Journal of Computational and Applied Mathematics 485 (2026) 117587

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

A fourth-order exponential time differencing scheme with real and distinct poles rational approximation for solving non-linear reaction-diffusion systems

W. K. Attipoe [a], A. Kleefeld [b,c], E. O. Asante-Asamani [a,*]

[a] Clarkson University, 8 Clarkson Ave, Potsdam, 13676, NY, USA
[b] Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH, 52425, Jülich, Germany
[c] Faculty of Medical Engineering and Technomathematics, University of Applied Sciences, Aachen, 52428, Jülich, Germany

ARTICLE INFO

Keywords:
Exponential time differencing
Semilinear parabolic problems
Non-Padé rational approximation
Reaction diffusion systems
Fourth-order time stepping

ABSTRACT

A novel fourth-order, L-stable, exponential time differencing Runge-Kutta type scheme is developed to solve non-linear systems of reaction-diffusion equations, particularly with non-smooth data. The new scheme, ETDRK4RDP, is constructed by approximating the matrix exponentials in the ETDRK4 scheme with a fourth-order, L-acceptable, non-Padé rational function having real and distinct poles. The L-acceptable rational approximation ensures efficient damping of spurious oscillations arising from non-smooth initial and/or boundary conditions. The real and distinct poles of the rational function eliminate the presence of complex arithmetic in the solution of linear systems and thus make the scheme more attractive for parallelization in low-level languages like Fortran and C++. We verify empirically that the new ETDRK4RDP scheme is fourth-order accurate for several reaction diffusion systems with Dirichlet and Neumann boundary conditions and show it to be more efficient than competing exponential time differencing schemes when implemented in parallel, with up to five times speed-up in CPU time.

1. Introduction

Time-dependent reaction-diffusion equations (RDEs) are mathematical models that describe the spatio-temporal dynamics of substances as they react and diffuse in a fluid medium. They are used as classical models of biological pattern formation [1,2], following pioneering work by Alan Turing. With the inclusion of an advection term, RDEs have been used to model the movement of pollutants through surface and ground water [3,4] as well as monitoring the onset and progression of tumors [5–7].

Mathematically, RDEs are described by a system of partial differential equations of the form

$$\begin{aligned} \frac{\partial u(x,t)}{\partial t} &= D\Delta u(x,t) + f(u(x,t),t), \quad \text{in} \quad \Omega_T, \\ u(x,t) &= g(x,t), \quad \text{on} \quad \Gamma_T, \end{aligned} \tag{1}$$

where Ω describes a bounded open subset of $\mathbb{R}^d$, $d \in \{1,2,3\}$ and $\Omega_T := \Omega \times (0,T]$ is a cylindrical domain with parabolic boundary $\Gamma_T := \overline{\Omega_T} \backslash \Omega_T$. The solution to the system $u : \overline{\Omega_T} \longrightarrow \mathbb{R}^s$, describes the spatio-temporal change in the concentration of $s-$species, defined to be $g(x,t)$ on the parabolic boundary and interacting through the nonlinear reaction function $f : \Omega_T \longrightarrow \mathbb{R}^s$, assumed to be sufficiently smooth. The species also diffuse at rates determined by the coefficient matrix $D = \text{diag}(d_1, d_2, \ldots, d_s),\ d_i > 0 \quad \forall i$.

* Corresponding author.
E-mail address: easantea@clarkson.edu (E.O. Asante-Asamani).

https://doi.org/10.1016/j.cam.2026.117587
Received 17 July 2025; Received in revised form 12 February 2026
Available online 4 March 2026

A classical approach for solving Eq. (1) uses the method of lines, where the Laplacian is initially discretized with m^d points to obtain the system of ordinary differential equations

$$\begin{aligned} \frac{dU}{dt} + AU &= F(U(t), t), \quad U \in \mathbb{R}^{s\times m^d}, \\ U(x,0) &= U_0(x), \end{aligned} \tag{2}$$

which can then be solved with an appropriate time-discretization scheme. Here, the matrix $A \in \mathbb{R}^{(s\times m^d)\times(s\times m^d)}$ represents the spatial discretization of the diffusion operator $-D\Delta$, $U(t) = (u_1(t), u_2(t), \dots, u_s(t), \dots, u_{s\times m^d}(t))^\top$ is the time-dependent solution vector and $F \in \mathbb{R}^{s\times m^d}$ a nonlinear map approximating f on the spatial grid.

There are a host of time-stepping methods that solve the resulting stiff ODE systems in Eq. (2). Popular among them are the Backward Euler, IMEX, and Explicit Runge-Kutta methods. These methods have various challenges that can make their computation expensive. They may be fully implicit, which requires an iterative solver at each time step; multistep, which requires a history of solutions which must be computed accurately or are explicit and require relatively small time steps to improve stability. Recently, high order IMEX Runge-Kutta methods as well as methods based on integrating factors have been developed that can preserve physical structures of interest in stiff system [8–12].

The family of exponential time differencing (ETD) schemes seeks to address some of these challenges. ETD schemes are developed from the exact solution to Eq. (2), given by the integral equation

$$U(t) = \mathrm{e}^{A(t-t_0)}U_0 + \int_{t_0}^{t} \mathrm{e}^{A(t-\tau)}F(U(\tau),\tau)\,\mathrm{d}\tau, \quad t \in [t_0, T], \tag{3}$$

obtained through Duhamel's principle. The significance of these schemes is in the accurate treatment of the linear (diffusion) term via matrix exponential leading to stable evolution of stiff problems [13]. The challenge is in integrating the nonlinear function F which depends on the unknown solution U. In 2002, Cox and Matthews introduced the class of ETD Runge-Kutta (ETDRK) schemes [14] which approximate the integral in Eq. (3) by first interpolating the non-linear function F with high-order polynomials and then integrating the resulting expression. The resulting scheme still suffered from numerical instability and inefficiencies resulting from the computation of the matrix exponential and high powers of matrix inverses. Kassam and Trefthan [15] attempted to resolve these instabilities via contour integration, however the resulting scheme required the precomputation of a considerable number of matrix inverses.

A subclass of ETDRK schemes, developed to improve on the computational efficiency of ETDRK schemes use rational functions to approximate the matrix exponentials and eliminate terms involving high powers of matrices through algebraic simplification. A number of these schemes use the standard Padé rational approximations ([16–19]) while others use non-Padé rational functions having real and distinct poles (RDP) [20]. Desirable rational approximations to the exponential e^z should have modulus bounded by 1 when $\mathrm{Re}(z) < 0$ (A-acceptable) and approach zero asymptotically as $|z| \to -\infty$ (L-acceptable). The class of L-acceptable Padé rational functions tend to have complex poles which require the programming of complex arithmetic on lower-level programming languages such as Fortran or C++. Complex arithmetic requires special treatment on such platforms and is known to increase the computational cost of solving linear systems of equation. The attraction of the class of RDP rational functions is that they are L-acceptable and have real and distinct poles, leading to schemes devoid of complex arithmetic and thus have more efficient linear solves. Whereas schemes involving Padé rational functions are parallelizable, the absence of complex arithmetic in RDP schemes make them more attractive for designing parallel algorithms.

In 2016, Asante-Asamani, Khaliq and Wade [20] developed the first class of ETDRK schemes using an RDP rational function to approximate the matrix exponential. The scheme is however of second order, limiting its application to RDEs discretized with higher order spatial discretization schemes such as spectral methods. Later in 2020, a more computationally efficient form of the scheme was introduced for multidimensional problems [13]. In this work, we develop a fourth-order ETD scheme referred to as ETDRK4RDP, which uses a fourth-order RDP rational function, developed by Voss and Khaliq in 1996 [21], to approximate the matrix exponentials in the ETDRK4 scheme. Empirical convergence analysis supports the fourth-order accuracy of the scheme across various scalar and multidimensional RDEs with Dirichlet and Neumann boundary conditions. Comparison with the competing L-stable, fourth-order ETDRK4P04 scheme, which employs the Padé (0,4) rational function, shows improved efficiency for multidimensional systems of RDEs and a significant speed up in CPU time when implemented in parallel.

In Section 2 we present a fourth-order finite difference scheme used to discretize the Laplacian, in order to obtain the diffusion matrix A. In Section 3 we introduce the RDP rational function and its properties, and present the derivation of the ETDRK4RDP scheme. In Section 4, we show and discuss the numerical experiments to validate the convergence and efficiency of serial and parallel versions of the scheme on various test problems. We share some concluding remarks and future work in Section 5.

2. Spatial discretization

We begin by discretizing the second-order spatial derivatives in 1D and extend the results using the Kronecker product of matrices to problems in 2D. To do this, consider Eq. (1) and discretize the Laplacian Δ on the domain $[a, b]$. Define a uniform mesh of size $h = \dfrac{b-a}{m+1}$ and divide the domain into $m+2$ points with $m \geq 3$, with each point on the grid set to $x_j = a + jh$ with $j = 0, 1, 2, \dots, m+1$. To ensure the overall scheme is fourth-order, we approximate the second-order partial derivatives using a standard fourth-order central difference scheme ([22], p. 339). By writing $w_j = w(x_j)$, for a given function $w(x), w : [a, b] \longrightarrow \mathbb{R}$, we can approximate its

second-order derivatives with respect to x at x_j as

$$w''|_{x_j} = \frac{1}{12h^2}(-w_{j-2} + 16w_{j-1} - 30w_j + 16w_{j+1} - w_{j+2}) + \mathcal{O}(h^4), \tag{4}$$

$j = 2, 3, \ldots, m-1$.
The approximation used at the nodes $j = 0, 1, m$ and $m+1$ depends on the boundary conditions imposed. For homogeneous Dirichlet boundary conditions, since the values at the endpoints x_0 and x_{m+1} are known, we extrapolate the values w_{-1} and w_{m+2} using a fourth-degree Lagrange interpolating polynomial centered at x_0 and x_{m+1}. The formulas for the derivatives at x_1 and x_m can be summarized as

$$\begin{aligned} w''|_{x_1} &= \frac{1}{12h^2}(11w_0 - 20w_1 + 6w_2 + 4w_3 - w_4), \\ w''|_{x_m} &= \frac{1}{12h^2}(-w_{m-3} + 4w_{m-2} + 6w_{m-1} - 20w_m + 11w_{m+1}). \end{aligned} \tag{5}$$

