Performance assessment of exponential Rosenbrock methods for large systems of ODE

E. J. Carr[1] T. J. Moroney[2] I. W. Turner[3]

(Received 1 November 2012; revised 27 March 2013)

Abstract

This article studies time integration methods for stiff systems of ordinary differential equations of large dimension. For such problems, implicit methods generally outperform explicit methods because the step size is usually less restricted by stability constraints. Recently, however, a family of explicit methods, called exponential integrators, have become popular for large stiff problems due to their favourable stability properties and the rapid convergence of non-preconditioned Krylov subspace methods for computing matrix-vector products involving exponential-like functions of the Jacobian matrix. In this article, we implement the so-called exponential Rosenbrock methods using Krylov subspaces. Numerical experiments on a challenging real-world test problem reveal that these methods are a promising preconditioner-free alternative to well-established approaches based on preconditioned

http://journal.austms.org.au/ojs/index.php/ANZIAMJ/article/view/6331

Newton–Krylov implementations of the backward differentiation formulas.
Subject class: 65L04
Keywords: exponential integrators, backward differentiation formulas, stiff ordinary differential equations, Krylov subspace methods, porous media

Contents

1 Introduction

We study time integration methods for initial value problems involving large stiff systems of nonlinear ordinary differential equations (ODEs) in autonomous form:

$$u'(t) = g(u)\,, \quad u(0) = u_0\,, \quad 0 < t \leqslant T\,, \tag{1}$$

where $u \in \mathbb{R}^N$ with N large, $g : \mathbb{R}^N \to \mathbb{R}^N$ is a nonlinear vector-valued function of u, u' is the time derivative of u, and u_0 is the known initial solution vector. Such problems frequently arise from spatial discretisations of

nonlinear time-dependent partial differential equations, where the entries of the vector u are the discrete solution values and the stiffness of the system is characterised by the presence of one or more eigenvalues of the Jacobian matrix $J = \partial g/\partial u$ with large negative real parts.

A time integration method applied to (1) produces an approximation to the continuous solution $u(t)$ taking the form of individual approximations $u_n \approx u(t_n)$ at a discrete set of times $\{t_n, n \in \mathbb{N}\}$ with step sizes $\tau_n = t_{n+1} - t_n$. Such methods are often classified as being explicit or implicit. In terms of the computation required per time step, explicit methods are generally more attractive because they do not require the solution of a linear or nonlinear system of equations at each time step. However, for stiff problems implicit methods generally outperform explicit methods. Roughly speaking, the reason for this is due to the eigenvalue distribution described above. Since implicit methods have a much larger region of absolute stability[1], the step size is less restricted by stability constraints.

Perhaps the most widely used implicit methods for stiff problems are the backward differentiation formulas (BDFs). The idea is to evaluate (1) at $t = t_{n+1}$, and then use a qth order backward difference approximation[2] to approximate $u'(t_{n+1})$:

$$u'(t_{n+1}) \approx \sum_{i=0}^{q} \alpha_{n,i} u_{n+1-i},$$

where the coefficients $\alpha_{n,i}$ depend on the order q, the recent step size history and the current step size τ_n. The qth order scheme[3] takes the form of a

[1] For a fixed step size τ, the region of absolute stability specifies any constraints on τ needed to ensure the approximate solution to the linear problem $y' = \lambda y$, where $\lambda \in \mathbb{C}$ with $\Re(\lambda) < 0$, matches the asymptotic behaviour of the exact solution, that is, $y(t) \to 0$ as $t \to \infty$. A scheme is said to be A-stable if it is stable for all $\Re(\lambda) < 0$.

[2] Such an approximation is derived by differentiating the interpolating polynomial (of degree at most q) that passes through the $q+1$ points (t_{n+1-i}, u_{n+1-i}), $i = 0, \ldots, q$.

