Exponential Almost Runge-Kutta Methods for Semilinear Problems

John Carroll Eóin O'Callaghan

19 January 2013

Abstract

We present a new class of one-step, multi-value Exponential Integrator (EI) methods referred to as Exponential Almost Runge-Kutta (EARK) methods which involve the derivatives of a nonlinear function of the solution. In order to approximate such derivatives to a sufficient accuracy, the EARK methods will be implemented within the broader framework of Exponential Almost General Linear Methods (EAGLMs) to accommodate past values of this nonlinear function and becoming multistep in nature as a consequence. Established EI methods, such as Exponential Time Differencing (ETD) methods, Exponential Runge-Kutta (ERK) methods and Exponential General Linear Methods (EGLMs) become special cases of EAGLMs. We present order conditions which facilitate the construction of two- and three-stage EARK methods and, when cast in an EAGLM format, we perform a stability analysis to enable a comparison with existing EI methods. We conclude with some numerical experiments which confirm the convergence order and also demonstrate the computational efficiency of these new methods.

1 Introduction

Over the last decade, we have observed significant interest in the construction of Exponential Integrator (EI) methods for semilinear initial-value problems (see [5, 11, 16, 23]) of the form

$$y' = Ly + N(t,y), \qquad y(0) = y_0, \tag{1}$$

where $y \in R \to R^d$, $L \in R^d \times R^d$ and $N(:) \in R^d$ represent the linear and nonlinear terms of the equation respectively. Spatial semi-discretisation of parabolic partial differential equations leads to systems of this type where d represents the number of spatial grid points and matrix L will typically be large and sparse.

Exponential integrators are characterised both by their ability to exactly evaluate the contribution of the linear part of the equation (i.e. if the nonlinear part is zero, then the numerical method simplifies to an evaluation of the exponential function of L) and also by stability properties that are comparable to those of typical implicit methods. We present a class of explicit exponential integrator methods which when re-formulated into Exponential Almost General Linear Methods (EAGLMs) can be viewed as an extension of the explicit Exponential General Linear Methods (EGLMs) [27] which combine the Exponential Runge-Kutta (ERK) methods [11, 15, 17] and Exponential Time Differencing (ETD) methods [5], providing high-order methods with good stability properties for problems described by (1).

We begin in §2 with a representation of the exact solution of (1) using exponential-like functions referred to as φ-functions. Exponential Time Differencing (ETD) methods and Exponential Runge-Kutta (ERK) methods are reviewed briefly in §3, where we also discuss how these methods can be combined to produce the family of Exponential General Linear Methods (EGLMs). For a more comprehensive review of exponential integrators and their history, we refer the reader to Minchev & Wright [23].

In §4, we present a new class of methods, which we call Exponential Almost Runge-Kutta (EARK) methods, which were motivated by the work on Almost Runge-Kutta (ARK) methods by Butcher [8] and, in §5, we show how they can be combined with ETD methods to produce a class of explicit Exponential Almost General

Linear Methods (EAGLMs). In §6, we study the stability properties of the new EARK family of methods and establish their correspondence to EGLMs.

In §7, we consider the implementation costs of EARK methods when compared with the more established families of methods as this is an important consideration in the design of efficient algorithms. We conclude in §8 by presenting the results of some numerical experiments applied to semi-discretised parabolic partial differential equations which support the convergence properties of these methods and also demonstrate their computational performance (in terns of both accuracy and efficiency) when compared with competitive exponential integrator methods over fixed integration step sizes.

2 Exponential Integrators & φ-Functions

The exact solution of (1) is given by the variation of constants formula.

$$y(t_n+h) = e^{hL}y_n + \int_0^h e^{(h-\tau)L}N\left(t_n+\tau, y(t_n+\tau)\right)d\tau \tag{2}$$

where $h = t_{n+1} - t_n$ is the integration step-size. If we substitute the Taylor expansion for $N\left(t_n+\tau, y(t_n+\tau)\right)$ about the point t_n

$$\sum_{m=0}^{\infty} \frac{\tau^m N^{(m)}\left(y(t_n)\right)}{m!} \tag{3}$$

into (2), and define

$$\varphi_j(hL) = \frac{1}{h^j}\int_0^h e^{(h-\tau)L}\frac{\tau^{j-1}}{(j-1)!}d\tau, \tag{4}$$

we can then represent the exact solution for (1) by the expansion

$$y(t_{n+1}) = e^{hL}y\left(t_n\right) + \sum_{i=1}^{\infty} h^i \varphi_i(hL) N^{(i-1)}\left(t_n, y\left(t_n\right)\right)$$

where $t_{n+1} = t_n + h$ (see [23, Lemma 5.1]).

A key element, and the main computational difficulty, in the implementation of exponential integrators is the evaluation of the matrix exponential and exponential-like functions (4) referred to as φ-functions. An additional complication lies in the observation that, if the matrix exponential or φ-function is computed explicitly, the resultant matrix will typically not retain any of the sparse properties of the original matrix [24], making the direct approach unsuitable for very large matrices due to excessive storage requirements. As a consequence, a number of alternative approaches have been developed which work with the application of the φ-functions on a vector, $\varphi_j(A)\times v$, without generating $\varphi_j(A)$ explicitly (see [1], for example). Algorithms which adopt this approach include Krylov subspace methods [12, 13, 25], real Leja points methods [3, 2, 9], contour integration methods based on rational approximations [18, 19, 28] and, more recently, contour approximation techniques which employ the Carathéodory-Fejér method [29]. We will conduct our numerical experiments in §8 using two of these approaches, namely the `PHIPM` implementation of a Krylov subspace-based approach by Niesen & Wright [25], and an implementation utilising the real Leja points method by Caliari & Ostermann [9], which we refer to as `ReLPM`.

3 Multi-step and Multi-stage Methods

3.1 Exponential Time Differencing Methods

Some of the earliest EI schemes to appear were members of the multistep Exponential Time Differencing (ETD) family of methods [23]. Following the approach of §2, ETDs methods can be constructed by approximating $N\left(t_n+\tau, y(t_n+\tau)\right)$ in (2) by the Newton interpolation polynomial (3) up to the required order and

then solving the resulting integral exactly. The simplest case is to take one term of the series and approximate $N(t_n+\tau, y(t_n+\tau))$ by the constant value $N_n = N(t_n, y_n)$, leading to the one-step exponential time differencing Euler method:

$$y_{n+1} = e^{hL} y_n + h\varphi_1 N_n, \tag{5}$$

which is referred to as ETD_1, or the ETD Euler method, as it reduces to the classical Euler method when $L = 0$. Taking the first two terms of the series, i.e. approximating $N(t_n+\tau, y(t_n+\tau))$ by a linear polynomial $N_n + \frac{\tau}{h}(N_n - N_{n-1})$ results in the two-step method:

$$y_{n+1} = e^{hL} y_n + h[\varphi_1 + \varphi_2] N_n - h\varphi_2 N_{n-1} \tag{6}$$

which will be referred to as ETD_2.

3.2 Exponential Runge-Kutta Methods

Cox & Matthews [11] developed a 1-step Runge-Kutta-type extension to ETD methods which they referred to as ETD Runge-Kutta Methods. These schemes are now considered as belonging to the family of Exponential Runge-Kutta (ERK) methods. The application of an ERK method to the general problem (1) results in the following equations:

$$\begin{aligned} y_{n+1} &= e^{hL} y_n + h \sum_{i=1}^{s} b_i(hL) K_i. \\ K_1 &= N_n = N(t_n, y_n) \\ K_i &= N\left(t_n + c_i h, e^{c_i hL} y_n + h \sum_{j=1}^{i-1} a_{ij}(c_i hL) K_j \right), \quad i = 2, \ldots, s \end{aligned}$$

where s is the number of stages.

The order of a classical Runge-Kutta method can be established by comparing a Taylor expansion of the exact solution $y(t_n+h)$ with its numerical approximation y_{n+1}. This approach is more complicated for ERK methods arising from coupling conditions between the nonlinear and linear parts of the problem, despite the fact that the linear part has been solved exactly [21].

Definition 1. [20, Definition 1] An exponential method has stiff order p if the local error has order $p+1$ with respect to h_{n+1} when the method is applied to an abstract semi-linear ODE (1) in which $N(t, y(t))$ is a sufficiently smooth function of t.

When deriving order conditions for constructing schemes of higher order, it is sometimes not possible to satisfy all the conditions unless we *weaken* some of them. We will therefore make a distinction between schemes of strong and weak stiff order.

Definition 2. An exponential method is said to be of *strong* order p if it satisfies the p stiff order conditions for the general semi-linear ODE (1).

Definition 3. An exponential method is said to be of *weak* order p if it has strong order $p-1$ and satisfies the p stiff order conditions only in the weakened case where $L = 0$.

Hochbruck & Ostermann [15] developed an approach based on trees for deriving stiff order conditions using the elementary differentials of $F(u) = Lu + N(u)$ which arise from a Taylor expansion of the exact solution. Such an approach facilitated the construction of conditions for arbitrary orders in a manner similar to that for classical RK methods. An example of a strongly 2^{nd} order, two-stage family of schemes is ERK_{22} [15, Scheme 5.3] parametrised by the free variable c_2:

$$\begin{array}{c|cc|c} 0 & & & I \\ c_2 & c_2\varphi_{1,2} & & e^{c_2 hL} \\ \hline & \varphi_1 - \frac{1}{c_2}\varphi_2 & \frac{1}{c_2}\varphi_2 & \end{array} \tag{7}$$

In the construction of higher order methods, the number of conditions to be satisfied grows rapidly. This makes it increasingly difficult to achieve higher orders, and this difficultly makes the notion of weak order, as defined in Definition 3, very important. For example, there are no strongly 3rd order 3-stage ERKs, only weakly 3rd order methods.

