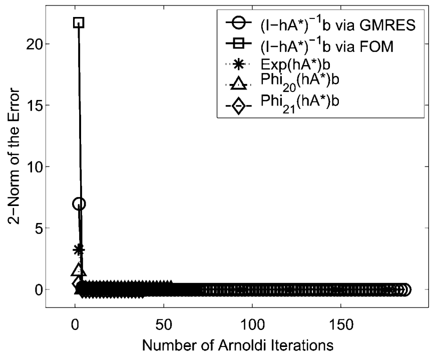
Thus for each of the numerical examples above we perform the following tests. First, we integrate the system up to some time $t = t_*$ using *Matlab's* routine `ode23s` with relative and absolute tolerances set to 10^{-6} to obtain $U(t_*)$. Then we compute the Jacobian matrix $A_* = A(U(t_*)) = \frac{DF}{DU}(U(t_*))$, pick a vector $b \in \mathbb{R}^N$ and plot the number of Arnoldi iterations it takes to estimate $(I - hA_*)^{-1}b$, $e^{hA_*}b$, $\phi_{20}(hA_*)b$, and $\phi_{21}(hA_*)b$ to within a given tolerance *tol*. Fig. 3 and Table 2 show the results of this test for the Brusselator example with $t_* = 5.0$. While we found similar behavior for b picked at random and $b = U_*$ the results shown below demonstrate the most relevant case where $b = F(U_*)$. For all these choices of b we found that the relative position of the curves in Fig. 3 remains the same, our choice of $b = F(U_*)$, h and N clarifies the trends as the stiffness of the problems increases. As we can see from Tables 2 and 3, for large step sizes $h = 0.05, 0.5$ the Krylov subspace projections



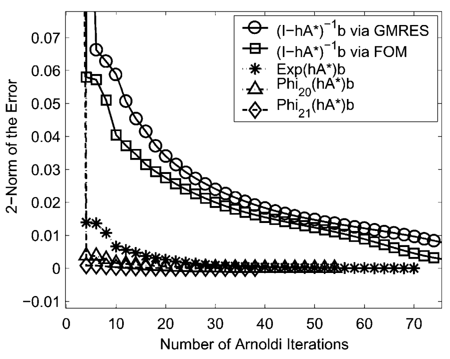
a $h = 0.05$



b $h = 0.05$, zoomed image of (a)



c $h = 0.5$



d $h = 0.5$, zoomed image of (c)

Fig. 3. We plot the 2-norm of the error in the Krylov subspace approximation of (i) $(I - hA_*)^{-1}b$ using GMRES, (ii) $(I - hA_*)^{-1}b$ using FOM, (iii) $e^{hA_*}b$, (iv) $\phi_{20}(hA_*)b$, and (v) $\phi_{21}(hA_*)b$ vs. the number of Arnoldi iterations it took to obtain this error for the BRUSS example. In these runs $N = 200$ and $t_* = 5.0$.

Table 2
This table lists the number of Arnoldi iterations it took to approximate the expressions (i) $(I - hA_*)^{-1}b$ using GMRES, (ii) $(I - hA_*)^{-1}b$ using FOM, (iii) $e^{hA_*}b$, (iv) $\phi_{20}(hA_*)b$, and (v) $\phi_{21}(hA_*)b$ for the Brusselator example to within $tol = 10^{-5}$

Problem size, $2N$	$(I - hA_*)^{-1}b$ via GMRES	$(I - hA_*)^{-1}b$ via FOM	$e^{hA_*}b$	$\phi_{20}(hA_*)b$	$\phi_{21}(hA_*)b$
200	10	9	5	4	4
400	21	18	8	4	4
800	43	37	13	5	4

In this example $h = 0.05$, $t_* = 5$, grid sizes $N = 100, 200, 400$, and $b = F(U_*)$. Note that the Courant–Friedrichs–Levy (CFL) condition restricted time steps corresponding to the grid sizes $N = 100, 200, 400$ are $\Delta t_{\text{stab}} = 1.22 \times 10^{-3}$, 3.09×10^{-4}, 7.77×10^{-5}.

Table 3
This table lists the number of Arnoldi iterations it took to approximate the expressions (i) $(I - hA_*)^{-1}b$ using GMRES, (ii) $(I - hA_*)^{-1}b$ using FOM, (iii) $e^{hA_*}b$, (iv) $\phi_{20}(hA_*)b$, and (v) $\phi_{21}(hA_*)b$ for the Brusselator example to within $tol = 10^{-5}$

Problem size, $2N$	$(I - hA_*)^{-1}b$ via GMRES	$(I - hA_*)^{-1}b$ via FOM	$e^{hA_*}b$	$\phi_{20}(hA_*)b$	$\phi_{21}(hA_*)b$
200	92	86	35	27	19
400	187	174	70	55	38
800	382	359	140	112	78

In this example $h = 0.5$, $t_* = 5$, grid sizes $N = 100, 200, 400$, and $b = F(U_*)$.

can be more efficiently used to complete the intermediate steps in EPI schemes (i.e., calculation of expressions of type $e^{hA_*}b$, $\phi_{20}(hA_*)b$, and $\phi_{21}(hA_*)b$) compared to the intermediate steps that have to be performed within the Newton iteration of an implicit scheme (i.e., using Arnoldi iteration to approximate $(I - hA_*)^{-1}b$). This trend remains the same as N or h increase and consequently the problems becomes more stiff (Fig. 3, Tables 2 and 3).

