A second-order exponential time differencing scheme for non-linear reaction-diffusion systems with dimensional splitting

E.O. Asante-Asamani[a], A. Kleefeld[b,*], B.A. Wade[c]

[a] *Department of Mathematics, Clarkson University, Potsdam, NY 13699, USA*
[b] *Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH, 52425 Jülich, Germany*
[c] *Department of Mathematics, University of Louisiana at Lafayette, Lafayette, LA 70504-3568, USA*

Abstract

A second-order L-stable exponential time-differencing (ETD) method is developed by combining an ETD scheme with approximating the matrix exponentials by rational functions having real distinct poles (RDP), together with a dimensional splitting integrating factor technique. A variety of non-linear reaction-diffusion equations in two and three dimensions with either Dirichlet, Neumann, or periodic boundary conditions are solved with this scheme and shown to outperform a variety of other second-order implicit-explicit schemes. An additional performance boost is gained through further use of basic parallelization techniques.

Keywords: Exponential time differencing, real distinct pole, dimensional splitting, reaction-diffusion systems, matrix exponential
2000 MSC: 65M12, 65M15, 65M20, 65F60

1. Introduction

Many applications, for example in biology [50–52], pattern formation [38], or medicine [21, 59] can be modelled with a system of s time-dependent non-linear reaction-diffusion equations in d dimensions given by

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{D}\Delta\mathbf{u} + \mathbf{F}(\mathbf{u}), \qquad \mathbf{x} \in \Omega \subset \mathbb{R}^d, \quad t \in (0,T) \tag{1}$$

with appropriate initial and boundary condition for the domain Ω and the time interval $(0,T)$. The numerical computation of such systems in two or three dimensions is challenging due to its high dimensionality, coupling, stiffness of the reaction and diffusion terms, and since the function $\mathbf{F}$ may be non-linear. Additionally, spurious oscillations in the approximate solution might occur if the initial data are non-smooth or are incompatible with the boundary data.

Many algorithms have been proposed to solve such systems after discretization in space. For example, linearly implicit methods [4, 33, 64], semi-implicit methods [12], projection methods [22], stabilized explicit Runge-Kutta methods such as ROCK2 and ROCK4 [1,

*corresponding author
Email addresses: nasmano@gmail.com (E.O. Asante-Asamani), a.kleefeld@fz-juelich.de (A. Kleefeld), bruce.wade@louisiana.edu (B.A. Wade)

2], PIROCK [3], SERK, SERK2V2, and SERK2V3 [36, 37, 41, 42], its extrapolated variants [43–45], explicit Runge-Kutta-Chebyshev (RKC) [60], implicit-explicit Runge-Kutta-Chebyshev (IRKC) [58] (one of many implicit-explicit methods (IMEX) [28,56]), factorized Runge-Kutta-Chebyshev (FRKC) or Runge-Kutta with Gegenbauer polynomials [53, 54], integration factor methods [66, 69] in combination with Krylov subspace methods [14, 39], and exponential propagation iterative methods of Runge-Kutta type (EPIRK) [61] and its variants such as IMEXP [40], to mention a few.

One important family to solve the problem at hand is exponential time differencing (ETD), which uses the Duhamel principle and solves the resulting initial value problem by approximating the integral appropriately [32, 46]. A class of ETD schemes based on Runge-Kutta methods has been introduced by Cox and Matthews [17]. Since then, many researchers have considered this scheme (see for example [19, 26, 27, 30, 32, 47]). However, the challenge of how to effectively approximate the matrix exponentials remains. Kleefeld, Khaliq, and Wade [31, 35] considered an ETD Crank-Nicolson (ETD-CN) scheme; that is, the matrix exponential has been approximated through a second-order Padé-$(1,1)$ approximation. Their fully discrete scheme is of second-order and highly efficient, but it is not L-stable. Other schemes use Krylov subspace methods to approximate the matrix exponential [10] or use higher order Padé approximations [34].

To speed up computations and to decrease memory demand, various dimensional splitting techniques such as, Strang simple and Strang symmetric splitting, as well as an integrating factor method have been considered by Asante-Asamani and Wade [8] for a variety of examples with Dirichlet and Neumann boundary conditions in two dimensions as a competitive alternative to the locally one-dimensional splitting of Bhatt and Khaliq [9]. It has been observed empirically that the integrating factor approach outperforms other splitting techniques.

An extension of the work for ETD-CN with Padé-$(1,1)$ is given by Yousuf, Khaliq, and Kleefeld [67] using a Padé-$(0,2)$ approximation of the matrix exponential leading to a second-order L-stable scheme. However, one has to use complex arithmetic in the implementation due to the two complex-valued poles in the Padé approximation. To avoid this, one can use a rational (non-Padé) approximation of second-order with real distinct poles (see [65]), which has recently been applied by Asante-Asamani, Khaliq and Wade [7] to the second-order ETD scheme, named hereafter ETD-RDP. It is shown that ETD-RDP is a competitive alternative to BDF2 [64] and ETD-Padé-$(0,2)$ for a variety of examples with Dirichlet and Neumann boundary conditions in two dimensions. Further, it has been remarked by the authors that the ETD-RDP algorithm is easily parallelizable.

Contribution

In this work, we apply a splitting technique using an integrating factor (IF) approach to the second-order L-stable ETD-RDP scheme, lifting the derivation from two dimensions to d dimensions, which we call ETD-RDP-IF. We show that besides Dirichlet and Neumann boundary conditions, also the periodic boundary condition case can be handled. Further, the algorithm can deal with non-smooth boundary conditions as well as with mismatching boundary conditions. Additionally, we explain how to solve the sparse linear systems in higher dimensions efficiently using an extension of the Thomas algorithm for Dirichlet and Neumann boundary conditions and using the Fourier transformation for periodic boundary conditions. Further, we describe how to implement a parallelization strategy in Fortran, and

show that we developed a second-order L-stable method that is able to outperform other second-order IMEX schemes as well as ETD-RDP.

Organization of the paper

In Section 2, the mathematical model of a system consisting of s non-linear reaction-diffusion equations is detailed. Section 3 illustrates how this system is discretized in d-dimensional space. The dimensional splitting using the integrating factor approach is given in Section 4. The discretization with respect to time using exponential time differencing is described in Section 5. In Section 6, we develop the approximation of the matrix exponential. The implementation and parallelization of the fully discrete ETD-RDP-IF scheme is given in Section 7. Extensive numerical results in two and three dimensions with Dirichlet, Neumann, and periodic boundary conditions are given in Section 8, which show the second-order accuracy and the efficiency of the new ETD-RDP-IF scheme. A short summary and a conclusion together with an outlook for possible future research is given in Section 9.

2. Time-dependent non-linear reaction-diffusion equation

The mathematical model of a system of s time-dependent non-linear reaction-diffusion equations in d dimensions is given as

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{D}\Delta\mathbf{u} + \mathbf{F}(\mathbf{u}), \qquad \mathbf{x} \in \Omega \subset \mathbb{R}^d, \quad t \in (0,T) \tag{2}$$

where the vector-valued concentration function $\mathbf{u} = \mathbf{u}(\mathbf{x},t) = (u_1(\mathbf{x},t),\ldots,u_s(\mathbf{x},t))^\top$ depends on the spatial variable $\mathbf{x} = (x_1,\ldots,x_d) \in \Omega$ and the temporal variable $t \in (0,T)$. Here, $\Omega \subset \mathbb{R}^d$ is a bounded domain with Lipschitz continuous boundary. Each component of the vector-valued function $\mathbf{F}(\mathbf{u}) = (f_1(\mathbf{x}),\ldots,f_s(\mathbf{x}))^\top$ is assumed to be a sufficiently smooth and bounded function which may be non-linear. The diagonal matrix $\mathbf{D} = \operatorname{diag}(D_1,\ldots,D_s) \in \mathbb{R}^{s\times s}$ contains given constant diffusion coefficients and $\Delta\mathbf{u}$ is the d-dimensional Laplacian taken component-wise. The boundary conditions on $\partial\Omega$ are either homogeneous Dirichlet $\mathbf{u}(\cdot\,,t) = 0$, homogeneous Neumann $\frac{\partial}{\partial\nu}\mathbf{u}(\cdot\,,t) = 0$ where ν denotes the exterior normal of Ω, or periodic boundary conditions. The initial condition is given by $\mathbf{u}(\mathbf{x},0) = \mathbf{u}_0(\mathbf{x})$ for $\mathbf{x} \in \Omega$.

3. Discretizing in space

As a first step, we discretize the spatial domain assuming an equidistant grid size h along all d directions. The parameter p denotes the number of spatial grid points along each direction. Then, the second-order derivatives within the expression $-\mathbf{D}\Delta\mathbf{u}$ are discretized by centered differences with second-order accuracy. As a result, one obtains for (2) the system of non-linear ordinary differential equations

$$\frac{\partial \mathbf{U}}{\partial t} + \mathbf{A}\mathbf{U} = \mathbf{f}(\mathbf{U}), \quad \mathbf{U}(0) = \mathbf{U}_0 \tag{3}$$

where $\mathbf{U}$ is a vector of size $m = s\cdot p^d$, $\mathbf{A}$ is a matrix of size $s\cdot p^d$ times $s\cdot p^d$, $\mathbf{f}(\mathbf{U})$ is a vector-valued function of size $s\cdot p^d$ ($\mathbf{F}(\mathbf{u})$ evaluated at the p^d spatial points), and the initial condition $\mathbf{U}_0$ is a vector of size $s\cdot p^d$ ($\mathbf{u}_0(\mathbf{x})$ evaluated at the p^d spatial points). The entries

of the matrix $\mathbf{A}$ depend on the particular boundary condition. Three cases are given in the following two examples.

Example 1. *The one-dimensional approximation of the* $-\Delta\mathbf{u}$ *is given by*

$$\mathbf{B}_p = -\frac{1}{h^2}\begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{pmatrix} \in \mathbb{R}^{p\times p}, \qquad h = \frac{1}{p+1},$$

$$\mathbf{B}_p = -\frac{1}{h^2}\begin{pmatrix} -2 & 2 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 2 & -2 \end{pmatrix} \in \mathbb{R}^{p\times p}, \qquad h = \frac{1}{p-1},$$

$$\mathbf{B}_p = -\frac{1}{h^2}\begin{pmatrix} -2 & 1 & & & 1 \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ 1 & & & 1 & -2 \end{pmatrix} \in \mathbb{R}^{p\times p}, \qquad h = \frac{1}{p},$$

for homogeneous Dirichlet, homogeneous Neumann, and periodic boundary conditions, respectively. Hence, in one dimension $\mathbf{A}$ *in (3) approximating* $-\mathbf{D}\Delta\mathbf{u}$ *is given by* $\mathbf{A} = \mathbf{A}_1$ *with* $\mathbf{A}_1 = \mathbf{B}_p \otimes \mathbf{D} \in \mathbb{R}^{(s\cdot p)\times(s\cdot p)}$*. Here,* $\otimes$ *denotes the Kronecker product (see [63] for the definition and its properties).*

Example 2. *In two dimensions,* $\mathbf{A}$ *in (3) is given by* $\mathbf{A} = \mathbf{A}_1 + \mathbf{A}_2 \in \mathbb{R}^{(s\cdot p^2)\times(s\cdot p^2)}$ *with* $\mathbf{A}_1 = \mathbf{I}_p \otimes \mathbf{B}_p \otimes \mathbf{D}$*,* $\mathbf{A}_2 = \mathbf{B}_p \otimes \mathbf{I}_p \otimes \mathbf{D}$*, where* $\mathbf{I}_p$ *denotes the identity matrix of size* p *(see also pp. 1345–1346 in [8] for the Dirichlet and Neumann case). The extension to three dimensions reads* $\mathbf{A} = \mathbf{A}_1 + \mathbf{A}_2 + \mathbf{A}_3 \in \mathbb{R}^{(s\cdot p^3)\times(s\cdot p^3)}$ *with*

$$\begin{aligned} \mathbf{A}_1 &= \mathbf{I}_p \otimes \mathbf{I}_p \otimes \mathbf{B}_p \otimes \mathbf{D}, \qquad \mathbf{A}_2 = \mathbf{I}_p \otimes \mathbf{B}_p \otimes \mathbf{I}_p \otimes \mathbf{D}, \\ \mathbf{A}_3 &= \mathbf{B}_p \otimes \mathbf{I}_p \otimes \mathbf{I}_p \otimes \mathbf{D}. \end{aligned} \tag{4}$$

Note that the extension to four dimensions or even higher follows an obvious pattern. Precisely, we have $\mathbf{A} = \sum_{i=1}^{d} \mathbf{A}_i$ *with*

$$\mathbf{A}_i = \underbrace{\mathbf{I}_p \otimes \cdots \otimes \mathbf{I}_p}_{(d-i)\,times} \otimes \underbrace{\mathbf{B}_p}_{pos\ d-i+1} \otimes \underbrace{\mathbf{I}_p \otimes \cdots \otimes \mathbf{I}_p}_{(i-1)\,times} \otimes \mathbf{D} \in \mathbb{R}^{(s\cdot p^d)\times(s\cdot p^d)}. \tag{5}$$

