mathematics

MDPI

Article

Serial and Parallel Iterative Splitting Methods: Algorithms and Applications to Fractional Convection-Diffusion Equations

Jürgen Geiser [1], Eulalia Martínez [2,*] and Jose L. Hueso [2]

[1] Department of Electrical Engineering and Information Technology, Ruhr-University of Bochum, 44801 Bochum, Germany; juergen.geiser@ruhr-uni-bochum.de

[2] Instituto Universitario de Matemática Multidisciplinar, Universitat Politècnica de València, 46022 Valencia, Spain; jlhueso@mat.upv.es

* Correspondence: eumarti@mat.upv.es

Received: 11 May 2020; Accepted: 22 October 2020; Published: 04 November 2020

Abstract: The benefits and properties of iterative splitting methods, which are based on serial versions, have been studied in recent years, this work, we extend the iterative splitting methods to novel classes of parallel versions to solve nonlinear fractional convection-diffusion equations. For such interesting partial differential examples with higher dimensional, fractional, and nonlinear terms, we could apply the parallel iterative splitting methods, which allow for accelerating the solver methods and reduce the computational time. Here, we could apply the benefits of the higher accuracy of the iterative splitting methods. We present a novel parallel iterative splitting method, which is based on the multi-splitting methods, The flexibilisation with multisplitting methods allows for decomposing large scale operator equations. In combination with iterative splitting methods, which use characteristics of waveform-relaxation (WR) methods, we could embed the relaxation behavior and deal better with the nonlinearities of the operators. We consider the convergence results of the parallel iterative splitting methods, reformulating the underlying methods with a summation of the individual convergence results of the WR methods. We discuss the numerical convergence of the serial and parallel iterative splitting methods with respect to the synchronous and asynchronous treatments. Furthermore, we present different numerical applications of fluid and phase field problems in order to validate the benefit of the parallel versions.

Keywords: multisplitting method; iterative splitting method; numerical analysis; operator-splitting method; initial value problem; iterative solver method; waveform relaxation method; convection-diffusion equation; viscous Burgers' equation; fractional diffusion equations

MSC: 35K45; 35K90; 47D60; 65M06; 65M55

1. Introduction

Nowadays, iterative splitting methods are important solver methods to solve large systems of ordinary, partial, or stochastic differential equations, see [1–6]. Iterative splitting methods are based on two solver ideas: In the first part, we separate the full operators into different sub-operators and reduce the computational time for such sub-computation. An additional benefit is the iterative technique, which allows for solving a relaxation problem, as in the waveform-relaxation method or Picard's iterative method, see [7–10]. Both parts reduce the computational time and the complexity as if we solved all parts (full operator and direct method) together, see [11]. Such iterative splitting methods can be used to compute, with less computational burden, an approximate solution of the ordinary differential equations (ODEs) or semi-discretized partial differential equations (PDEs), see [10,11].

Moreover, we consider parallel splitting methods, which are nowadays important to solve large problems, see [3]. Such parallel splitting methods are applied to the splitted subproblems and computed independently by the different processors. Therefore, in a second part we consider the parallelization-techniques, which are given with the multi-splitting approach, see [5]. Such approaches allow for embedding the splitting techniques of the first part into a parallelized version, see [12]. Based on the parallel versions of our methods, we can consider large computational problems.

For the computational problems, we consider the interesting part of higher dimensional and nonlinear PDEs as nonlinear and fractional convection-diffusion equations, see [13,14]. The nonlinear PDEs are applied in nonlinear flow problems, for example, to simulate traffic-flow problems, see [15] and flow problems that are related to Navier–Stokes equations, see [16], which can be modeled with the Burgers' equation.

Furthermore, we take into account the fractional PDEs, e.g., fractional convection-diffusion equations. Nowadays, the fractional calculus is applied to several fields of science and engineering, for example, in visco-elastic and thermal diffusion in fractal domains, see [17,18], or in phase-field models with mixtures of fluids [19,20], or in biological models to deal with fractional multi-models [21]. It is also interesting on a theoretical aspect, e.g., physical and geometrical interpretation [22], fractional Dirac operators [23].

The treatment of such delicate fractional and nonlinear partial differential equations (PDEs) needs additional spatial and nonlinear approximations of higher order, see [24]. We apply flexible iterative splitting methods, see [13,25], which can be extended to parallel algorithms.