Figs. 3–5 show plots of the 2-norm of the error vs. the corresponding number of Arnoldi iterations for different time step sizes. We found that the relative positions of the curves do not change as the grid size N is

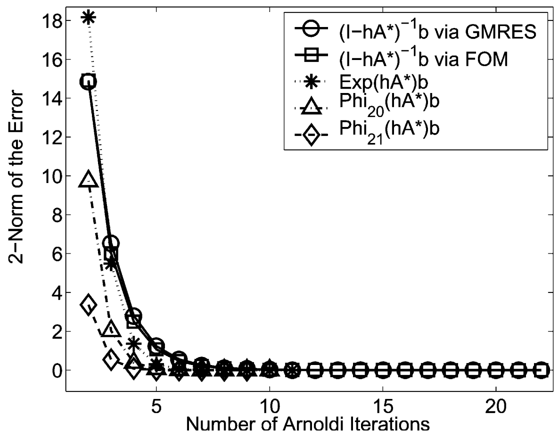


Fig. 4. These graphs plot the 2-norm of the error in the Krylov subspace approximation of (i) $(I - hA_*)^{-1}b$ using GMRES, (ii) $(I - hA_*)^{-1}b$ using FOM, (iii) $e^{hA_*}b$, (iv) $\phi_{20}(hA_*)b$, and (v) $\phi_{21}(hA_*)b$ vs. the number of Arnoldi iterations it took to obtain this error for the BURGERS example. For this calculation $N = 1000$, $t_* = 1.0$.

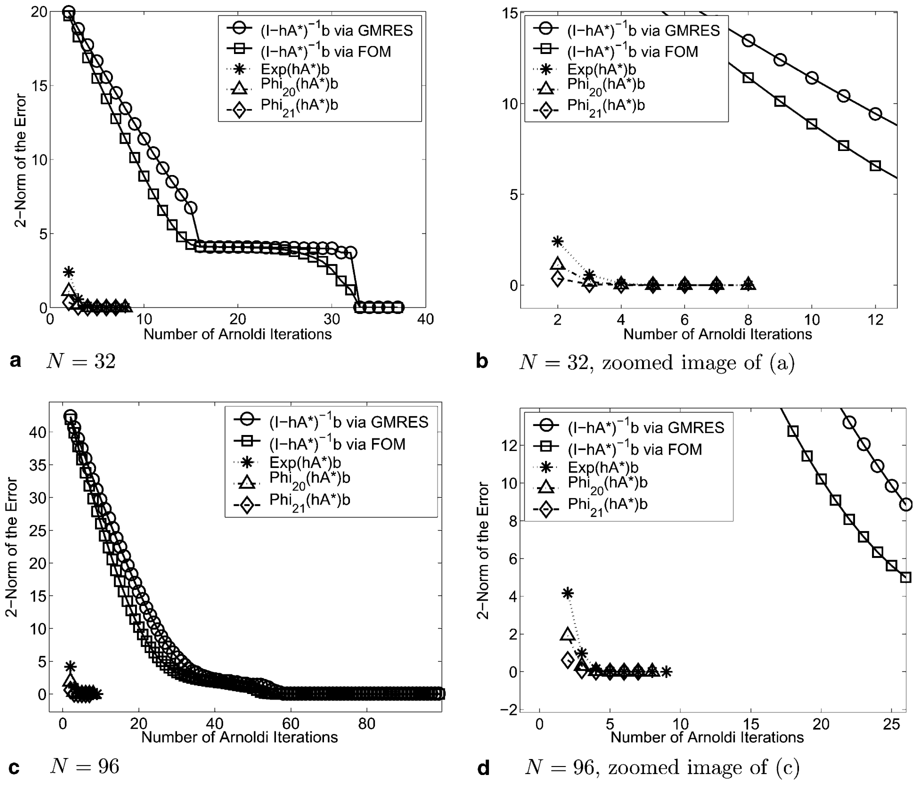


Fig. 5. These graphs plot the 2-norm of the error in the Krylov subspace approximation of (i) $(I - hA_*)^{-1}b$ using GMRES, (ii) $(I - hA_*)^{-1}b$ using FOM, (iii) $e^{hA_*}b$, (iv) $\phi_{20}(hA_*)b$, and (v) $\phi_{21}(hA_*)b$ vs. the number of Arnoldi iterations it took to obtain this error for the CUSP example. In these runs $N = 32,96$, $h = 5 \times 10^{-5}$, $t_* = 10^{-5}$.

increased. Tables 2, 4 and 5 show how many iterations it took to reduce the error below 10^{-3}. The figures and tables confirm the claim above by showing that there are regimes of large time steps where the Krylov projections used within an EPI scheme will be more efficient than Krylov projection based inversions of $(I - hA_n)$ which are part of an implicit method.

The Burgers example yields behavior very similar to the Brusselator problem. We use the time step $h = 0.005$, $t_* = 1.0$, $v = 0.0003$, and $N = 500, 1000, 2000$. Fig. 4 and Table 4 show the results of the test and once again demonstrate the efficiency of Krylov subspace projection technique in computing the functions involving exponential compared to more computationally intensive approximation of the product of an inverse of the matrix $(I - hA_*)$ and a vector b.