For comparison purposes in numerical experiments to follow in §8, we will use Krogstad's 4-stage scheme [17]:

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2}\varphi_{1,2} & & & \\ \frac{1}{2} & \frac{1}{2}\varphi_{1,3}-\varphi_{2,3} & \varphi_{2,3} & & \\ 1 & \varphi_{1,4}-2\varphi_{2,4} & 0 & 2\varphi_{2,4} & \\ \hline & \varphi_1-3\varphi_2+4\varphi_3 & 2\varphi_2-4\varphi_3 & 2\varphi_2-4\varphi_3 & -\varphi_2+4\varphi_3 \end{array} \tag{8}$$

We refer to this scheme as ERK_{34}. It is a strongly 3rd order, weakly 4th order scheme and was consistently the best performing 4-stage scheme among those considered by Hochbruck & Ostermann [15].

3.3 Exponential General Linear Methods

General Linear Methods (GLMs) are a class of multi-step, multi-stage explicit methods introduced by Butcher [6]. They are a generalisation of linear multi-step methods with the multi-stage nature of Runge-Kutta methods. The advantage which they offer is that they allow one to easily construct higher-order schemes while retaining an inherent Runge-Kutta stability [7].

Ostermann, Thalhammer & Wright [27] introduced a class of explicit Exponential General Linear Methods (EGLMs) based on the Adams-Bashforth schemes. EGLMs are an extension of GLMs into the exponential framework and contain, as special cases, the ETD and ERK method families. For given start values $y_0, y_1, \ldots, y_{q-1}$, the internal stages K_i are defined by

$$K_i = N\left(e^{hL}y_n + h\sum_{j=1}^{i-1} a_{ij}(hL)K_j + h\sum_{j=1}^{q} u_{ij}(hL)N_{n-j}\right)$$

for $1 < i \leq s$, and the numerical approximation y_{n+1} at time t_{n+1} is given by

$$y_{n+1} = e^{hL}y_n + h\sum_{i=1}^{s} b_i(hL)K_i + h\sum_{j=1}^{q} v_i(hL)N_{n-j}.$$

Following the ERK notation, we identify EGLMs through the use of the subscripts p, s and q, where p is the order of the method, s represents the number of stages while q is the number of steps.

Ostermann, Thalhammer & Wright [27] concluded that EGLMs, like their classical counterparts, combine the ease of construction of high-order methods with the superior stability properties of ERK methods. They derived order conditions for 2-stage schemes [27, (2.3) & (2.7)] and considered the case where $c_2 = 1$. This choice for c_2 reduces the number of distinct φ's thereby improving the computational efficiency of the schemes. These schemes are referred to as EGLM_{322} [27, Table 4.1] and EGLM_{423} [27, Table 4.2]

They are particular cases of the more general families parametrised by c_2 [26] namely the strongly 3rd order family EGLM_{322c_2} family of methods:

$$\begin{array}{c|cc|cc} c_2 & a_{2,1} & & e^{c_2hL} & u_{2,1} \\ \hline & b_1 & b_2 & e^{hL} & v_1 \end{array} \tag{9}$$

$$a_{2,1} = c_2\varphi_{1,2} + c_2^2\varphi_{2,2}$$

$$b_1 = \varphi_1 + \frac{c_2-1}{c_2}\varphi_2 + \frac{-2}{c_2}\varphi_3 \qquad b_2 = \frac{1}{c_2^2+c_2}\varphi_2 + \frac{2}{c_2^2+c_2}\varphi_3$$

$$u_{2,1} = -c_2^2\varphi_{2,2} \qquad v_1 = \frac{-c_2}{c_2+1}\varphi_2 - \frac{2}{c_2+1}\varphi_3$$

and the strongly 4[th] order EGLM$_{423}c_2$ family of methods:

$$\begin{array}{c|cc|ccc} c_2 & a_{21} & & e^{c_2 hL} & u_{21} & u_{22} \\ \hline & b_1 & b_2 & e^{hL} & v_1 & v_2 \end{array} \tag{10}$$

$$a_{2,1} = c_2\varphi_{1,2} + \frac{3c_2^2}{2}\varphi_{2,2} + c_2^3\varphi_{3,2}$$

$$u_{2,1} = -2c_2^2\varphi_{2,2} - 2c_2^3\varphi_{3,2} \qquad u_{2,2} = \frac{c_2^2}{2}\varphi_{2,2} + c_2^3\varphi_{3,2}$$

$$b_1 = \varphi_1 + \frac{\frac{3c_2-2}{2}\varphi_2 + c_2 - 3\varphi_3 - 3\varphi_4}{c_2} \qquad b_2 = \frac{2\varphi_2 + 6\varphi_3 + 6\varphi_4}{c_2^3 + 3c_2^2 + 2c_2}$$

$$v_1 = \frac{-2c_2\varphi_2 - 2c_2 - 4\varphi_3 + 6\varphi_4}{c_2 + 1} \qquad v_2 = \frac{\frac{c_2}{2}\varphi_2 + c_2 - 1\varphi_3 - 3\varphi_4}{c_2 + 2}$$

4 Exponential Almost Runge-Kutta Methods

Almost Runge-Kutta (ARK) methods were introduced by Butcher in 1997 [8]. They are a special case of GLMs in that they retain the multi-stage nature of RK methods but allow for the passing of more than one value from step to step. In this way, they are multi-value rather than multi-step schemes. For ARK methods, three values form the inputs and outputs at each step, namely the approximation to the solution, together with approximations to its first and second derivatives.

However, in the design of an Exponential Almost Runge-Kutta (EARK) method, the input and output values passed from step-to-step are the function evaluations of the approximate solution, N_{n+1}, together with the derivatives $N_{n+1}^{(i)}$ of increasing order. Using a similar notation to EGLMs, we will use the subscripts p, s, q and r to denote a p-order, $q = 1$-step s-stage, r-value method. The general form of an EARK$_{psqr}$ tableau is:

$$\begin{array}{c|cccc|ccc} c_2 & a_{21} & & & & w_{21} & \cdots & w_{2r} \\ \vdots & \vdots & \ddots & & & \vdots & & \vdots \\ c_{s-1} & a_{s-1,1} & \cdots & a_{s-1,s-2)} & & w_{s-1,1} & \cdots & w_{s-1,r} \\ 1 & b_1 & \cdots & b_{s-1} & 0 & z_1 & \cdots & z_r \\ \hline & b_1 & \cdots & b_{s-1} & 0 & z_1 & \cdots & z_r \\ & \beta_{11} & \cdots & \beta_{1,s-1} & \beta_{1s} & \delta_{11} & \cdots & \delta_{1r} \\ & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ & \beta_{r1} & \cdots & \beta_{r,s-1} & \beta_{rs} & \delta_{r1} & \cdots & \delta_{rr} \end{array} \tag{11}$$

The w_{ij} and z_i entries represent the $\varphi-$functions operating on the incoming derivatives approximations, $N_n^{(i)}$, while the β and δ entries are scalars used in the approximation of the outgoing derivative approximations.

As an illustration, we will consider a 3-stage EARK method. The repetition in the final row of the tableau's upper half and the first row of its lower half means that an s-stage EARK method has more in common with an $(s-1)$-stage EGLM. The simplest form of an EARK method is a 3-stage, 1-value method whose tableau has the form:

$$\begin{array}{c|ccc|c} c_2 & a_{21} & & & w_{21} \\ 1 & b_1 & b_2 & & z_1 \\ \hline & b_1 & b_2 & 0 & z_1 \\ & \beta_{21} & \beta_{22} & \beta_{23} & \delta_{21} \end{array}$$

representing the equations

$$\begin{aligned}
K_1 &= N_n = N(t_n, y_n) \\
Y_2 &= e^{c_2 hL} y_n + h\left(a_{21} K_1 + w_{21} h N'_n\right) \\
K_2 &= N\left(t_n + c_2 h, Y_2\right) \\
Y_3 &= e^{hL} y_n + h\left(b_1 K_1 + b_2 K_2 + z_1 h N'_n\right) \\
y_{n+1} &= Y_3 \\
N_{n+1} &= N\left(t_n + h, Y_3\right) \\
h N'_{n+1} &= \beta_{21} K_1 + \beta_{22} K_2 + \beta_{23} N_{n+1} + \delta_{21} h N'_n
\end{aligned} \tag{12}$$

We can see from (12), that the value of Y_3 can be reused for the output values y_{n+1} and N_{n+1}, as such, the final extra row on the upper tableau does not have any detrimental effect on performance.

To produce the vector of outgoing approximations, $v = (N'_{n+1}, \ldots, N^{(r)}_{n+1})$, we construct a matrix $M = (\beta\ \delta)$, where β and δ, from the lower section of (11), are $r \times s$ and $r \times r$ matrices respectively, such that,

$$M \begin{pmatrix} N_n \\ N_{n+c_2} \\ \vdots \\ N_{n+c_s} \\ N_{n+1} \\ h N'_n \\ \vdots \\ h^P N^{(r)}_n \end{pmatrix} = \begin{pmatrix} h N'_{n+1} \\ \vdots \\ h^P N^{(r)}_{n+1} \end{pmatrix}$$

This matrix is constructed by solving a set of linear equations generated from Taylor expansions of the elements of the incoming vector of approximations, together with the N_{n+c_i} and N_{n+1} values produced by the internal stages. Specifically, m_j, the j^{th} row of the matrix M, is the solution to the linear system

$$X m_j^T = e_{j+1}$$

where e_i is the standard basis vector and the $(r+s) \times (r+s)$ matrix $X = [W, Z]$ where,

$$W = \begin{pmatrix} 1 & c_2 & \cdots & c_i & \cdots & c_{s-1} & 1 \\ -1 & -c_2 & \cdots & -c_i & \cdots & c_{s-1} & 0 \\ \frac{1}{2} & \frac{1}{2}c_2 & \cdots & \frac{1}{2}c_i & \cdots & \frac{1}{2}c_{s-1} & 0 \\ \vdots & \vdots & & \vdots & & \vdots & \vdots \\ -1^{j-1}\frac{1}{(j-1)!} & & \cdots & -1^{j-1}\frac{1}{(j-1)!}c_i & \cdots & & 0 \\ \vdots & & & \vdots & & & \vdots \end{pmatrix}$$

$$Z = \begin{pmatrix} 0 & 0 & & 0 \\ 1 & 0 & & \\ -1 & 1 & \ddots & \vdots \\ \frac{1}{2} & -1 & \ddots & \\ \vdots & \frac{1}{2} & \ddots & 0 \\ & \vdots & \ddots & 1 \\ & & & \frac{-1^{j-r-1}}{(j-r-1)!} \\ & & & \vdots \end{pmatrix}$$

We summarise the convergence properties of EARK methods using the following results.