4. Dimensional splitting

In this section, we explain how to apply a dimensional splitting technique to solve (3). We first consider the three-dimensional case. Precisely, we here use $\mathbf{A} = \mathbf{A}_1 + \mathbf{A}_2 + \mathbf{A}_3$. Consider a new time-dependent function of the form $\mathbf{V} = \mathrm{e}^{\mathbf{A}_1 t}\mathbf{U}$ (here the term $\mathrm{e}^{\mathbf{A}_1 t}$ is called an integrating factor). The matrix exponential is described in [49]. Then, the derivative

with respect to time t of $\mathbf{V} = \mathrm{e}^{\mathbf{A}_1 t}\mathbf{U}$ is given by

$$\mathbf{V}_t = \mathrm{e}^{\mathbf{A}_1 t}\mathbf{U}_t + \mathbf{A}_1 \mathrm{e}^{\mathbf{A}_1 t}\mathbf{U} \tag{6}$$

recalling that $\mathbf{U}$ depends on t as well. Inserting (3) into (6) yields

$$\begin{aligned}
\mathbf{V}_t &= \mathrm{e}^{\mathbf{A}_1 t}\left(\mathbf{f}(\mathbf{U}) - \mathbf{A}\mathbf{U}\right) + \mathbf{A}_1 \mathrm{e}^{\mathbf{A}_1 t}\mathbf{U} = \mathrm{e}^{\mathbf{A}_1 t}\mathbf{f}(\mathbf{U}) - \mathrm{e}^{\mathbf{A}_1 t}\mathbf{A}\mathbf{U} + \mathbf{A}_1 \mathrm{e}^{\mathbf{A}_1 t}\mathbf{U} \\
&= \mathrm{e}^{\mathbf{A}_1 t}\mathbf{f}(\mathbf{U}) - \mathbf{A}\mathrm{e}^{\mathbf{A}_1 t}\mathbf{U} + \mathbf{A}_1 \mathrm{e}^{\mathbf{A}_1 t}\mathbf{U} = \mathrm{e}^{\mathbf{A}_1 t}\mathbf{f}(\mathbf{U}) - \left(\mathbf{A}_2 + \mathbf{A}_3\right)\mathrm{e}^{\mathbf{A}_1 t}\mathbf{U} \\
&= \mathrm{e}^{\mathbf{A}_1 t}\mathbf{f}(\mathrm{e}^{-\mathbf{A}_1 t}\mathbf{V}) - \left(\mathbf{A}_2 + \mathbf{A}_3\right)\mathbf{V}
\end{aligned}$$

where we used in the third step that $\mathbf{A}$ and $\mathbf{A}_1$ commute, hence it follows that $\mathbf{A}$ and $\mathrm{e}^{\mathbf{A}_1 t}$ commute. This is proved below. Hence, we obtain

$$\mathbf{V}_t + \left(\mathbf{A}_2 + \mathbf{A}_3\right)\mathbf{V} = \mathbf{g}(\mathbf{V}), \quad \mathbf{V}(0) = \mathbf{U}_0 \tag{7}$$

with $\mathbf{g}(\mathbf{V}) = \mathrm{e}^{\mathbf{A}_1 t}\mathbf{f}(\mathrm{e}^{-\mathbf{A}_1 t}\mathbf{V})$ and the initial condition $\mathbf{V}(0) = \mathrm{e}^{\mathbf{A}_1 0}\mathbf{U}_0 = \mathbf{I}_m\mathbf{U}_0 = \mathbf{U}_0$ with $m = s\cdot p^3$. Similarly, we now define $\mathbf{W} = \mathrm{e}^{\mathbf{A}_2 t}\mathbf{V}$, compute $\mathbf{W}_t$, insert $\mathbf{V}_t$ from (7) into $\mathbf{W}_t$, and use the fact that $\mathrm{e}^{\mathbf{A}_2}$ commutes with $\mathbf{A}_2 + \mathbf{A}_3$ which is true since $\mathbf{A}_2$ and $\mathbf{A}_2 + \mathbf{A}_3$ commute as shown below. We obtain

$$\mathbf{W}_t + \mathbf{A}_3\mathbf{W} = \mathbf{h}(\mathbf{W}), \quad \mathbf{W}(0) = \mathbf{U}_0, \tag{8}$$

where $\mathbf{h}(\mathbf{W}) = \mathrm{e}^{\mathbf{A}_2 t}\mathbf{g}(\mathrm{e}^{-\mathbf{A}_2 t}\mathbf{W})$. Once $\mathbf{W}(t)$ is found, we can obtain $\mathbf{V}(t)$ using the formula $\mathbf{V}(t) = \mathrm{e}^{-\mathbf{A}_2 t}\mathbf{W}(t)$ and $\mathbf{U}(t)$ via $\mathbf{U}(t) = \mathrm{e}^{-\mathbf{A}_1 t}\mathbf{V}(t)$. Hence, the original three-dimensional problem is converted to a problem involving only one-dimensional components.

Remark 1. *The solution of (3) with $\mathbf{A} = \sum_{i=1}^{d}\mathbf{A}_i$ where $\mathbf{A}_i$ is given by (5) can be obtained via splitting through*

$$\mathbf{Z}_t + \mathbf{A}_d\mathbf{Z} = \mathbf{k}(\mathbf{Z}), \quad \mathbf{Z}(0) = \mathbf{U}_0,$$

where $\mathbf{k}(\mathbf{Z}) = \left(\mathrm{e}^{\mathbf{A}_{d-1}t} \bullet \ldots \bullet \mathrm{e}^{\mathbf{A}_1 t}\right)\mathbf{F}(\left(\mathrm{e}^{-\mathbf{A}_1 t} \bullet \ldots \bullet \mathrm{e}^{-\mathbf{A}_{d-1}t}\right)\mathbf{Z})$, where $\bullet$ denotes matrix multiplication (whenever needed for clarity). Then, the function $\mathbf{U}(t)$ is given by the expression $\mathbf{U}(t) = \left(\mathrm{e}^{-\mathbf{A}_1 t} \bullet \ldots \bullet \mathrm{e}^{-\mathbf{A}_{d-1}t}\right)\mathbf{Z}(t)$ where it is assumed that the operators commute accordingly. We will see later that this is the case for our examples.

All that is left to show is that $\mathbf{A}$ and $\mathbf{A}_1$ commute as well as that $\mathbf{A}_2$ and $\mathbf{A}_2 + \mathbf{A}_3$ commute for the three-dimensional case.

Lemma 1. *Let $\mathbf{A} = \mathbf{A}_1 + \mathbf{A}_2 + \mathbf{A}_3$, where $\mathbf{A}_i$ is defined in (4). Then $\mathbf{A}$ and $\mathbf{A}_1$ commute. Also $\mathbf{A}_2$ and $\mathbf{A}_2 + \mathbf{A}_3$ commute.*

Proof. It suffices to show that $\mathbf{A}_2\mathbf{A}_1 = \mathbf{A}_1\mathbf{A}_2$, $\mathbf{A}_1\mathbf{A}_3 = \mathbf{A}_3\mathbf{A}_1$, and $\mathbf{A}_2\mathbf{A}_3 = \mathbf{A}_3\mathbf{A}_2$. Using the product property of the Kronecker product, we obtain

$$\mathbf{A}_1\mathbf{A}_2 = \mathbf{I}_p\mathbf{I}_p \otimes \mathbf{I}_p\mathbf{B}_p \otimes \mathbf{B}_p\mathbf{I}_p \otimes \mathbf{D}^2 = \mathbf{I}_p \otimes \mathbf{B}_p \otimes \mathbf{B}_p \otimes \mathbf{D}^2$$

and similarly

$$\mathbf{A}_2\mathbf{A}_1 = \mathbf{I}_p\mathbf{I}_p \otimes \mathbf{I}_p\mathbf{B}_p \otimes \mathbf{B}_p\mathbf{I}_p \otimes \mathbf{D}^2 = \mathbf{I}_p \otimes \mathbf{B}_p \otimes \mathbf{B}_p \otimes \mathbf{D}^2.$$

Similarly, it can be shown that $\mathbf{A}_1\mathbf{A}_3 = \mathbf{A}_3\mathbf{A}_1$ and $\mathbf{A}_2\mathbf{A}_3 = \mathbf{A}_3\mathbf{A}_2$ holds □

Lemma 2. *For any $i, j \in \{1, 2 \ldots, d\}$, the matrices $\mathbf{A}_i$ commute with $\mathbf{A}_j$, where they are given in (5).*

Proof. We have on the one hand

$$\begin{aligned}
&\mathbf{A}_i\mathbf{A}_j \\
&= \mathbf{I}_p \otimes \cdots \otimes \mathbf{I}_p \otimes \underbrace{\mathbf{B}_p}_{\text{pos } d-\min(i,j)+1} \otimes \mathbf{I}_p \otimes \cdots \otimes \mathbf{I}_p \otimes \underbrace{\mathbf{B}_p}_{\text{pos } d-\max(i,j)+1} \otimes \mathbf{I}_p \otimes \cdots \otimes \mathbf{I}_p \otimes \mathbf{D}^2 \\
&= \mathbf{I}_p \otimes \cdots \otimes \mathbf{I}_p \otimes \underbrace{\mathbf{B}_p}_{\text{pos } d-\min(j,i)+1} \otimes \mathbf{I}_p \otimes \cdots \otimes \mathbf{I}_p \otimes \underbrace{\mathbf{B}_p}_{\text{pos } d-\max(j,i)+1} \otimes \mathbf{I}_p \otimes \cdots \otimes \mathbf{I}_p \otimes \mathbf{D}^2
\end{aligned}$$

and on the other hand, we have

$$\begin{aligned}
&\mathbf{A}_j\mathbf{A}_i \\
&= \mathbf{I}_p \otimes \cdots \otimes \mathbf{I}_p \otimes \underbrace{\mathbf{B}_p}_{\text{pos } d-\min(j,i)+1} \otimes \mathbf{I}_p \otimes \cdots \otimes \mathbf{I}_p \otimes \underbrace{\mathbf{B}_p}_{\text{pos } d-\max(j,i)+1} \otimes \mathbf{I}_p \otimes \cdots \otimes \mathbf{I}_p \otimes \mathbf{D}^2
\end{aligned}$$

which finishes the proof. □

5. Discretization in time

What remains is to discretize (8) with respect to time. We now focus on the three-dimensional case, but the results easily extend to higher dimensions. The case for two or even one-dimension is then a special case of the results for the three-dimensional case. Consider solving (8) using Duhamel's principle (see Theorem 2.5.4 in [70]); that is

$$\mathbf{W}(t) = \mathrm{e}^{-\mathbf{A}_3 t}\mathbf{W}(0) + \int_0^t \mathrm{e}^{-\mathbf{A}_3(t-s)}\mathbf{h}(\mathbf{W}(s))\,\mathrm{d}s$$

which satisfies the semi-discrete equation in the interval $[t_n, t_{n+1}]$ with $k = t_{n+1} - t_n$ the positive time step. Thus,

$$\mathbf{W}(t_{n+1}) = \mathrm{e}^{-k\mathbf{A}_3}\mathbf{W}(t_n) + \int_{t_n}^{t_{n+1}} \mathrm{e}^{-\mathbf{A}_3(t_{n+1}-s)}\mathbf{h}(\mathbf{W}(s))\,\mathrm{d}s\,.$$

and, upon change of variables $s = t_n + k\tau$ for $\tau \in [0, 1]$, we have

$$\mathbf{W}(t_{n+1}) = \mathrm{e}^{-k\mathbf{A}_3}\mathbf{W}(t_n) + k\int_0^1 \mathrm{e}^{-k\mathbf{A}_3(1-\tau)}\mathbf{h}(\mathbf{W}(t_n + k\tau))\,\mathrm{d}\tau\,. \tag{9}$$

Set $\widehat{\mathbf{h}}(\tau) = \mathbf{h}(\mathbf{W}(t_n + k\tau))$ and note that the linear approximation of the non-linear function $\widehat{\mathbf{h}}(\tau)$ is given by $\widehat{\mathbf{h}}(0) + (\widehat{\mathbf{h}}(1) - \widehat{\mathbf{h}}(0))\tau$ for $\tau \in [0, 1]$. Substituting this into (9) yields

$$\mathbf{W}(t_{n+1}) = \mathrm{e}^{-k\mathbf{A}_3}\mathbf{W}(t_n) + k\int_0^1 \mathrm{e}^{-k\mathbf{A}_3(1-\tau)}\,\mathrm{d}\tau\widehat{\mathbf{h}}(0) + k\int_0^1 \mathrm{e}^{-k\mathbf{A}_3(1-\tau)}\tau\,\mathrm{d}\tau(\widehat{\mathbf{h}}(1) - \widehat{\mathbf{h}}(0)). \tag{10}$$

Now, we rewrite the integrals in (10) using the following lemma. The detailed proof can be found in [6, Lemma 2.1.1].