For homogeneous Neumann boundary conditions, the value of the function at w_{-1} and w_{m+1} are unknown hence, we consider Eq. (4) and set $w_{-1} = w_1$ and $w_m = w_{m+2}$ from the boundary condition

$$\frac{w_1 - w_{-1}}{2h} = 0 \implies w_1 = w_{-1} \quad \text{and} \quad \frac{w_m - w_{m-2}}{2h} = 0 \implies w_m = w_{m+2}.$$

Thus, the approximation at the boundaries summarizes to

$$\begin{aligned} w''|_{x_1} &\approx \frac{1}{12h^2}(16w_0 - 31w_1 + 16w_2 - w_3), \\ w''|_{x_m} &\approx \frac{1}{12h^2}(-w_{m-2} + 16w_{m-1} - 30w_m + 16w_{m+1} - w_{m+2}). \end{aligned} \tag{6}$$

The derivatives at x_0 and x_{m+1} are set to

$$\begin{aligned} w''|_{x_0} &\approx \frac{1}{12h^2}(-30w_0 + 32w_1 - 2w_2), \\ w''|_{x_{m+1}} &\approx \frac{1}{12h^2}(-2w_{m-1} + 32w_m - 30w_{m+1}). \end{aligned} \tag{7}$$

A similar discretization has been used in Gibou and Fedkiw [23] to achieve fourth-order accuracy on irregular domains. Let I_p be a p-dimensional identity matrix and B_p the matrix obtained from the 1D discretization of the Laplacian using Eq. (4). For systems in 2D, we make use of the Kronecker product for the discrete Laplacian presented in Hundsdorfer and Verwer [24] to formulate the system matrix. In particular, for discrete Laplacian A in 2D, we have that $A = A_1 + A_2$, where

$$A_1 = B_p \otimes I_p \quad \text{and} \quad A_2 = I_p \otimes B_p, \tag{8}$$

with B_p and I_p as the one-dimensional finite difference discretization of the second derivative and identity matrix of equivalent dimension p, respectively. For Dirichlet boundary conditions and $s-$species, $p = s \times m$, whereas $p = s \times (m+2)$ for Neumann boundary conditions.

3. Development of the ETDRK4RDP scheme

3.1. Background

Cox-Matthews [14] and Kassam-Trefethen [15] in 2002 developed a class of fourth-order ETD Runge-Kutta (ETDRK4) schemes which approximate the integral in the exact solution Eq. (3) by interpolating the non-linear function F with higher-order polynomials. The scheme is most preferred due to the accurate treatment of the linear (diffusion) term via matrix exponential leading to stable evolution of stiff problems. The approximation is given by the recurrence relation,

$$\begin{aligned} U_{n+1} = {} & \mathrm{e}^{-kA}U_n + \frac{1}{k^2}(-A)^{-3}[-4I + kA + \mathrm{e}^{-kA}(4I + 3kA + k^2A^2)](F(u_n, t_n) \\ & + \frac{2}{k^2}(-A)^{-3}[2I - kA - \mathrm{e}^{-kA}(2I + kA)])(F(a_n, t_n + k/2) + F(b_n, t_n + h/2)) \\ & + \frac{1}{k^2}(-A)^{-3}[-4I + 3kA - k^2A^2 + \mathrm{e}^{-kA}(4I + kA)]F(c_n, t_n + k) \end{aligned} \tag{9}$$

where,

$$\begin{aligned} a_n &= \mathrm{e}^{-kA/2}U_n - A^{-1}\left(\mathrm{e}^{-kA/2} - I\right)F(u_n, t_n) \\ b_n &= \mathrm{e}^{-kA/2}U_n - A^{-1}\left(\mathrm{e}^{-kA/2} - I\right)F(a_n, t_n + k/2) \\ c_n &= \mathrm{e}^{-kA/2}a_n - A^{-1}\left(\mathrm{e}^{-kA/2} - I\right)[2F(b_n, t_n + k/2) - F(u_n, t_n)]. \end{aligned}$$

The presence of higher powers of matrix inverses and the matrix exponentials, coupled with the reciprocal of the time step (k) in the scheme, makes it computationally expensive and unstable. Some authors [13,17–19], have derived fourth-order ETDRK4 schemes using fourth-order Padé (0,4) and Padé (2,2) rational functions to approximate the matrix exponential. While Padé (0,4) rational functions are L-acceptable they have complex poles, making their algebraic computations less efficient. On the other hand Padé (2,2) rational functions are only A-acceptable, and thus are unable to efficiently damp spurious oscillations for problems with mismatched initial and boundary conditions without the application of presmoothing steps [13]. To curb these issues, we require a rational function that is both L-acceptable and has real and distinct poles to approximate the matrix exponential.
To make the scheme easy to develop, we make the following substitutions to Eq. (9),

$$\begin{aligned} a_n &= \mathrm{e}^{-\frac{k}{2}A}U_n + \tilde{P}(kA)F(U_n,t_n) \\ b_n &= \mathrm{e}^{-\frac{k}{2}A}U_n + \tilde{P}(kA)F(a_n,t_n+\frac{k}{2}) \\ c_n &= \mathrm{e}^{-\frac{k}{2}A}a_n + \tilde{P}(kA)[2F(b_n,t_n+\frac{k}{2}) - F(U_n,t_n)] \\ U_{n+1} &= \mathrm{e}^{-kA}U_n + P_1(kA)F(U_n,t_n) + 2P_2(kA)(F(a_n,t_n+\frac{k}{2}) + F(b_n,t_n+\frac{k}{2})) \\ &\quad + P_3(kA)F(c_n,t_n+k), \end{aligned} \tag{10}$$

with

$$P_1(kA) = \frac{1}{k^2}(-A)^{-3}[-4I + kA + \mathrm{e}^{-kA}(4I + 3kA + k^2A^2)]. \tag{11}$$

$$P_2(kA) = \frac{1}{k^2}(-A)^{-3}[2I - kA - \mathrm{e}^{-kA}(2I + kA)]. \tag{12}$$

$$P_3(kA) = \frac{1}{k^2}(-A)^{-3}[-4I + 3kA - k^2A^2 + \mathrm{e}^{-kA}(4I + kA)]. \tag{13}$$

$$\tilde{P}(kA) = -A^{-1}(\mathrm{e}^{-\frac{k}{2}A} - I). \tag{14}$$

3.2. Real and distinct pole rational function for the matrix exponential

To approximate the matrix exponentials in Eq. (10), we consider the fourth-order RDP rational approximation to e^z developed by D. A. Voss and A. Q. M. Khaliq in Voss and Khaliq [21] in 1996. The compact form of the rational function is

$$R(z) = \frac{N(z)}{D(z)} = \frac{1 + a_1z + a_2z^2 + a_3z^3}{(1-b_1z)(1-b_2z)(1-b_3z)(1-b_4z)}, \tag{15}$$

where $a_i, b_i \in \mathbb{R}$. Note that the motivation for requiring the poles and coefficients to be real is to eliminate the need for complex arithmetic within the final scheme and its associated linear systems with the expectation to speed up run time. The choice of repeated poles, i.e. $b_i = b$ leads to a partial fraction decomposition of the form

$$R(z) = \sum_{i=1}^{4} \frac{w_i}{(1-b_iz)} \tag{16}$$

which was suggested by Khaliq and Voss to result in essentially serial schemes [21]. Requiring the poles to be distinct leads to the partial fraction decomposition

$$R(z) = \sum_{i=1}^{4} \frac{w_i}{1-b_iz}, \tag{17}$$

where the expansion coefficients are estimated using, $w_i = \lim\limits_{z \longrightarrow \frac{1}{b_k}} (1-b_kz)R(z)$. Such separation of poles results in independent backward Euler-type linear systems which admit a natural parallelization, as will be shown in the sections that follow.

In order for the rational approximation $R(z)$ in Eq. (15) to be A-acceptable (ie. $|R(z)| < 1$ when $Re(z) < 0, z \in \mathbb{C}$) it suffices to choose the coefficients $b_i > 0$ and a_i, b_i such that $E(y) := |D(\mathrm{i}y)|^2 - |N(\mathrm{i}y)|^2 \geq 0, \forall y \in \mathbb{R}$. This follows from the maximum principle and the fact that $b_i > 0$ ensures that $R(z)$ is analytic on the left-half complex plane. L-acceptablity (A-acceptable and $|R(z)| \longrightarrow 0$ as $z \longrightarrow -\infty$) follows from the fact that the degree of the numerator $N(z)$ is greater than that of the denominator $D(z)$. In addition to these constraints, the requirement that the rational fraction be a fourth order approximation to the exponential led to the coefficients given in Table 1. Note that the weights w_i refer to the partial fraction form of $R(z)$ in Eq. (17). The superiority of the RDP rational functions, over L-acceptable Padé schemes of similar order, lies in its real and distinct poles which avoid complex arithmetic in the solution of linear systems. The graph of $R(-z)$ using the parameters in Table 1 is shown in Fig. 1.

To derive the ETDRK4RDP scheme, we define a corresponding rational approximation of the matrix exponential. Recall from Cauchy's integral formula for matrix functions that if $f(z)$ is a complex analytic function inside and on a simple closed contour Γ, and A is a square matrix whose spectrum lies entirely inside Γ, then the function of the matrix $f(A)$ defined by:

$$f(A) = \frac{1}{2\pi\mathrm{i}} \int_\Gamma f(z)(zI - A)^{-1}\, \mathrm{d}z, \tag{18}$$

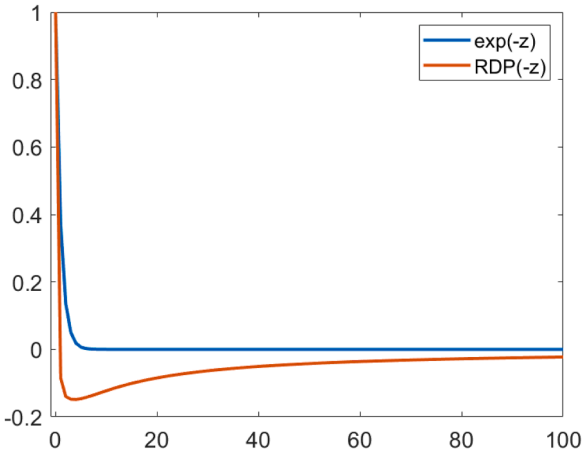

Fig. 1. Approximation of e^{-z} with $R(-z)$.