In the CUSP example in addition to demonstrating the trend that we observed in the previous two problems we also check whether the behavior is repeated for two different times $t_* = 10^{-5}$ and 10^{-4}. The Matlab routine `ode23s` uses a time step on the order of 10^{-7}–10^{-6} with the relative and absolute tolerances set to 10^{-6}, so we use $h = 5 \times 10^{-5}$. We also set $b = hF(U_*)$. Table 5 and Fig. 5 show the number of Arnoldi iterations for $t_* = 10^{-5}$ and Table 6 and Fig. 6 lists results for $t_* = 10^{-4}$. Recall that unlike the previous two examples in this problem the stiffness is governed not only by the discretization of the diffusion terms but also by the nonlinear term with the factor $1/\varepsilon$. It is interesting to note that the Krylov subspace approximation of the functions involving an exponential seems much less sensitive to a slight ($\sim 10^{-1}$) increase of stiffness as N grows. Fig. 6 also demonstrates that even when the initial error at the first Arnoldi iteration is the largest for the exponential function, it converges faster and thus it still requires fewer iterations than approximating $(I - hA_*)^{-1}b$.

The second set of tests directly addresses the question of performance of the EPI schemes as compared to standard integrators. The complexity of the implementation and the number of possible quadrature rules, and consequently numerical scheme grows as the order of the method increases. Thus in order to illuminate

Table 4
This table lists the number of Arnoldi iterations it took to approximate the expressions (i) $(I - hA_*)^{-1}b$ using GMRES, (ii) $(I - hA_*)^{-1}b$ using FOM, (iii) $e^{hA_*}b$, (iv) $\phi_{20}(hA_*)b$, and (v) $\phi_{21}(hA_*)b$ for the Burgers example to within $tol = 10^{-5}$

Problem size, N	$(I - hA_*)^{-1}b$ via GMRES	$(I - hA_*)^{-1}b$ via FOM	$e^{hA_*}b$	$\phi_{20}(hA_*)b$	$\phi_{21}(hA_*)b$
500	15	15	9	8	7
1000	22	22	11	10	9
2000	40	39	15	13	12

In this example $h = 0.005$, $t_* = 1.0$, grid sizes $N = 500, 1000$, and $b = F(U_*)$.

Table 5
This table lists the number of Arnoldi iterations it took to approximate the expressions (i) $(I - hA_*)^{-1}b$ using GMRES, (ii) $(I - hA_*)^{-1}b$ using FOM, (iii) $e^{hA_*}b$, (iv) $\phi_{20}(hA_*)b$, and (v) $\phi_{21}(hA_*)b$ for the CUSP example to within $tol = 10^{-5}$

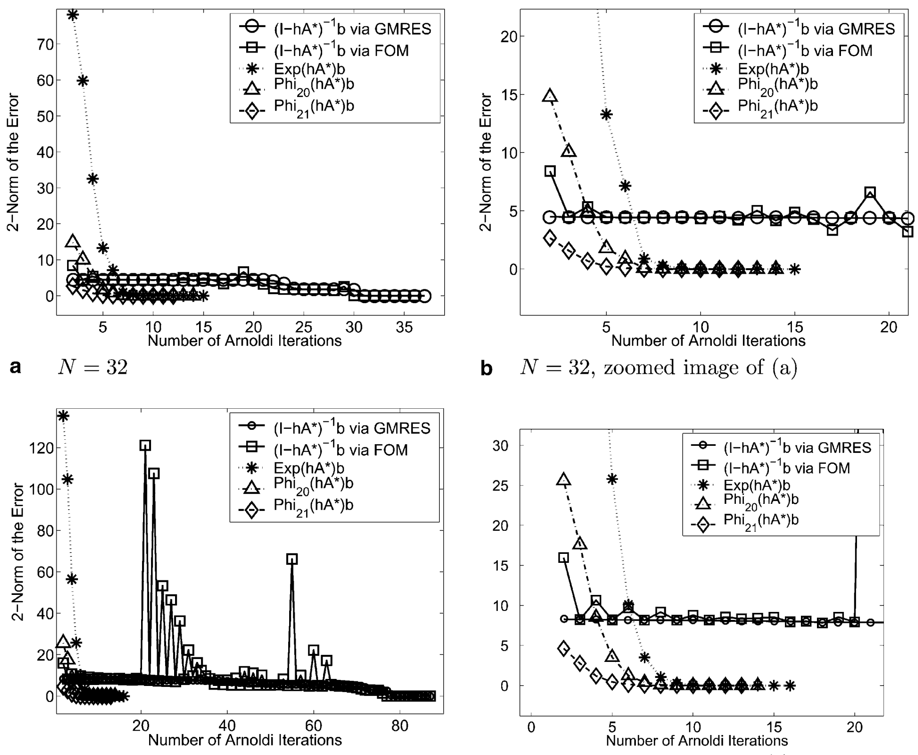
Problem size, $3(N+2)$	$(I - hA_*)^{-1}b$ via GMRES	$(I - hA_*)^{-1}b$ via FOM	$e^{hA_*}b$	$\phi_{20}(hA_*)b$	$\phi_{21}(hA_*)b$
102	37	37	8	8	7
198	68	68	8	8	7
294	99	99	8	8	7

In this example $h = 5 \times 10^{-5}$, $t_* = 10^{-5}$, grid sizes $N = 32, 64, 96$, and $b = hF(U_*)$.