Lemma 3. *Let* $\mathbf{A}$ *be an arbitrary non-singular square matrix, then*

$$\begin{aligned} k\int_0^1 \mathrm{e}^{-k\mathbf{A}(1-\tau)}\,\mathrm{d}\tau &= \mathbf{A}^{-1}(\mathbf{I}-\mathrm{e}^{-k\mathbf{A}})\,, \\ k\int_0^1 \mathrm{e}^{-k\mathbf{A}(1-\tau)}\tau\,\mathrm{d}\tau &= k^{-1}\mathbf{A}^{-2}(k\mathbf{A}-\mathbf{I}+\mathrm{e}^{-k\mathbf{A}})\,. \end{aligned}$$

Applying the results of Lemma 3 to (10) and setting $W(t_n)=W_n$, we obtain

$$\begin{aligned} \mathbf{W}_{n+1} &= \mathrm{e}^{-k\mathbf{A}_3}\mathbf{W}_n + \mathbf{A}_3^{-1}(\mathbf{I}_m-\mathrm{e}^{-k\mathbf{A}_3})\mathbf{h}(\mathbf{W}_n) \\ &+ k^{-1}\mathbf{A}_3^{-2}(k\mathbf{A}_3-\mathbf{I}_m+\mathrm{e}^{-k\mathbf{A}_3})(\mathbf{h}(\mathbf{W}_{n+1})-\mathbf{h}(\mathbf{W}_n))\,. \end{aligned} \tag{11}$$

Note, however that the current scheme (11) is fully implicit and hence one would need to employ Newton-type solvers. We therefore use the (easy to derive) first order approximation of (11) given by

$$\mathbf{W}^*_{n+1} = \mathrm{e}^{-k\mathbf{A}_3}\mathbf{W}_n + \mathbf{A}_3^{-1}(\mathbf{I}_m-\mathrm{e}^{-k\mathbf{A}_3})\mathbf{h}(\mathbf{W}_n) \tag{12}$$

known as ETD1 (exponential time differencing of order one which is the same as the IMEX backward Euler method). Next, we set $\mathbf{h}(\mathbf{W}_{n+1})=\mathbf{h}(\mathbf{W}^*_{n+1})$. Hence, our second-order semi-discrete exponential time differencing (ETD) scheme is (see also p. 25 in [7])

$$\begin{aligned} \mathbf{W}_{n+1} &= \mathrm{e}^{-k\mathbf{A}_3}\mathbf{W}_n + \mathbf{A}_3^{-1}(\mathbf{I}_m-\mathrm{e}^{-k\mathbf{A}_3})\mathbf{h}(\mathbf{W}_n) \\ &+ k^{-1}\mathbf{A}_3^{-2}(k\mathbf{A}_3-\mathbf{I}_m+\mathrm{e}^{-k\mathbf{A}_3})(\mathbf{h}(\mathbf{W}^*_{n+1})-\mathbf{h}(\mathbf{W}_n)) \\ \mathbf{W}^*_{n+1} &= \mathrm{e}^{-k\mathbf{A}_3}\mathbf{W}_n + \mathbf{A}_3^{-1}(\mathbf{I}_m-\mathrm{e}^{-k\mathbf{A}_3})\mathbf{h}(\mathbf{W}_n)\,. \end{aligned} \tag{13}$$

Finally, the unwinding back to $\mathbf{U}$ is needed. However, we explain this later, since we first concentrate on the approximation of the matrix exponential.

6. Approximating the matrix exponential

The final step is to approximate the matrix exponential. There are several ways to achieve this (see [35,48,49,67]). However, we focus on the second-order L-acceptable rational approximation with simple real distinct poles (RDP) originally proposed in [65] and used in [6, 7] to derive the ETD-RDP scheme. A second-order approximation of the matrix exponential using rational approximation [7, p. 26] is given by

$$\mathbf{R}_{\mathrm{RDP}}(k\mathbf{A_3}) = \left(\mathbf{I}_m-\frac{5k}{12}\mathbf{A}_3\right)\left[\left(\mathbf{I}_m+\frac{k}{3}\mathbf{A}_3\right)\left(\mathbf{I}_m+\frac{k}{4}\mathbf{A}_3\right)\right]^{-1} \approx \mathrm{e}^{-k\mathbf{A_3}} \tag{14}$$

and the Padé-$(0,1)$ first-order approximation is given by

$$\mathbf{R}_{01}(k\mathbf{A_3}) = (\mathbf{I}_m+k\mathbf{A}_3)^{-1} \approx \mathrm{e}^{-k\mathbf{A_3}}\,. \tag{15}$$

Using (14) and (15) for (13) yields

$$\begin{aligned}\mathbf{W}_{n+1} &= \mathbf{R}_{\mathrm{RDP}}(k\mathbf{A_3})\mathbf{W}_n + \mathbf{A}_3^{-1}(\mathbf{I}_m - \mathbf{R}_{\mathrm{RDP}}(k\mathbf{A_3}))\mathbf{h}(\mathbf{W}_n)\\ &+ k^{-1}\mathbf{A}_3^{-2}(k\mathbf{A}_3 - \mathbf{I}_m + \mathbf{R}_{\mathrm{RDP}}(k\mathbf{A_3}))(\mathbf{h}(\mathbf{W}^*_{n+1}) - \mathbf{h}(\mathbf{W}_n))\\ \mathbf{W}^*_{n+1} &= \mathbf{R}_{01}(k\mathbf{A_3})\mathbf{W}_n + \mathbf{A}_3^{-1}(\mathbf{I}_m - \mathbf{R}_{01}(k\mathbf{A_3}))\mathbf{h}(\mathbf{W}_n)\,. \end{aligned} \tag{16}$$

The first-order predictor $\mathbf{W}^*_{n+1}$ can be simplified to

$$\begin{aligned}\mathbf{W}^*_{n+1} &= \mathbf{R}_{01}(k\mathbf{A_3})\mathbf{W}_n + \mathbf{A}_3^{-1}\left(\mathbf{R}_{01}(k\mathbf{A_3})^{-1}\mathbf{R}_{01}(k\mathbf{A_3}) - \mathbf{R}_{01}(k\mathbf{A_3})\right)\mathbf{h}(\mathbf{W}_n)\\ &= \mathbf{R}_{01}(k\mathbf{A_3})\mathbf{W}_n + \mathbf{A}_3^{-1}(\mathbf{I}_m + k\mathbf{A}_3 - \mathbf{I}_m)\mathbf{R}_{01}(k\mathbf{A_3})\mathbf{h}(\mathbf{W}_n)\\ &= \mathbf{R}_{01}(k\mathbf{A_3})\left(\mathbf{W}_n + k\mathbf{h}(\mathbf{W}_n)\right) = \left(\mathbf{I}_m + k\mathbf{A}_3\right)^{-1}\left(\mathbf{W}_n + k\mathbf{h}(\mathbf{W}_n)\right)\,.\end{aligned}$$

Next, we focus on the corrector $\mathbf{W}_{n+1}$. We have

$$\mathbf{A}_3^{-1}(\mathbf{I}_m - \mathbf{R}_{\mathrm{RDP}}(k\mathbf{A_3})) = k\left(\mathbf{I}_m + \frac{k}{12}\mathbf{A}_3\right)\left(\mathbf{I}_m + \frac{k}{4}\mathbf{A}_3\right)^{-1}\left(\mathbf{I}_m + \frac{k}{3}\mathbf{A}_3\right)^{-1}$$

and likewise

$$\begin{aligned}&k^{-1}\mathbf{A}_3^{-2}(k\mathbf{A}_3 - \mathbf{I}_m + \mathbf{R}_{\mathrm{RDP}}(k\mathbf{A_3}))\\ =\;& \frac{k}{2}\left(\mathbf{I}_m + \frac{k}{6}\mathbf{A}_3\right)\left(\mathbf{I}_m + \frac{k}{4}\mathbf{A}_3\right)^{-1}\left(\mathbf{I}_m + \frac{k}{3}\mathbf{A}_3\right)^{-1}\,.\end{aligned}$$

The corrector can now be written as

$$\begin{aligned}\mathbf{W}_{n+1} &= \left(\mathbf{I}_m - \frac{5k}{12}\mathbf{A}_3\right)\left(\mathbf{I}_m + \frac{k}{4}\mathbf{A}_3\right)^{-1}\left(\mathbf{I}_m + \frac{k}{3}\mathbf{A}_3\right)^{-1}\mathbf{W}_n\\ &+ \frac{k}{2}\left(\mathbf{I}_m + \frac{k}{4}\mathbf{A}_3\right)^{-1}\left(\mathbf{I}_m + \frac{k}{3}\mathbf{A}_3\right)^{-1}\mathbf{h}(\mathbf{W}_n)\\ &+ \frac{k}{2}\left(\mathbf{I}_m + \frac{k}{6}\mathbf{A}_3\right)\left(\mathbf{I}_m + \frac{k}{4}\mathbf{A}_3\right)^{-1}\left(\mathbf{I}_m + \frac{k}{3}\mathbf{A}_3\right)^{-1}\mathbf{h}(\mathbf{W}^*_{n+1})\,.\end{aligned}$$

These three terms can be nicely rewritten using the following three partial fraction decompositions (cf. [6, p. 32] for the details)

$$\begin{aligned}\left(\mathbf{I}_m - \frac{5k}{12}\mathbf{A}_3\right)\mathbf{S}_{m,k}\mathbf{T}_{m,k} &= 9\mathbf{T}_{m,k} - 8\mathbf{S}_{m,k}\\ \mathbf{S}_{m,k}\mathbf{T}_{m,k} &= 4\mathbf{T}_{m,k} - 3\mathbf{S}_{m,k}\\ \left(\mathbf{I}_m + \frac{k}{6}\mathbf{A}_3\right)\mathbf{S}_{m,k}\mathbf{T}_{m,k} &= 2\mathbf{T}_{m,k} - 1\mathbf{S}_{m,k}\,,\end{aligned}$$

where we used the abbreviations

$$\mathbf{S}_{m,k} = \left(\mathbf{I}_m + \frac{k}{4}\mathbf{A}_3\right)^{-1}\,,\quad \mathbf{T}_{m,k} = \left(\mathbf{I}_m + \frac{k}{3}\mathbf{A}_3\right)^{-1}$$

to shorten the notation for these decompositions. Utilizing this partial fraction decomposition, the corrector term can be expressed more compactly as

$$\begin{aligned}\mathbf{W}_{n+1} &= \mathbf{T}_{m,k}\left(9\mathbf{W}_n + 2k\mathbf{h}(\mathbf{W}_n) + k\mathbf{h}(\mathbf{W}^*_{n+1})\right)\\ &- \mathbf{S}_{m,k}\left(8\mathbf{W}_n + \frac{3k}{2}\mathbf{h}(\mathbf{W}_n) + \frac{k}{2}\mathbf{h}(\mathbf{W}^*_{n+1})\right)\end{aligned}$$

Hence, the fully discrete ETD-RDP scheme is given by

$$\begin{aligned}\mathbf{W}_{n+1} &= \left(\mathbf{I}_m + \frac{k}{3}\mathbf{A}_3\right)^{-1}\left[9\mathbf{W}_n + 2k\mathbf{h}(\mathbf{W}_n) + k\mathbf{h}(\mathbf{W}^*_{n+1})\right]\\ &- \left(\mathbf{I}_m + \frac{k}{4}\mathbf{A}_3\right)^{-1}\left[8\mathbf{W}_n + \frac{3k}{2}\mathbf{h}(\mathbf{W}_n) + \frac{k}{2}\mathbf{h}(\mathbf{W}^*_{n+1})\right]\\ \mathbf{W}^*_{n+1} &= (\mathbf{I}_m + k\mathbf{A}_3)^{-1}\left(\mathbf{W}_n + k\mathbf{h}(\mathbf{W}_n)\right). \qquad (17)\end{aligned}$$

Note that the stability regions of this second-order L-stable scheme (see [6, Theorems 4.0.2–4.0.4] for the detailed proof of second order accuracy and [7, p. 26] for the L-stability, respectively) are given in [6, Figure 2.2] and compared against ETD-CN, ETD-Padé-$(0,2)$, and implicit-explicit Adams-Moulton/Bashforth scheme.