[3] For $q = 1$ the well-known backward Euler method, a first order one-step method, is

system of nonlinear equations,

$$f(u_{n+1}) \equiv u_{n+1} - \gamma_n g(u_{n+1}) + a_n = 0\,, \tag{2}$$

where $a_n = \sum_{i=1}^{q} (\alpha_{n,i}/\alpha_{n,0}) u_{n+1-i}$ and $\gamma_n = \tau_n/\alpha_{n,0}$, which must be solved at every time step for u_{n+1}.

The computational cost of solving (2) is dominated by the solution of a system of linear equations at each Newton iteration:

$$[I - \gamma_n J(u_{n+1,k})]\, x_k = -f(u_{n+1,k})\,, \tag{3}$$

where $x_k = u_{n+1,k+1} - u_{n+1,k}$ and $u_{n+1,k}$ denotes the kth Newton iterate.

For large problems, Krylov subspace methods are favoured because they require only matrix-vector products with the Jacobian matrix. This is attractive because such products are approximated cheaply using finite difference approximations [7, §3.2.1, e.g.]. Such methods for solving systems of nonlinear equations are called *Jacobian-free Newton–Krylov methods* [8]. The preferred Krylov method is usually the generalised minimal residual method (GMRES), since it minimises the 2-norm of the residual vector at each iteration. The catch is that convergence is usually unsatisfactory without some form of preconditioning. Using *right* preconditioning, the linear system (3) is solved in two stages by introducing a preconditioner matrix M:

- solve $[I - \gamma_n J(u_{n+1,k})]\, M^{-1} \tilde{x}_k = -f(u_{n+1,k})$ for $\tilde{x}_k$ using GMRES;
- solve $M x_k = \tilde{x}_k$ for x_k.

The choice of M is highly problem-dependent: a good preconditioner for one problem is not necessarily a good preconditioner for another [9]. A standard approach, which works well in many cases, is to choose M equal to the coefficient matrix of (3) but 'freeze' the matrix over multiple time steps [8]. Note that for this particular choice of M, the approach is not *truly*

obtained, whereas for $q > 1$, the backward differentiation formulas are all higher order multistep methods as u_{n+1} depends on not only u_n but also $u_{n-1}, \ldots, u_{n+1-q}$.

Jacobian-free since the Jacobian matrix is formed and stored whenever M is updated.

Over the past decade, a family of time integration methods known as exponential integrators have emerged as a serious contender to implicit methods for large, stiff problems. In the simplest of terms, an exponential integrator for (1) is any time integration method that involves the exponential of the Jacobian matrix. The prototype method (in essence, the backward Euler equivalent for exponential integrators) is known as the exponential Rosenbrock–Euler method [6]. It is summarised as follows. Given the approximate solution at $t = t_n$, denoted by u_n, linearise the nonlinear function $g(u)$ to obtain

$$u'(t) = g_n + J_n(u - u_n),$$

where $g_n = g(u_n)$ and $J_n = J(u_n)$. Solving this system of linear ODEs exactly using the integrating factor:

$$\int_{t_n}^{t_{n+1}} \frac{d}{dt}\left(e^{-tJ_n}u\right)\,dt = \int_{t_n}^{t_{n+1}} e^{-tJ_n}\left(g_n - J_n u_n\right)\,dt,$$

produces the scheme

$$u_{n+1} = u_n + \tau_n \varphi_1(\tau_n J_n) g_n, \tag{4}$$

where

$$\varphi_1(z) = \frac{e^z - 1}{z}. \tag{5}$$

Note that (4) is explicit: u_{n+1} is defined explicitly in term of u_n, so there is no nonlinear or linear system to solve. Instead, computation of the matrix function vector product $\varphi_1(\tau_n J_n) g_n$ is required per time step.