Table 1
Parameter values: Partial fraction decomposition of the RDP rational function.

k	b_k	w_k	a_k
1	0.4751834017787114	20.10707940496431	1.579627631895178
2	1.0000000000000000	0.5229558818011362	0.303085087930133
3	0.3888888888888889	–15.21083750434353	–0.324250474367700
4	0.7155553412275962	–4.419197782421921	

is well-defined by integrating around the eigenvalues of A [25,26]. The corresponding matrix notation of the RDP for the matrix exponential is given as

$$R(-kA) = (I - a_1 kA + a_2 k^2 A^2 - a_3 k^3 A^3)\left(\prod_{i=1}^{4}(I + b_i kA)\right)^{-1}, \tag{19}$$

with corresponding partial fraction decomposition of the form

$$R(-kA) = \sum_{i=1}^{4}(w_i)(I + b_i kA)^{-1}. \tag{20}$$

The RDP approximation to $e^{-\frac{k}{2}A}$ in matrix notation is given by

$$R(-kA/2) = \left(I - \frac{k}{2}a_1 A + \frac{k^2}{4}a_2 A^2 - \frac{k^3}{8}a_3 A^3\right)\left(\prod_{i=1}^{4}\left(I + \frac{b_i}{2}kA\right)\right)^{-1}, \tag{21}$$

with partial fraction decomposition of the form

$$R(-kA/2) = \sum_{i=1}^{4} w_i\left(I + \frac{b_i}{2}kA\right)^{-1}. \tag{22}$$

3.3. Derivation of the ETDRK4RDP scheme

We begin the derivation by considering each of the approximations a_n, b_n, c_n and U_{n+1} separately and simplify their results when the matrix exponentials are substituted with the corresponding RDP approximation. Consider $\tilde{P}(kA)$ as defined in Eq. (14) and substitute the RDP approximation $R(-kA/2)$ in Eq. (21) with $e^{-kA/2}$. We have

$$\tilde{P}(kA) = -A^{-1}(e^{-\frac{k}{2}A} - I).$$

That is,

$$\begin{aligned}\tilde{P}(kA) &= -A^{-1}(R(-kA/2) - I)\\ &= -A^{-1}\left((I - \frac{k}{2}a_1 A + \frac{k^2}{4}a_2 A^2 - \frac{k^3}{8}a_3 A^3)\left(\prod_{i=1}^{4}\left(I + \frac{b_i}{2}kA\right)\right)^{-1} - I\right).\end{aligned}$$

Then, we factor out the matrix inverses and simplify algebraically and write the partial fraction decomposition of $\tilde{P}(kA)$ as

$$\tilde{P}(kA) = \sum_{i=1}^{4} q_i \left(I + \frac{b_i}{2} kA \right)^{-1}, \tag{23}$$

where

$$q_i = \frac{k\left(\mu_1 - 2\frac{\mu_2}{b_i} + 4\frac{\mu_3}{b_i^2} - 8\frac{\mu_4}{b_i^3} \right)}{\prod_{j=1, j\neq i}^{4} \left(1 - \frac{b_j}{b_i} \right)} \qquad i = 1, 2, 3, 4.$$

and,

$$\mu_1 = \frac{1}{2}(a_1 - (b_1 + b_2 + b_3 + b_4)),\ \mu_2 = \frac{1}{4}(a_2 - (b_1 b_2 + b_3 b_4 + (b_1 + b_2)(b_3 + b_4)),$$
$$\mu_3 = \frac{1}{8}(a_3 - ((b_1 + b_2) b_3 b_4 + (b_3 + b_4) b_1 b_2)),\ \mu_4 = -\frac{1}{16}(b_1 b_2 b_3 b_4).$$

To simplify a_n, we substitute the partial fraction decomposition of $\tilde{P}(-kA)$ in Eq. (23) into Eq. (10) and the RDP approximation $R(-kA/2)$ in Eq. (22) with $\mathrm{e}^{-kA/2}$. Thus,

$$\begin{aligned} a_n &= \mathrm{e}^{(-kA/2)} U_n + \tilde{P}(kA) F(U_n, t_n) \\ &= R(-kA/2) U_n + \tilde{P}(kA) F(U_n, t_n) \\ &= \sum_{i=1}^{4} w_i \left(I + \frac{b_i}{2} kA \right)^{-1} U_n + \sum_{i=1}^{4} q_i \left(I + \frac{b_i}{2} kA \right)^{-1} F(U_n, t_n). \end{aligned}$$

Next, we factor out the sum of the matrix inverse and simplify the resulting term in partial fraction form as

$$a_n = \sum_{i=1}^{4} J_{ia}, \tag{24}$$

where,

$$J_{ia} = \left(I + \frac{b_i}{2} kA \right)^{-1} (w_i U_n - q_i F(U_n, t_n)).$$

Similarly for b_n, we substitute the partial fraction decomposition of $\tilde{P}(-kA)$ in Eq. (23) into Eq. (10) and the RDP approximation $R(-kA/2)$ in Eq. (22) with $\mathrm{e}^{-kA/2}$.

$$\begin{aligned} b_n &= \mathrm{e}^{-\frac{k}{2}A} U_n + \tilde{P}(kA) F(a_n, t_n + \frac{k}{2}) \\ &= R(-kA/2) U_n + \tilde{P}(kA) F(a_n, t_n + \frac{k}{2}) \\ &= \sum_{i=1}^{4} w_i \left(I + \frac{b_i}{2} kA \right)^{-1} U_n + \sum_{i=1}^{4} q_i \left(I + \frac{b_i}{2} kA \right)^{-1} F\left(a_n, t_n + \frac{k}{2} \right). \end{aligned}$$

Again, we factor out the matrix inverses and simplify the resulting term in partial fraction form as,

$$b_n = \sum_{i=1}^{4} J_{ib}, \tag{25}$$

where,

$$J_{ib} = \left(I + \frac{b_i}{2} kA \right)^{-1} (w_i U_n - q_i F(a_n, t_n + k/2)).$$

Similarly, for c_n we substitute $\tilde{P}(-kA)$ in Eq. (23) into Eq. (10) and the RDP approximation $R(-kA/2)$ in Eq. (22) with $\mathrm{e}^{-kA/2}$. The resulting equation is,

$$\begin{aligned} c_n &= \mathrm{e}^{-\frac{k}{2}A} a_n + \tilde{P}(kA)[2F(b_n, t_n + k/2) - F(U_n, t_n)] \\ &= R(-kA/2) a_n + \tilde{P}(kA)[2F(b_n, t_n + k/2) - F(U_n, t_n)] \\ &= \sum_{i=1}^{4} w_i \left(I + \frac{b_i}{2} kA \right)^{-1} a_n + \sum_{i=1}^{4} q_i \left(I + \frac{b_i}{2} kA \right)^{-1} [2F(b_n, t_n + k/2) - F(U_n, t_n)]. \end{aligned}$$

We again factor out the matrix inverses and simplify the resulting term in partial fraction form as,

$$c_n = \sum_{i=1}^{4} J_{ic}, \tag{26}$$

where

$$J_{ic} = \left(I + \frac{b_i}{2}kA\right)^{-1}(w_i a_n - q_i[2F(b_n, t_n + k/2) - F(U_n, t_n)]).$$

Finally, to evaluate U_{n+1}, we begin by substituting the RDP approximation $R(-kA)$ in Eq. (19) with e^{-kA} and simplify the coefficients of each of the terms in the expression independently. To begin, observe that

$$(I + b_1kA)(I + b_2kA)(I + b_3kA)(I + b_4kA) = I + \alpha kA + \beta k^2A^2 + \gamma k^3A^3 + \rho k^4A^4,$$

where

$$\alpha = b_1 + b_2 + b_3 + b_4, \qquad \beta = b_1b_2 + b_3b_4 + (b_1 + b_2)(b_3 + b_4),$$
$$\gamma = (b_1 + b_2)b_3b_4 + (b_3 + b_4)b_1b_2, \qquad \rho = b_1b_2b_3b_4.$$

Consider $P_1(kA)$ and substitute e^{-kA} with $R(-kA)$ as defined in Eq. (22), thus

$$\begin{aligned} P_1(kA) &= \frac{1}{k^2}(-A)^{-3}[-4 + kA + R(-kA)(4 + 3kA + k^2A^2)] \\ &= \frac{1}{k^2}(-A)^{-3}[-4 + kA + (I + a_1kA + a_2k^2A^2 + a_3k^3A^3)\left(\prod_{i=1}^{4}(I + b_ikA)\right)^{-1} \\ &\quad \times(4 + 3kA + k^2A^2)]. \end{aligned}$$

Next, factor out the product of the matrix inverse and simplify the resulting algebraic expression,

$$P_1(kA) = (w_1kI + w_2k^2A + w_3k^3A^2)\left(\prod_{i=1}^{4}(I + b_ikA)\right)^{-1}.$$

Next, we write this rational function in partial fraction form as

$$P_1(kA) = \sum_{i=1}^{4} r_i\left(I + b_ikA\right)^{-1}, \tag{27}$$

where,

$$r_i = \frac{\left(w_1 - \frac{w_2}{b_i} + \frac{w_3}{b_i^2}\right)k}{\prod_{j=1, j\neq i}^{4}\left(1 - \frac{b_j}{b_i}\right)}, \quad i = 1, 2, 3, 4,$$

with

$$w_1 = 4\gamma - \beta - a_1 - 3a_2 - 4a_3, \quad w_2 = 4\rho - \gamma - a_2 - 3a_3, \quad w_3 = -\rho - a_3.$$

Similar to how we obtained $P_1(kA)$, we simplify $P_2(kA)$ as,

$$P_2(kA) = \sum_{i=1}^{4} g_i\left(I + b_ikA\right)^{-1}, \tag{28}$$

where,

$$g_i = \frac{\left(l_1 - \frac{l_2}{b_i} + \frac{l_3}{b_i^2}\right)k}{\prod_{j=1, j\neq i}^{4}\left(1 - \frac{b_j}{b_i}\right)}, \qquad i = 1, 2, 3, 4$$

with,

$$l_1 = -(2\gamma - \beta - (2a_3 + a_2)), \quad l_2 = -(2\rho - \gamma - a_3), \quad l_3 = \rho.$$