Now, we substitute $\mathbf{W} = \mathrm{e}^{\mathbf{A}_2 t}\mathbf{V}$ and $\mathbf{h}(\mathbf{W}) = \mathrm{e}^{\mathbf{A}_2 t}\mathbf{g}(\mathbf{V})$, which means we have the expressions $\mathbf{W}_n = \mathrm{e}^{\mathbf{A}_2 nk}\mathbf{V}_n$, $\mathbf{W}_{n+1} = \mathrm{e}^{\mathbf{A}_2 nk}\mathrm{e}^{\mathbf{A}_2 k}\mathbf{V}_{n+1}$, $\mathbf{h}(\mathbf{W}_n) = \mathrm{e}^{\mathbf{A}_2 nk}\mathbf{g}(\mathbf{V}_n)$, and further $\mathbf{h}(\mathbf{W}_{n+1}) = \mathrm{e}^{\mathbf{A}_2 nk}\mathrm{e}^{\mathbf{A}_2 k}\mathbf{g}(\mathbf{V}_{n+1})$. Then, (17) can be written as

$$\begin{aligned}\mathbf{V}_{n+1} &= \left(\mathbf{I}_m + \frac{k}{3}\mathbf{A}_3\right)^{-1}\left[\mathrm{e}^{-\mathbf{A}_2 k}\left\{9\mathbf{V}_n + 2k\mathbf{h}(\mathbf{V}_n)\right\} + k\mathbf{g}(\mathbf{V}^*_{n+1})\right]\\ &- \left(\mathbf{I}_m + \frac{k}{4}\mathbf{A}_3\right)^{-1}\left[\mathrm{e}^{-\mathbf{A}_2 k}\left\{8\mathbf{V}_n + \frac{3k}{2}\mathbf{h}(\mathbf{V}_n)\right\} + \frac{k}{2}\mathbf{h}(\mathbf{V}^*_{n+1})\right]\\ \mathbf{V}^*_{n+1} &= (\mathbf{I}_m + k\mathbf{A}_3)^{-1}\mathrm{e}^{-\mathbf{A}_2 k}\left(\mathbf{V}_n + k\mathbf{h}(\mathbf{V}_n)\right).\end{aligned}$$

Now, we approximate $\mathrm{e}^{-\mathbf{A}_2 k}$ in the corrector with $\mathbf{R}_{\mathrm{RDP}}$ and in the predictor with $\mathbf{R}_{01}$. This gives

$$\begin{aligned}\mathbf{V}_{n+1} &= \left(\mathbf{I}_m + \frac{k}{3}\mathbf{A}_3\right)^{-1}\left[\left\{9\left(\mathbf{I}_m + \frac{k}{3}\mathbf{A}_2\right)^{-1} - 8\left(\mathbf{I}_m + \frac{k}{4}\mathbf{A}_2\right)^{-1}\right\}\right.\\ &\bullet\ \left.\{9\mathbf{V}_n + 2k\mathbf{g}(\mathbf{V}_n)\} + k\mathbf{g}(\mathbf{V}^*_{n+1})\right]\\ &- \left(\mathbf{I}_m + \frac{k}{4}\mathbf{A}_3\right)^{-1}\left[\left\{9\left(\mathbf{I}_m + \frac{k}{3}\mathbf{A}_2\right)^{-1} - 8\left(\mathbf{I}_m + \frac{k}{4}\mathbf{A}_2\right)^{-1}\right\}\right.\\ &\bullet\ \left.\left\{8\mathbf{V}_n + \frac{3k}{2}\mathbf{g}(\mathbf{V}_n)\right\} + \frac{k}{2}\mathbf{g}(\mathbf{V}^*_{n+1})\right]\\ \mathbf{V}^*_{n+1} &= (\mathbf{I}_m + k\mathbf{A}_3)^{-1}(\mathbf{I}_m + k\mathbf{A}_2)^{-1}\left(\mathbf{V}_n + k\mathbf{g}(\mathbf{V}_n)\right). \qquad (18)\end{aligned}$$

Next, we recall the substitution $\mathbf{V} = \mathrm{e}^{\mathbf{A}_1 t}\mathbf{U}$ and $\mathbf{g}(\mathbf{V}) = \mathrm{e}^{\mathbf{A}_1 t}\mathbf{f}(\mathbf{U})$, which means we have $\mathbf{V}_n = \mathrm{e}^{\mathbf{A}_1 nk}\mathbf{U}_n$, $\mathbf{V}_{n+1} = \mathrm{e}^{\mathbf{A}_1 nk}\mathrm{e}^{\mathbf{A}_1 k}\mathbf{U}_{n+1}$, $\mathbf{g}(\mathbf{V}_n) = \mathrm{e}^{\mathbf{A}_1 nk}\mathbf{f}(\mathbf{U}_n)$, and $\mathbf{g}(\mathbf{V}_{n+1}) = \mathrm{e}^{\mathbf{A}_1 nk}\mathrm{e}^{\mathbf{A}_1 k}\mathbf{f}(\mathbf{U}_{n+1})$. We approximate $\mathrm{e}^{-\mathbf{A}_1 k}$ in the corrector with $\mathbf{R}_{\mathrm{RDP}}$ and in the predic-

tor with $\mathbf{R}_{01}$, which finally yields the fully discrete iterative scheme ($d = 3$) given in the following frame.

ETD-RDP-IF scheme in d dimensions
For $n = 0, 1, 2, \ldots, T/k - 1$ compute

$$
\begin{aligned}
\mathbf{U}_{n+1} &= \left(\mathbf{I}_m + \frac{k}{3}\mathbf{A}_d\right)^{-1} \left[\left\{9\left(\mathbf{I}_m + \frac{k}{3}\mathbf{A}_{d-1}\right)^{-1} - 8\left(\mathbf{I}_m + \frac{k}{4}\mathbf{A}_{d-1}\right)^{-1}\right\} \bullet \ldots \right. \\
&\bullet \left. \left\{9\left(\mathbf{I}_m + \frac{k}{3}\mathbf{A}_1\right)^{-1} - 8\left(\mathbf{I}_m + \frac{k}{4}\mathbf{A}_1\right)^{-1}\right\} \{9\mathbf{U}_n + 2k\mathbf{f}(\mathbf{U}_n)\} + k\mathbf{f}(\mathbf{U}^*_{n+1})\right] \\
&- \left(\mathbf{I}_m + \frac{k}{4}\mathbf{A}_d\right)^{-1} \left[\left\{9\left(\mathbf{I}_m + \frac{k}{3}\mathbf{A}_{d-1}\right)^{-1} - 8\left(\mathbf{I}_m + \frac{k}{4}\mathbf{A}_{d-1}\right)^{-1}\right\} \bullet \ldots \right. \\
&\bullet \left. \left\{9\left(\mathbf{I}_m + \frac{k}{3}\mathbf{A}_1\right)^{-1} - 8\left(\mathbf{I}_m + \frac{k}{4}\mathbf{A}_1\right)^{-1}\right\} \left\{8\mathbf{U}_n + \frac{3k}{2}\mathbf{f}(\mathbf{U}_n)\right\} + \frac{k}{2}\mathbf{f}(\mathbf{U}^*_{n+1})\right] \\
\mathbf{U}^*_{n+1} &= \left(\mathbf{I}_m + k\mathbf{A}_d\right)^{-1} \bullet \ldots \bullet \left(\mathbf{I}_m + k\mathbf{A}_1\right)^{-1} \left(\mathbf{U}_n + k\mathbf{f}(\mathbf{U}_n)\right) . \qquad (19)
\end{aligned}
$$

with $\mathbf{U}_0 = \mathbf{U}(0)$.

Note that we list here the fully discrete iterative ETD-RDP-IF scheme in d dimensions since it is easily derived in a similar fashion.

7. Implementation and parallelization of the fully discrete scheme

An implementation of this scheme given by (19) in parallel utilizing only three threads for the three-dimensional case is illustrated in the flow chart given in Figure 1. One could theoretically also use five threads to speed up the computations. However, we use for the numerical results a computer which has at most four threads as explained in the next section. The two-dimensional case is illustrated in Figure 2 and can be derived using the system given in (18) replacing $\mathbf{V}$ with $\mathbf{U}$, $\mathbf{g}$ with $\mathbf{f}$, $\mathbf{A}_2$ with $\mathbf{A}_1$ and $\mathbf{A}_3$ with $\mathbf{A}_2$.

The most time consuming part is the numerical solution of the large linear systems of the form $(\mathbf{I}_m + \frac{k}{\gamma}\mathbf{A}_i)x = b$ with $\gamma \in \{1, 3, 4\}$ and $\mathbf{A}_i$ given by (4) for the three-dimensional case and (5) for the general case. We focus our explanations directly on the matrices $\mathbf{A}_i \in \mathbb{R}^{m\times m}$ with $m = s\cdot p^d$. These are sparse having only a few diagonals occupied with non-zero elements. Hence, only those diagonals have to be stored. In fact, it is easy to see that the matrix $\mathbf{A}_1$ has three diagonals for both the Dirichlet and Neumann boundary condition case. Precisely, it has a main diagonal (the location is zero) as well as an upper and lower diagonal located at $\pm s$ elements apart from the main diagonal, written compactly here as $\mathbf{A}_1 = \text{sparse}(\mathbf{T}, [-s, 0, s])$ where the matrix $\mathbf{T}$ contains the three vectors $\ell^{(1)} = (0, \ldots, 0, \ell_{s+1}, \ldots, \ell_m)^\top$, $d^{(1)} = (d_1, \ldots, d_s)^\top$, and $u^{(1)} = (u_1, \ldots, u_{m-s}, 0, \ldots, 0)^\top$ all of size $s\cdot p^3$ padded with zeros accordingly. Similarly, the matrices $\mathbf{A}_2$ and $\mathbf{A}_3$ for the Dirichlet and Neumann boundary case are sparse and of the form $\mathbf{A}_2 = \text{sparse}(\mathbf{T}, [-s\cdot p, 0, s\cdot p])$ and $\mathbf{A}_3 = \text{sparse}(\mathbf{T}, [-s\cdot p^2, 0, s\cdot p^2])$, respectively. For the periodic boundary condition case, we have sparse matrices with five diagonals of the form $\mathbf{A}_1 = \text{sparse}(\mathbf{T}, [-s\cdot(p-1), -s, 0, s, s\cdot(p-1)])$ with $\mathbf{T} = [\ell^{(2)}, \ell^{(1)}, d^{(1)}, u^{(1)}, u^{(2)}]$. Similarly, we have the matrix

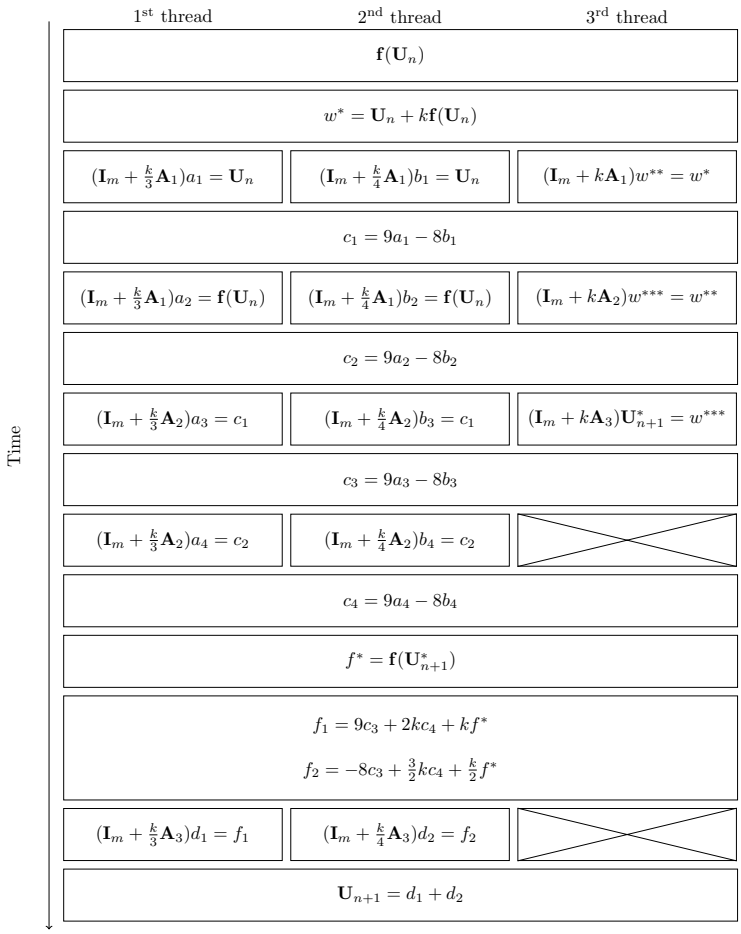

Figure 1: Flow chart for the implementation of the ETD-RDP-IF scheme in parallel using three threads for the three dimensional case.

$\mathbf{A}_2 = \text{sparse}(\mathbf{T}, [-s\cdot p(p-1), -s\cdot p, 0, s\cdot p, s\cdot p(p-1)])$, and finally $\mathbf{A}_3 = \text{sparse}(\mathbf{T}, [-s\cdot p^2(p-1), -s\cdot p^2, 0, s\cdot p^2, s\cdot p^2(p-1)])$.

The band structure for the d-dimensional case is now obvious. For $i \in \{1, \ldots, d\}$, we have $\mathbf{A}_i = \text{sparse}(\mathbf{T}, [-s\cdot p^{i-1}, 0, s\cdot p^{i-1}])$ for the Dirichlet and Neumann boundary case and $\mathbf{A}_i = \text{sparse}(\mathbf{T}, [-s\cdot p^{i-1}(p-1), -s\cdot p^{i-1}, 0, s\cdot p^{i-1}, s\cdot p^{i-1}(p-1)])$ for the periodic boundary case.

A straight-forward $\mathbf{LU}$-decomposition of the matrices $\mathbf{I}_m + \frac{k}{\gamma}\mathbf{A}_1$, $\mathbf{I}_m + \frac{k}{\gamma}\mathbf{A}_2$, and $\mathbf{I}_m + \frac{k}{\gamma}\mathbf{A}_3$ without pivoting gives lower and upper matrices with the same structure without any fill-ins, since the matrices are all diagonal dominant if k is small enough. Hence, we can adapt the well-known Thomas algorithm to solve tridiagonal systems of the form $\text{sparse}(\mathbf{T}, [-1, 0, 1])$ (see for example [16]) to derive the following algorithm in order to solve systems of the form $\text{sparse}(\mathbf{T}, [-w, 0, w])$.