Table 6
This table lists the number of Arnoldi iterations it took to approximate the expressions (i) $(I - hA_*)^{-1}b$ using GMRES, (ii) $(I - hA_*)^{-1}b$ using FOM, (iii) $e^{hA_*}b$, (iv) $\phi_{20}(hA_*)b$, and (v) $\phi_{21}(hA_*)b$ for the CUSP example to within $tol = 10^{-5}$

Problem size, $3(N+2)$	$(I - hA_*)^{-1}b$ via GMRES	$(I - hA_*)^{-1}b$ via FOM	$e^{hA_*}b$	$\phi_{20}(hA_*)b$	$\phi_{21}(hA_*)b$
102	37	36	15	14	12
198	62	61	16	14	13
294	87	87	16	14	13

In this example $h = 5 \times 10^{-5}$, $t_* = 10^{-4}$, grid sizes $N = 32, 64, 96$, and $b = hF(U_*)$.



a $N = 32$ **b** $N = 32$, zoomed image of (a)

c $N = 96$ **d** $N = 96$, zoomed image of (c)

Fig. 6. These graphs plot the 2-norm of the error in the Krylov subspace approximation of (i) $(I - hA_*)^{-1}b$ using GMRES, (ii) $(I - hA_*)^{-1}b$ using FOM, (iii) $e^{hA_*}b$, (iv) $\phi_{20}(hA_*)b$, and (v) $\phi_{21}(hA_*)b$ vs. the number of Arnoldi iterations it took to obtain this error for the CUSP example. In these runs $N = 32, 96$, $h = 5 \times 10^{-5}$, $t_* = 10^{-4}$.

relative performance of explicit, implicit, and exponential methods we will compare the simplest second-order methods of each type. We will also show that even the third-order EPI schemes can be less costly than the second-order explicit and implicit integrators. In these tests we integrate each of the three examples above using the following methods.

AB2 is the second-order explicit Adams–Bashforth method

$$U_{n+1} = U_n + \frac{h}{2}(3F_n - F_{n-1}). \tag{77}$$

Since the method is two-step we use the second-order Runge–Kutta (Midpoint) method to obtain the starting values for the scheme. Obviously, AB2 requires the least amount of computations per time step out of all the methods we test but the time step of this method is limited by the stability requirement.

AM2 is the second-order implicit Adams–Moulton method (or Trapezoidal rule)

$$U_{n+1} = U_n + \frac{h}{2}(F_n + F_{n+1}). \tag{78}$$

It is well known that this method is *A*-stable and therefore the time step size is only restricted by accuracy. However, since system (78) is nonlinear the good stability properties of this method come at the expense of an increase in the number of computations per time step. Thus at each time step we will employ a Newton method to solve for U_{n+1} and a Krylov subspace projection method, GMRES, to invert the matrix $(I - (h/2)\frac{DF}{DU}(U))$ within each Newton iteration. Therefore at each time step the number of Newton iterations required to achieve a given tolerance is equal to the number of times the Arnoldi algorithm has to be executed.

EPI2 is the second-order multistep type EPI scheme

$$U_{n+1} = U_n + g_0(A_n h)hF_n, \tag{79}$$

where $g_0(z) = (e^z - 1)/z$. Out of all possible second -order exponential propagation schemes this method requires the least amount of computation per times step since only one evaluation of $F(U)$ and one Krylov projection have to be performed.

EPI3: this third-order multistep type EPI method was introduced in Section 3.2.1:

$$U_{n+1} = U_n + g_0(A_n h)hF_n + \frac{2}{3}g_1(A_n h)hR_{n-1}, \tag{80}$$

where

$$\begin{aligned} g_0(z) &= \frac{e^z - 1}{z}, \\ g_1(z) &= \frac{e^z - (1+z)}{z^2}. \end{aligned} \tag{81}$$

This two step method requires at most one evaluation of $F(U)$ (the value of F_{n-1} can be saved from the previous time step) and two Krylov subspace projections to compute $g_0(A_h h)hF_n$ and $g_1(A_n h)hR_{n-1}$ at each time step. As noted earlier, in general if the residuals allow to do so, it is possible to optimize the scheme by reusing the Krylov basis from the evaluation of $g_0(A_h h)hF_n$ to calculate $g_1(A_n h)hR_{n-1}$. However, in this implementation we consider the worst case scenario and perform two Krylov projections at each time step.

EPIRK3: this third-order Runge–Kutta type EPI method was introduced in Section 3.2.2 and is given by

$$\begin{aligned} r_1 &= U_n + 2\phi_{20}(A_n \frac{h}{2})\frac{h}{2}F_n, \\ U_{n+1} &= U_n + \phi_{20}(A_n h)hF_n + \frac{1}{3}\phi_{21}(A_n h)hR(r_1), \end{aligned} \tag{82}$$

where

$$\begin{aligned} \phi_{20}(z) &= \frac{e^z - 1}{z}, \\ \phi_{21}(z) &= 2\frac{e^z - (1+z)}{z^2}. \end{aligned} \tag{83}$$

This method has about the same operations count per time step as the scheme EPI3 above plus one additional function evaluation $F(r_1)$. Note that only one Krylov projection is needed to compute both $\phi_{20}(A_n h/2)(h/2)F_n$ and $\phi_{20}(A_n h)hF_n$ with the latter used for calculating the residual.