The underlying modeling equations are given as nonlinear fractional convection-diffusion equations:

$$\partial_t u = -\nu\, u\nabla u + \mu\, (\mathcal{L}_x^\alpha u + \mathcal{L}_y^\beta u) + f(x,y,t),\ (x,y,t) \in \Omega \times [0,T], \tag{1}$$

$$u(x,y,0) = u_0(x,y,0),\ (x,y) \in \Omega, \tag{2}$$

$$u(x,y,t) = 0,\ (x,y,t) \in \partial\Omega \times [0,T], \tag{3}$$

where $\Omega \in \mathbb{R}^d$ is the spatial domain with d as the dimension and $T \in \mathbb{R}^+$. We denote by $\nu \in \mathbb{R}^+$ a scalar parameter of the convection part and by $\mu \in \mathbb{R}^+$ the viscosity of the diffusion part, while f is a right hand side function.

We have different cases:

- $\alpha = \beta = 2$: viscous Burgers' equation, see [2,26],
- $\alpha = \beta = 2$ and $\nu = 0$: Diffusion equation, see [2,26], and
- $\alpha, \beta \in (1,2)$: fractional diffusion equation, see [14,27].

Further interesting examples that are related to fractional and nonlinear PDE can be found in [24,28].

Equations (1)–(3) can be derived in a more general setting with the idea of the least action principle, see [29,30]. Here, we are not restricted by the specific Lagrangian or Hamiltonian principle, such that we can account for a much larger number of fractional differential equations.

Here, the fractional Laplace operator replaces the standard Laplacian operator, see [14], and it is denoted as a sum of one-dimensional spatial operators, $\mathcal{L}_{\alpha,\beta} = \mathcal{L}_x^\alpha + \mathcal{L}_y^\beta$:

$$\mathcal{L}_x^\alpha u := \frac{\partial^\alpha u}{\partial x^\alpha}, \tag{4}$$

where we have defined the Riemann fractional derivative of order α, as:

$$\frac{\partial^\alpha u}{\partial x^\alpha} = \frac{1}{\Gamma(2-\alpha)} \frac{\partial^2}{\partial x^2} \int_L^x \frac{u(\xi)}{(x-\xi)^{\alpha-1}} d\xi \tag{5}$$

In the next step, we apply a semi-discretisation with higher order finite-difference methods for $\alpha = \beta = 2$, see [31] or with higher order Grünwald formula for the fractional operators with $\alpha, \beta \in (1,2)$, see [27,32]. Thus, we could reduce the computational cost of the time-splitting approaches, which are given as an iterative splitting approach, see [33]. With such an approximation and the consideration of the semi-discretization with higher order schemes, we obtain the nonlinear differential equation system in a Cauchy-form, which is given as:

$$\partial_t c = \tilde{A}(c)\, c + \tilde{B}\, c + F(t),\ t \in [0,T], \tag{6}$$

$$c(0) = c_0, \tag{7}$$

where $c \in \mathcal{X} \subset \mathbb{R}^M$, while $M = m^d$, where d is the dimension and m is the number of grid-points in each spatial direction. Furthermore, $\tilde{A}(c) : \mathcal{X} \to \mathcal{X} \times \mathcal{X}$ is a nonlinear matrix, which includes the spatial discretization of the nonlinear convection in Equation (1) with the boundary conditions and with higher order spatial finite difference schemes, see [31]. The operator $\tilde{B} \in \mathcal{X} \times \mathcal{X}$ is a linear matrix, which includes the spatial discretization of the fractional diffusion term with the boundary conditions and with the underlying fractional discretization methods, see [24,32]. The function $F(t)$: $\mathbb{R}^+ \to \mathcal{X}$ is the discretized right-hand side. We assume $\mathcal{X}$ to be an appropriate Banach space with a vector and induced matrix norm $||\cdot||$, see [10,34,35].

For the nonlinear term, we apply the linearisation that is based on the Picard's iteration, which is also a special iterative splitting approach, see [7,36]. We have:

$$\partial_t v_j = \tilde{A}(v_{j-1})\, v_j + \tilde{B}\, v_j + F(t),\ t \in [0,T], \tag{8}$$

$$v(0) = c_0, \tag{9}$$

where $j = 1,2,\ldots,J$ with some $J \in \mathbb{N}$ (number of the iterations for the nonlinear part) and starting condition $v_0(t) = c_0$. Furthermore, we assume to have a sufficiently large J, such that $||v_J - v_{J-1}|| \leq err$, where err is an error-bound. For simplification, we assume $F(t) = 0$, while $F(t)$ is a right hand side, which can be approached by an integral equation, see [35].

Therefore, we obtain a linearised differential equation system, which is solved by the outer-iterations with $j = 1,\ldots,J$. In the following, we concentrate on solving such linearised evolution equations by decomposing them with respect to their underlying matrix-operator in submatrix-operators, so that we obtain the following differential equation,

$$\partial_t\, c = Ac = \sum_{l=1}^{L} A_l\, c,\ c(0) = c_0, \tag{10}$$

where $A = \tilde{A}(v_J) + \tilde{B} \in \mathbb{R}^M \times \mathbb{R}^M$ is the full operator, and A_l are the sub-operators. Further, $c \in C^1([0,T];\mathbb{R}^M)$ is the solution and $c_0 \in \mathbb{R}^M$ is the initial condition.