Precisely, we have to compute for the sparse matrices from above with given diagonals

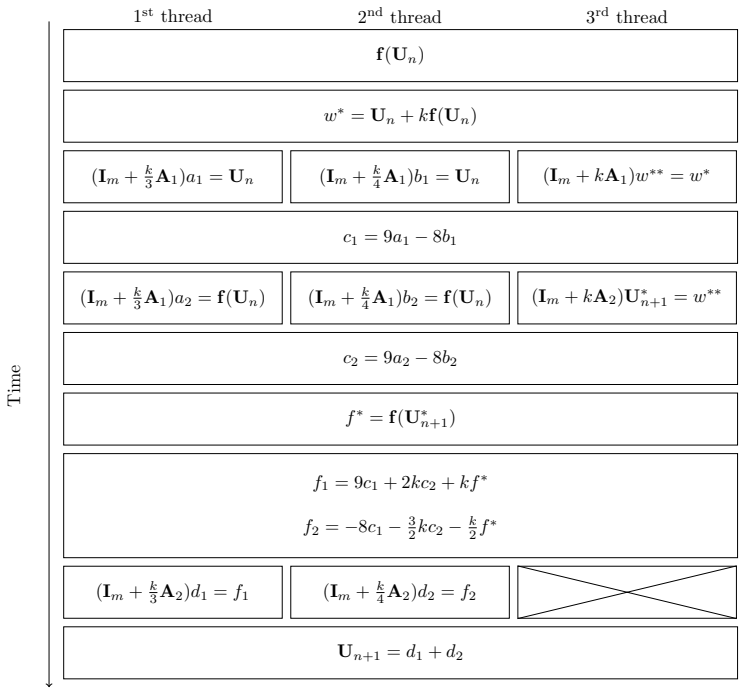

Figure 2: Flow chart for the implementation of the ETD-RDP-IF scheme in parallel using three threads for the two dimensional case.

$\ell^{(1)}$, $d^{(1)}$, and $u^{(1)}$ with offset w and given right-hand side b the following steps. For $i = 1, \ldots, w$, we compute $\alpha_i = u_i^{(1)}/d_i^{(1)}$ and $\beta_i = b_i/d_i^{(1)}$ as well as for $i = w+1, \ldots, m-w$ we have $\alpha_i = u_i^{(1)}/\{d_i^{(1)} - \alpha_{i-w} \cdot l_i^{(1)}\}$ and for $i = w+1, \ldots, m$ we have $\beta_i = \{b_i - \beta_{i-w} \cdot l_i^{(1)}\}/\{d_i^{(1)} - \alpha_{i-w} \cdot l_i^{(1)}\}$. Then, we assign $x_{m-i+1} = \beta_{m-i+1}$ for $i = 1, \ldots, w$ and compute $x_i = \beta_i - \alpha_i \cdot x_{i+w}$ for the decreasing $i = m-w, \ldots, 1$.

Now, we focus on the periodic boundary case. As shown above, the matrices have five diagonals. If the diagonals have the same distance from each other, one could derive the factorization explicitly, and use forward and backward substitution similar to the variant of the Thomas algorithm explained before. However, this is not the case here. Using an **LU**-decomposition shows that a few fill-ins are generated and therefore a variant of the Thomas algorithm is not applicable. One could use the well-known Sherman-Morrison-Woodbury formula (see [24]), since the periodic case in 1D with $s = 1$ is a rank-2 update of the Dirichlet case. Hence, in this special situation, it is possible to obtain an algorithm to compute explicitely $(\mathbf{I}_m + \frac{k}{\gamma}\mathbf{A}_1)$ using the variant of the Thomas algorithm as explained before. But the extension to 2D already gives a rank-$2p$ update (that means one has to solve a linear systems with $2p$ right hand sides using the variant of the Thomas algorithm). The situation gets even more complicated considering $(\mathbf{I}_m + \frac{k}{\gamma}\mathbf{A}_2)$ in 2D or in higher dimensions with $s > 1$.

If we consider $s = 1$ and $d = 2$, then it is easy to see that $(\mathbf{I}_m + \frac{k}{\gamma}\mathbf{A}_1)$ is a diagonal

block matrix, where each block is a cyclic Toeplitz matrix. Likewise, $(\mathbf{I}_m + \frac{k}{\gamma}\mathbf{A}_2)$ is a cyclic Toeplitz matrix. That means, we can apply the Fourier transformation to efficiently solve the linear system. Precisely, if a is the first colum of a cyclic Toeplitz matrix, then it holds $(\mathcal{F}_m a)\cdot(\mathcal{F}_m x) = \mathcal{F}_m b$ and hence x is given by $x = \mathcal{F}_m^{-1}((\mathcal{F}_m b)./(\mathcal{F}_m a))$ (see for example [13]). Here, ./ means component-wise division. This is faster in computation than using the variant of the Thomas algorithm after applying the Sherman-Morrison-Woodbury formula. Likewise, we have for $s = 1$ and $d = 3$ that $(\mathbf{I}_m + \frac{k}{\gamma}\mathbf{A}_1)$ and $(\mathbf{I}_m + \frac{k}{\gamma}\mathbf{A}_2)$ are cyclic block Toeplitz matrices and $(\mathbf{I}_m + \frac{k}{\gamma}\mathbf{A}_3)$ is a cyclic Toeplitz matrix.

Remark 2. *Again, we would like to stress the fact that we never store the complete matrices in memory, but instead only the elements from the three or five diagonals depending on the problem at hand. The solution of the linear systems for the Dirichlet and Neumann boundary case can be directly obtained through the use of a generalization of the Thomas algorithm taking a serious advantage of the dimensional splitting altogether. Hence, we can avoid to solve the sparse linear systems iteratively through Jacobi, Gauß-Seidel or Krylov methods. However, the situation is different for the periodic boundary case in the general setting. For the special case $s = 1$, we use the Fourier transform to efficiently solve the sparse linear system instead of applying the Sherman-Morrison-Woodbury formula and solving directly by a variant of the Thomas algorithm.*

8. Numerical results and comparison

In this section, numerical results are presented for a variety of examples both in two and three dimensions and compared with existing methods. All numerical results are performed on a PC equipped with four Intel cores (i7-4790 CPU @ 3.60GHz) on a socket each of which can have two threads (architecture: x86_64, CPU operation modes: 32-bit and 64-bit, and byte order: little endian). The machine has 32GB of memory. We use the gfortran Fortran compiler gcc version 7.4.0 on SUSE Linux (version 15.1) with the optimization option `-O3 -march=native` and the OpenMP (version 4.5) option `-fopen_mp` and the Matlab version R2018a. More information of the mentioned compiler, software, and operation system can be obtained at the following links: https://www.intel.de https://gcc.gnu.org https://www.suse.com https://www.openmp.org https://de.mathworks.com Additionally, we used the Fortran library FFTPACK5.1 for the fast Fourier transformation available under https://people.sc.fsu.edu/~jburkardt/f77_src/fftpack5.1/fftpack5.1.html

8.1. Enzyme kinetics of Michaelis-Menten type

The enzyme kinetics of Michaelis-Menten type reaction-diffusion equation in one dimension has been considered in [15] and since then researchers have considered it in higher dimensions as a testing scenario (see for example [9]). It is of the form

$$\frac{\partial u}{\partial t} = \gamma \Delta u - \frac{u}{1+u}$$

where the positive constant γ is given. The domain Ω is assumed to be a unit square and the time interval is $[0, 1]$. The initial condition is prescribed to be constant one and the boundary condition is assumed to be homogeneous Dirichlet. Solving this problem numerically is known to be challenging due to the discontinuity of the initial and boundary conditions as it can cause spurious oscillations in the solution (cf. [9]). Clearly, the problem at hand fits into the format (2) with $s = 1$, $D = \gamma$, and $f(u) = -u/(1 + u)$.

First, we show in Table 1 that the new method ETD-RDP-IF is indeed second-order as ETD-RDP without splitting (see [7] for ETD-RDP). We use $D = 0.2$, fix h quite small as 0.0125, and vary k from 0.05 to 0.00625 to compute the error $\epsilon(k)$ measured in the L^∞ norm and the estimated order of convergence $\text{EOC} = \log(\epsilon(k)/\epsilon(k/2))/\log(2)$ for the final time $T = 1$. Additionally, we include the CPU times. As we see in Table 1, we obtain

k	$\epsilon(k)_{\text{ETD-RDP}}$	EOC	CPU Time	$\epsilon(k)_{\text{ETD-RDP-IF}}$	EOC	CPU time
0.05000	5.3337×10^{-3}		0.38305	7.1838×10^{-3}		0.02738
0.02500	1.5935×10^{-3}	1.74	0.76822	1.5829×10^{-3}	2.18	0.04211
0.01250	4.3420×10^{-4}	1.88	1.51854	3.6926×10^{-4}	2.10	0.07334
0.00625	1.1356×10^{-4}	1.93	3.02891	8.8718×10^{-5}	2.06	0.13775

Table 1: Estimated order of convergence EOC for both the ETD-RDP and ETD-RDP-IF for the enzyme kinetics example using the parameters $D = 0.2$, $h = 0.0125$, $T = 1$ within the unit square. Additionally, the CPU times in seconds of the Matlab program are listed.

a second-order convergence rate with comparable errors for both methods. Above all, we notice a much better performance of the ETD-RDP-IF compared to ETD-RDP. We obtain a speed-up factor of 20.

Next, we show in Figure 3 the efficiency of the new method ETD-RDP-IF compared with other second-order methods such as ETD-RDP, IMEX-BDF2, IMEX-TR, and IMEX-Adams2 (all methods are implemented in Matlab and run serial).

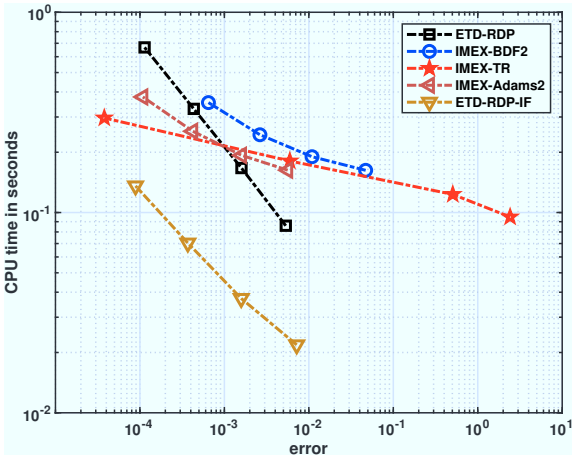

Figure 3: Log-Log efficiency plot comparing ETD-RDP and ETD-RDP-IF with different second-order IMEX schemes implemented in Matlab running serial for the enzyme kinetics example using the parameters $D = 0.2$ and $T = 1$ within the unit square.

As we see in Figure 3, the ETD-RDP method has a comparable efficiency as the different second-order IMEX schemes for this problem. However, the application of dimensional splitting outperforms the IMEX schemes. Hence, the ETD-RDP-IF is the most efficient method for this enzyme kinetics example.

Note also that we tried other splitting techniques for the enzyme kinetics example, such as Strang simple splitting and Strang symmetric splitting (see [6]). It turns out that the integrating factor outperforms the other two splitting techniques (as this was also the case in [8]).

Now, we show in Table 2 the performance of the ETD-RDP-IF method implemented in Matlab in comparison with the Fortran implementation without and with parallelization for a variety of parameter choices for h and k using the parameters $D = 0.2$ and $T = 1$. Although the algorithm ETD-RDP-IF is already very efficient in serial compared to other

$k = h$	Matlab	Fortran (1 thread)	Fortran (3 threads)
1/100	0.16	0.14	0.06
1/200	1.20	1.19	0.48
1/400	9.76	9.87	4.54
1/800	83.26	73.80	35.81

Table 2: CPU times in seconds for the ETD-RDP-IF method implemented in Matlab and Fortran (serial and parallelized version) for the enzyme kinetics example using the parameter $D = 0.2$ and $T = 1$ within the unit square.

methods, it can be greatly improved first through conversion to Fortran and then through parallelization. We obtain a speed-up factor of two. Note that we list the timings of the complete program, i.e. the preparation of matrices (allocation, initialization, etc.) and the timing of the ETD-RDP-IF algorithm.

Note that one could further increase the performance of the Fortran program by using floating point arithmetic with single precision instead of double precision as used above. Using three threads and single precision arithmetic within the Fortran program, we obtain the timings 0.04, 0.33, 2.76, and 23.65 seconds, respectively. This gives a total speed-up factor of four compared to the Matlab implementation.