The exponential Rosenbrock–Euler method is a second order method. Higher order methods for (1) have been developed and include the exponential Rosenbrock methods [6] and the exponential propagation iterative methods [9]. Such methods require multiple matrix function vector products per time

step involving the so-called *phi functions*, which are defined by (5) and the recurrence relation:

$$\varphi_k(z) = \frac{\varphi_{k-1}(z) - \varphi_{k-1}(0)}{z}, \quad k = 2, 3, \ldots .$$

The attraction of using an exponential integrator is due to two main reasons. Firstly, they have excellent stability properties—they are usually exact for linear problems (that is, problems in the form of (1) with $g(u) = Au + b$, where A and b have fixed entries), which means that they are A stable. In contrast, only the first and second order backward differentiation formulas possess this property. The second reason is that Krylov subspace methods for computing matrix function vector products involving $\varphi_k(\tau_n J_n)$ typically converge rapidly, meaning that the integrators can be implemented efficiently without preconditioning.

The aim of this article is to compare the performance of higher order exponential integrators to the backward differentiation formulas. Similar comparisons for first and second order methods [3] and fixed step size implementations [9] have featured previously. We focus on the exponential Rosenbrock methods of Hochbruck, Ostermann and Schweitzer [6]. We briefly discuss their derivation, present methods of order three and four, and explain the implementation of the schemes using Krylov subspace methods (see Section 2). Numerical experiments comparing the performance of each method are carried out in Section 3.

2 Exponential Rosenbrock methods

2.1 General framework

We write the initial value problem (1) in the form

$$u'(t) = g_n + J_n(u - u_n) + d(u),$$

where $d(u) = g(u) - g_n - J_n(u - u_n)$. Using the integrating factor e^{-tJ_n}, the following expression is obtained for the exact solution at $t = t_{n+1}$,

$$u(t_{n+1}) = u_n + \tau_n \varphi_1[\tau_n J(u_n)] g(u_n) + \int_{t_n}^{t_{n+1}} e^{(t_{n+1}-t)J(u_n)} d[u(t)]\, dt\,.$$

This formula is the starting point for developing higher order one-step exponential integrators for (1). The basic idea is to approximate the integral using an appropriate quadrature rule. Since the integrand depends on the solution $u(t)$, in general, a time integration method is obtained consisting of s internal stages $u_{n,i}$ that approximate the solution at $t = t_n + c_i\tau_n$:

$$\begin{aligned} u_{n,i} &= u_n + c_i\tau_n \varphi_1(c_i\tau_n J_n) g_n + \tau_n \sum_{j=1}^{i-1} a_{ij}(c_i\tau_n J_n) d_{n,j}\,, \quad i = 1, \ldots, s\,, \\ u_{n+1} &= u_n + \tau_n \varphi_1(\tau_n J_n) g_n + \tau_n \sum_{i=1}^{s} b_i(\tau_n J_n) d_{n,i}\,, \end{aligned} \tag{6}$$

where $d_{n,i} = d(u_{n,i})$ and the scalars c_i are called the nodes of the method. The quadrature weights $b_i(z)$ and coefficients $a_{ij}(z)$ are linear combinations of the functions $\varphi_k(z)$ and $\varphi_k(c_i z)$. Conditions on these functions determine the order of the method [6].

Note that the exponential Rosenbrock–Euler method (4) is an exponential Rosenbrock method with $s = 0$.

2.2 Embedded methods

A critical component of time integration methods for large, stiff problems is the use of adaptive step size control, whereby the local error is estimated at every time step and the step size adjusted to satisfy absolute and relative error tolerances. The idea is to construct a pair of embedded methods to obtain two approximate solutions: u_{n+1} and an error estimator $\hat{u}_{n+1}$. The

difference between these solutions is taken as an estimate of the local error at each time step. To reduce the computation introduced, the error estimator relies on the same internal stages $u_{n,i}$, $i = 1, \ldots, s$ and takes the general form

$$\hat{u}_{n+1} = u_n + \tau_n \varphi_1(\tau_n J_n) g_n + \tau_n \sum_{i=1}^{s} \hat{b}_i(\tau_n J_n) d_{n,i} .$$

The following pairs of embedded methods were given by Hochbruck et al. [6].