Finally, we simplify $P_3(kA)$ in a similar procedure as,

$$P_3(kA) = \sum_{i=1}^{4} h_i\left(I + b_ikA\right)^{-1}, \tag{29}$$

where,

$$h_i = \frac{\left(m_1 - \frac{m_2}{b_i} + \frac{m_3}{b_i^2} - \frac{m_4}{b_i^3}\right)k}{\prod_{j=1, j\neq i}^{4}\left(1 - \frac{b_j}{b_i}\right)}, \qquad i = 1, 2, 3, 4$$

with,

$$m_1 = -(-4\gamma + 3\beta - \alpha + (a_2 + 4a_3)), m_2 = -(-4\rho + 3\gamma - \beta + a_3), m_3 = -(3\rho - \gamma), m_4 = \rho.$$

Substituting the simplified coefficients $P_1(kA)$, $P_2(kA)$ and $P_3(kA)$ in Eqs. (27)–(29) respectively, into the final approximation U_{n+1} in Eq. (9), we have that

$$\begin{aligned} U_{n+1} = &\left(\sum_{i=1}^{4}(w_i)(I + b_i kA)^{-1}\right)U_n + \left(\sum_{i=1}^{4} r_i\left(I + b_i kA\right)^{-1}\right)F(U_n, t_n) \\ &+2\left(\sum_{i=1}^{4} g_i\left(I + b_i kA\right)^{-1}\right)(F(a_n, t_n + k/2) + F(b_n, t_n + k/2) \\ &+\left(\sum_{i=1}^{4} h_i\left(I + b_i kA\right)^{-1}\right)F(c_n, t_n + k). \end{aligned}$$

We then group the terms with similar quotients and summarize as follows

$$\begin{aligned} U_{n+1} = \sum_{i=1}^{4}\Big\{ &\left(I + b_i kA\right)^{-1}(w_i U_n + r_i F(U_n, t_n) + 2g_i[F(a_n, t_n + k/2) + F(b_n, t_n + k/2)] \\ &+ h_i F(c_n, t_n + k))\Big\}. \end{aligned}$$

Rewriting the solution U_{n+1} as a linear system, we solve for Y_i for $i = 1, 2, 3, 4$ in the following systems

$$(I + b_i kA)Y_i = w_i U_n + r_i F(U_n, t_n) + 2g_i[F(a_n, t_n + k/2) + F(b_n, t_n + k/2)] + h_i F(c_n, t_n + k),$$

and obtain U_{n+1} as

$$U_{n+1} = \sum_{i=1}^{4} Y_i.$$

3.4. Implementation of the ETDRK4RDP algorithm

Putting all the results together, the ETDRK4RDP scheme can be implemented as follows:

1. Solve: $\left(I + \frac{b_i}{2}kA\right)J_{ia} = (w_i U_n - q_i F(U_n, t_n)), \quad$ for $i = 1, 2, 3, 4$
2. Set $a_n = \sum_{i=1}^{4} J_{ia}$.
3. Solve: $\left(I + \frac{b_i}{2}kA\right)J_{ib} = (w_i U_n - q_i F(a_n, t_n + k/2)), \quad$ for $i = 1, 2, 3, 4$.
4. Set $b_n = \sum_{i=1}^{4} J_{ib}$.
5. Solve: $\left(I + \frac{b_i}{2}kA\right)J_{ic} = (w_i a_n - q_i[2F(b_n, t_n + k/2) - F(U_n, t_n)]), \quad$ for $i = 1, 2, 3, 4$.
6. Set $c_n = \sum_{i=1}^{4} J_{ic}$.
7. Solve:

$$\begin{aligned} (I + b_i kA)Y_i = &w_i U_n + r_i F(U_n, t_n) + 2g_i[F(a_n, t_n + k/2) + F(b_n, t_n + k/2)] \\ &+ h_i F(c_n, t_n + k), \quad \text{for } i = 1, 2, 3, 4 \end{aligned}$$

8. Set $U_{n+1} = \sum_{i=1}^{4} Y_i$.

Notice that computing each stage value a_n, b_n, c_n as well as the final solution U_{n+1} requires four independent linear solves. Thus, the algorithm can be implemented in parallel using four processors to solve the linear systems associated with each stage concurrently. We will compare the serial and parallel performance of our ETDRK4RDP scheme with the existing fourth-order ETDRK4P22 and ETDRK4P04 schemes. The algorithms for the respective schemes are provided below.

3.5. Implementation of the ETDRK4P04 algorithm

The algorithm for ETDRK4P04 is as follows. For each time step:

1. Solve: $(kA - \tilde{c}_i I)a_{ni} = \tilde{w}_i U_n + k\tilde{\Omega}_i F(U_n, t_n)$, $i = 1, 2$.
2. Set: $a_n = 2\sum_{i=1}^{2} Re(a_{ni})$
3. Solve: $(kA - \tilde{c}_i I)b_{ni} = \tilde{w}_i U_n + k\tilde{\Omega}_i F(a_n, t_n + \frac{k}{2})$, $i = 1, 2$.
4. Set: $b_n = 2\sum_{i=1}^{2} Re(b_{ni})$
5. Solve: $(kA - \tilde{c}_i I)c_{ni} = \tilde{w}_i a_n + k\tilde{\Omega}[2F(b_n, t_n + \frac{k}{2}) - F(U_n, t_n)]$, $i = 1, 2$.
6. Set: $c_n = 2\sum_{i=1}^{2} Re(c_{ni})$
7. Solve: $(kA - \hat{c}_i)u_{n1} = \hat{w}_i U_n + \hat{w}_{1i} kF(U_n, t_n) + 2k\hat{w}_{21} G(a_n, b_n, t_n + \frac{k}{2}) + k\hat{w}_{3i} F(c_n, t_n + k)$, $i = 1, 2$.
8. Set: $U_{n+1} = 2\sum_{i=1}^{2} Re(u_{ni})$.

The exact values for $\tilde{c}_i, \hat{c}_i, \tilde{w}_i, \tilde{\Omega}_i, \hat{w}_{li}, l = 1, 2, 3$ can be found in Yousuf [18] Pg. 126–127. Notice that computing each stage value a_n, b_n, c_n as well as the final solution U_{n+1} requires two independent linear solves. Thus, the algorithm can be implemented in parallel using two processors for solve the linear systems.

3.6. Implementation of the ETDRK4P22 algorithm

The algorithm for ETDRK4P22 is as follows. For each time step:

1. Solve: $(kA - \tilde{c}_2 I)a_{n1} = 2w_{11} U_n + 24w_{51} kF(U_n, t_n)$
2. Set: $a_n = U_n + 2Re(a_{n1})$
3. Solve: $(kA - \tilde{c}_2 I)b_{n1} = 2w_{11} U_n + 24w_{51} kF(a_n, t_n + \frac{k}{2})$
4. Set: $b_n = U_n + 2Re(b_{n1})$
5. Solve: $(kA - \tilde{c}_2 I)c_{n1} = 2w_{11} a_n + 24w_{51} k[2F(b_n, t_n + \frac{k}{2}) - F(U_n, t_n)]$
6. Set: $c_n = a_n + 2Re(c_{n1})$
7. Solve: $(kA - \tilde{c}_1 I)u_{n1} = w_{11} U_n + w_{21} kF(U_n, t_n) + 4w_{31} kG(a_n, b_n, t_n + \frac{k}{2}) + w_{41} kF(c_n, t_n + k)$.
8. Set: $U_{n+1} = U_n + 2Re(u_{n1})$.

The precise values for $\tilde{c}_i, w_{ij} \in \mathbb{C}$ are given in Khaliq et al. [19], Pg 381. Observe that computing each stage value a_n, b_n, c_n as well as the final solution U_{n+1} requires one linear solve. Thus, the algorithm is inherently serial.

4. Numerical experiments and results

The ETDRK scheme on which our new ETDRK4RDP scheme is based is a fourth-order accurate method [14]. Since the matrix exponential in the ETDRK scheme has been replaced with a fourth-order rational approximation in ETDRK4RDP, we expect the new scheme to be fourth-order accurate. Additionally, we have discretized our spatial derivatives using a fourth-order accurate finite difference scheme, thus we expect the fully discrete scheme to be fourth-order accurate in both space and time. We will verify the order of accuracy empirically here, and defer a full theoretical analysis to future work.

To investigate the order of accuracy and efficiency of the proposed scheme, we tested the performance of the scheme on a variety of test problems with various initial and boundary conditions. We compared the accuracy of the new scheme with existing fourth-order ETDRK4P22 and ETDRK4P04 schemes developed in Yousuf [18] and Asante-Asamani et al. [13], respectively. The ETDRK4P22 scheme developed by M. Yousuf for semilinear parabolic problems uses the Padé(2,2) rational function, which is not L-acceptable while the ETDRK4P04 uses the Padé(0, 4) scheme to approximate the matrix exponentials in the ETDRK4 scheme developed by Cox and Matthews [14]. We evaluate the performance of the schemes on problems in two spatial dimensions as well as for scalar and systems of reaction-diffusion equations with homogeneous Dirichlet and Neumann boundary conditions. The error, $E(k)$ at time step k, is measured with the $L_\infty-$ norm and the estimated order of convergence is calculated using the formula

$$p = \frac{\log[E(k)/E(k/2)]}{\log 2}.$$

For problems with known exact solution, we estimate the error as $E(k) = \|U(k) - \hat{U}(k)\|_\infty$, where $\hat{U}(k)$ and $U(k)$ denote the approximate grid solution and exact solutions, respectively at the final time T using a temporal step size of k. The spatial derivatives are approximated with a fourth-order finite difference scheme on a mesh with spacing h described in Section 2. By setting $k \approx h$, we were able to derive the correct order of convergence of the fully discretized system. For problems with unknown exact solution, we use the numerical solution on the next level of refinement as the reference solution for computing errors and the order of convergence, $\tilde{p}$. We only evaluated the convergence of the time discretization by fixing the spatial mesh size, h. The errors are computed by using the numerical solution on the next fine mesh as the reference solution. For a grid refinement study with temporal step sizes $k, k/2, k/4, k/8$ we use the formula

$$\tilde{p} = \frac{\log[\tilde{E}(k)/\tilde{E}(k/2)]}{\log 2},$$

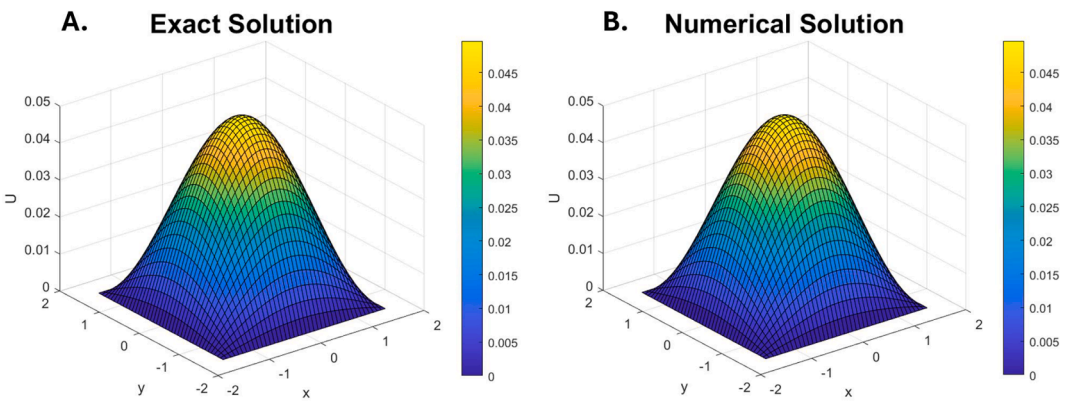

Fig. 2. Solution profiles. (A) Exact solution and (B) Numerical solution using ETDRK4RDP for test-problem 4.1.

where, $\tilde{E}(k) = \|\hat{U}(k) - \hat{U}(k/2)\|$. It's been shown in Leveque [1] that the estimation of the error this way determines the correct order of convergence of the scheme for sufficiently small k.