First, we consider the test results for the Brusselator equation. We use the five methods above to integrate the Brusselator system over the time interval $[0, 1]$ with the grid size $N = 100$ and $\alpha = 1/50$. Table 7 lists the time steps sizes h used as well as the total time (in seconds) that it took each of the methods to integrate

Table 7
This table lists time (in seconds, rounded) it took for each of the methods to integrate the Brusselator system over the time interval $[0, 1]$ with a given time step size h

Time step, h	AB2	AM2	EPI2	EPI3	EPIRK3
10^{-4}	213	650	256	337	337
2×10^{-4}	54	318	75	115	116
4×10^{-4}	14	170	24	45	45
8×10^{-4}	3	106	9	19	19
1.6×10^{-3}	Unstable	86	6	11	11
3.2×10^{-3}	Unstable	79	3	6	6
6.4×10^{-3}	Unstable	55	2	3.45	3.98
1.28×10^{-2}	Unstable	39	2	2.49	2.58
2.56×10^{-2}	Unstable	30.7	1.5	2.2	2.3
5.12×10^{-2}	Unstable	28.1	1.4	2.3	2.5
1.024×10^{-1}	Unstable	29.4	1.4	2.4	2.6

In these runs the tolerance is set to $tol = 10^{-7}$.

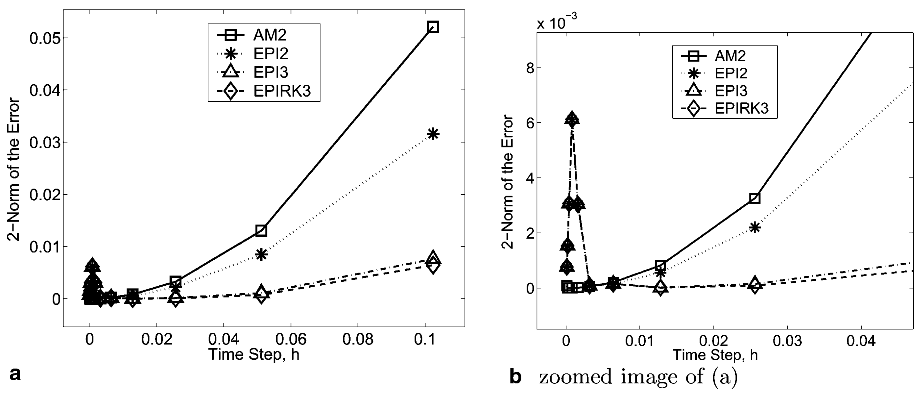


Fig. 7. The graphs show the 2-norm of the error vs. the time step size h for integration of the Brusselator system with methods AM2, EPI2, EPI3, and EPIRK3. In these runs the tolerance is set to $tol = 10^{-7}$.

the system over the given time interval. In these calculations the tolerance used for both the Newton iteration as well as for all Krylov projections is set to 10^{-7}. Fig. 7 shows the 2-norm of the corresponding errors which are calculated using the solution computed with Matlab's `ode15s` routine with 10^{-13} relative and absolute tolerances as reference. Note that while for large values of the time step the graphs in Fig. 7 exhibit the expected order of the methods, for small values of the time step the graphs for schemes EPI3 and EPIRK3 show a curious peak in error (Fig. 7). This is explained by looking at the results of the same calculations with the lowered tolerance of $tol = 10^{-9}$ in Figs. 8(a) and (b). As we can see from these graphs the peak is shifted to the left. The reason for this phenomenon is that in the range of time step values where the peak is located the global error is dominated by the error of the Krylov subspace approximations. When the tolerance is lowered and the exponential functions are computed with more precision the time discretization error prevails in this range as can by see by comparing Figs. 7(a) and (b) and 8(a) and (b). In either of these cases, however, the exponential propagation schemes are shown to be more efficient than the implicit Trapezoidal Rule method AM2 as can be seen from the results in Table 7 and more efficient than the explicit method AB2 which becomes unstable for time steps larger than $h \approx 0.0012$.

Let us also examine the effect the magnitude of the tolerances has on the comparative performance of the implicit and EPI methods above. For this test we are interested in the large time step regimes and therefore we set $h = 0.0512$. Since very loose tolerances are used sometimes in the Krylov projections step of a Newton–Krylov method to improve efficiency, we will vary the value of tol and record the integration time and the error for the schemes AM2, EPI2, EPI3, and EPIRK3. First, we tried to lower the tolerance on the Krylov projection step in AM2 but leave the tolerance for the outer Newton iteration at $tol = 10^{-6}$. However, we