8.2. The Brusselator system

In this section, we consider the two- and three-dimensional generalization of the one-dimensional reaction-diffusion Brusselator system. For more explanations regarding the well-studied model for a hypothetical tri-molecular reaction, we refer the reader to [68, p. 526] and the references cited therein for the details and the background of this model. The system in two dimensions reads

$$\begin{aligned} \frac{\partial u_1}{\partial t} &= D\Delta u_1 + u_1^2 u_2 - (A+1)u_1 + B \\ \frac{\partial u_2}{\partial t} &= D\Delta u_2 - u_1^2 u_2 + A u_1 \end{aligned}$$

where the constants D, A, and B are given. In our experiment, we consider the domain $\Omega = [0,1]^2$ and $t \in (0,2)$ and the parameters $D = 2 \times 10^{-3}$, $A = 3.4$, and $B = 1$ as done in [7]. We prescribe homogeneous Neumann boundary conditions for both u_1 and u_2. The initial condition is given by

$$u_1(x,y,0) = 1/2 + y \qquad u_2(x,y,0) = 1 + 5x\,.$$

Again, we can see in Table 3 that the estimated order of convergence agrees with the theoretical convergence order of two where we fixed $h = 0.0125$. In Figure 4 we also show that the ETD-RDP-IF outperforms ETD-RDP and the second-order IMEX schemes IMEX-BDF2, IMEX-TR, and IMEX-Adams2 (all methods are implemented in Matlab and run serial). The CPU timings can be greatly improved through the Fortran implementation and

k	$\epsilon(k)_{\text{ETD-RDP}}$	EOC	CPU Time	$\epsilon(k)_{\text{ETD-RDP-IF}}$	EOC	CPU time
0.1000	2.3309×10^{-1}		0.19429	2.3852×10^{-1}		0.03740
0.0500	5.8468×10^{-2}	2.00	0.38407	4.8944×10^{-2}	2.28	0.06368
0.0250	1.5750×10^{-2}	1.89	0.76545	1.2898×10^{-2}	1.92	0.12128
0.0125	4.0884×10^{-3}	1.95	1.53955	3.3223×10^{-3}	1.96	0.24152

Table 3: Estimated order of convergence EOC for both the ETD-RDP and ETD-RDP-IF for the Brusselator example using the parameters $D = 2 \times 10^{-3}$, $A = 3.4$, $B = 1$, $h = 0.0125$, and $T = 2$ within the unit square. Additionally, the CPU times in seconds of the Matlab program are listed.

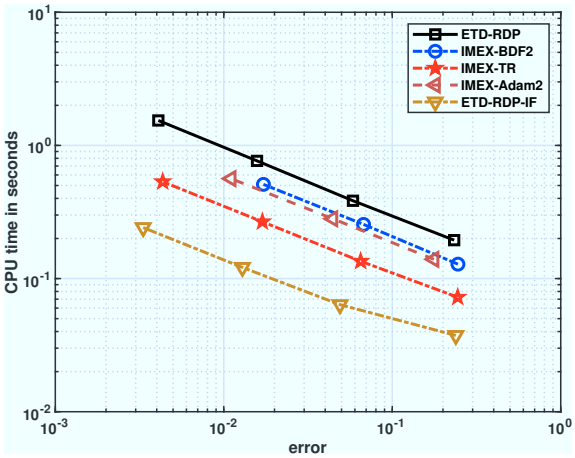

Figure 4: Log-Log efficiency plot comparing ETD-RDP and ETD-RDP-IF with different second-order IMEX schemes implemented in Matlab running serial for the Brusselator example using the parameters $D = 2 \times 10^{-3}$, $A = 3.4$, $B = 1$, and $T = 2$ within the unit square.

additionally through parallelization. The CPU timings are listed in Table 4. As we can see,

$k = h$	Matlab	Fortran (1 thread)	Fortran (3 threads)
1/100	0.51	0.36	0.16
1/200	3.87	3.32	1.60
1/400	34.05	28.99	13.92
1/800	286.06	190.75	114.19

Table 4: CPU times in seconds for the ETD-RDP-IF method implemented in Matlab and Fortran (serial and parallelized version) for the Brusselator example using the parameters $D = 2 \times 10^{-3}$, $A = 3.4$, $B = 1$, and $T = 2$.

the serial Fortran program is faster than the Matlab program as expected. The parallelized version is almost twice as fast as the serial version.

Finally, we also consider the three-dimensional Brusselator example on the domain $\Omega = [0, 1]^3$ with the same initial and boundary conditions and parameters as in test problem 4 within [9]. Precisely, we use homogeneous Neumann boundary conditions and as initial conditions $u_1(x, y, z, 0) = 1 + \sin(2\pi x)\sin(2\pi y)\sin(2\pi z)$ and $u_2(x, y, z, 0) = 3$. The parameters are $D = 0.02$, $A = 1$, $B = 2$, and $T = 5$. The spatial and time step are given by $h = 1/10$ and $k = 1/1000$. In Figure 5, we show a 2D slice through the 3D domain of u_1 and u_2 for $z = 1$ and the final time $T = 5$. The two plots agree with the theory of the terminal

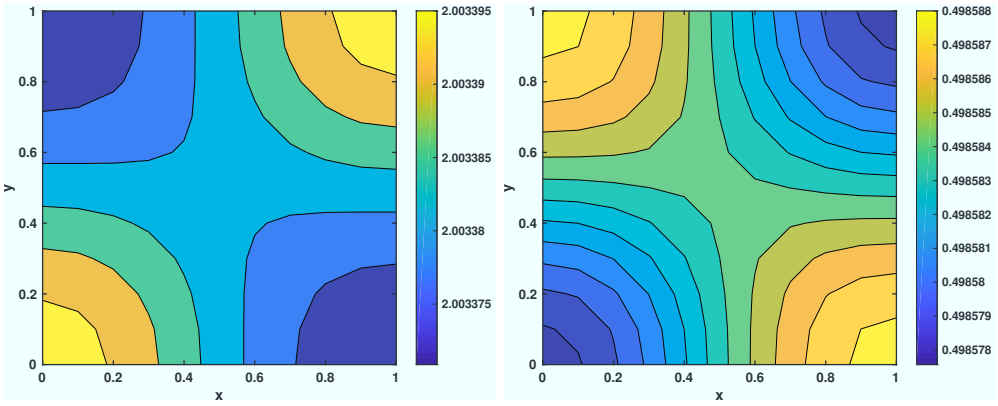

Figure 5: The concentration profiles of u_1 and u_2 for $z = 1$ at $T = 5$ using the parameters $D = 0.02$, $A = 1$, $B = 2$, $h = 1/10$ and $k = 1/1000$.

behavior of the equation proposed in the paper [62] since $1 - A + B^2 \geq 0$ holds. Note that the Matlab program needs 2.87 seconds to compute the result. The serial and parallelized Fortran version only need 1.27 and 0.65 seconds, respectively. In Figure 6 (left) we also show the solutions u_1 and u_2 at the point $(0.3, 0.3, 0.3)$ for the time interval $[0, 5]$ using the same parameters as before. Additionally, we also show in Figure 6 (right) the solutions u_1 and u_2 at the point $(1/3, 1/3, 1/3)$ for the time interval $[0, 40]$ using the parameters $D = 0.02$, $A = 3$, $B = 1$, $h = 1/10$ and $k = 1/1000$. Hence, the equation $1 - A + B^2 \geq 0$ is violated. As expected, the values approach B and A/B (here 2 and $1/2$) as t increases, since

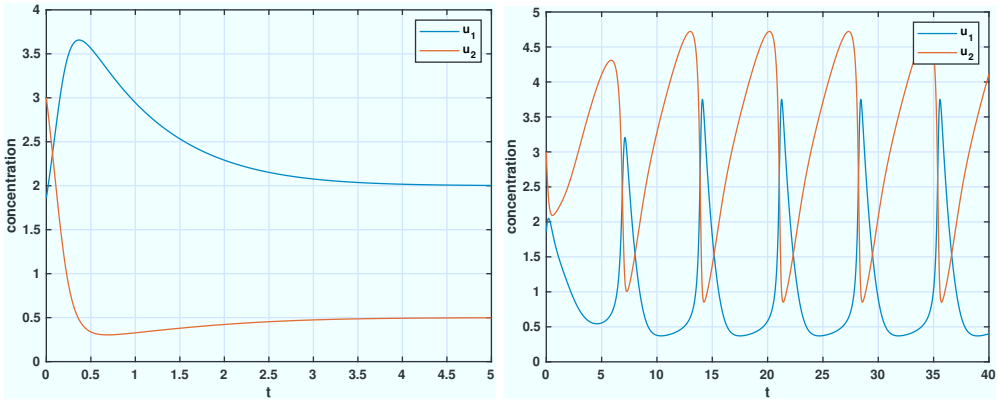

Figure 6: Left: profile of u_1 and u_2 at the point $(1/3, 1/3, 1/3)$ for the time interval $[0, 5]$ using the parameters $D = 0.02$, $A = 1$, $B = 2$, $h = 1/10$ and $k = 1/1000$. Right: profile of u_1 and u_2 at the point $(1/3, 1/3, 1/3)$ for the time interval $[0, 40]$ using the parameters $D = 0.02$, $A = 3$, $B = 1$, $h = 1/10$ and $k = 1/1000$.

$1 - A + B^2 \geq 0$ is satisfied. However, if we choose $A = 3$ and $B = 1$, then $1 - A + B^2 \geq 0$ is violated and we obtain an oscillatory solution. This again is in agreement with [62] that the solution does not converge to a fixed concentration. The plots shown in Figure 6 are also in agreement with Figures 10 and 11 within [9].

8.3. The complex Ginzburg-Landau equation on a periodic domain

The complex Ginzburg-Landau equation [23] has been extensively studied in the physics community. It describes a variety of phenomena such as non-linear waves to second-order phase transitions, from superconductivity, superfluidity, and Bose-Einstein condensation to liquid crystals and strings in field theory (see [5] for a general overview). The complex Ginzburg-Landau equation is given by

$$\frac{\partial u}{\partial t} = u + (1 + \mathrm{i}\alpha)\Delta u - (1 + \mathrm{i}\beta)\, u\, |u|^2$$

where the constants α and β are given real-valued parameters describing linear and non-linear dispersion, respectively. Here, the prescribed initial condition is given by a normal random field with mean zero and standard deviation one, but could also be a smooth function such as a series of Gaussian pulses. The boundary condition is assumed to be periodic. Hence, it fits into the format (2) with $s = 1$, $D = 1 + \mathrm{i}\alpha$, and $f(u) = u - (1 + \mathrm{i}\beta)\, u\, |u|^2$.

For our experiment, we consider the domain $\Omega = [0, 200]^2$ and $t \in (0, 100)$ and the parameters $\alpha = 0$ and $\beta = 1.3$. For the discretization in space, we use $p = 400$ ($h = 1/2$). For the time step, we use $k = 1/20$. To produce the result that is shown in Figure 7, we need about 96 seconds with our implementation in Matlab without any parallelization. Thus, we obtain a comparable performance with the numerical method that is based on

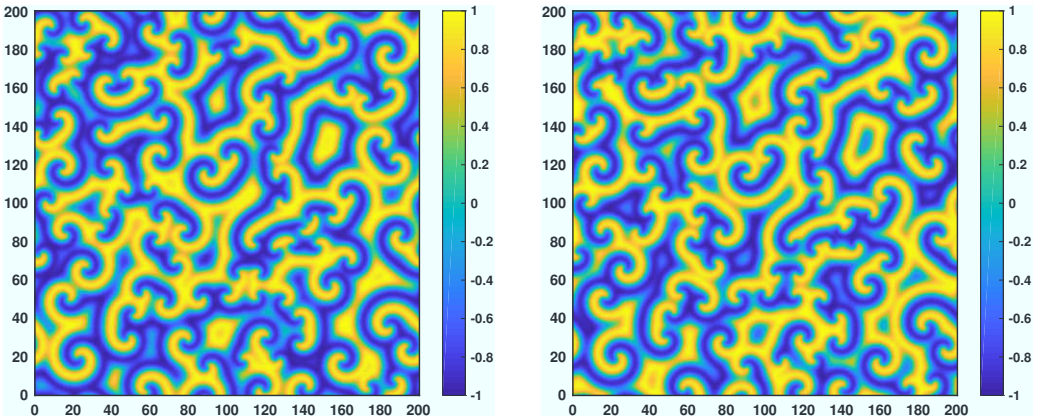

Figure 7: Real (left image) and imaginary part (right image) of the solution of the complex Ginzburg-Landau equation on the periodic domain $\Omega = [0, 200]^2$ for time $T = 100$ with the parameters $\alpha = 0$, $\beta = 1.3$, $p = 400$, and $k = 1/20$ using a standard normal random field as initial condition.

Fourier spatial discretization and a fourth-order exponential time differencing Runge-Kutta (ETDRK4) method that is usually faster than the standard finite difference scheme (see [30]). However, special attention has to be drawn to the time stepping in Fourier space to avoid aliasing effects. The implementation of the Fourier spectral ETDRK4 method in Matlab with the same set of parameters as above needs only 28 seconds (see p. 11 for the Matlab implementation within [29]). Of course, we have only compared the timing and not the approximation quality of the solution as one should. Further, our method is L-stable in comparison to the EDTRK4 and one could create a situation where ETD-RDP-IF outperforms ETDRK4 in the sense of approximation quality or more precisely in the sense of CPU time versus accuracy.