Numerical experiments involving serial implementation of all algorithms were run in MATLAB R2023b on a Dell Latitude 5440 with 13th Gen Intel(R) Core(TM) i7-1355U 1.70 GHz with 32 GB RAM. Parallel versions of the algorithms were run in Fortran using OpenMP with 2–4 processors on a PC with 32 × 12th Gen Intel Core i9-14900 with 32 GiB of RAM and 1.0TB of memory running Ubuntu 22.04. Linear systems were solved using UMFPACK package (see [27]) by first storing sparse matrices in compressed column storage format (see [28]). A similar implementation has been used in Asante-Asamani et al. [29] to achieve similar results. We chose to implement parallelization in Fortran because efforts to do this in MATLAB produced less efficient schemes. We believe this is primarily because MATLAB uses multithreading to solve linear sytsems efficiently. Efforts to directly parallelize the solution of independent linear systems limits multithreading to a single processor, leading to reduced performance. The MATLAB and Fortran codes for implementing ETDRK4RDP for the respective test problems have been made publicly available and can be accessed at https://github.com/wisdomattipoe/ETDRK4RDP-Scheme-Serial.git.

4.1. Model problem with a Dirichlet boundary condition

We investigate the performance of the proposed scheme by considering the following two-dimensional reaction-diffusion equation with corresponding homogeneous Dirichlet boundary and initial condition,

$$\begin{aligned} \frac{\partial u}{\partial t} &= \Delta u - u, \quad \frac{-\pi}{2} < x, y < \frac{\pi}{2}, \quad t \in [0, T]. \\ u(x, y, 0) &= \cos(x)\cos(y). \end{aligned} \tag{30}$$

The exact solution is given by $u(x, y, t) = \mathrm{e}^{-3t}\cos(x)\cos(y)$. We discretize each dimension of the spatial domain with $m + 2$ nodes with spatial step size $h = \frac{\pi}{m+1}$. The spatial discretization is performed as discussed in Section 2. As seen in Fig. 2 the numerical solution mimics the exact solution profile. The convergence plot in Fig. 3A and dummyTXdummy- grid refinement study in Table 2 confirm the fourth-order accuracy of the fully discrete scheme, since we vary both the time step (k) and spatial mesh size (h) in our analysis. For all time steps examined, the serial implementation of ETDRK4RDP scheme in MATLAB is less accurate compared to ETDRK4P04 and ETDRK4P22 schemes and requires more CPU time (See Fig. 3 and Table 2). The better accuracy observed for ETDRK4P04 and the ETDRK4P22 can be attributed to smaller error constants in their corresponding rational approximations, with Padé(2,2) having the smallest error constant.

It is noteworthy that the ETDRK4RDP scheme performs twice the number of linear solves as the ETDRK4P04 scheme and four times the number of linear solves as the ETDRK4P22 but only increases the CPU time by 26% compared with ETDRK4P04 and 158% compared with ETDRK4P22, which are both well below expectation and suggests that ETDRK4RDP is more efficiently in solving each linear systems. This is not surprising since our scheme is designed without complex arithmetic, which is known to increase the complexity of solving linear systems. These results suggest a potential to significantly improve the CPU time of ETDRK4RDP through parallelization.

As noted earlier, we implemented parallel algorithms in Fortran because MATLAB's use of multithreading to solve linear systems was restricted by our efforts to dedicate single processors to solve independent linear systems through direct parallelization. Observe from Table 2 that a serial implementation of all algorithms run significantly faster in Fortran compared to MATLAB with a 3 times speed up for ETDRK4RDP, 2.5 times speed up for ETDRK4P04 and 2 times speed up for ETDRK4P22.

Since ETDRK4P22 is inherently a serial algorithm no parallelization was necessary. We implemented the remaining schemes in parallel within Fortran with four processors for ETDRK4RDP and two processors for ETDRK4P04, as required by their respective parallel algorithms. As can be observed from Table 2, parallelization of ETDRK4RDP results in a two times speed up in its CPU time at the finest time resolution (k = 0.0125) (compared with the serial implementation in Fortran). A similar speed up was observed for ETDRK4P04. Comparatively, the parallelized ETDRK4RDP is faster than ETDRK4P04 and ETDRK4P22.

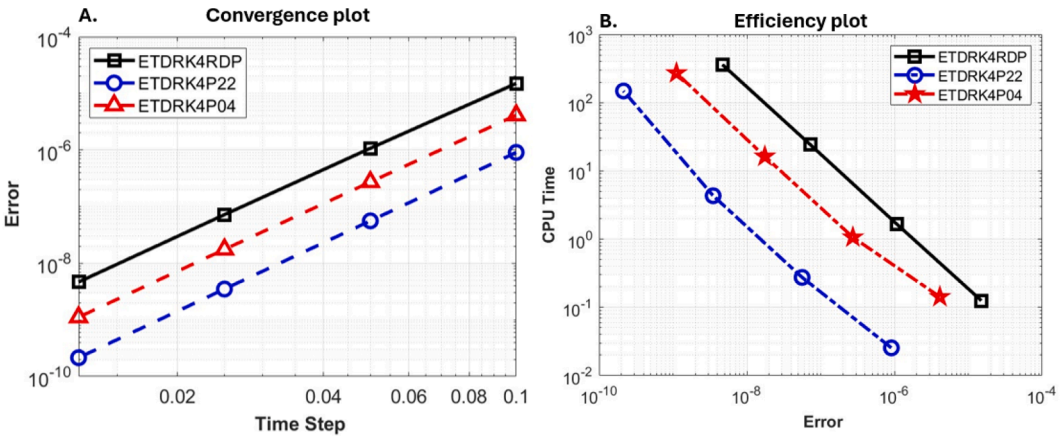

Fig. 3. Convergence and efficiency plots for Example 4.1. (A) Convergence and (B) Efficiency of the new fourth-order ETDRK4RDP scheme compared with the existing fourth-order ETDRK4P22, and ETDRK4P04 schemes.

Table 2
Convergence table showing fourth-order convergence of the new ETDRK4RDP scheme compared with the existing ETDRK4P04 and ETDRK4P22 schemes for the model test problem in Example 4.1 with $T = 1$. Table also displays the CPU time for serial implementation of all schemes in MATLAB as well as single core (1-core) and multicore implementation in Fortran.

Method	k	h	Error	p	CPU Time MATLAB	1-core	Multicore
ETDRK4RDP	0.10	0.08	1.50×10^{-5}	–	0.066	0.031	0.010
	0.05	0.04	1.07×10^{-6}	3.80	0.808	0.381	0.143
	0.025	0.02	7.23×10^{-8}	3.89	12.056	4.842	2.391
	0.0125	0.01	4.66×10^{-9}	3.96	178.107	53.832	27.501
ETDRK4P04	0.10	0.08	4.12×10^{-6}	–	0.069	0.032	0.016
	0.05	0.04	2.73×10^{-7}	3.92	0.724	0.432	0.216
	0.025	0.02	1.76×10^{-8}	3.96	9.791	5.112	2.556
	0.0125	0.01	1.11×10^{-9}	3.98	140.289	56.136	30.285
ETDRK4P22	0.10	0.08	9.07×10^{-7}	–	0.025	0.018	–
	0.05	0.04	5.62×10^{-8}	4.01	0.301	0.190	–
	0.025	0.02	3.50×10^{-9}	4.01	4.659	3.294	–
	0.0125	0.01	2.12×10^{-10}	4.03	69.603	33.342	–

Table 3
Convergence table showing fourth-order convergence of the new ETDRK4RDP scheme compared with the existing ETDRK4P04 and ETDRK4P22 schemes for the model test problem in Example 4.2 with $T = 1$. Table also displays the CPU time for serial implementation of all schemes in MATLAB as well as single core (1-core) and multicore implementation in Fortran.