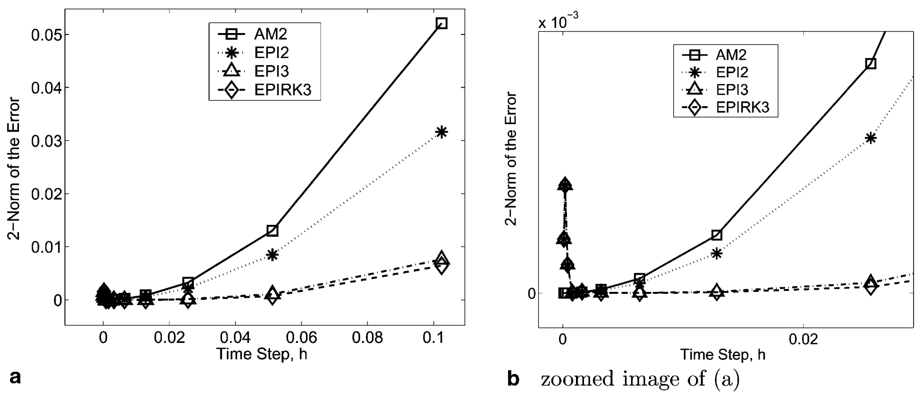


Fig. 8. The graphs show the 2-norm of the error vs. the time step size h for integration of the Brusselator system with methods AM2, EPI2, EPI3 and EPIRK3. In these runs the tolerance is set to $tol = 10^{-9}$.

found that in this case the Newton iteration in AM2 stops converging and the solution cannot be obtained. Thus we ran tests where the tolerances for both the Krylov projections and the Newton iteration are set to a range of values $tol = 10^{-3}, 10^{-4}, 10^{-5}$. The AM2 algorithm consists of a straightforward implementation of the Newton–Krylov iteration with fixed values of tolerances for both the Newton and Krylov iterations. Such implementation provides the most direct way to compare the implicit and EPI methods. However, the Newton method has existed for a long time and various improvements to the basic algorithm has been developed. In particular, for large systems an inexact Newton iteration has been introduced and shown to be more cost efficient for many problems [37]. In the future it might be possible to develop similar improvements for the EPI methods. If so comparing such improved EPI schemes with the implicit scheme with inexact Newton method would be more fair. However, for the sake of completeness we have also implemented and included in our comparison study a version of the implicit Adams–Moulton method with embedded inexact Newton method as it is described in [38]. The inexact Newton iteration replaces the original algorithm for solving $G(x) = 0$ with iteration $x_{k+1} = x_k + s_k$ where $G'(x_k)s_k = -\ G(x_k) + r_k,\ k = 1, 2, \ldots$ and r_k satisfies condition $\|G(x_k) + G'(x_k)s_k\| \leqslant \eta_k \|G(x_k)\|$. Thus to fully specify an inexact Newton algorithm one has to choose the forcing term η_k. We follow [38] and choose η_k to be

$$\eta_k = \frac{\|G(x_k) - G(x_{k-1}) - G\prime(x_{k-1})s_{k-1}\|}{\|G(x_{k-1})\|}. \tag{84}$$

Table 8 summarizes the results of these tests. As we can see even for low tolerances the EPI schemes still outperform the implicit method both in speed and in the accuracy of the obtained solution. In fact, the EPI schemes also outperform the implicit method AM2IN although by a smaller compared to AM2. While the goal of this paper is primarily to introduce EPI schemes and demonstrate their performance, further fine tuning of these methods and comparisons with other state-of-the-art implicit algorithms will be addressed in our future work.

As we can see from Table 9 and Fig. 9 performance of the methods for the BURG example is very similar to the case of the Brusselator system. Parameters used for the Burgers equation were the grid size $N = 500$,

Table 8
This table compares performance of the methods AM2, AM2IN, EPI2, EPI3 and EPIRK3 with respect to different values of tolerance; in each of the tests the Brusselator system was integrated up to time $t = 1$ with $N = 100$ and $h = 0.0512$

Method	$tol = 10^{-3}$		$tol = 10^{-4}$		$tol = 10^{-5}$	
	Time	Error	Time	Error	Time	Error
AM2	7.81	0.014	9.72	0.013	12.8	0.013
AM2IN	1.93	0.012	2.62	0.013	3.49	0.013
EPI2	0.44	0.011	0.56	0.008	0.67	0.085
EPI3	0.56	0.010	0.67	0.00171	0.87	0.001
EPIRK3	0.59	0.011	0.71	0.00231	0.99	0.0008

The total execution times are given in seconds and rounded, and the 2-norms of the errors in the final solution are provided.

Table 9
This table lists times it took for each of the methods to integrate the BURG system over the time interval $[0, 0.5]$ with a given time step size h

Time step, h	AB2	AM2	EPI2	EPI3	EPIRK3
10^{-4}	137	2376	306	620	626
2×10^{-4}	36	1284	120	276	281
4×10^{-4}	9	774	73	15	15
8×10^{-4}	2.5	492	50	89.5	90
1.6×10^{-3}	Unstable	297	33	56.8	57.5
3.2×10^{-3}	Unstable	244	22	37.3	39.8
6.4×10^{-3}	Unstable	178.5	15	27.4	28.1

In these runs the tolerance is set to $tol = 10^{-7}$.

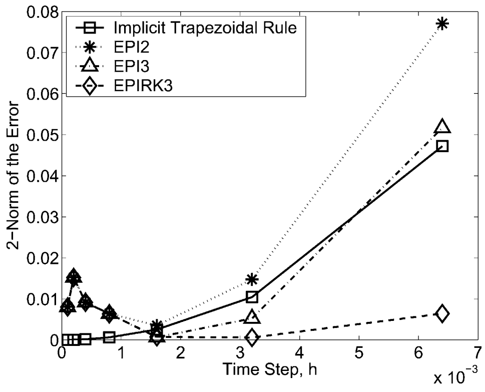


Fig. 9. The graphs show the 2-norm of the error vs. the time step size h for integration of the BURG system with methods AM2, EPI2, EPI3, and EPIRK3. In these runs the tolerance is set to $tol = 10^{-7}$.