Theorem 1. *EARK methods of the form (11) will be exact in the case $N_n = N$, a constant, if the following conditions are met*

$$\sum_{j=1}^{i-1} a_{ij} = c_i \varphi_{1,i} \tag{13a}$$

$$\sum_i b_i = \varphi_1 \tag{13b}$$

for $i = 1, \ldots, s$

Proof. See [26, Theorem 13]. □

Theorem 2. *EARK methods of the form (11) where $s = 2$ or 3, can achieve order p if they satisfy (13) and*

$$\sum_{k=2}^{j-1} a_{jk} \tfrac{c_k^i}{i!} + w_{ki} = c_k^{i+1} \varphi_{i+1,k}$$

for $i = 1, \ldots, p-3$, $j = 1, \ldots, s$,

$$\sum_{j=2}^{s} b_j \left(\sum_{k=2}^{j-1} a_{jk} \tfrac{c_k^i}{i!} + w_{ji} - c_j^{i+1} \varphi_{i+1,j} \right) = 0$$

for $i = p-2$, and

$$\sum_{j=2}^{s} \tfrac{c_j^i}{i!} b_j + z_i = \varphi_{i+1}$$

for $i = 1, \ldots, p-1$

Proof. See [26, Theorem 13]. □

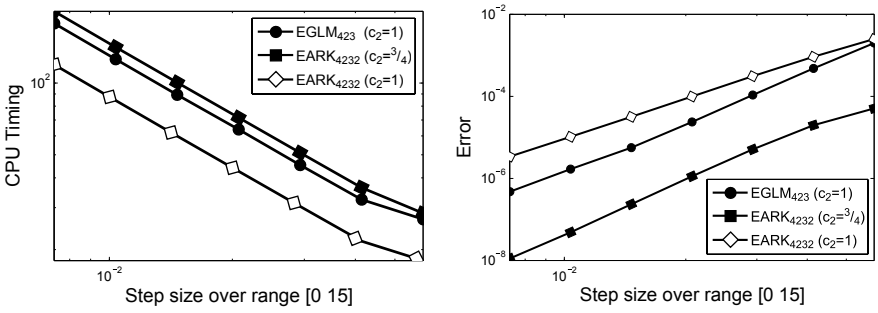
If we pass the two values hN' and h^2N'' from step to step we can construct the 4th order family of schemes knows as EARK_{4312c_2} , which is parametrised by the free variable c_2:

$$\begin{array}{c|ccc|cc}
c_2 & c_2\varphi_{1,2} & & & c_2^2\varphi_{2,2} & c_2^3\varphi_{3,2} \\
1 & \varphi_1 - \frac{6}{c_2^3}\varphi_4 & \frac{6}{c_2^3}\varphi_4 & 0 & \varphi_2 - \frac{6}{c_2^2}\varphi_4 & \varphi_3 - \frac{3}{c_2}\varphi_4 \\
\hline
 & \varphi_1 - \frac{6}{c_2^3}\varphi_4 & \frac{6}{c_2^3}\varphi_4 & 0 & \varphi_2 - \frac{6}{c_2^2}\varphi_4 & \varphi_3 - \frac{3}{c_2}\varphi_4 \\
 & -\frac{2{c_2}^2-3c_2}{{c_2}^2-2c_2+1} & -\frac{1}{{c_2}^3-2{c_2}^2+c_2} & \frac{2c_2+1}{c_2} & -\frac{c_2}{c_2-1} & 0 \\
 & \frac{2{c_2}^2-6}{{c_2}^2-2c_2+1} & -\frac{4}{{c_2}^3-2{c_2}^2+c_2} & \frac{2c_2+4}{c_2} & -\frac{2c_2+2}{c_2-1} & 0
\end{array} \tag{14}$$

Numerical experiments have shown that a value of $c_2 = 0.75$ in EARK_{4312c_2} is the optimal choice in terms of accuracy against stepsize performance. The choice of $c_2 = 1$ would allow for a more efficient implementation. However, this would result in the stage approximation at $t = t_{n+c_2}$ and the method approximation at $t = t_{n+1}$ being evaluated at the same point in time. In (14), this results in division-by-zero within the 2nd and 3rd lower tableau rows. To construct an EARK scheme with $c_2 = 1$, a new lower tableau section must be derived, namely one which does not involve K_2, i.e. a scheme where $\beta_{12} = \beta_{22} = 0$:

$$\begin{array}{ccc|cc}
\varphi_1 - \frac{6}{c_2^3}\varphi_4 & \frac{6}{c_2^3}\varphi_4 & 0 & \varphi_2 - \frac{6}{c_2^2}\varphi_4 & \varphi_3 - \frac{3}{c_2}\varphi_4 \\
-2 & 0 & 2 & -1 & 0 \\
-2 & 0 & 2 & -2 & 0
\end{array} \tag{15}$$

Note: (15) shows only the lower section of the tableau — the upper section is identical to that of (14).



(a) CPU timing against Stepsize (b) Global error against Stepsize

Figure 1: Fixed step integration of Brusselator System

To study the accuracy performance of both (14) and (15), we conduct a fixed stepsize numerical experiment for the Brusselator System [14], described in greater detail in §8.1. For comparison purposes, we also include EGLM$_{423}c_2$ (10) with $c_2 = 1$, as it's the most computationally efficient 4$^{\text{th}}$ order EGLM.

From Figure 1(a), we can see that the choice of $c_2 = 1$ allows for a significantly faster implementation of (14). However, Figure 1(b) shows that the modified lower tableau (15) has caused a serious impact on accuracy. This is because the approximations to hN' and h^2N'' are calculated by (15) to a lower order than that provided for in the lower tableau section of (14) and, as a consequence, compromise the scheme's accuracy.

If we wish to persevere with the more computationally-efficient choice of $c_2 = 1$, we will need additional data to produce sufficiently high-order approximations. An obvious source for such additional data are previous values of N. To this end, we introduce a broader framework of multi-stage, multi-step and multi-value methods, providing the flexibility required to construct EARK methods which exhibit high accuracy without compromising implementation efficiency.

5 Combining EARK methods with EGLMs

5.1 Exponential Almost General Linear Methods

To begin constructing schemes of a multi-stage, multi-step and multi-value nature, it is necessary to use an expanded tableau to represent all method families considered so far. This representation gives rise to a broader family of methods, which we refer to as Exponential Almost General Linear Methods (EAGLMs):

$$\begin{array}{c|ccc|c|ccc|ccc} 0 & & & & & & & & & & \\ c_2 & a_{21} & & & e^{c_2 hL} & u_{21} & \cdots & u_{2q} & w_{21} & \cdots & w_{2r} \\ \vdots & \vdots & \ddots & & \vdots & & \vdots & & & \vdots & \\ c_s & a_{s1} & \cdots & a_{s(s-1)} & e^{c_s hL} & u_{s1} & \cdots & u_{sq} & w_{s1} & \cdots & w_{sr} \\ \hline & b_1 & \cdots & b_s & e^{hL} & v_1 & \cdots & v_q & z_1 & \cdots & z_r \end{array} \tag{16}$$

having equations

$$\begin{aligned} K_1 &= N_n = N(t_n, y_n) \\ K_i &= N\left(t_n + c_i h,\ e^{hL} y_n + h\sum_{j=1}^{i-1} a_{ij}(hL)K_j + h\sum_{k=1}^{q} u_{ik}(hL)N_{n-k} + h\sum_{j=1}^{r} w_{ij}h^j N_n^{(j)}\right) \\ y_{n+1} &= e^{hL} y_n + h\left(\sum_{i=1}^{s} b_i(hL)K_i + \sum_{k=1}^{q} v_k(hL)N_{n-k} + \sum_{j=1}^{r} z_j(hL)h^j N_n^{(j)}\right) \end{aligned} \tag{17}$$

for $1 \le i \le s$. Such methods combine the multi-stage, multi-step nature of EGLMs with the multi-value format of EARK methods. We summarise the convergence properties of EAGLMs using the following results.

Theorem 3. *EAGLMs of the form* (16) *will be exact in the case $N_n = N$, a constant, if the following* consistency *conditions are met*

$$\sum_{j=1}^{i-1} a_{ij} + \sum_{j=1}^{q} u_{ij} = c_i \varphi_{1,i} \tag{18a}$$

for $i = 1, \ldots, s$, and

$$\sum_{i}^{s} b_i + \sum_{i=1}^{q} v_i = \varphi_1 \tag{18b}$$

Proof. See [26, Theorem 3]. □

Theorem 4. *EAGLMs of the form* (17)*, where $s = 2$ or 3, that is 2 or 3-stage methods, that meet the consistency conditions* (18)*, can achieve order p if*

$$\sum_{k=2}^{j-1} a_{jk} \frac{c_k^i}{i!} + \sum_{k=1}^{q} \frac{(-k)^i}{i!} u_{kj} + w_{ki} = c_k^{i+1} \varphi_{i+1,k} \tag{19a}$$

for $i = 1, \ldots, p-3$, $j = 1, \ldots, s$,

$$\sum_{j=2}^{s} b_j \left(\sum_{k=2}^{j-1} a_{jk} \frac{c_k^i}{i!} + \sum_{k=1}^{q} \frac{(-k)^i}{i!} u_{jk} + w_{ji} - c_j^{i+1} \varphi_{i+1,j} \right) = 0 \tag{19b}$$

where $i = p-2$, and

$$\sum_{j=2}^{s} \frac{c_j^i}{i!} b_j + \sum_{j=1}^{q} \frac{(-j)^i}{i!} v_j + z_i = \varphi_{i+1} \tag{19c}$$

for $i = 1, \ldots, p-1$

Proof. See [26, Theorem 4]. □

In what follows, we will concentrate on 2-stage schemes:

c_2	a_{21}		$e^{c_2 hL}$	u_{21}	u_{22}	w_{21}	w_{22}
	b_1	b_2	e^{hL}	v_1	v_2	z_1	z_2

$$K_1 = N_n = N(t_n, y_n)$$
$$K_2 = N\left(t_n + c_2 h,\ e^{c_2 hL} y_n + h\left[a_{21} K_1 + u_{21} N_{n-1} + u_{22} N_{n-2} + w_{21} h N_n' + w_{22} h^2 N_n''\right]\right)$$

for which we can simplify conditions (19) to

$$\sum_{k=2}^{j-1} a_{jk} \frac{c_k^i}{i!} + \sum_{k=1}^{q} \frac{(-k)^i}{i!} u_{kj} + w_{ki} = c_k^{i+1} \varphi_{k,i+1}$$

for $1 < i \leq 2$, $1 \leq j \leq i$, and

$$\frac{c_2^i}{i!} b_2 + \sum_{j=1}^{q} \frac{(-j)^i}{i!} v_j + z_i = \varphi_{i+1}$$

for $1 \leq i \leq 2$.