Method	k	h	Error	p	CPU Time (Serial)	1-core	Multi-core
ETDRK4RDP	0.10	0.16	4.37×10^{-6}	–	0.004	0.004	0.003
	0.05	0.08	4.05×10^{-7}	3.53	0.103	0.079	0.021
	0.025	0.04	3.03×10^{-8}	3.74	1.416	0.927	0.232
	0.0125	0.02	2.07×10^{-9}	3.87	23.926	15.874	3.969
ETDRK4P04	0.10	0.16	1.48×10^{-5}	–	0.010	0.030	0.015
	0.05	0.08	9.43×10^{-7}	3.97	0.085	0.410	0.045
	0.025	0.04	5.95×10^{-8}	3.99	1.145	1.684	1.043
	0.0125	0.02	3.74×10^{-9}	3.99	17.779	14.969	8.461
ETDRK4P22	0.10	0.16	1.16×10^{-5}	–	0.006	0.019	–
	0.05	0.08	7.27×10^{-7}	3.99	0.033	0.081	–
	0.025	0.04	4.54×10^{-8}	4.00	0.558	0.681	–
	0.0125	0.02	2.84×10^{-9}	4.00	8.663	7.528	–

4.2. Model problem with a Neumann boundary

Next, we consider the following scalar reaction-diffusion system with homogeneous Neumann boundary conditions given by

$$\begin{aligned} \frac{\partial u}{\partial t} &= \Delta u - u, \quad -\pi < x, y < \pi, \quad t \in [0, T]. \\ u(x, y, 0) &= \cos(x)\cos(y). \end{aligned} \tag{31}$$

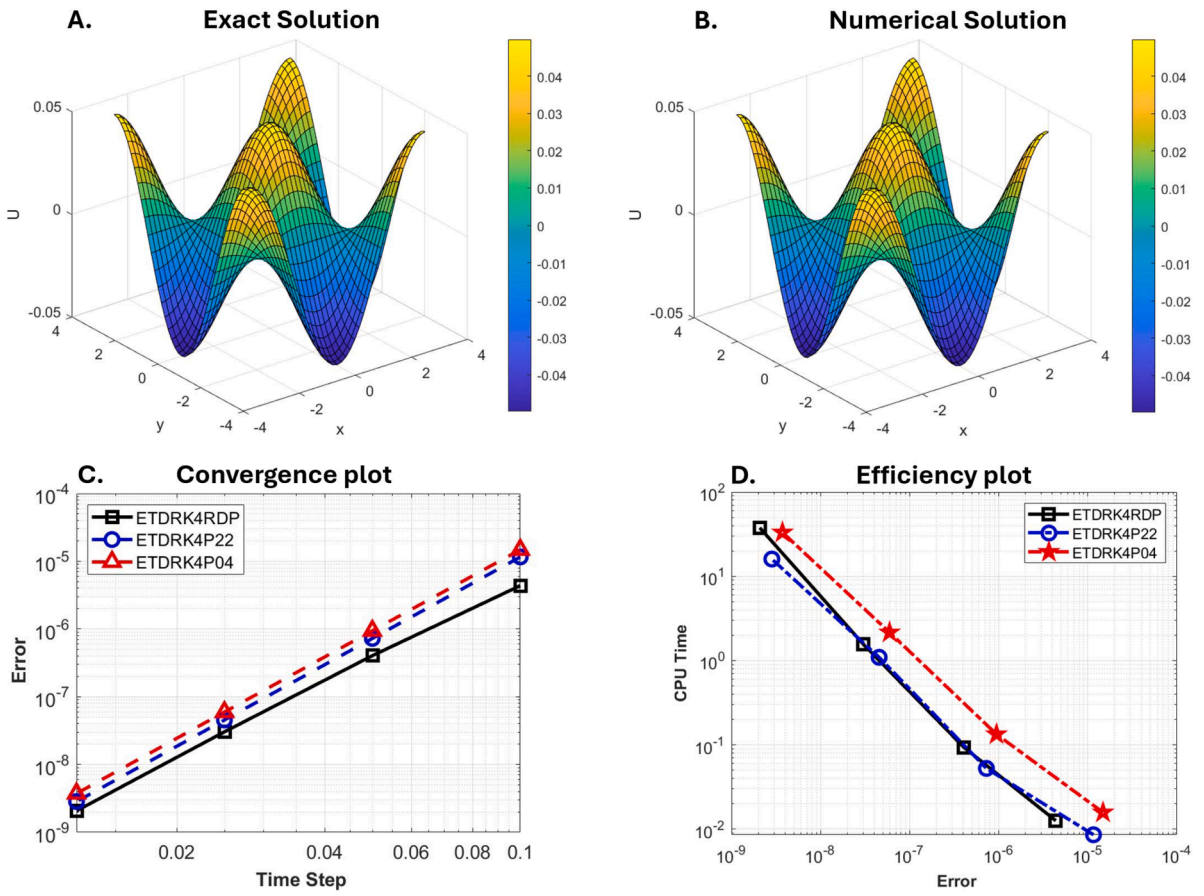

Fig. 4. Convergence and efficiency plots for Example 4.2. (A)Exact and (B) numerical solution. (C) Convergence and (D) efficiency of the new fourth-order ETDRK4RDP scheme compared with the existing fourth-order ETDRK4P22, and ETDRK4P04 schemes.

Table 4
Convergence table showing fourth-order convergence of the new ETDRK4RDP scheme compared with the existing ETDRK4P04 and ETDRK4P22 schemes for the model test problem in Example 4.3 with $T = 1$. Table also displays the CPU time for serial implementation of all schemes in MATLAB as well as single core (1-core) and multicore implementation in Fortran.

Method	k	h	Error	p	CPU Time (Serial)	1-core	Multi-core
ETDRK4RDP	0.10	0.00625	1.2×10^{-9}	–	9.6170	1.862	0.466
	0.05	0.00625	3.1×10^{-10}	2.21	12.2440	3.530	0.884
	0.025	0.00625	3.6×10^{-11}	3.31	24.4578	6.848	1.712
	0.0125	0.00625	3.2×10^{-12}	3.50	56.1013	14.217	3.556
ETDRK4P04	0.10	0.00625	1.3×10^{-9}	–	6.3058	1.443	0.722
	0.05	0.00625	1.4×10^{-10}	3.35	9.4759	2.514	1.262
	0.025	0.00625	1.2×10^{-11}	3.48	19.1614	5.033	2.513
	0.0125	0.00625	8.9×10^{-13}	3.72	37.7672	10.091	5.053
ETDRK4P22	0.10	0.00625	4.06×10^{-1}	–	2.7541	0.687	–
	0.05	0.00625	4.03×10^{-1}	0.01	5.9302	1.384	–
	0.025	0.00625	3.63×10^{-1}	0.15	9.8251	3.198	–
	0.0125	0.00625	1.18×10^{-1}	1.00	20.2239	9.404	–

The corresponding exact solution is $u(x, y, t) = \mathrm{e}^{-3t}\cos(x)\cos(y)$. We discretize each dimension of the spatial domain with $m + 2$ nodes and set a spatial step size $h = \dfrac{2\pi}{m+1}$. The spatial discretization is done as described in Section 2. Fig. 4A and B show the exact solution and numerical solution using ETDRK4RDP. Here we see a display of how close the numerical solution is to the exact one.

Again, our empirical convergence analysis in Table 3 and convergence plot in Fig. 4C show that the the fully discrete scheme with ETDRK4RDP is fourth order accurate. Surprisingly, our scheme is more accurate than all other schemes for this test problem. Again,

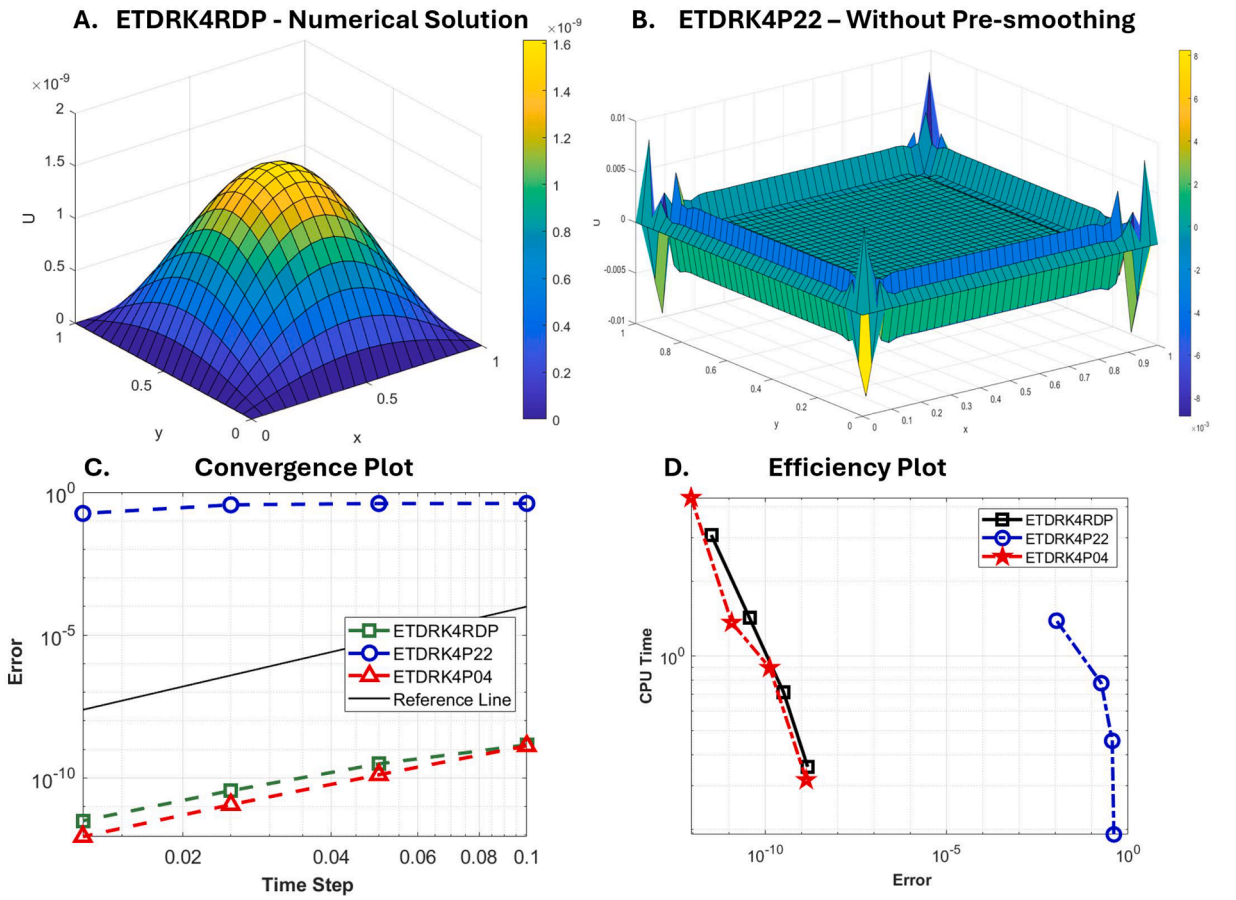

Fig. 5. Convergence and efficiency plots for Example 4.3. (A) Numerical-solution using ETDRK4RDP, (B) ETDRK4P22-solution without Pre-smoothing (C) Convergence Plot (D) Efficiency of the new fourth-order ETDRK4RDP scheme compared with the existing fourth-order ETDRK4P22, and ETDRK4P04 schemes.

our serial implementation in MATLAB has better timing than expected, given the number of linear solves performed per time step compared with the other schemes. The speed up obtained from a serial implementation in Fortran is not as dramatic as was observed in the previous test problem with speed up of 1.5X, 1.2X and 1.1X for ETDRK4RDP, ETDRK4P04 and ETDRK4P22 respectively. As can be observed from the last two columns of Table 3, parallelization in Fortran results in a two times speed in the CPU times for ETDRK4RDP and ETDRK4P04 (compared with their serial implementation on Fortran). Comparatively, the parallelized ETDRK4RDP is twice as fast as ETDRK4P04 and 1.8 times faster than ETDRK4P22 at the finest temporal resolution (k = 0.0125).