The bottleneck of the iterative methods is due to the large sizes of the iteration matrices, see [4,5]; therefore, we consider parallel versions of the iterative splitting method, see [3,12]. Based on the decomposition of the large scale differential equation with operator A into different smaller sub-differential equations with operators A_l, where $l = 1,\ldots,L$, and L is the number of processors, we distribute the computational time to many processors and reduce the overall computational time, see [5]. Furthermore, we modify the synchronous parallel splitting method with chaotic (asynchronous) ideas, such that the computation and communication of the various processors can be done independently, see [37].

In addition, we have considered different examples of the literature that also discuss optimized computational cost and error bounds, see [28]. Here, we deal with models of viscous Burgers' equation, which are applied in flow-problems, and fractional convection-diffusion equations, which are applied in diffusion interface phase field problems, see [20].

The outline of this paper is as follows. Section 2 explains the serial iterative splitting method. Section 3 introduces the parallel iterative splitting method. We discuss the theoretical results in Section 4. The numerical examples are presented in Section 5. In Section 6, we discuss the theoretical and practical results.

2. Serial Iterative Splitting Method

We consider a two-level iterative splitting method, which is discussed for two operators in [2] and for L operators in [38].

In this work, we also apply the recent theoretical results of our work in [26]. While in that work we took the adaptivity of the splitting results into account, here we consider the application of multi-operator splitting approaches and multi-splitting methods.

Based on the differential Equation (10), we have the following decomposition of the operator A:

- $A = \sum_{l=1}^{L} A_l$, while A_l are the sub-operators for the iterative part (solver part), and
- and $B_l = A - A_l$ are the sub-operators for the relaxation part (right hand side part).

where we have $l = 1, \ldots, L$.

The serial iterative splitting method is given in the following algorithm. Here, we consider discretization step-size $\tau = t^{n+1} - t^n$, which can also be set adaptively, see [26]. We assume to have time-interval $[t^n, t^{n+1}]$ with $n = 0, 1, \ldots, N$, where t_0 is the initial time and $t_N = T$ is the end time. We consecutively solve the following sub-problems for $i = 0, L, \ldots, (m-1)L$:

$$\frac{\partial c_{i+1}(t)}{\partial t} = A_1 c_{i+1}(t) \; + \; B_1 c_i(t), \text{ with } \; c_{i+1}(t^n) = c^n, \tag{11}$$

$$\frac{\partial c_{i+2}(t)}{\partial t} = A_2 c_{i+2}(t) \; + \; B_2 c_{i+1}(t), \text{ with } \; c_{i+2}(t^n) = c^n, \tag{12}$$

$$\ldots \tag{13}$$

$$\frac{\partial c_{i+L}(t)}{\partial t} = A_L c_{i+L}(t) \; + \; B_L c_{i+L-1}(t), \text{ with } \; c_{i+L}(t^n) = c^n \; , \tag{14}$$

where we can assume for the first initialisation $c_0(t) = 0$, or a different initial-function, see [2]. $c^n = c(t^n)$ is the known split approximation, which is computed in the previous iterative procedure. The current split approximation at time $t = t^{n+1}$ is defined as $c^{n+1} = c_{(m-1)L}(t^{n+1})$, where $(m-1)L$ is the maximal number of iterative steps. The stopping criterion is $||c_{(m-1)L} - c_{(m-2)L}|| \leq err$ and, then we have the solution $c(t^{n+1}) = c_{(m-1)L}(t^{n+1})$.

The solutions are given as:

$$c_{i+1}(t) = \exp(A_1 \; (t - t^n)) c^n + \int_{t^n}^{t} \exp(A_1 \; (t-s)) \; B_1 \; c_i(s) \; ds, \tag{15}$$

$$c_{i+2}(t) = \exp(A_2 \; (t - t^n)) c^n + \int_{t^n}^{t} \exp(A_2 \; (t-s)) \; B_2 \; c_{i+1}(s) \; ds, \tag{16}$$

$$\ldots, \tag{17}$$

$$c_{i+L}(t) = \exp(A_L \; (t - t^n)) c^n + \int_{t^n}^{t} \exp(A_L \; (t-s)) \; B_L \; c_{i+L-1}(s) \; ds, \tag{18}$$

where $t \in [t^n, t^{n+1}]$.