We consider a particular form of EAGLMs whose structure is much closer to EARK methods than to EGLMs. For such schemes, the u_{ij} or v_i tableau entries are zero and therefore the previous timesteps are not involved in φ-function operations. The lower section of the tableau has non-zero γ_{ij} entries and, as such, uses information from the previous timesteps in the approximation of the derivatives. We have adopted the

convention of referring to such schemes as EARK$_{psqr}$ methods. This emphasises the EARK structure of the upper tableau while retaining the q subscript to indicate the number of steps.

Within this restricted format, we present the strongly 3$^{\text{rd}}$ Order family of methods: EARK$_{3221}c_2$

$$\begin{array}{c|ccc|cc}
c_2 & c_2\varphi_{1,2} & & & 0 & c_2^2\varphi_{2,2} \\
1 & \varphi_1 - \frac{2\varphi_3}{c_2^2} & \frac{2\varphi_3}{c_2^2} & & 0 & \varphi_2 - \frac{2\varphi_3}{c_2} \\
\hline
 & \varphi_1 - \frac{2\varphi_3}{c_2^2} & \frac{2\varphi_3}{c_2^2} & 0 & 0 & \varphi_2 - \frac{2\varphi_3}{c_2} \\
 & -2 & 0 & \frac{3}{2} & \frac{1}{2} & 0
\end{array} \tag{20}$$

and EARK$_{4232}c_2$, the strongly 4$^{\text{th}}$ Order family of methods:

$$\begin{array}{c|ccc|cc|cc}
c_2 & c_2\varphi_{1,2} & & & 0 & 0 & c_2^2\varphi_{2,2} & c_2^3\varphi_{3,2} \\
1 & \varphi_1 - \frac{6}{c_2^3}\varphi_4 & \frac{6}{c_2^3}\varphi_4 & & 0 & 0 & \varphi_2 - \frac{6}{c_2^2}\varphi_4 & \varphi_3 - \frac{3}{c_2}\varphi_4 \\
\hline
 & \varphi_1 - \frac{6}{c_2^3}\varphi_4 & \frac{6}{c_2^3}\varphi_4 & 0 & 0 & 0 & \varphi_2 - \frac{6}{c_2^2}\varphi_4 & \varphi_3 - \frac{3}{c_2}\varphi_4 \\
 & -3 & 0 & \frac{11}{6} & \frac{3}{2} & -\frac{1}{3} & 0 & 0 \\
 & -5 & 0 & 2 & 4 & -4 & 0 & 0
\end{array} \tag{21}$$

We will also restrict our attention to the choice $c_2 = 1$ as this allows for the most efficient implementation.

5.2 Equivalence between EGLMs and EARK methods

We will illustrate the equivalence between EGLMs and EARK methods, by comparing EGLM$_{322}c_2$ (9) with EARK$_{3221}c_2$ (20). The EGLM$_{322}c_2$ (9) equations are

$$K_2 = N\left(t_n + c_2 h, e^{c_2 hL} y_n + h\left[\left(c_2\varphi_{21} + c_2^2\varphi_{22}\right) N_n - c_2^2\varphi_{22} N_{n-1}\right]\right)$$

$$y_{n+1} = e^{hL} y_n + h\left[\left(\varphi_1 + \tfrac{c_2-1}{c_2}\varphi_2 + \tfrac{-2}{c_2}\varphi_3\right) N_n + \left(\tfrac{1}{c_2^2+c_2}\varphi_2 + \tfrac{2}{c_2^2+c_2}\varphi_3\right) K_2 + \left(\tfrac{-c}{c_2+1}\varphi_2 - \tfrac{2}{c_{2+1}}\varphi_3\right) N_{n-1}\right]$$

and we can see that both N_n and N_{n-1} are multiplied by a linear combination of φ-functions. If we then consider the EARK$_{3221}c_2$ (20) scheme, when written explicitly in the form of (17), we obtain

$$K_2 = N\left(t_n + c_2 h,\ e^{c_2 hL} y_n + h\left[c_2\varphi_{21} N_n - c_2^2\varphi_{22} h N_n'\right]\right) \tag{22}$$

$$y_{n+1} = e^{hL} y_n + h\left[\left(\varphi_1 - \tfrac{2}{c_2^2}\varphi_3\right) N_n + \tfrac{2}{c_2^2}\varphi_3 K_2 + \left(\varphi_2 - \tfrac{2}{c_2}\varphi_3\right) h N_n'\right] \tag{23}$$

If we then approximate N_n' by $\frac{3}{2}N_n - 2N_{n-1} + \frac{1}{2}N_{n-2}$, we can see that (22) and (23) become

$$\begin{aligned}
Y_2 &= e^{c_2 hL} y_n + h\left[c_2\varphi_{21} N_n - c_2^2\varphi_{22}\left(\tfrac{3}{2}N_n - 2N_{n-1} + \tfrac{1}{2}N_{n-2}\right)\right] \\
&= e^{c_2 hL} y_n + h\left[\left(c_2\varphi_{21} - \tfrac{2}{3}c_2^2\varphi_{22}\right) N_n + 2\varphi_{22} c_2^2 N_{n-1} - \tfrac{1}{2}\varphi_{22} c_2^2 N_{n-2}\right] \\
K_2 &= N\left(t_n + c_2 h, Y_2\right) \\
y_{n+1} &= e^{hL} y_n + h\left[\left(\varphi_1 + \tfrac{3}{2}\varphi_2 - \tfrac{3c_2+2}{c_2^2}\varphi_3\right) N_n + \tfrac{2}{c_2^2}\varphi_3 K_2 + \left(\tfrac{4}{c_2}\varphi_3 - 2\varphi_2\right) N_{n-1} + \left(\tfrac{1}{2}\varphi_2 - \tfrac{1}{c_2}\varphi_3\right) N_{n-2}\right].
\end{aligned}$$

This is a 3$^{\text{rd}}$ Order EGLM, albeit, in this case, one which uses two previous steps within the method rather than just one as (9) does.

6 Stability Analysis

6.1 Stability of ERK methods

Cox & Matthews [11] studied the stability of several 2nd-order schemes and in particular, some linearly-implicit schemes such as an Adams-Moulton / Adams-Bashforth method, and the standard Integrating Factor methods [22]. These are compared with a number of ETD and ERK methods. Following [5], we begin with the model nonlinear, scalar, autonomous ODE,

$$u' = \alpha u + f(u) \tag{24}$$

and linearise (24) about a fixed point u_0, such that $\alpha u_0 + f(u_0) = 0$ leading to

$$u' = \alpha u + \lambda u \tag{25}$$

where u is now the perturbation to u_0 and $\lambda = f'(u_0)$. The fixed point u_0 is stable if

$$Re(\alpha + \lambda) < 0$$

for all eigenvalues λ. It must be noted that this technique can only provide a comparison of the relative stabilities of various schemes. In addition, the stability analysis for a scalar test equation is of limited significance for parabolic PDEs.

In general, both α and λ are complex so the resulting stability region is four-dimensional. Following [11], in order to plot two-dimensional stability regions, we consider two cases. First, by assuming that λ is complex and α is fixed, negative and real, we can plot the resulting stability regions in the complex plane. Second, we look at the case where α is not fixed and both α and λ are real.

The analysis is performed by applying a scheme to the test problem (25) with λu regarded as the nonlinear term. For ERK_{22} (7) with $c_2 = 1$, this results in the following expression:

$$u_{n+1} = u_n e^{h\alpha} + h\left(\frac{\lambda\left(\frac{e^{h\alpha}-1}{h\alpha} - 1\right)\left(u_n e^{h\alpha} + \frac{\lambda u_n\left(e^{h\alpha}-1\right)}{\alpha}\right)}{h\alpha}\right) + h\left(\lambda u_n\left(\frac{e^{h\alpha}-1}{h\alpha} - \frac{\frac{e^{h\alpha}-1}{h\alpha} - 1}{h\alpha}\right)\right)$$

By setting $r = u_{n+1}/u_n$, $x = h\lambda$ and $y = h\alpha$, we then obtain

$$r = \frac{\left(xy + x^2\right)e^{2y} + \left(y^3 + \left(-x^2 - 2x\right)y - 2x^2\right)e^{y} + \left(x^2 + x\right)y + x^2}{y^3} \tag{26}$$

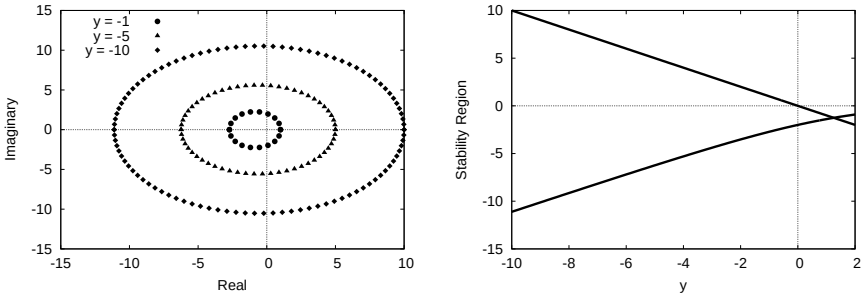
In the first case, we will fix $y < 0 \in \mathbf{R}$ and plot the boundary of the stability region which occurs when $r = 1$. To plot this in the complex plane, we set $r = e^{i\theta}$ and solve for x on the interval $\theta \in [0, 2\pi)$. Figure 2(a) shows the stability region boundaries for ERK_{22} (7) when $y = -1, -2, -5$ respectively.