4.3. Nonlinear problem with mismatched initial and boundary data

Next, we consider a nonlinear reaction-diffusion equation with homogeneous Dirichlet conditions set at the boundaries. The model problem is the two-dimensional analogue of the one-dimensional reaction-diffusion equation with Michaelis-Menten enzyme kinetics [30–32]. The model is the simplest case of enzyme kinetics, applied to an enzyme-catalyzed reaction involving the transformation of one substrate into one product [30]. It is described mathematically as,

$$\begin{aligned} \frac{\partial u}{\partial t} &= d\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) - \frac{u}{(1+u)}, \quad 0 < x, y < 1, \quad t > 0. \\ u(x, y, 0) &= 1, \quad 0 \le x, y \le 1. \end{aligned} \tag{32}$$

where the diffusion coefficient $d = 1$. The problem has no known exact solution. We observe that the initial condition is discontinuous at the boundary where the Dirichlet boundary conditions are applied. Such mismatches of the initial and boundary conditions are known to generate spurious oscillations, which, if not properly damped at the initial stages of the simulation, can significantly affect the accuracy of solutions [13]. Without pre-smoothing, Padé schemes which are not L-acceptable (ie. ETDRK4P22) fail to damp out spurious oscillations (See Fig. 5B), and struggle to maintain fourth-order accuracy (Table 4 and Fig. 5D). This phenomenon is

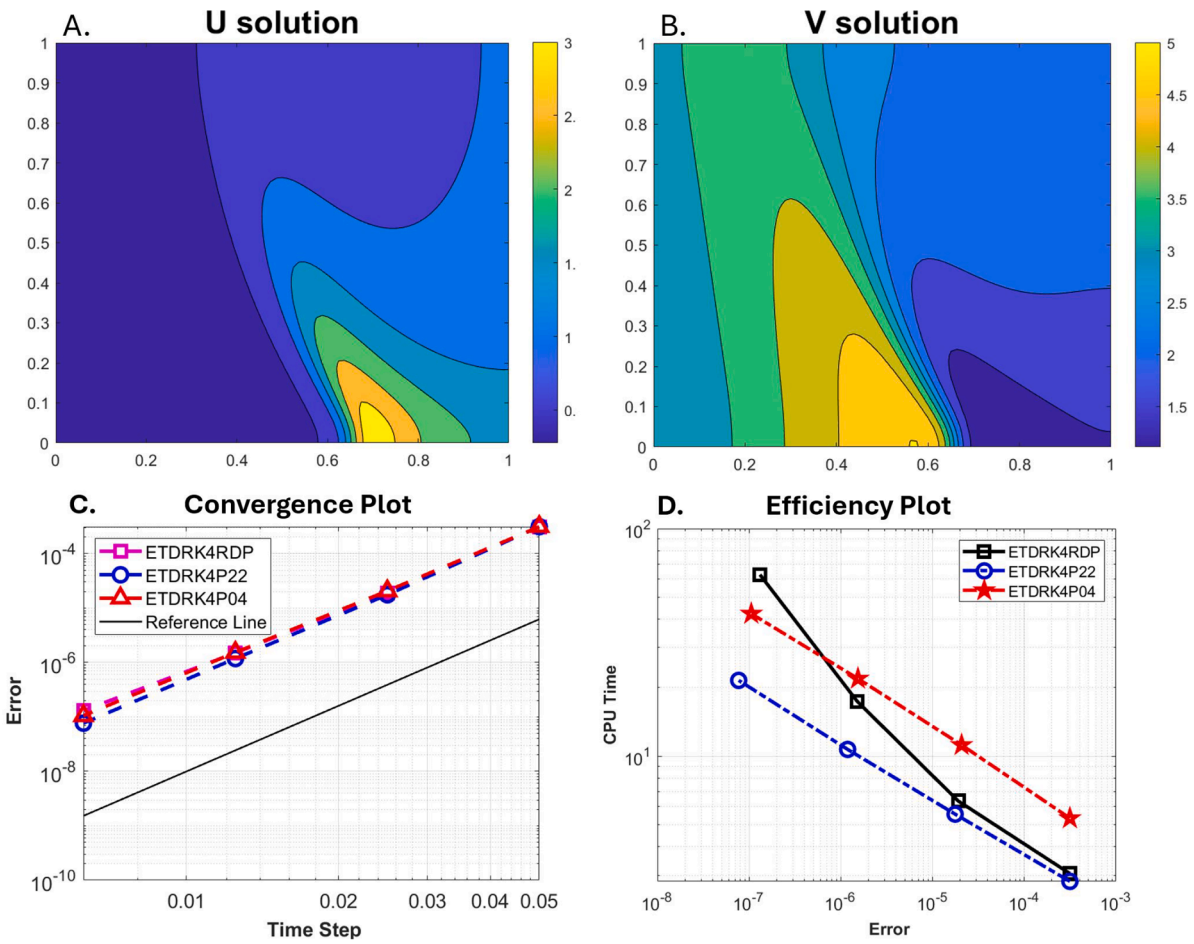

Fig. 6. Convergence and efficiency plots for Example 4.4. (A) U-solution and (B) V-Solution (C) Convergence Plot (D) Efficiency of the new fourth-order ETDRK4RDP scheme compared with the existing fourth-order ETDRK4P22 and ETDRK4P04 schemes.

also established in Asante-Asamani et al. [13]. The L-acceptability of the proposed scheme permits the smoothing of the boundary (see Fig. 5A), ensuring that the scheme converges with fourth-order accuracy (Table 4). In Fig. 5C, we see the inconsistency in the convergence results of the ETDRK4P22 without pre-smoothing, whereas the L-acceptable schemes all show better performance.

As can be observed from Table 4 and Fig. 5C, the accuracy of ETDRK4RDP is comparable to ETDRK4P04 but ETDRK4P04 is 1.5 times faster when both schemes are implemented in MATLAB. Serial implementation in Fortran results in a four times speed up in both schemes with a 2.2 times speed up for ETDRK4P22 (Table 4). Parallelization results in a 3.5 times speed up for ETDRK4RDP and a two times speed up for ETDRK4P04, compared to their serial implementations in Fortran. Comparing performance across schemes, we observe that the parallelized ETDRK4RDP, at the finest temporal resolution, is 1.5 times faster than the parallelized ETDRK4P04 and 2.7 times faster than ETDRK4P22.

4.4. The Brusselator model in 2D: A system of reaction-diffusion equations

Finally, we investigate the performance of the proposed scheme ETDRK4RDP in simulating a system of non-linear reaction-diffusion equations, with the Brusselator model [33] as a test case. It is a mathematical model of chemical reaction dynamics developed by Ilya Prigogine and colleagues in the 1960s. The model is a nonlinear two-component reaction system that exhibits complex behavior, including oscillations, spatial patterns, and chaos. The nonlinearity in the model coupled with oscillating dynamics, makes numerical computations challenging. The model is given by

$$u_t = \epsilon_1 \Delta u + A + u^2 v - (B+1)u,$$
$$v_t = \epsilon_2 \Delta v + Bu - u^2 v,$$

Table 5
Convergence table showing fourth-order convergence of the new ETDRK4RDP scheme compared with the existing ETDRK4P04 and ETDRK4P22 schemes for the model test problem in Example 4.4 with $T = 1$. Table also displays the CPU time for serial implementation of all schemes in MATLAB as well as single core (1-core) and multicore implementation in Fortran.

Method	k	h	Error	p	CPU Time (Serial)	1-core	Multi-core
ETDRK4RDP	0.05	0.0125	3.14×10^{-4}	–	3.060	0.832	0.209
	0.025	0.0125	1.91×10^{-5}	4.04	6.080	1.669	0.417
	0.0125	0.0125	1.51×10^{-6}	3.76	12.925	3.276	0.819
	0.00625	0.0125	1.31×10^{-7}	3.83	29.769	6.579	1.646
ETDRK4P04	0.05	0.0125	3.14×10^{-4}	–	5.811	1.992	1.001
	0.025	0.0125	2.07×10^{-5}	3.92	10.843	3.925	1.962
	0.0125	0.0125	1.53×10^{-6}	3.75	22.212	7.883	3.942
	0.00625	0.0125	1.06×10^{-7}	3.86	47.161	15.788	7.893
ETDRK4P22	0.05	0.0125	3.81×10^{-4}	–	1.348	0.859	–
	0.025	0.0125	1.72×10^{-5}	4.16	2.562	1.745	–
	0.0125	0.0125	1.54×10^{-6}	3.89	5.535	3.459	–
	0.00625	0.0125	1.32×10^{-8}	3.95	11.087	6.898	–

with, $t \in (0, 2]$, diffusion coefficients $\epsilon_1 = \epsilon_2 = 2 \cdot 10^{-3}$, and chemical parameters $A = 1$, $B = 3.4$ in 2D. Homogeneous Neumann boundary conditions are imposed at the boundary of the domain. The initial conditions are

$$u(x, y, 0) = \frac{1}{2} + y, \quad v(x, y, 0) = 1 + 5x, \qquad 0 < x, y < 1.$$

The problem has no exact solution hence we consider investigating the order and accuracy with a fine grid solution. The spatial derivatives are discretized as described in Section 2. The numerical solutions, convergence, and efficiency plots are shown in Fig. 6. The numerical solution profiles in U and V shown in Fig. 6 A and B, display the oscillating dynamics and spatial patterns of the model at the final time. As shown in Table 5 and Fig. 6C, the proposed ETDRK4RDP scheme converges with fourth-order accuracy, as do all other schemes under consideration, and demonstrates comparable error. Surprisingly ETDRK4RDP is about 1.5 times faster than ETDRK4P04 when implemented in serial using MATLAB. It is however slower than ETDRK4P22 (Table 6). Serial implementation in Fortran results in a 4.6 times speed up in CPU time for ETDRK4RDP, 3 times speed up for ETDRK4P04 and 1.6 times speed up for ETDRK4P22. Parallelization further speeds up ETDRK4RDP by a factor of four and ETDRK4P04 by a factor of two, compared with their serial implementations in Fortran (Table 5, last two columns). Comparing performance across schemes, it can be observed from Table 5 that ETDRK4RDP is five times faster than ETDRK4P04 and four times faster than ETDRK4P22. This is remarkable, and shows a clear advantage in computational speed of ETDRK4RDP over existing fourth order ETD Runge-Kutta schemes with Padé rational approximations, when solving systems of reaction-diffusion equation.