In sum, we obtain a stable solution although the initial data are non-smooth. We obtain similar solution patterns, when we use smooth initial boundary conditions such as a series of Gaussian pulses in the form $u(x,y,0) = \mathrm{e}^{-((x-50)^2+(y-50)^2)/1000} - \mathrm{e}^{-((x-100)^2+(y-100)^2)/1000} + \mathrm{e}^{-((x-100)^2+(y-50)^2)/1000}$ with the same set of parameters as before. We again need about 96 (95.65 to be precise) seconds in Matlab without parallelization to produce the images shown in Figure 8. The serial Fortran version needs 4.99 seconds and the parallelized Fortran

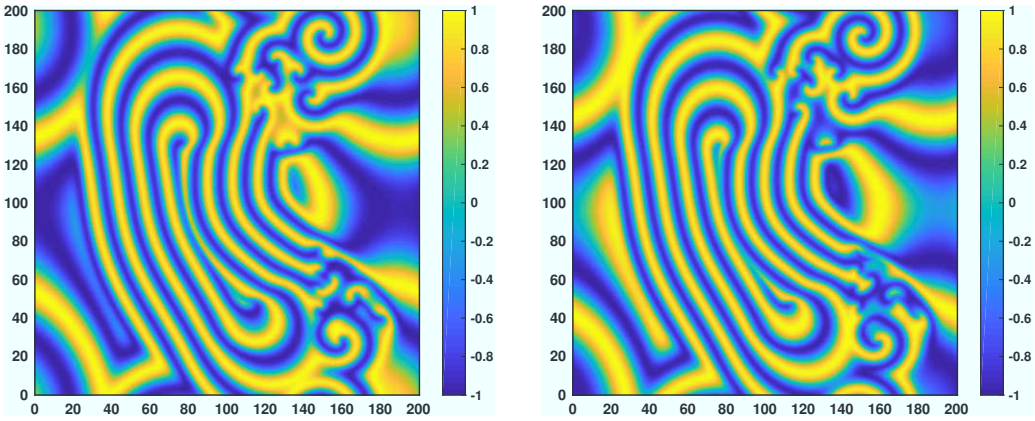

Figure 8: Real (left image) and imaginary part (right image) of the solution of the complex Ginzburg-Landau equation on the periodic domain $\Omega = [0,200]^2$ for time $T = 100$ with the parameters $\alpha = 0$, $\beta = 1.3$, $p = 400$, and $k = 1/20$ using a series of Gaussian pulses field as initial condition.

version needs only 2.76 seconds. Refer also to the last row in Table 5.

p	h	Matlab	Fortran (1 thread)	Fortran (3 threads)
50	4	1.24	1.23	0.51
100	2	5.17	4.95	2.04
200	1	23.11	20.85	10.34
400	1/2	95.65	86.66	53.98

Table 5: CPU times in seconds for the ETD-RDP-IF method implemented in Matlab and Fortran (serial and parallelized version) for the Ginzburg-Landau example on periodic domain $\Omega = [0,200]^2$ using the parameters $\alpha = 0$, $\beta = 1.3$, $k = 1/20$, and $T = 100$.

A similar situation arises when we consider the three-dimensional case. The ETDRK4 using a spectral method needs only 27 seconds in Matlab with the parameters $\alpha = 0$, $\beta = 1.3$, $p = 50$ ($h = 2$), and $k = 1/20$ for time $T = 100$ and $\Omega = [0,100]^3$ whereas our method in Matlab (without any parallelization) needs about 119 seconds, which is longer as expected due to the second-versus-fourth order of the schemes. The serial and parallelized Fortran program need 108.21 and 57.87 seconds, respectively.

We now consider $\Omega = [0,200]^3$ with the same parameters as before and increase the parameter p to 200. That means, we have $8,000,000$ spatial discretization points. Gaussian pulses as initial condition of the form $u(x,y,z,0) = \mathrm{e}^{-((x-50)^2+(y-50)^2+(z-50)^2)/1000} - \mathrm{e}^{-((x-100)^2+(y-100)^2+(z-100)^2)/1000}$ are used, which yields Figure 9. The Matlab program needs 9673.13 seconds whereas the serial and parallelized Fortran program need 8260.91 and 5287.64 seconds, respectively. Refer also to the last row in Table 6.

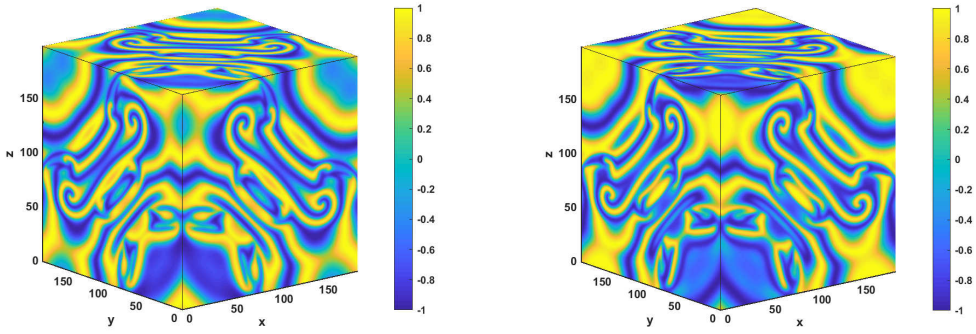

Figure 9: Real (left image) and imaginary part (right image) of the solution of the complex Ginzburg-Landau equation on the periodic domain $\Omega = [0, 200]^3$ for time $T = 100$ with the parameters $\alpha = 0$, $\beta = 1.3$, $p = 200$, and $k = 1/20$ using a series of Gaussian pulses field as initial condition.

p	h	Matlab	Fortran (1 thread)	Fortran (3 threads)
50	4	119.61	108.37	57.94
100	2	1033.63	912.41	585.91
200	1	9673.13	8260.91	5287.64

Table 6: CPU times in seconds for the ETD-RDP-IF method implemented in Matlab and Fortran (serial and parallelized version) for the Ginzburg-Landau example on periodic domain $\Omega = [0, 200]^3$ using the parameters $\alpha = 0$, $\beta = 1.3$, $k = 1/20$, and $T = 100$.

8.4. Schrödinger equation

The Schrödinger equation [57] is a linear PDE and it describes a state function of a quantum-mechanical system (see for example [25] for more details). However, in this section, we consider an extension of it. Precisely, we focus on the d-dimensional non-linear cubic Schrödinger equation of the form

$$\mathrm{i}\Psi_t + \Delta\Psi = q\left(|\Psi|^2\right)\Psi\,, \qquad \mathbf{x} \in [0,1]^d\,, \quad t \in (0,T) \tag{20}$$

with given initial condition $\Psi_0(\mathbf{x}) = \Psi(\mathbf{x}, 0)$ and homogeneous Neumann boundary condition. Here, q is a given function of $|\Psi|^2$. The wave function $\Psi(\mathbf{x}, t)$ can be written as $v(\mathbf{x}, t) + \mathrm{i}w(\mathbf{x}, t)$ and hence, we obtain the coupled system of equations

$$\begin{aligned} v_t + \Delta w &= \quad q(v^2 + w^2)w \\ w_t - \Delta v &= -q(v^2 + w^2)v \end{aligned}$$

which can be written as

$$\begin{pmatrix} v_t \\ w_t \end{pmatrix} + \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} \Delta v \\ \Delta w \end{pmatrix} = q(u^2 + w^2) \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} v \\ w \end{pmatrix}.$$

By setting $\mathbf{u} = \begin{pmatrix} v \\ w \end{pmatrix}$ and $\mathbf{D} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$ we can rewrite the system as

$$\mathbf{u}_t + \mathbf{D}\Delta\mathbf{u} = q(|\mathbf{u}|^2)\mathbf{D}\mathbf{u}$$

which fits into the format (2). However, note that $\mathbf{D}$ is not a diagonal matrix. Using the approximation of the Laplacian as in Example 1 and Example 2, one can again verify similarly to Lemma 2 that the dimensional splitting commutes and hence the ETD-RDP-IF method can be used.

First, we present numerical results for the one-dimensional non-linear cubic Schrödinger equation of the form (see also [11, Section 4])

$$\mathrm{i}\Psi_t + \Psi_{xx} + |\Psi|^2\Psi = 0\,, \qquad -L_0 < x < L_1\,, \quad 0 < t < T \tag{21}$$

with initial condition

$$\Psi(x,0) = \sqrt{2a}\exp\left(\mathrm{i}\,\frac{c}{2}x\right)\operatorname{sech}(\sqrt{a}x)$$

and homogeneous Neumann boundary condition. The function

$$\Psi(x,t) = \sqrt{2a}\exp\left[\mathrm{i}\left(\frac{c}{2}x - \left\{\frac{c^2}{4} - a\right\}t\right)\right]\operatorname{sech}(\sqrt{a}(x-ct))$$

satisfies the Schrödinger equation and the Neumann boundary condition as $|x|$ approaches infinity. The total mass $M(t)$ and energy $E(t)$ of the wave function are given by

$$\begin{aligned} M(t) &= \int_{\mathbb{R}} |\Psi(x,t)|^2\,\mathrm{d}x\,, \\ E(t) &= \int_{\mathbb{R}} \left(|\Psi_x(x,t)|^2 - \frac{1}{2}|\Psi(x,t)|^4\right)\,\mathrm{d}x\,, \end{aligned}$$

respectively. For the purpose of numerically verifying the conservation of these quantities, we approximate the integrals by the trapezoidal rule (as in [11, Section 4.4])

$$\begin{aligned} M(t_n) &\approx \frac{h}{2}\left[|\mathbf{U}_0^n|^2 + 2\sum_{i=1}^{p}|\mathbf{U}_i^n|^2 + |\mathbf{U}_{p+1}^n|^2\right] \\ E(t_n) &\approx \frac{h}{2}\left[-\frac{1}{2}(|\mathbf{U}_0^n|^4 + |\mathbf{U}_{p+1}^n|^4) + 2\sum_{i=1}^{p}\left(\left|\frac{\mathbf{U}_{i+1}^n - \mathbf{U}_{i-1}^n}{2h}\right|^2 - \frac{1}{2}|\mathbf{U}_i^n|^4\right)\right]. \end{aligned}$$

For the numerical results we used the parameters $a = 0.01$, $c = 0.1$, $T = 108$, $h = 1/2$, and $\Delta t = 1/100$ within the interval $[L_0, L_1] = [-80, 100]$. A plot of the exact solution (defined on $\mathbb{R}$) and the approximated solution (defined on $[-80, 100]$) with computing time less than five seconds in Matlab reveals visually no difference and agrees with the one given in [11, Figure 4.1]. The exact values for $M(0)$ and $M(108)$ can be obtained by basic integration techniques from calculus. They are both $2/5 = 0.4$. Numerically, we obtain $0.399\,999\,954\,123\,281$ and $0.399\,999\,954\,128\,036$, respectively. For the energy $E(0)$ and $E(108)$, we obtain the exact value $-1/3000 \approx -0.000\,333\,333\,333\,333$. We obtain numerically $-0.000\,336\,760\,546\,294$ and $-0.000\,336\,768\,976\,796$. Hence, the mass and the energy are conserved asymptotically for the approximated solution obtained via ETD-RDP-IF.

Next, we consider the two-dimensional Schrödinger equation within the unit square and the function $q(|\mathbf{u}|^2) = B(x,y) + C(x,y)|\mathbf{u}|^2$ with $B(x,y) = (1-2\pi^2)(1-\cos^2(\pi x)\cos^2(\pi y))$ and $C(x,y) = (1-2\pi^2)$. The exact solution is given by $\Psi(x,y,t) = \mathrm{e}^{-\mathrm{i}t}\cos(\pi x)\cos(\pi y)$ (see also [18, p. 754]). We use the parameters $T = 1$, $k = 0.0125$, and $h = 1/78$ to create the same plot as in [18, Figure 5] within one second. Note that we obtain similar timing improvements for the parallelized Fortran version in comparison to the serial Matlab program as in the Brusselator example. For some recent results on more complex domains, we refer the reader to [55].

9. Conclusion

In this article, we have combined the second-order exponential time differencing method (ETD) of second-order with approximating the matrix exponential through (non-Padé) rational approximation having real simple and distinct poles (RDP) and the integrating factor (IF) dimensional splitting technique to obtain the second-order L-stable ETD-RDP-IF scheme in two and three dimensions. With this scheme, one can solve non-linear reaction-diffusion equations with either Dirichlet, Neumann, or periodic boundary conditions in either two or three dimensions. The new scheme outperforms other second-order IMEX schemes such as IMEX-BDF2, IMEX-TR, and IMEX-Adams2 as well as ETD-RDP. The excellent performance for 2D and above all 3D examples can be further enhanced through the conversion from Matlab to Fortran. Using basic parallelization techniques increases the efficiency further. The source code is available under the link

`https://github.com/kleefeld80/ETDRDPIF`

In the future, we intend to extend these ideas to develop methods with higher order. Additionally, incorporating an adaptive time selection procedure to these methods might further increase the performance. As an alternative one might want to consider and apply parallel-in-time procedures (see for example [20] for an overview).

Acknowlegement

The collaboration of the authors arose in Cádiz, Spain during the international Conference on Computational and Mathematical Methods in Science and Engineering (CMMSE'19). We would like to thank the organizer, Jesús Vigo-Aguiar, for valuable support and encouragement.

References

[1] A. Abdulle, *Fourth order Chebyshev methods with recurrence relation*, SIAM Journal on Scientific Computing **23** (2002), no. 6, 2041–2054.

[2] A. Abdulle and A.A. Medovikov, *Second order Chebyshev methods based on orthogonal polynomials*, Numerische Mathematik **90** (2001), no. 1, 1–18.