For the second case, we fix $r = 1$, and plot the growth of the real extents of the stability regions against varying y. Under this restriction, the solutions to (26) are

$$x = -\frac{y^2}{e^y - y - 1} \qquad x = -y$$

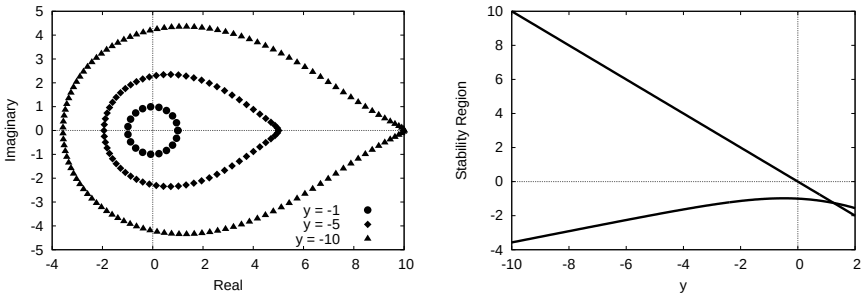
and Figure 2(b) shows a graph of those solutions. We can see that, as y grows in magnitude, the stability region's real extents also grow approximately linearly. Figure 3(a) plots boundary curves for ETD2 (6). The region of stability is significantly smaller than that of ERK_{22} (7) and Figure 3(b) highlights the slower growth of the real extents of that stability region as y grows.

Krogstad [17] investigated the stability regions of a number of 4th order schemes, notably that of ERK_{44} (8), and some multi-step generalisation of IF methods, leading to the conclusion that (8) had the largest stability



(a) Complex plane stability regions (b) Stability region real extents

Figure 2: ERK_{22} (7) stability boundaries



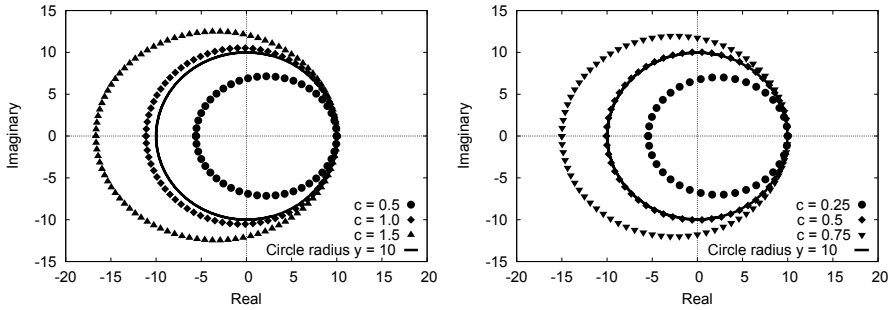
(a) Complex plane stability regions (b) Stability region real extents

Figure 3: ETD_2 (6) stability boundaries

region. Maset & Zennaro [20] studied the requirements for unconditional stability in ERK methods. Focusing on ERK_{22} (7) and a companion scheme, ERK_{12}:

$$\begin{array}{c|cc|c} 0 & & & I \\ c_2 & c_2\varphi_{1,2} & & e^{c_2 h\alpha} \\ \hline & (1-\frac{1}{2c_2})\varphi_1 & \frac{1}{2c_2}\varphi_1 & \end{array} \tag{27}$$

they demonstrated that the schemes are unconditionally stable when $c_2 \geq 1$ and $c_2 \geq \frac{1}{2}$ respectively. Looking at the stability plots of both schemes for varying c_2 at a fixed $\alpha = 10$ we can that see the stability regions in Figures 4(a) and 4(b) exceed α when the conditions on c_2, as derived by Maset & Zennaro, are satisfied.

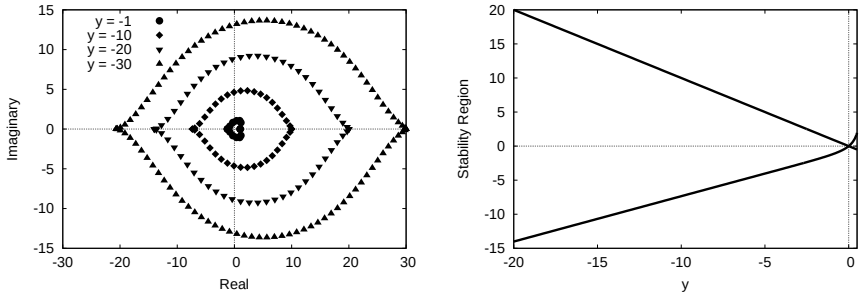


(a) ERK_{22} (7) (b) ERK_1 (27)

Figure 4: The stability boundaries for ERK methods (7) and (27)

6.2 Stability of EGLMs

EGLMs can also be analysed with this approach. Studying the boundary plots of one such method, EGLM$_{322}c_2$ (9), we can see that, in Figure 5(a), the stability region extends back into the negative complex plane. This is a property observed for ERK$_2$, but not for ETD$_2$ (see Figures 2(a) and 3(a)).



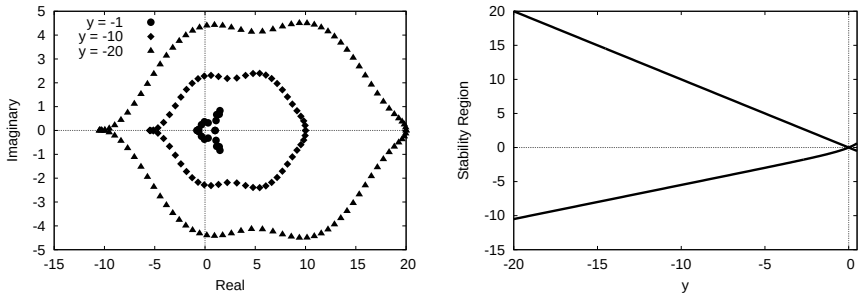
(a) Complex Plane Region Boundaries (b) Real Stability Region Extents

Figure 5: EGLM$_{322}c_2$ (9) stability boundaries

Figure 5(b) shows the desired linear growth of the EGLM$_{322}c_2$ (9) stability region's real extents.

6.3 Stability of EAGLMs

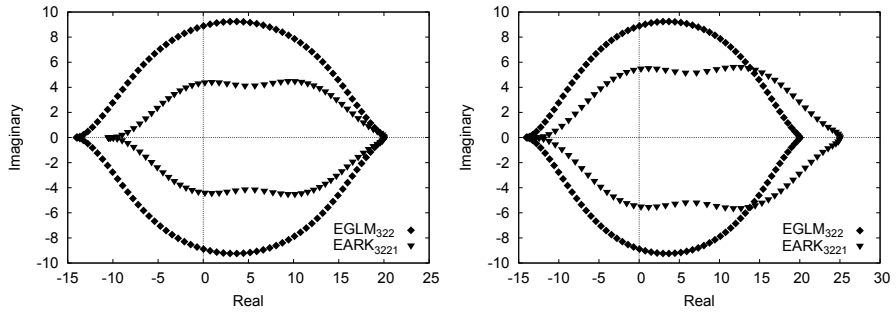
This method of analysis can not be applied directly to EARK methods due to the complexity arising from the presence of the derivatives of N. Instead we perform the analysis on the EAGLM formulation of EARK methods where we use values of N at previous timesteps to approximate the derivatives of N needed by individual methods. Arising from this, some clear inferences may be made relating to the corresponding stability properties of EARK methods. Figure 6 plots the stability regions for EARK$_{3221}c_2$ (20) with $c_2 = 1$. Note that the growth of the stability regions is again linear with α. In this particular case, the value of N'_n was approximated by $-2N_{n-2} + \frac{1}{2}N_{n-1} + \frac{3}{2}N_n = N'_n + O(h^3)$.



(a) Complex Plane Region Boundaries (b) Real Stability Region Extents

Figure 6: EARK$_{3221}c_2$ (20) stability boundaries

To gain a better understanding of the comparative stabilities between EGLMs and EAGLMs, we plot the stability regions together. Figure 7(a) shows the stability regions of the two primary 3$^{\text{rd}}$ order schemes, namely EGLM$_{322}c_2$ (9) and EARK$_{3221}c_2$ (20) for fixed $h\alpha = y = 20$. It is evident that the EAGLM scheme has a smaller stability region than the EGLM. However, we will see in the next section, that EAGLMs have a lower computational cost per step over EGLMs when measuring $\varphi\times$ vector operations (see Tables 2 and 3). Therefore, we also plot both schemes with the EARK$_{3221}c_2$ region scaled by the savings in φ-vector products. This is similar to a technique used by Butcher [7] where the stability regions were scaled relative to the number of internal stages of the classical schemes being compared. Figure 7(b) shows the new stability region and we can see now that the real axis stability extents of the EAGLM scheme exceed those of the EGLM scheme.



(a) Region Boundaries not scaled

(b) Region Boundaries scaled

Figure 7: EGLM$_{322}c_2$ (9) and EARK$_{3221}c_2$ (20) side-by-side stability boundaries

7 Computational Cost

The number of distinct φ-functions required by exponential integrator methods is a key parameter when making a relative comparison of their computational efficiencies. In Section 2, we noted some important algorithms developed for φ-function evaluation and, in particular, identified an approach common to many of those algorithms, namely the operation of the φ-function upon a vector.

Ostermann, Thalhammer & Wright [27] counted separately the numbers of φ-functions and $\varphi(A)\times v$ operations for a number of EI schemes [27, Table 5.1] to provide a relative indication of where the EGLM schemes ranked in terms of computational efficiency. We will follow this approach when comparing the relative efficiencies of the EARK methods and EAGLMs.

Niesen and Wright [25] generalised the operation of a φ-function upon a vector into the implementation of a linear combination of φ-functions upon input vectors $v_0,\dots,v_p$:

$$e^A v_0+\varphi_1(A)v_1+\varphi_2(A)v_2+\cdots+\varphi_p(A)v_p \tag{28}$$

in an single operation. To take this approach into account, we include a third count in our efficiency comparison, namely the number of linear combination function calls necessary to implement each scheme.