Table 10
The table shows results of the comparison of the methods for integrating the CUSP example over the time interval $[0, 10^{-4}]$ with time steps $h = 10^{-5}$ and $h = 5 \times 10^{-6}$

$h = 1E-5$	Execution time	Error	$h = 5E-6$	Execution time	Error
AB2	0.01	1.73E − 2	AB2	0.01	4.32E − 3
AM2	Diverges		AM2	6.78	8.61E − 4
EPI2	0.04	1.65E − 1	EPI2	0.09	8.22E − 2
EPI3	0.1	6.6E − 1	EPI3	0.21	1.68E − 1
EPIRK3	0.11	2.1E − 1	EPIRK3	0.3	9.46E − 2

$\nu = 3 \times 10^{-4}$, the time interval $[0, 0.5]$ and the tolerance $tol = 10^{-7}$. A slight difference with the Brusselator example is that in this case it appears that an explicit method is more efficient than any other technique. However, as N increases and so does the stiffness the stability bound becomes severely restrictive, while as we can see from the table the EPI schemes remain efficient at high values of h. The results of the table also clearly indicate the superior efficiency of the EPI schemes compared to the implicit solver AM2.

In the CUSP example the methods exhibit a different behavior which is instructive to consider. We perform the calculations with grid size $N = 32$, parameter values $\varepsilon = 10^{-4}$, $\sigma = 1/144$, $tol = 10^{-7}$ over the time interval $[0, 10^{-4}]$. Here we find that the accuracy considerations outweigh those of stability (Table 10). The stability bound on the time step for the explicit scheme AB2 in this case is of the order 10^{-5}. For time steps smaller than this bound the comparison of the total time of integration of the CUSP system by implicit and EPI schemes shows the same results as for the previous two examples. The errors in this case are also comparable even though the exponential propagation schemes give a somewhat larger error. However, when the time step exceeds the stability bound we find that not only AB2 goes unstable, but the Newton iteration within the implicit method AM2 no longer converges to within a given tolerance and the EPI schemes also yield unreasonably large errors. Thus for this problem an explicit method will always be the fastest way to compute the solution as confirmed by our numerical experiments and integration with either implicit or exponential propagation schemes is not appropriate.

5. Conclusions

In this paper, we have introduced a new class of exponential integrators which we called exponential propagation iterative (EPI) methods. We have discussed the methodology for the construction of these schemes and studied their performance on several test problems. We have demonstrated that the faster convergence of Arnoldi iterations needed by EPI schemes provide computational savings compared to standard implicit Newton–Krylov integrators with no preconditioning. It is not clear at this point whether EPI schemes can be used in conjunction with a preconditioner and we plan to investigate this question in the future. We have also showed the superior stability properties of these methods compared to explicit schemes. Thus the EPI schemes can provide an efficient alternative to standard integrators if no good preconditioner is available for a large-scale stiff problem, its time integration is challenging because of a stability bound on the time step and the accuracy requirement allows a time step far exceeding that bound, While the numerical examples in this paper provide some guidance as to what type of problems can benefit from the use of EPI schemes more research is needed to determine the classes of systems for which the EPI methods are advantageous. The study of the performance and application of the higher-order EPI methods proposed in this paper will be presented elsewhere.

Acknowledgments

The author thanks Prof. Alexandre Chorin and Prof. G.I. Barenblatt for introducing her to an inspiring and supportive scientific environment, Prof. Ole Hald and Dr. David Bernstein for helpful discussions, and Mr. John Barber for advice on Mathematica software. This work was supported in part by the NSF VIGRE Grant DMS-9819728, NSF/DOE partnership in plasma physics Grant DMS-0317511, and University of California President's Postdoctoral Fellowship Program.