The `exprb32` scheme is a one stage pair of embedded exponential Rosenbrock methods of orders two and three:

$$\begin{aligned} u_{n,1} &= u_n + \tau_n \varphi_1(\tau_n J_n) g_n , \\ u_{n+1} &= u_n + \tau_n \varphi_1(\tau_n J_n) g_n + \tau_n b_1(\tau_n J_n) d_{n,1} , \\ \hat{u}_{n+1} &= u_n + \tau_n \varphi_1(\tau_n J_n) g_n , \end{aligned}$$

where $b_1(z) = 2\varphi_3(z)$. The `exprb43` scheme is a two stage pair of embedded exponential Rosenbrock methods of orders three and four,:

$$\begin{aligned} u_{n,1} &= u_n + \tfrac{1}{2}\tau_n \varphi_1(\tfrac{1}{2}\tau_n J_n) g_n , \\ u_{n,2} &= u_n + \tau_n \varphi_1(\tau_n J_n) g_n + \tau_n a_{21}(\tau_n J_n) d_{n,1} , \\ u_{n+1} &= u_n + \tau_n \varphi_1(\tau_n J_n) g_n + \tau_n b_1(\tau_n J_n) d_{n,1} + \tau_n b_2(\tau_n J_n) d_{n,2} , \\ \hat{u}_{n+1} &= u_n + \tau_n \varphi_1(\tau_n J_n) g_n + \tau_n \hat{b}_1(\tau_n J_n) d_{n,1} + \tau_n \hat{b}_2(\tau_n J_n) d_{n,2} , \end{aligned}$$

where $a_{21} = \varphi_1(z)$, $b_1(z) = 16\varphi_3(z) - 48\varphi_4(z)$, $b_2(z) = -2\varphi_3(z) + 12\varphi_4(z)$, $\hat{b}_1(z) = 16\varphi_3(z)$ and $\hat{b}_2(z) = -2\varphi_3(z)$. In both schemes the higher order method is used to continue the integration.

2.3 Krylov subspace methods

In general, an s stage exponential Rosenbrock method contains (at most) $(s+1)(s+2)/2$ unique products involving a matrix function and a vector. However,

since all products involve the matrix J_n and either of the vectors g_n or $d_{n,k}$, $k = 1, \ldots, s$, only $s + 1$ Krylov subspaces are required. For example, the scheme `exprb43` requires seven unique products but only three Krylov subspaces per time step.

In what follows, we describe the computation of $f(\tau_n J_n) b_k$, where

$$b_k = \begin{cases} g_n, & k = 0, \\ d_{n,k}, & k = 1, \ldots, s, \end{cases}$$

and $f(z)$ is any one of the functions $\varphi_1(z)$, $b_i(z)$, $a_{ij}(z)$ or $\hat{b}_i(z)$. Approximations are extracted from the small dimensional Krylov subspaces $\mathcal{K}_m(J_n, b_k)$, $k = 0, \ldots, s$ using $s+1$ applications of Arnoldi's method, yielding the relations

$$J_n V_m^{(k)} = V_m^{(k)} H_m^{(k)} + \beta_m^{(k)} v_{m+1}^{(k)} e_m^T, \quad k = 0, \ldots, s,$$

where $v_1 = b_k / \|b_k\|_2$, the columns of $V_m^{(k)} = [v_1^{(k)}, v_2^{(k)}, \ldots, v_m^{(k)}] \in \mathbb{R}^{N \times m}$ form an orthornormal basis for $\mathcal{K}_m(J_n, b_k)$, $H_m^{(k)} = V_m^{(k)T} J_n V_m^{(k)} \in \mathbb{R}^{m \times m}$ is an upper Hessenberg matrix and e_m is the mth canonical basis vector in $\mathbb{R}^m$ [1]. Since $d_{n,1}$ depends on g_n and $d_{n,k}$ depends on $d_{n,k-1}$ for $k = 2, \ldots, s$ all applications of Arnoldi's method must be performed serially.