As an illustration, we present in detail the evaluation of these different efficiency measures when applied to the ERK scheme, ERK$_{33}$ [15, Scheme 5.8] :

$$\begin{array}{c|ccc|c} 0 & & & & I \\ c_2 & c_2\varphi_{1,2} & & & e^{c_2hL} \\ \frac{2}{3} & \frac{2}{3}\varphi_{1,3}-\frac{4}{9c_2}\varphi_{2,3} & \frac{4}{9c_2}\varphi_{2,3} & & e^{\frac{2}{3}hL} \\ \hline & \varphi_1-\frac{3}{2}\varphi_2 & 0 & \frac{3}{2}\varphi_2 & \end{array} \tag{29}$$

- **Explicit Evaluation**. This is the approach used by the `EXPINT` package [4], which employs Padé approximations. When the φ-functions are computed explicitly, they can be used for a number of matrix-vector products at negligible cost. As such, a count of the number of distinct matrix exponentials and φ-functions provides a good relative indicator of a scheme's complexity. Also, in what follows, we exploit a key optimisation applicable to the majority of EI schemes, using the identity:

$$e^{c_ihL}\times y_n+hc_i\varphi_{1,i}(hL)\times v=y_n+hc_i\varphi_{1,i}(hL)\times(Ly_n+v) \tag{30}$$

  to eliminate the need to perform the matrix-exponential by a vector operation, (`expm` $\times$ vector) and this gives a significant performance boost. Under this measure, ERK$_{33}$ (29) requires 5 φ-function evaluations.

- **$\varphi\times$ vector Operations**. We noted in Section 2 that a number of techniques have been developed which work with the operation of the φ-function upon a vector. For some schemes, it may be necessary to

compute a $\varphi\times$ vector product with the same φ-function at two different stages within a step. Therefore, the number of distinct φ-functions represents only a lower bound on the estimated number of operations necessary to compute one step. In practice, each scheme must be considered separately to determine the true and optimal number of operations needed. A pseudo code example illustrates this procedure:

1: $K_2 \leftarrow N\left(t_n + c_2 h, e^{c_2 hL} \times y_n + h c_2 \varphi_{1,2} \times N_n\right)$
2: $K_3 \leftarrow N\left(t_n + \frac{2}{3}h, e^{\frac{2}{3}hL} \times y_n + h\left[\frac{2}{3}\varphi_{1,3} \times N_n + \frac{4}{9c_2}\varphi_{1,3} \times (K_2 - N_n)\right]\right)$
3: $y_{n+1} \leftarrow N\left(t_n + h, e^{hL} \times y_n + h\left[\frac{2}{3}\varphi_1 \times N_n + \varphi_2 \times (K_3 - \frac{3}{2}N_n)\right]\right)$

This algorithm performs a single step for ERK_{33} (29). There are a total of seven matrix-vector operations. By incorporating the (30) optimisation, we can reduce this to 5 operations in total:

1: $K_2 \leftarrow N\left(t_n + c_2 h, y_n + h c_2 \varphi_{1,2} \times (L y_n + N_n)\right)$
2: $K_3 \leftarrow N\left(t_n + \frac{2}{3}h, y_n + \frac{2}{3}h\varphi_{1,3} \times (L y_n + N_n) + h\frac{4}{9c_2}\varphi_{1,3} \times (K_2 - N_n)\right)$
3: $y_{n+1} \leftarrow N\left(t_n + h, y_n + h\varphi_1 \times (L y_n + N_n) + h\varphi_2 \times (\frac{3}{2}K_3 - N_n)\right)$

- $\varphi\times$ **vector Linear Combinations**. Finally, this is the technique introduced by the `PHIPM` code [25] whereby each stage can be completed in a single operation. For ERK_{33} (29), the pseudo code algorithm is:

1: $M_1 \leftarrow \begin{pmatrix} y_n & N_n \end{pmatrix}$
2: $K_2 \leftarrow N\left(t_n + c_2 h, \texttt{phipm}(M_1)\right)$
3: $M_2 \leftarrow \begin{pmatrix} y_n & N_n & \frac{-1}{9c_2}N_2 + \frac{1}{9c_2}K_2 \end{pmatrix}$
4: $K_3 \leftarrow N\left(t_n + c_2 h, \texttt{phipm}(M_2)\right)$
5: $M_3 \leftarrow \begin{pmatrix} y_n & N_n & \frac{-3}{2}N_2 + \frac{3}{2}K_2 \end{pmatrix}$
6: $y_{n+1} \leftarrow \texttt{phipm}(M_3)$

The operations to construct the M_i matrices, whose dimensions we denote as (n, m_i), are computationally negligible. In practice, any of the $\varphi\times$ vector techniques could be modified to take M_i as a parameter. However, each vector would need to be processed individually and therefore the cost of processing the matrix would scale linearly by the number of vectors, m_i. The $\texttt{phipm}(M_i)$ operation cost is independent of m_i and therefore, we simply count the number of stages in the scheme. Under this measurement, ERK_{33} (29) receives a count of 3 operations.

Table 1 summarises the computational cost analysis for the ETD schemes (5) and (6) and the ERK schemes (29) and (8), under each of the three categories of φ-function evaluation. ETD Euler is 1-stage and requires only one φ-function. It's computational cost, therefore, is 1 under each of the categories and it serves as a useful baseline. The cost should be interpreted as a relative measure of the CPU time needed to take a time step. For example, under the explicit evaluation of φ-functions, we would expect ERK_{44} (8) to take 3 times longer to complete the same number of time steps as ETD_2 (6) (the ratio of distinct φ's is 6:2).

Scheme		Explicit Evaluation	$\varphi\times$ vector	Linear Combination
ETD Euler	(5)	1	1	1
ETD_2	(6)	2	2	1
ERK_{33}	(29)	5	5	3
ERK_{44} Krogstad	(8)	6	6	4

Table 1: Relative performance measure for the 2 and 3-Stage ERKs

We provide similar summaries for the EGLM and EARK/EAGLM schemes respectively. In addition, we can deduce the costs for the higher order 2-stage schemes in both families.

The conclusion which we can draw from Tables 2 and 3 is that, under the $\varphi\times$ vector operation count, the EARK methods are more efficient that their EGLMs counterparts for the $c_2 = 1$ case. In addition, the efficiency gap widens with higher-order schemes since, for a single increase in order, the EGLMs incur two additional $\varphi\times$ vector operations while the EARK methods incur only one.

Scheme		c_2	Explicit Evaluation	$\varphi\times$ vector	Linear Combination
EGLM$_{322}c_2$	(9)	$c_2 \neq 1$	5	5	2
EGLM$_{322}c_2$	(9)	$c_2 = 1$	3	4	2
EGLM$_{423}c_2$	(10)	$c_2 \neq 1$	7	7	2
EGLM$_{423}c_2$	(10)	$c_2 = 1$	4	6	2
EGLM$_{p2(p-1)}c_2$		$c_2 \neq 1$	2p - 1	2p - 1	2
EGLM$_{p2(p-1)}$		$c_2 = 1$	p	2p - 2	2

Table 2: Relative performance measure for the four EGLMs

Scheme		c_2	Explicit Evaluation	$\varphi\times$ vector	Linear Combination
EARK$_{3221}c_2$	(20)	$c_2 \neq 1$	5	5	2
EARK$_{3221}c_2$	(20)	$c_2 = 1$	3	3	2
EARK$_{4232}c_2$	(21)	$c_2 \neq 1$	7	7	2
EARK$_{4232}c_2$	(21)	$c_2 = 1$	4	4	2
EARK$_{p2(p-1)(p-2)}c_2$		$c_2 \neq 1$	2p - 1	2p - 1	2
EARK$_{p2(p-1)(p-2)}$		$c_2 = 1$	p	p + 2	2

Table 3: Relative performance measure for the four EAGLMs

8 Numerical Experiments

To highlight the competitiveness of EARK methods, we present experimental benchmarks against a number of established parabolic PDE test problems. All experiments were performed in Matlab 2011b 64bit running on Windows 7 x64. The CPU was an Intel Core 2 Quad Q9450 clocked at 2.66GHz and the system had 8GB of RAM.

For each problem, we perform two numerical experiments, one using the 3$^{\text{rd}}$ order schemes (ERK$_{33}$ (29), EGLM$_{322}$ (9) with $c_2 = 1$ and EARK$_{3221}$ (20) with $c_2 = 1$) and the second using the 4$^{\text{th}}$ order schemes (ERK$_{44}$ (8), EGLM$_{423}c_2$ (10) with $c_2 = 1$ and EARK$_{4232}c_2$ (21) with $c_2 = 1$). In each case, we plot two measurements. The first pair of figures plot step-size against accuracy which illustrates the differences in the per-step accuracy differences between schemes of the same convergence order. The second set of figures plot computational cost against accuracy. First, we use the `phipm` implementation which provides a relative picture of the cost of each scheme when implemented with the linear combination approach to $\varphi-$function evaluations (28). Secondly we plot the same measurements using the Real Leja Points Method (`ReLPM`) [9]. This illustrates a scheme's efficiency when implemented with $\varphi\times$ vector operations.

8.1 Brusselator System

The Brusselator System [14]

$$u_i' = 1 + u_i^2 v_i - 4u_i + \alpha (N+1)^2 (u_{i-1} - 2u_i + u_{i+1})$$
$$v_i' = 3u_i + u_i^2 v_i + \alpha (N+1)^2 (v_{i-1} - 2u_i + v_{i+1})$$

on the interval $x \in [0,1]$ and $t \in [0,15]$, with $\alpha = 1/50$ and initial conditions,

$$u_i(0) = 1 + \sin(2\pi x)$$
$$v_i(0) = 3$$

models diffusion in a chemical reaction. The system is discretised in space, with $M = 127$ resulting in $2M$ equations. The Jacobian is banded with a constant width 5 if the equations are ordered as $u_1, v_1, u_2, v_2, \ldots$. As M increases, the problem becomes increasingly stiff.