5. Conclusion

In this article, we have developed a fourth-order, L-stable exponential time differencing scheme of Runge-Kutta type (ETDRK4RDP) which uses a real and distinct poles (RDP) rational function to approximate the matrix exponential. A fully discrete scheme, with a fourth-order finite difference scheme in space and ETDRK4RDP in time, is empirically verified to converge with fourth-order accuracy using two linear problems with exact solutions. Additionally, the fourth-order convergence of the time discretization scheme is verified using two non-linear problems.

We compared the performance of the new scheme, in terms of accuracy and CPU time, with the existing fourth-order, A-stable ETDRK4P22 and fourth-order, L-stable ETDRK4P04 scheme. As expected, the serial implementation of ETDRK4RDP demonstrates superior accuracy over the A-stable ETDRK4P22 scheme for non-smooth problems and comparable accuracy with the L-stable ETDRK4P04 for both smooth and non-smooth problems. The serial implementation of the scheme in MATLAB was however less computationally efficient than ETDRK4P22 and ETDRK4P04 for most of the test problems considered. This is primarily because ETDRK4RDP requires 16 linear solves per time step (four independent solves per stage) compared with 8 linear solves for ETDRK4P04 and 4 for ETDRK4P22.

We evaluated the comparative performance of parallel versions of the new scheme over parallel versions of the competing ETDRK4P04 and ETDRK4P22 schemes. Whereas ETDRK4P04 has been previously introduced with a parallel algorithm, our work is the first direct investigation of the performance of this parallel algorithm in a multiprocessor environment. Across all test problems, we observed that parallelization in Fortran significantly improved the CPU time for ETDRK4RDP and ETDRK4P04, with the most significant improvement observed for ETDRK4RDP. Our scheme was consistently faster than all competing schemes across all the test problems considered, with the greatest advantage observed for the Brusselator problem, where the parallelized ETDRK4RDP achieved up to five times speed up over existing schemes. We attribute the computational advantage of ETDRK4RDP to the absence of complex arithmetic within its linear systems, a consequence of the real poles of the RDP rational function. By distributing the work of solving four real linear systems per stage over four processors, we reduce the workload to one linear solve per stage without of complex arithmetic, whereas the other methods perform one linear solve involving complex arithmetic.

We have focused primarily on test problems with either Dirichlet or Neumann boundary conditions. We did not consider periodic boundary conditions in this work as there are more efficient ways to compute the matrix exponential in this case, say via Fourier spectral methods, as was done in the implementation of the ETDRK4 scheme in Cox and Matthews [14]. However, the L-stability of our scheme and the use of Fourier transformation to solve our linear systems, as was done in Asante-Asamani et al. [29], could provide both accuracy and speed advantages over ETDRK4 for nonsmooth problems. This will be the subject of future work along side the development of a dimensional splitting version of the ETDRK4RDP scheme to further improve computational efficiency for multidimensional problems.

Data availability

I have shared a link to my code in the manuscript

Funding

The work is supported by funding to EAA from the NSF/NIGMS Mathematical Biology Program (# R01GM152810). The content of this work is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health .

Declaration of competing interests

I have nothing to declare

Acknowledgment

The authors would like to thank the editor and three referees whose suggestions lead to a considerably improved version of the initial manuscript.

References

[1] R.J. Leveque, Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems, 2007.
[2] R. Tyson, S. Lubkin, J.D. Murray, Model and analysis of chemotactic bacterial patterns in a liquid medium, J. Math. Biol. 38 (1999) 359–375.
[3] J.K. Kamel, Chemical vapor deposition/chemical vapor infiltration of pyrocarbon in porous carbon, Technical Report, University of Notre Dame, 2007.
[4] K. Zhang, Y. Zuo, H. Zhao, X. Ma, J. Gu, J. Wang, Y. Yang, C. Yao, J. Yao, Fourier neural operator for solving subsurface oil/water two-phase flow partial differential equation, SPE J. 27 (03) (2022) 1815–1830.
[5] A.R. Anderson, M.A. Chaplain, E.L. Newman, R.J. Steele, A.M. Thompson, Mathematical modelling of tumour invasion and metastasis, Comput. Math. Method. Med. 2 (2) (2000) 129–154.
[6] M.A. Chaplain, A.M. Stuart, A model mechanism for the chemotactic response of endothelial cells to tumour angiogenesis factor, J. Math. Med. Biol. 10 (3) (1993) 149–168.
[7] R.A. Gatenby, E.T. Gawlinski, A reaction-diffusion model of cancer invasion, J. Cancer Res. 56 (24) (1996) 5745–5753.
[8] H. Zhang, L. Liu, X. Qian, S. Song, Large time-stepping, delay-free, and invariant-set-preserving integrators for the viscous Cahn-Hilliard-Oono equation, J. Comput. Phy. 499 (2024) 112708.
[9] H. Zhang, H. Wang, X. Teng, A second-order, global-in-time energy stable implicit-explicit Runge-Kutta scheme for the phase field crystal equation, SIAM J. Numer. Anal. 62 (6) 2667–2697.
[10] X. Teng, H. Zhang, A third-order energy stable exponential-free Runge-Kutta framework for the nonlocal Cahn-Hilliard equation, J. Sci. Comput. 103 (3) (2025) 74.
[11] H. Wang, J. Sun, H. Zhang, X. Qian, A novel up to fourth-order equilibria-preserving and energy-stable exponential Runge-Kutta framework for gradient flows, CSIAM Trans. Appl. Math. 6 (2025) 106–147.
[12] H. Zhang, X. Qian, S. Song, Third-order accurate, large time-stepping and maximum-principle-preserving schemes for the Allen-Cahn equation, Numer. Algorith. 95 (3) (2024) 1213–1250.
[13] E. Asante-Asamani, A. Kleefeld, B.A. Wade, A fourth-order exponential time differencing scheme with dimensional splitting for non-linear reaction-diffusion systems, 2025, p. 116568.
[14] S.M. Cox, P.C. Matthews, Exponential time differencing for stiff systems, J. Comput. Phys. 176 (2) (2002) 430–455.
[15] A.-K. Kassam, L.N. Trefethen, Fourth-order time-stepping for stiff PDEs, SIAM J. Sci. Comput. 26 (4) (2005) 1214–1233.
[16] B. Kleefeld, A. Khaliq, B. Wade, An ETD Crank-Nicolson method for reaction-diffusion systems, Numer. Method. Part. Differ. Eq. 28 (4) (2012) 1309–1335.
[17] M. Yousuf, A. Khaliq, B. Kleefeld, The numerical approximation of nonlinear Black-Scholes model for exotic path-dependent American options with transaction cost, Int. J. Comput. Math. 89 (9) (2012) 1239–1254.
[18] M. Yousuf, Efficient L-stable method for parabolic problems with application to pricing American options under stochastic volatility, Appl. Math. Comput. 213 (1) (2009) 121–136.
[19] A. Khaliq, J. Martin-Vaquero, B. Wade, M. Yousuf, Smoothing schemes for reaction-diffusion systems with nonsmooth data, J. Comput. Appl. Math. 223 (1) (2009) 374–386.
[20] E. Asante-Asamani, A. Khaliq, B.A. Wade, A real distinct poles exponential time differencing scheme for reaction-diffusion systems, J. Comput. Appl. Math. 299 (2016) 24–34.
[21] D.A. Voss, A.Q.M. Khaliq, Time-stepping algorithms for semidiscretized linear parabolic PDEs based on rational approximants with distinct real poles, Adv. Comput. Math. 6 (1996) 353–363.
[22] J.H. Mathews, K.D. Fink, et al,Numerical Methods Using MATLAB, 4, Pearson Prentice Hall Upper Saddle River, 2004. NJ .
[23] F. Gibou, R. Fedkiw, A fourth-order accurate discretization for the Laplace and heat equations on arbitrary domains, with applications to the Stefan problem, J. Comput. Phys. 202 (2) (2005) 577–601.
[24] W. Hundsdorfer, J.G. Verwer, Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations, 33, Springer Science & Business Media, 2013.
[25] P.M. Gauthier, Lectures on Several Complex Variables, Springer, 2014.
[26] X. Huang, Z. Liu, C. Wu, Derivative and higher-order cauchy integral formula of matrix functions, Open Math. 19 (1) (2021) 1771–1778.
[27] T.A. Davis, A Suite of Sparse matrix packages, 2025. https://github.com/DrTimothyAldenDavis/SuiteSparse/releases.

[28] L. Hanyk, UMFPACK Fortran Interface, 2014. https://geo.mff.cuni.cz/lh/Fortran/UMFPACK/.
[29] E. Asante-Asamani, A. Kleefeld, B.A. Wade, A second-order exponential time differencing scheme for non-linear reaction-diffusion systems with dimensional splitting, J. Comput. Phys. 415 (2020) 109490.
[30] J. Müller, C. Kuttler, Methods and models in mathematical biology, in: Lect. Notes Math., 2015.
[31] H.P. Bhatt, A.Q.M. Khaliq, The locally extrapolated exponential time differencing lod scheme for multidimensional reaction-diffusion systems, J. Comput. Appl. Math. 285 (2015) 256–278.
[32] Y. Cherruault, M. Choubane, J. Valleton, J. Vincent, Stability and asymptotic behavior of a numerical solution corresponding to a diffusion-reaction equation solved by a finite difference scheme (Crank-Nicolson), Comput. Math. Appl. 20 (11) (1990) 37–46.
[33] P.A. Zegeling, H. Kok, Adaptive moving mesh computations for reaction-diffusion systems, J. Comput. Appl. Math. 168 (1–2) (2004) 519–528.