References

[1] D.A. Knoll, D.E. Keyes, Jacobian-free Newton–Krylov methods: a survey of approaches and applications, J. Comput. Phys. 193 (2004) 357–397.
[2] E. Hairer, S.P. Nørsett, G. Wanner, Solving Ordinary Differential Equations, Springer, Berlin, 1987.
[3] S.M. Cox, Exponential time differencing (ETD) references. Available from: <www.maths.adelaide.edu.au/people/scox/etd.html>.
[4] J. Certaine, The solution of ordinary differential equations with large time constants, in: A. Ralston, H.S. Wilf (Eds.), Mathematical Methods for Digital Computers, 1960, pp. 128–132.
[5] D.A. Pope, An exponential method of numerical integration of ordinary differential equations, Commun. ACM 6 (8) (1963) 491–493.
[6] J.D. Lawson, Generalized Runge–Kutta processes for stable systems with large Lipschitz constants, SIAM J. Numer. Anal. 4 (1967) 372–380.
[7] S.P. Nørsett, An A-stable modification of the Adams–Bashforth methods, Lecture Notes Math. 109 (1969) 214–219.
[8] C.B. Moler, C.F. Van Loan, Nineteen dubious ways to compute the exponential of a matrix, SIAM Rev. 20 (4) (1978) 801–836.
[9] C.B. Moler, C.F. Van Loan, Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later, SIAM Rev. 45 (1) (2003) 3–49.
[10] E. Gallopoulos, Y. Saad, Efficient solution of parabolic equations by Krylov approximation methods, SIAM J. Sci. Stat. Comp. 13 (1992) 1236–1264.
[11] J.D. Lawson, S.J. Thomas, R.V.M. Zahar, Weighted quadrature in Krylov methods, Utilitas Mathematica 51 (1997) 165–182.
[12] G. Beylkin, J.M. Keiser, L. Vozovoi, A new class of time discretization schemes for the solution of nonlinear PDEs, J. Comput. Phys. 147 (1998) 362–387.
[13] S.M. Cox, P.C. Matthews, Exponential time differencing for stiff systems, J. Comput. Phys. 176 (2002) 430–455.
[14] A.K. Kassam, L.N. Trefethen, Fourth-order time stepping for stiff PDEs, SIAM J. Sci. Comp. (2004).
[15] S. Krogstad, Generalized integrating factor methods for stiff PDEs, J. Comput. Phys. 203 (1) (2005) 72–88.
[16] M. Hochbruck, A. Ostermann, Exponential Runge–Kutta methods for parabolic problems, Appl. Numer. Math. 53 (2005) 323–339.
[17] Y. Saad, Iterative Methods for Sparse Linear Systems, PWS Publishing Company, 1996.
[18] C.W. Gear, Y. Saad, Iterative solution of linear equations in ODE codes, SIAM J. Sci. Stat. Comput. 4 (4) (1983) 583–601.
[19] A. Nauts, R.E. Wyatt, New approach to many-state quantum dynamics: the recursive-residue-generation method, Phys. Rev. Lett. 51 (5) (1983) 2238–2241.
[20] T.J. Park, J.C. Light, Unitary quantum time evolution by iterative Lanczos reduction, J. Chem. Phys. 85 (1986) 5870–5876.
[21] H.A. Van der Vorst, An iterative solution method for solving $f(a)x = b$ using Krylov subspace information obtained for the symmetric positive definite matrix a, J. Comput. Appl. Math. 18 (1987) 249–263.
[22] R.A. Friesner, L.S. Tuckerman, B.C. Dornblaser, T.V. Russo, A method for exponential propagation of large systems of stiff nonlinear differential equations, J. Sci. Comput. 4 (1989) 327–354.
[23] M. Tokman, Magnetohydrodynamic modeling of solar coronal arcades using exponential propagation methods, Ph.D. Thesis, Caltech, 2000.
[24] M. Hochbruck, Ch. Lubich, On Krylov subspace approximations to the matrix exponential operator, SIAM J. Numer. Anal. 34 (1997) 1911–1925.
[25] M. Hochbruck, Ch. Lubich, H. Selhofer, Exponential integrators for large systems of differential equations, SIAM J. Sci. Comput. 19 (1998) 1552–1574.
[26] M. Tokman, P.M. Bellan, Three-dimensional model of the structure and evolution of coronal mass ejections, Astrophys. J. 567 (2) (2002) 1202–1210.

[27] W.E. Arnoldi, The principle of minimized iteration in the solution of the matrix eigenvalue problem, Quart. Appl. Math. 9 (1951) 17–29.
[28] L.A. Knizhnerman, Calculation of functions of unsymmetric matrices using Arnoldi method, Comput. Math. Math. Phys. 31 (1) (1991) 1–9.
[29] Y. Saad, Analysis of some Krylov subspace approximations to the matrix exponential operator, SIAM J. Numer. Anal. 29 (1992) 209–228.
[30] R.B. Sidje, Expokit: a software package for computing matrix exponentials, ACM T. Math. Software 24 (1) (1998) 130–156.
[31] R.H. Merson, An operational method for the study of integration processes, in: Proceedings of the Symposium on Data Processing, Weapons Research Establishment, Salisbury, Australia, 1957, pp. 110-1–110-25.
[32] J.C. Butcher, Coefficients for the study of Runge–Kutta integration processes, J. Austr. Math. Soc. 3 (1963) 185–201.
[33] R. Lefever, G. Nicolis, Chemical instabilities and sustained oscillations, J. Theor. Biol. 30 (1971) 267–284.
[34] R. FitzHugh, Mathematical models of excitation and propagation in nerve, in: H.P. Schwan (Ed.), Biological Engineering, McGraw-Hill, New York, 1969, pp. 1–85.
[35] J. Nagumo, S. Arimoto, S. Yoshizawa, An active pulse transmission line simulating nerve axon, Proc. IRE 50 (1962) 2061–2070.
[36] E.C. Zeeman, Differential equations for the heartbeat and nerve impulse, in: C.H. Waddington (Ed.), Towards a Theoretical Biology, vol. 4, Edinburgh University press, 1972, pp. 8–67.
[37] R.S. Dembo, S.C. Eisenstat, T. Steihaug, Inexact Newton methods, SIAM J. Numer. Anal. 19 (2) (1982) 400–408.
[38] S.C. Eisenstat, H.F. Walker, Choosing the forcing terms in an inexact Newton method, SIAM J. Sci. Comput. 17 (1) (1996) 16–32.