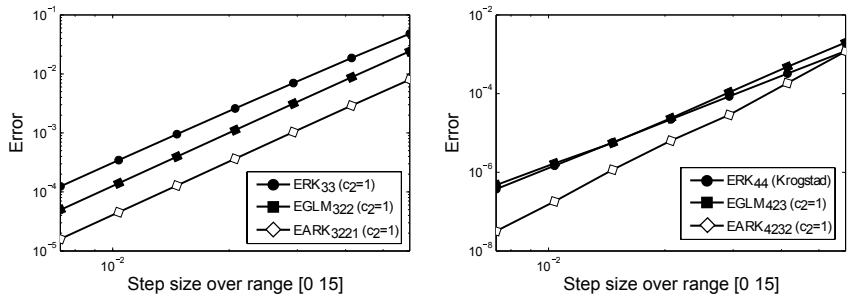
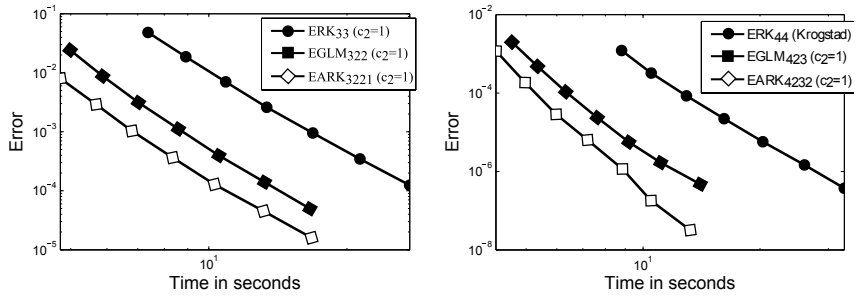
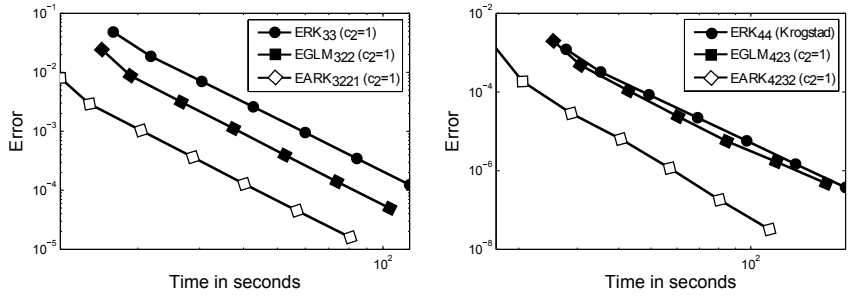


Figure 8: Step-size against Accuracy for the Brusselator System



`phipm` Implementation



ReLPM Implementation

Figure 9: Brusselator System CPU Timing

8.2 The Allen-Cahn Equation

The Allen-Cahn equation [10] is a 1D parabolic PDE

$$y_t = \lambda y_{xx} + y - y^3,$$

on the interval $x \in [-1,1]$ and $t \in [0,50]$, with initial condition,

$$y(0,x) = 0.53x + 0.47\sin(-1.5\pi x)$$

and boundary conditions,

$$y(t,-1) = -1, \qquad y(t,1) = 1$$

We make use of the implementation supplied by the `EXPINT` Matlab package [4, Section 4.2.6] which discretises the linear part, λw_{xx}, using a Chebyshev differentiation matrix to produce a dense L matrix.

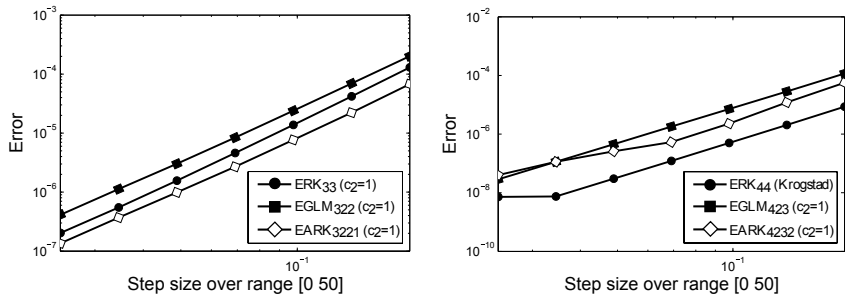
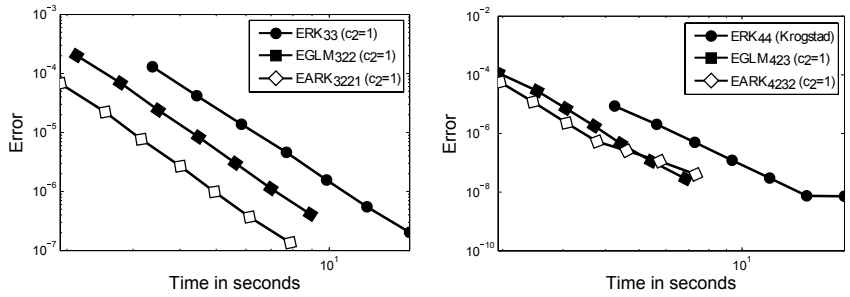
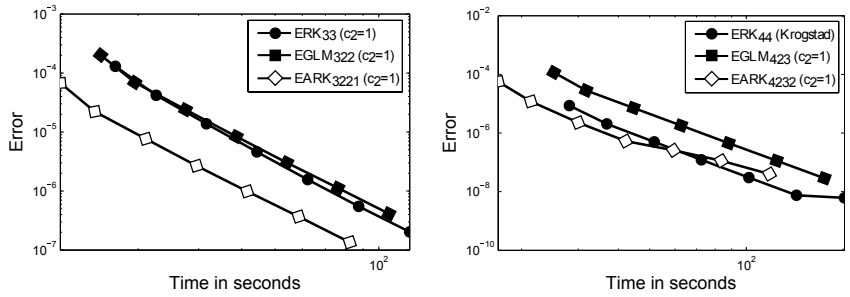


Figure 10: Step-size against Accuracy for the Allen-Cahn problem



`phipm` Implementation



ReLPM Implementation

Figure 11: Allen-Cahn System CPU Timing

8.3 The RDA 2D Equation

The 2D RDA is a stiff problem presented by Caliari & Ostermann [9]. The standard semi-discretisation using finite differences on a uniform spatial mesh gives rise to an L matrix which is pentadiagonal in structure. The equation describes reaction-diffusion-advection

$$u_t = \varepsilon\left(u_{xx}+u_{yy}\right)-\alpha\left(u_x+u_y\right)+\rho u\left(u-\tfrac{1}{2}\right)\left(1-u\right)$$

with $x \in [0,1]^2$ and $t \in [0,0.3]$, subject to homogeneous Neumann boundary conditions and initial condition,

$$u(t=0,x,y) = 0.3+256\left(x(1-x)y(1-y)\right)^2.$$

We will run our experiments for $\varepsilon = 0.05$, $\alpha = -1$ and $\rho = 100$, the same combination of parameters as those used by Caliari & Ostermann [9]. The problem is discretised in space, with $M = 31$. Being a 2D problem, the L matrix can be very large, even for moderately coarse mesh discretisation and, depending on the integrator, can have excessive memory requirements. For example, in Table 4 we list the memory required for different problem discretisations, comparing EIs to Matlab's `ODE15s`. It is clear that memory usage is significantly lower for both the `PHIPM` and `ReLPM` approaches to implementing an EI.

Dimension	32	64	128	256	512
`ODE15s`	52	595	3950	N/A	N/A
`PHIPM`	53	71	170	585	1382
ReLPM	25	63	147	533	1094

Table 4: Memory Usage in Megabytes (MBs)

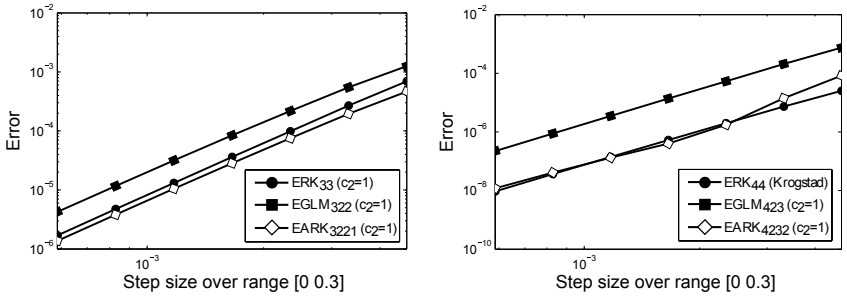


Figure 12: Step-size against Accuracy for the RDA2 problem

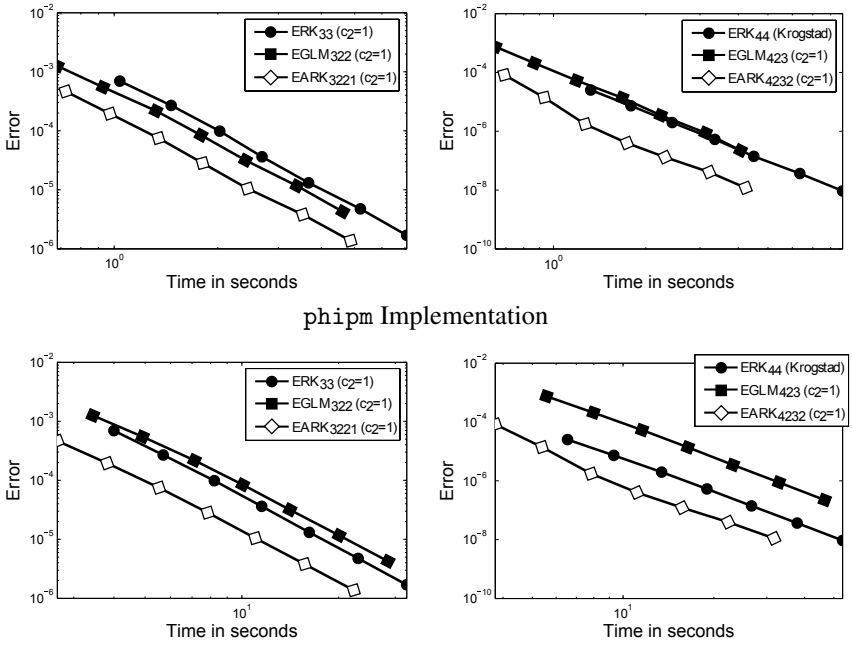


Figure 13: RDA2 System CPU Timing

8.4 Interpreting the Results

Looking at the plots, we can draw a number of conclusions relating to the EARK methods. Figures 8, 10 and 12 plot accuracy against stepsize for each scheme when applied to the three test problems respectively. The EARK schemes demonstrate greater accuracy over the EGLMs, and equal or surpass the performance of the ERKs. Figures 9, 11 and 13 plot accuracy against CPU timings using firstly `PHIPM` and secondly `ReLPM`. With `PHIPM`, the competitive accuracy demonstrated by the ERK methods is offset by their higher computational costs and cause them to significantly under-perform the EARK methods. The EGLMs, like the EARK methods,

are 2-stage schemes and so the computational costs of the two families are nearly identical. The superior accuracy, therefore, of the EARK methods remains their key differentiating factor in outperforming EGLMs. For the `ReLPM` experiments, the fewer $\varphi\times$ vector operations needed to implement the EARK methods results in additional performance improvements when compared to both the ERK methods and EGLMs. Tables 5 - 7 provide global error and cpu measurements from the experiments which reinforce the conclusions drawn from the figures.