[3] A. Abdulle and G. Vilmart, *PIROCK: A swiss-knife partitioned implicit–explicit orthogonal Runge–Kutta–Chebyshev integrator for stiff diffusion–advection–reaction problems with or without noise*, Journal of Computational Physics **242** (2013), 869–888.

[4] G. Akrivis, M. Crouzeiz, and C. Makridakis, *Implicit-explicit multistep methods for quasilinear parabolic equations*, Numer. Math **82** (1999), no. 4, 521–541.

[5] I.S. Aranson and L. Kramer, *The world of the complex Ginzburg-Landau equation*, Rev. Mod. Phys. **74** (2002), 99–143.

[6] E.O. Asante-Asamani, *An exponential time differencing scheme with a real distinct poles rational function for advection-diffusion-reactions systems*, Ph.D. Thesis, 2016.

[7] E.O. Asante-Asamani, A.Q.M. Khaliq, and B.A. Wade, *A real distinct poles exponential time differencing scheme for reaction-diffusion systems*, Journal of Computational and Applied Mathematics **299** (2016), 24–34.

[8] E.O. Asante-Asamani and B.A. Wade, *A dimensional splitting of ETD schemes for reaction-diffusion systems*, Communications in Computational Physics **19** (2016), no. 5, 1343–1356.

[9] H.P. Bhatt and A.Q.M. Khaliq, *The locally extrapolated exponential time differencing LOD scheme for multidimensional reaction-diffusion systems*, Journal of Computational and Applied Mathematics **285** (2015), 256–278.

[10] H.P. Bhatt, A.Q.M. Khaliq, and B.A. Wade, *Efficient Krylov-based exponential time differencing method in application to 3D advection-diffusion-reaction systems*, Applied Mathematics and Computation **338** (2018), 260–273.

[11] A.G. Bratsos and A.Q.M. Khaliq, *A conservative exponential time differencing method for the nonlinear cubic Schrödinger equation*, International Journal of Computer Mathematics **94** (2017), no. 2, 230–251.

[12] L.Q. Chen and J. Shen, *Applications of semi-implicit Fourier-spectral method to phase field equations*, Comput. Physics Comm **108** (1998), no. 2–3, 147–158.

[13] M. Chen, *On the solution of circulant linear systems*, SIAM Journal on Numerical Analysis **24** (1987), no. 3, 668–683.

[14] S. Chen and Y.-T. Zhang, *Krylov implicit integration factor methods for spatial discretization on high dimensional unstructured meshes: Application to discontinuous Galerkin methods*, Journal of Computational Physics **230** (2011), no. 11, 4336–4352.

[15] Y. Cherruault, M. Choubane, J.M. Valleton, and J.C. Vincent, *Stability and asymptotic behavior of a numerical solution corresponding to a diffusion-reaction equation solved by a finite difference scheme (Crank-Nicolson)*, Computers & Mathematics with Applications **20** (1990), no. 11, 37–46.

[16] S.D. Conte and C.W. De Boor, *Elementary numerical analysis: An algorithmic approach*, 3rd ed., McGraw-Hill Higher Education, 1980.

[17] S.M. Cox and P.C. Matthews, *Exponential time differencing for stiff systems*, Journal of Computational Physics **176** (2002), no. 2, 430–455.

[18] M. Dehghan and D. Mirzaei, *The meshless local Petrov-Galerkin (MLPG) method for the generalized two-dimensional non-linear Schrödinger equation*, Engineering Analysis with Boundary Elements **32** (2008), no. 9, 747–756.

[19] Q. Du and W. Zhu, *Analysis and applications of the exponential time differencing schemes and their contour integration modifications*, BIT Numer Math **45** (2005), no. 2, 307–328.

[20] M.J. Gander, *50 years of Time Parallel Time Integration*, Multiple shooting and time domain decomposition, 2015.

[21] R.A. Gatenby and E.T. Gawlinski, *A reaction-diffusion model of cancer invasion*, Cancer Research **56** (1996), no. 24, 5745–5753.

[22] C.W. Gear and I.G. Kevrekidis, *Projective methods for stiff differential equations: Problems with gaps in their eigenvalue spectrum*, SIAM Journal on Scientific Computing **24** (2003), no. 4, 1091–1106.

[23] V.L. Ginzburg and L.D. Landau, *On the theory of superconductivity*, Zh. Eksp. Teor. Fiz. **20** (1950), 1064–1082.

[24] G.H. Golub and C.F. Van Loan, *Matrix computations*, 4th ed., Johns Hopkins University Press, 2013.

[25] D.J. Griffiths and D.F. Schroeter, *Introduction to quantum mechanics*, 3rd ed., Cambridge University Press, 2018.

[26] M. Hochbruck and A. Ostermann, *Exponential Runge-Kutta methods for parabolic problems*, Applied Numerical Mathematics **53** (2005), no. 2, 323–339.

[27] M. Hochbruck and A. Ostermann., *Explicit exponential Runge-Kutta methods for semilinear parabolic problems*, SIAM J. Numer. Anal. **43** (2006), no. 3, 1069–1090.

[28] W. Hundsdorfer and J. Verwer, *Numerical solution of time-dependent advection-diffusion-reaction equations*, Springer Series in Computational Mathematics, Springer, Berlin, 2003.

[29] A.-K. Kassam, *Solving reaction-diffusion equations 10 times faster*, 2003.

[30] A.-K. Kassam and L. Trefethen, *Fourth-order time-stepping for stiff PDEs*, SIAM Journal on Scientific Computing **26** (2005), no. 4, 1214–1233.

[31] A.Q.M. Khaliq, B. Kleefeld, and R.H. Liu, *Solving complex PDE systems for pricing American options with regime-switching by efficient exponential time differencing schemes*, Numerical Methods for Partial Differential Equations **29** (2013), no. 1, 320–336.

[32] A.Q.M. Khaliq, J. Martín-Vanquero, B.A. Wade, and M. Yousuf, *Smoothing schemes for reaction-diffusion systems with nonsmooth data*, Journal of Computational and Applied Mathematics **223** (2009), no. 1, 374–386.

[33] A.Q.M. Khaliq and B.A. Wade, *On smoothing of the Crank-Nicolson scheme for nonhomogeneous parabolic problems*, J. Comput. Meths. in Sci. Eng. **1** (2001), no. 1, 107–123.

[34] A.Q.M. Khaliq, B.A. Wade, M. Yousuf, and J. Vigo-Aguiar, *High order smoothing schemes for inhomogeneous parabolic problems with applications in option pricing*, Numerical Methods for Partial Differential Equations **23** (2007), no. 5, 1249–1276.

[35] B. Kleefeld, A.Q.M. Khaliq, and B.A. Wade, *An ETD Crank-Nicolson method for reaction-diffusion systems*, Numerical Methods for Partial Differential Equations **28** (2012), 1309–1335.

[36] B. Kleefeld and J. Martín-Vaquero, *SERK2v2: A new second-order stabilized explicit Runge-Kutta method for stiff problems*, Numerical Methods for Partial Differential Equations **29** (2013), no. 1, 170–185.

[37] ———, *SERK2v3: Solving mildly stiff nonlinear partial differential equations*, Journal of Computational and Applied Mathematics **299** (2016), 194–206.

[38] S. Kondo and T. Miura, *Reaction-diffusion model as a framework for understanding biological pattern formation*, Science **329** (2010), no. 5999, 1616–1620.

[39] D. Lu and Y.-T. Zhang, *Krylov integration factor method on sparse grids for high spatial dimension convection–diffusion equations*, J. Sci. Comput. **69** (2016), no. 2, 736–763.

[40] V.T. Luan, M. Tokman, and G. Rainwater, *Preconditioned implicit-exponential integrators (IMEXP) for stiff PDEs*, Journal of Computational Physics **335** (2017), 846–864.

[41] J. Martín-Vaquero and B. Janssen, *Second-order stabilized explicit Runge-Kutta methods for stiff problems*, Computer Physics Communications **180** (2009), 1802–1810.

[42] J. Martín-Vaquero, A.Q.M. Khaliq, and B. Kleefeld, *Stabilized explicit Runge-Kutta methods for multi-asset american options*, Computers & Mathematics with Applications **67** (2014), no. 6, 1293–1308.

[43] J. Martín-Vaquero and A. Kleefeld, *ESERK5: A fifth-order extrapolated stabilized explicit Runge-Kutta method*, Journal of computational and applied mathematics **356** (2019), 22–36.

[44] ———, *Extrapolated stabilized explicit Runge-Kutta methods*, Modelling for engineering and human behaviour 2018, 2019, pp. 325–331.

[45] J. Martín-Vaquero and B. Kleefeld, *Extrapolated stabilized explicit Runge–Kutta methods*, Journal of Computational Physics **326** (2016), 141–155.

[46] J. Martín-Vaquero and B.A. Wade, *On efficient numerical methods for an initial-boundary value problem with nonlocal boundary conditions*, Appl. Math. Modelling **36** (2012), no. 8, 3411–3418.

[47] B. Minchev and W. Wright, *A review of exponential integrators for first order semi-linear problems*, Technical Report **2** (2005).

[48] C. Moler and C.Van. Loan, *Nineteen dubious ways to compute the exponential of a matrix*, Siam Review **20** (1978), 801–826.

[49] C. Moler and C.F. van Loan, *Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later*, SIAM Review **45** (2003), no. 1, 3–49.

[50] J. Müller and C. Kuttler, *Methods and models in mathematical biology: Deterministic and stochastic approaches*, Springer, Heidelberg, 2015.

[51] J.D. Murray, *Mathematical biology I: An introduction*, 3rd ed., Interdisciplinary Applied Mathematics, vol. 17, Springer, New York, 2002.

[52] ———, *Mathematical biology II: Spatial models and biomedical applications*, 3rd ed., Interdisciplinary Applied Mathematics, vol. 18, Springer, New York, 2003.

[53] S. O'Sullivan, *A class of high-order Runge–Kutta–Chebyshev stability polynomials*, Journal of Computational Physics **300** (2015), 665–678.

[54] ———, *Runge-Kutta-Gegenbauer explicit methods for advection-diffusion problems*, J. Comput. Physics **388** (2019), 209–223.

[55] H. Rittich and R. Speck, *Time-parallel simulatin of the Schrödinger equation*, arXiv:1912.03312 (2019), 1–36.

[56] S.J. Ruuth, *Implicit-explicit methods for reaction-diffusion problems in pattern formation*, Journal of Mathematical Biology **34** (1995), no. 2, 148–176.

[57] E. Schödinger, *Quantisierung als Eigenwertproblem*, Annalen der Physik **384** (1926), no. 4, 361–377.

[58] L.F. Shampine, B.P. Sommeijer, and J.G. Verwer, *IRKC: An IMEX solver for stiff diffusion-reaction PDEs*, Journal of Computational and Applied Mathematics **196** (2006), no. 2, 485–497.

[59] J.A. Sherratt and J.D. Murray, *Models of epidermal wound healing*, Proc. R. Soc. Lond. B. **241** (1990), no. 1300, 29–36.

[60] B.P. Sommeijer, L.F. Shampine, and J.G. Verwer, *RKC: An explicit solver for parabolic PDEs*, Journal of Computational and Applied Mathematics **88** (1998), no. 2, 315–326.

[61] M. Tokman, *A new class of exponential propagation iterative methods of Runge-Kutta type (EPIRK)*, Journal of Computational Physics **230** (2011), no. 24, 8762–8778.

[62] E.H. Twizell, A.B. Gumel, and Q. Cao, *A second-order scheme for the "Brusselator" reaction–diffusion system*, Journal of Mathematical Chemistry **26** (1999), no. 4, 297–316.

[63] C.F. van Loan, *The ubiquitous Kronecker product*, Journal of Computational and Applied Mathematics **123** (2000), no. 1, 85–100.

[64] J. Vigo-Aguiar, J. Martín-Vaquero, and B.A. Wade, *Adapted BDF algorithms applied to parabolic problems*, Numerical Methods for Partial Differential Equations **23** (2007), no. 2, 350–365.

[65] D.A. Voss and A.Q.M. Khaliq, *Parallel LOD methods for second order time dependent PDEs*, Computers Math. Applic. **30** (1995), no. 10, 25–35.

[66] D. Wang, L. Zhang, and Q. Nie, *Array-representation integration factor method for high-dimensional systems*, Journal of Computational Physics **258** (2014), 585–600.

[67] M. Yousuf, A.Q.M. Khaliq, and B. Kleefeld, *The numerical approximation of nonlinear Black–Scholes model for exotic path–dependent American options with transaction cost*, International Journal of Computer Mathematics **89** (2012), no. 9, 1239–1254.

[68] P.A. Zegeling and H.P. Kok, *Adaptive moving mesh computations for reaction-diffusion systems*, Journal of Computational and Applied Mathematics **168** (2004), no. 1, 519–528.

[69] S. Zhao, J. Ovadia, X. Liu, Y.-T. Zhang, and Q. Nie, *Operator splitting implicit integration factor methods for stiff reaction-diffusion-advection systems*, Journal of Computational Physics **230** (2011), no. 15, 5996–6009.

[70] S. Zheng, *Nonlinear evolution equations*, Chapman & Hall/CRC Monographs and Surveys in Pure and Applied Mathematics **133** (2004).