The Krylov approximation is then [2, 5]

$$f(\tau_n J_n) b_k \approx \|b_k\|_2 V_m^{(k)} f(H_m^{(k)}) e_1, \quad k = 0, \ldots, s, \tag{7}$$

which reduces the evaluation of f to a small $m \times m$ matrix that is computed using standard approaches such as Padé approximation [5]. Beginning at $m = 1$ the subspace is expanded by one additional vector at each iteration producing an updated Arnoldi relation and approximation (7). Termination of the iterative procedure is based on the *generalized residual vector* [2, 5].

3 Results

Our interest in initial value problems in the form of (1) lies in the field of simulating transport in porous media and therefore we use the well-known two dimensional test problem, Richards' equation. We solve the two-dimensional Richards' equation,

$$\frac{\partial \theta(h)}{\partial t} = \frac{\partial}{\partial x}\left[K(h)\frac{\partial h}{\partial x}\right] + \frac{\partial}{\partial z}\left[K(h)\left(\frac{\partial h}{\partial z} + 1\right)\right],$$

on $\Omega = \{(x, z) \mid 0 < x < 5,\ 0 < z < 3\}$ and $0 < t < 12.5$, subject to the initial condition and boundary conditions:

$$\begin{aligned}
h &= -500\,, \quad && t = 0\,,\\
K(h)\frac{\partial h}{\partial x} &= 0\,, \quad && x = 0\,,\ x = 5\,,\\
K(h)\frac{\partial h}{\partial z} + K(h) &= 0\,, \quad && z = 0\,,\\
K(h)\frac{\partial h}{\partial z} + K(h) &= 0\,, \quad && z = 3\,,\ 0 < x < 2\,,\ 3 < x < 5\,,\\
K(h)\left(\frac{\partial h}{\partial x} + 1\right) &= 0.05\,, \quad && z = 3\,,\ 2 < x < 3\,.
\end{aligned}$$

The moisture content θ, hydraulic conductivity K and effective saturation S_e are all defined in terms of the pressure head $h < 0$ as given by the van Genuchten relations [1]:

$$\begin{aligned}
\theta(h) &= \theta_{res} + (\theta_{sat} - \theta_{res})S_e(h)\,,\\
K(h) &= K_{sat}\sqrt{S_e(h)}\left\{1 - \left[1 - S_e(h)^{1/m}\right]^m\right\}^2\,,\\
S_e(h) &= \left[1 + (-\alpha h)^n\right]^{-m}\,,
\end{aligned} \tag{8}$$

where $m = 1 - 1/n$.

The specific test problem concerns the flow of water into a very dry rectangular region divided into nine alternating blocks of two heterogeneous soils: sand and clay (see Figure 1 and Table 3). All boundaries are zero flux apart from a 1 m strip in the centre of the top block of sand, where a constant influx of water at 0.05 m per day is applied.

A spatial discretisation was performed using the finite volume method on a rectangular grid consisting of 129 nodes in both the x and z directions. Horizontal symmetry of the problem is not exploited. This produces an initial value problem in the form of (1) with $N = 129^2 = 16641$, where the vector u contains the unknown values of the pressure head h at each of the nodes. Carr, Moroney and Turner [1] give full details.