Steps	Schemes					
2^8	ERK_{33}	$EGLM_{322}$	$EARK_{3221}$	ERK_{44}	$EGLM_{423}$	$EARK_{4232}$
Error	4.8×10^{-2}	2.4×10^{-2}	8.1×10^{-3}	1.2×10^{-3}	2×10^{-3}	1.2×10^{-3}
`phipm`	7.37	4.9	4.74	8.81	4.6	4.18
ReLPM	17.03	15.8	12.2	27.81	25.42	17.09
2^9						
Error	7×10^{-3}	3.1×10^{-3}	1.1×10^{-3}	8.5×10^{-5}	1.1×10^{-4}	2.9×10^{-5}
`phipm`	10.89	7.02	6.8	12.94	6.33	5.97
ReLPM	30.33	26.51	20.14	49.45	42.26	28.76

Table 5: Brusselator System: Global errors and cpu timings for `phipm` and ReLPM

Steps	Schemes					
2^8	ERK_{33}	$EGLM_{322}$	$EARK_{3221}$	ERK_{44}	$EGLM_{423}$	$EARK_{4232}$
Error	1.3×10^{-4}	2×10^{-4}	6.8×10^{-5}	8.6×10^{-6}	1.2×10^{-4}	5.6×10^{-5}
`phipm`	3.38	2.12	1.92	4.25	1.94	3.16
ReLPM	17.17	15.56	11.88	28.18	25.19	16.91
2^9						
Error	1.4×10^{-5}	2.4×10^{-5}	7.7×10^{-6}	4.9×10^{-7}	7.1×10^{-6}	2.3×10^{-6}
`phipm`	5.84	3.50	3.16	7.28	3.06	3.07
ReLPM	31.61	27.69	21.91	51.79	44.38	29.18

Table 6: Allen-Cahn problem: Global errors and cpu timings for `phipm` and ReLPM

Steps	Schemes					
2^8	ERK_{33}	$EGLM_{322}$	$EARK_{3221}$	ERK_{44}	$EGLM_{423}$	$EARK_{4232}$
Error	1.3×10^{-5}	3.2×10^{-5}	1.1×10^{-5}	1.4×10^{-7}	3.5×10^{-6}	1.3×10^{-7}
`phipm`	3.68	2.41	2.44	4.5	2.24	2.31
ReLPM	16.62	14.46	11.01	26.21	23.33	15.69
2^9						
Error	1.7×10^{-6}	4.2×10^{-6}	1.4×10^{-6}	9.4×10^{-9}	2.2×10^{-7}	1.2×10^{-8}
`phipm`	7.08	4.62	4.86	8.76	4.07	4.22
ReLPM	32.56	28.49	22.27	53.64	46.91	31.52

Table 7: RDA2 problem: Global errors and cpu timings for `phipm` and ReLPM

9 Conclusions

The ARK motivation of including the derivatives of N in the construction of exponential integrator Almost Runge-Kutta (EARK) schemes results in a very compact structure and efficient use of φ-functions while ensuring good stability properties and high order. Although one-step in derivation and analysis, it proved necessary to utilise past values of N when implementation the EARK schemes in order to ensure that the required derivatives of N are approximated to a sufficiently-high accuracy. This will also be a key competitive feature of local error estimation when implementing variable step-size EARK algorithms (to be reported in future work). The inclusion of past values of N gave rise to the development of the exponential integrator Almost General Linear Methods (EAGLMs) which provided a convenient unifying framework for ETD methods, ERK methods, EARK methods and EGLMs. The results of some numerical experiments provide evidence that the EARK methods are competitive with both their ERK and EGLM counterparts in terms of accuracy and computational efficiency when solving semilinear problems using fixed integration stepsizes. In a future paper, we will show that such an EARK/EAGLM combination provides a highly-efficient algorithm for variable stepsize integration of semilinear problems in which the local error estimate becomes available without any additional computation (e.g. φ-function evaluations) unlike the ERK and EGLM counterparts, resulting in a significant saving in computational expense.

References

[1] Al-Mohy, A., Higham, N.: Computing the action of the matrix exponential, with an application to exponential integrators. SIAM J. Sci. Comput. **33**(2), 488–511 (2011)

[2] Bergamaschi, L., Caliari, M., Martínez, A., Vianello, M.: Comparing Leja and Krylov approximations of large scale matrix exponentials. In: International Conference on Computational Science (4), pp. 685–692 (2006)

[3] Bergamaschi, L., Vianello, M.: Efficient computation of the exponential operator for large, sparse, symmetric matrices. Numerical Linear Algebra with Applications **7**(1), 27–45 (2000)

[4] Berland, H., Skaflestad, B., Wright, W.: Expint – A Matlab package for exponential integrators. ACM Trans. Math. Softw. **33**(1), 1–17 (2007)

[5] Beylkin, G., Keiser, J., Vozovoi, L.: A new class of time discretization schemes for the solution of nonlinear PDEs. J. Comput. Phys. **147**(2), 362–387 (1998)

[6] Butcher, J.: On the convergence of numerical solutions to ordinary differential equations. Math. Comp. **20**, 1–10 (1966)

[7] Butcher, J.: General linear methods. Computers & Mathematics with Applications **31**(4-5), 105–112 (1996)

[8] Butcher, J.: An introduction to "Almost Runge-Kutta" methods. Appl. Numer. Math. **24**(2-3), 331–342 (1997)

[9] Caliari, M., Ostermann, A.: Implementation of exponential Rosenbrock-type integrators. Appl. Numer. Math. **59**(3-4), 568–581 (2009)

[10] Calvo, M., Portillo, A.: Variable step implementation of ETD methods for semilinear problems. Applied Mathematics and Computation **196**(2), 627–637 (2008)

[11] Cox, S., Matthews, P.: Exponential time differencing for stiff systems. J. Comput. Phys. **176**(2), 430–455 (2002)

[12] Friesner, R., Tuckerman, L., Dornblaser, B., Russo, T.: A method for exponential propagation of large systems of stiff nonlinear differential equations. J. Sci. Comput. **4**(4), 327–354 (1989)

[13] Gallopoulos, E., Saad, Y.: Efficient solution of parabolic equations by Krylov approximation methods. SIAM J. Sci. Stat. Comput. **13**(5), 1236–1264 (1992)

[14] Hairer, E., Norsett, S., Wanner, G.: Solving Ordinary Differential Equations II. Stiffand Differential-Algebraic Problems, Second Revised Edition with 137 Figures, vol. 2. Springer-Verlag (2002)

[15] Hochbruck, M., Ostermann, A.: Explicit exponential Runge-Kutta methods for semilinear parabolic problems. SIAM J. Numer. Anal. **43**(3), 1069–1090 (2005)

[16] Hochbruck, M., Ostermann, A.: Exponential integrators. Acta Numerica **19**, 209–286 (2010)

[17] Krogstad, S.: Generalized integrating factor methods for stiff PDEs. J. Comput. Phys. **203**(1), 72–88 (2005)

[18] Lu, Y.: Exponentials of symmetric matrices through tridiagonal reductions. Linear Algebra and its Applications **279**, 317–324 (1998)

[19] Lu, Y.: Computing a matrix function for exponential integrators. J. Comput. Appl. Math. **161**(1), 203–216 (2003)

[20] Maset, S., Zennaro, M.: Unconditional stability of explicit exponential Runge-Kutta methods for semilinear ordinary differential equations. Math. Comp. **78**, 957–967 (2009)

[21] Minchev, B.: Exponential integrators for semilinear problems. Ph.D. thesis, Department of Informatics, University of Bergen (2004)

[22] Minchev, B.: Integrating factor methods as exponential integrators. In: Proceedings of the 5th international conference on Large-Scale Scientific Computing, LSSC'05, pp. 380–386. Springer-Verlag, Berlin, Heidelberg (2006)

[23] Minchev, B., Wright., W.: A review of exponential integrators for first order semi-linear problems. Tech. Rep. 2/05, NTNU (2005)

[24] Moler, C., Loan, C.V.: Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. SIAM Review **45**(1), 3–49 (2003)

[25] Niesen, J., Wright, W.M.: A Krylov subspace algorithm for evaluating the phi-functions appearing in exponential integrators. Tech. Rep. arXiv: 0907.4631 (2009)

[26] O'Callaghan, E.: The analysis and implementation of exponential almost Runge-Kutta methods for semilinear problems. Ph.D. thesis, School of Mathematical Sciences, Dublin City University (2011)

[27] Ostermann, A., Thalhammer, M., Wright, W.: A class of explicit exponential general linear methods. BIT **46**(2), 409–431 (2006)

[28] Schmelzer, T.: Talbot quadratures and rational approximations. BIT Numerical Mathematics **46**, 653–670 (September 2006)

[29] Schmelzer, T., Trefethen., L.: Evaluating matrix functions for exponential integrators via Carathéodory-Fejér approximations and contour integrals. ETNA **29**, 1–18 (2007)