The schemes `exprb32` and `exprb43` were implemented in Matlab with Krylov subspace methods. In addition, we implemented the exponential Rosenbrock–Euler method using the step size control strategy featured in our previous article [1]. We refer to this method as `exprem22` henceforth. Performance comparisons were made with the Matlab interface to the CVODE module of the suite of nonlinear/differential-algebraic solvers (SundialsTB v2.5.0). For stiff problems, CVODE employs a variable order implementation of the backward differentiation formulas, meaning that the order is adjusted throughout the time integration with the goal of maximising the step size. Implementations with upper bounds of two, three and four on the order are denoted by `bdf2`, `bdf3` and `bdf4`. CVODE provides several choices for the solution of the linear systems at each Newton iteration. In this work, we used the in-built GMRES solver with the 'freezing' preconditioning strategy outlined in Section 1. The specific solver options are given in Table 1. Full details are provided in the Sundials documentation [4].

All methods tested were implemented in a Jacobian-free framework, that is, Jacobian matrix-vector products were approximated using

$$J(u)v \approx [g(u + \varepsilon v) - g(u)]/\varepsilon\,, \tag{9}$$

where ε is a suitably-defined perturbation [1, 4]. For CVODE, when updating

Table 1: Solver options used in CVODE.

Option	Value
`LinearSolver`	GMRES
`PrecType`	`Right`
`PrecModule`	`BandPre`
`LowerBwidth`	29
`UpperBwidth`	129

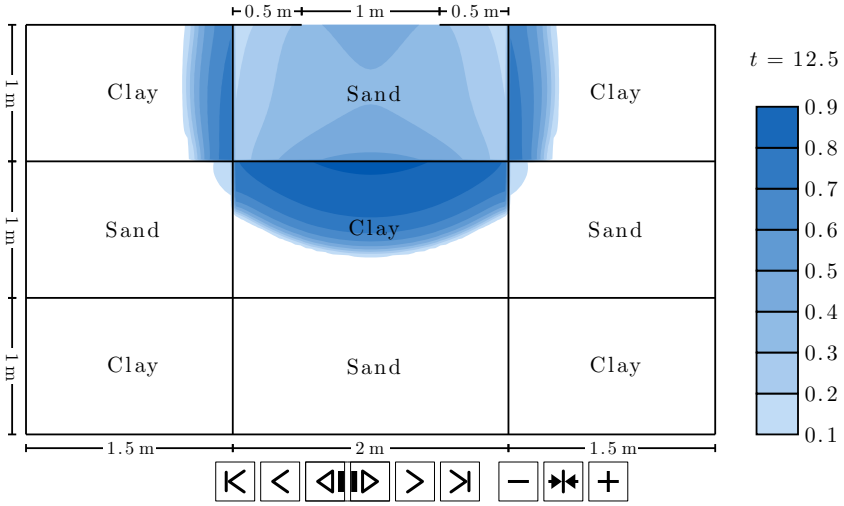

Figure 1: Numerical solution of the test problem. Contour plot of the saturation field (θ/θ_{sat}). Animation over varying time viewable in at least Adobe Reader version 9.

the preconditioner, the full Jacobian was computed numerically taking advantage of its banded structure by using (9) with appropriate choices for the vector ν [7, §2.3].

A benchmark solution was obtained using CVODE (with maximum order five) with very small absolute and relative error tolerances. For each of the six methods tested, simulations were performed in Matlab 2012b with absolute

Table 2: Hydraulic properties of sand and clay used in the van Genucten relations in equation (8).

Material	θ_{res}	θ_{sat}	K_{sat} [ms^{-1}]	α [m^{-1}]	n
Clay	0.106	0.469	1.52×10^{-6}	1.04	1.40
Sand	0.029	0.366	6.26×10^{-5}	2.80	2.23

and relative error tolerances (`tol`) equal to $10^{-4}, 10^{-5}, \ldots, 10^{-8}$, respectively. Table 3 lists the relative error of the solution at $t = 12.5$ days, the total number of function calls to $g(u)$ (for `bdf2`, `bdf3` and `bdf4` this includes the function evaluations required to form the Jacobian matrix when updating the preconditioner), the total number of accepted time steps and the CPU time for each simulation. Although we understand that CPU times strongly depend on the available computer architecture, we feel that the comparisons are of interest.

Comparing the second order methods, we conclude that `exprem22` comprehensively outperforms `bdf2` because it requires fewer function calls, fewer time steps and less CPU time to compute a solution of equivalent accuracy. For the third order methods, `exprem32` and `bdf3`, we observe that both schemes are comparable in terms of computation required to obtain a solution of desired accuracy. Finally, the fourth order backward differentiation formula, `bdf4`, performs better than the fourth order exponential Rosenbrock method, `exprb43`.

An important observation is the increase in computation per step when increasing the order of the exponential Rosenbrock methods. For a comparable number of time steps, the number of function calls increases as evidenced by the simulations: `exprem22` (2437 steps, 18335 calls), `exprb32` (2783 steps, 25217 calls) and `exprb43` (2692 steps, 36200 calls). Such a trend is not observable for the the backward differentiation formulas: `bdf2` (2860 steps, 20183 calls), `bdf3` (2926 steps, 20603 calls), `bdf4` (2386 steps, 16191 calls). The main reason for this is that the higher order exponential Rosenbrock

Table 3: Simulation statistics for the test problem. For each of the six methods, approximate solutions are obtained using absolute and relative error tolerances (`tol`). Statistics listed are the relative error of the approximate solution at $t = 12.5$ days, the total number of function calls to $g(u)$ (for `bdf2`, `bdf3` and `bdf4` this includes the function evaluations required to form the Jacobian matrix when updating the preconditioner), the total number of accepted time steps and the CPU time.

tol	exprem22	bdf2	exprb32	bdf3	exprb43	bdf4
Relative error						
10^{-4}	8.7E−3	1.4E−3	1.6E−2	3.3E−4	5.2E−2	2.1E−5
10^{-5}	4.1E−4	3.2E−4	3.4E−4	6.0E−5	5.2E−3	1.5E−6
10^{-6}	7.4E−6	5.8E−5	3.9E−6	8.8E−6	1.6E−4	4.2E−7
10^{-7}	3.4E−6	1.4E−5	3.5E−6	1.9E−6	6.7E−7	1.0E−7
10^{-8}	8.7E−7	3.2E−6	8.0E−7	3.3E−7	2.8E−7	2.9E−8
Total number of function calls						
10^{-4}	11688	20183	15408	12267	20945	11134
10^{-5}	18335	40286	25217	20603	25721	16191
10^{-6}	31062	85000	45404	36026	36200	24292
10^{-7}	56778	177813	89813	62152	55758	38410
10^{-8}	112346	382927	186361	107069	92538	59556
Total number of accepted time steps						
10^{-4}	1234	2860	1360	1707	1014	1561
10^{-5}	2437	6039	2783	2926	1598	2386
10^{-6}	5156	13365	5906	5484	2692	3660
10^{-7}	10660	28022	12652	9717	4664	5960
10^{-8}	22233	60273	27221	16826	8214	9329
CPU time [mins]						
10^{-4}	2.6	8.2	3.4	5.4	5.5	4.6
10^{-5}	4.0	15.3	5.3	8.4	6.2	6.5
10^{-6}	6.6	31.3	9.5	14.2	8.2	9.7
10^{-7}	11.9	64.6	18.5	24.2	12.7	14.2
10^{-8}	23.6	142.0	38.4	41.2	21.0	22.1

methods require computation of additional matrix function vector products per time step, whereas, for the backward differentiation formulas, the solution of one nonlinear system is required per step, regardless of the order used.

Another key finding is the relatively low accuracy of `exprb32` and `exprb43` compared with `bdf3` and `bdf4` for a given value of the tolerance (`tol`). This indicates that the exponential Rosenbrock methods are under estimating the local error at each time step. The reason for this is almost certainly that the embedded methods, used to estimate the local error, are of lower order.

Finally, interestingly, the computational cost required to compute a solution with relative error less than 10^{-5}. Surprisingly, `exprem22` (6.6 mins, 31062 calls) places second in this category, just behind `bdf4` (6.5 mins, 16191 calls) and well in front of `exprb32` (9.5 mins, 45404 calls), `exprb43` (12.7 mins, 55758 calls), `bdf3` (14.2 mins, 36026 calls) and `bdf2` (142 mins, 382927 calls).

4 Conclusions

Exponential Rosenbrock methods are competitive with backward differentiation formulas when applied to stiff problems arising from spatial discretisations of time-dependent partial differential equations. For very large problems, these methods are particularly attractive because they do not require preconditioning. Our numerical experiments indicate that, although the third and fourth order exponential Rosenbrock methods are competitive with the third and fourth order BDFs, they suffer from an increase in the number of function evaluations due to the multiple matrix function vector products (and therefore multiple Krylov subspaces) required per time step. A key point is that all of these subspaces involve the same Jacobian matrix. Future work will therefore focus on reducing the function evaluation count by recycling some of the spectral information from one subspace to the next.

Acknowledgements This work was supported by the Australian Research Council Discovery Project grant DP120103770. We thank the two anonymous reviewers for their detailed analysis and suggestions which improved the quality of the final version of this article.

References

[1] E. J. Carr, T. J. Moroney and I. W. Turner. Efficient simulation of unsaturated flow using exponential time integration. *Appl. Math. Comput.*, 217(14): 6587–6596, 2011. doi:10.1016/j.amc.2011.01.041 C110, C111, C112

[2] E. J. Carr, I. W. Turner and M. Ilic. Krylov subspace approximations for the exponential Euler method: error estimates and the harmonic Ritz approximant. *ANZIAM Journal*, 52: C612–C627, 2011. http://journal.austms.org.au/ojs/index.php/ANZIAMJ/article/view/3938 C110

[3] E. J. Carr, I. W. Turner and P. Perré. A variable step size Jacobian-free exponential integrator for simulating transport in heterogeneous porous media: application to wood drying. *J. Comput. Phys.*, 233: 66-82, 2013. doi:10.1016/j.jcp.2012.07.024 C107

[4] Sundials user documentation. https://computation.llnl.gov/casc/sundials/documentation/documentation.html C112

[5] M. Hochbruck, C. Lubich and H. Selhofer. Exponential integrators for large systems of differential equations. *SIAM J. Sci. Comput.*, 19(5): 1552–1574, 1998. doi:10.1137/S1064827595295337 C110

[6] M. Hochbruck, A. Ostermann and J. Schweitzer. Exponential Rosenbrock-type methods. *SIAM J. Numer. Anal.*, 47(1): 786–803, 2009. doi:10.1137/080717717 C106, C107, C108, C109

[7] C. T. Kelley. *Solving nonlinear equations with Newton's method*. SIAM, USA, 2003. C105, C113

[8] D. A. Knoll and D. E. Keyes. Jacobian-free Newton–Krylov methods: a survey of approaches and applications. *J. Comput. Phys.*, 193(2): 357–397, 2004. doi:10.1016/j.jcp.2003.08.010 C105

[9] M. Tokman. Efficient integration of large stiff systems of ODEs with exponential propagation iteration (EPI) methods. *J. Comput. Phys.*, 213: 748–776, 2006. doi:10.1016/j.jcp.2005.08.032 C105, C106, C107

Author addresses

1. **E. J. Carr**, Mathematical Sciences School, Queensland University of Technology, Brisbane, Queensland 4000, Australia.
mailto:elliot.carr@qut.edu.au

2. **T. J. Moroney**, Mathematical Sciences School, Queensland University of Technology, Brisbane, Queensland 4000, Australia.
mailto:t.moroney@qut.edu.au

3. **I. W. Turner**, Mathematical Sciences School, Queensland University of Technology, Brisbane, Queensland 4000, Australia.
mailto:i.turner@qut.edu.au