Remark 1. *We approximate the integral operator with a higher order integration scheme, e.g., exp-matrix computations, see [39,40], exp-Runge–Kutta methods, see [41], Pade- or Magnus-expansions, see [42,43] or with higher order numerical integration methods, as Trapezoidal- or Simpsons-rule, see [44].*

We define the error-function as $e_i(t) = c(t) - c_i(t)$ with $e_0(t) = c(t) - c_0$, the maximum-norm $||e_i|| = \max_{t \in [0,T]} ||e_i(t)||_\infty$, and the maximum operator norm $||A_l|| = ||A_l||_\infty$.

Theorem 1. *Consider the bounded operators $A_l \in \mathbb{R}^m \times \mathbb{R}^m$ for $l = 1, \ldots, L$, where L is the number of operators. The iterations (11)–(14) for the Cauchy-problem (10), which are applied with $i = 0, L, \ldots, (m-1)L$, are of order $\mathcal{O}(\tau^{mL})$.*

Proof. The result for the 2-level method is given in [38]. We apply a recursive argument to the iterative scheme with L operators and obtain:

$$||e_{mL}|| \leq \left(\Pi_{l=1}^{L} C_l ||B_l||\right)^m ||e_0||, \tag{19}$$

where C_l is given with $C_l = \mathcal{O}(\tau)$, see also [38]. □

Remark 2. *The operators are based on the spatial discretization with higher order methods, e.g., [27,31]. Further, we assume that all of the operators, including the fractional discretized operators, are bounded, see [24]. Based on such assumption, we could generalize the results with respect to fractional operators.*

3. Parallel Iterative Splitting Method

In the following, we parallelise the serial iterative splitting approach.
We present the following approaches:

- multi-splitting iterative approach, and
- two operator iterative splitting approach.

3.1. Multi-Splitting Iterative Approach

The problem is given as $\frac{\partial c}{\partial t} = Ac(t) = \sum_{l=1}^{L} A_l c(t),\ c = c_0$.
The idea is a multiple decomposition of

$$A = A_l + B_l,\ l = 1, \ldots, L, \tag{20}$$
$$B_l = A - A_l, \tag{21}$$

where A_l is a non-singular and B_l is the rest matrix.

Further, we have the decomposition of the parallel computable vectors:

$$c_i = \sum_{l=1}^{L} E_l c_{i,l}, \text{ and } E = \sum_{l=1}^{L} E_l, \tag{22}$$

where c_i is the i-th iterative solution and $c_{i,l}$ are the parallel computable solutions in the i-th iterative step. E is the identity matrix and E_l are diagonal matrices with positive entries.

The multisplitting iterative approach is given as:

$$\begin{aligned} &\frac{\partial c_{i,l}(t)}{\partial t} = A_l c_{i,l}(t) \ + \ B_l c_{i-1}(t), \text{ for } t \in [t^n, t^{n+1}] \\ &\text{with } \ c_{i,l}(t^n) = c(t^n), l = 1, \ldots L, \end{aligned} \tag{23}$$

where the initialisation is $c_0(t) = c(t^n)$, and $i = 1, \ldots, I$ are the iterative steps. The stopping criterion is $||c_i - c_{i-1}|| \leq err$ and then we have the solution $c(t^{n+1}) = c_i(t^{n+1})$.

The splitting error of the iterative splitting is of $k+1$ order, i.e. $O(\tau^{k+1})$, with

$$||err_{i+1}|| = K_i \tau_n^i ||err_0|| + O(\tau_n^{i+1}) \ , \tag{24}$$

where $K_i = \sum_{l=1}^{L} \frac{1}{\omega_l} \ ||B_l||$ and $\sum_{l=1}^{L} \frac{1}{\omega_l} = 1$ are the weights ($\omega_l > 1$) and $||err_0|| = ||c_0(t)||$, while $c_{-1}(t) = 0$.

Benefit:

- Parallel implementation (the method is designed for parallel contributions), and
- Good error balance between the different operators

Drawback:

- Balances in the decomposition of E_l important to damp large errors

3.2. Parallel Splitting: Classical Version (Synchronous Version)

Here, we consider the classical version of the parallel splitting algorithm that is applied with synchronisation, see also [45]. We assume to deal with L processors and $i = 1, \ldots, I$ are the iterative steps, while I is the maximum number of iterative steps.

We deal with the parallel splitting in the synchronous version as:

$$\frac{\partial c_{i,1}(t)}{\partial t} = A_1 c_{i,1}(t) \; + \; B_1 c_{i-1}(t), \text{ with } \; c_{i,1}(t^n) = c^n, \tag{25}$$

$$\frac{\partial c_{i,2}(t)}{\partial t} = A_2 c_{i,2}(t) \; + \; B_2 c_{i-1}(t), \text{ with } \; c_{i,2}(t^n) = c^n, \tag{26}$$

$$\ldots \tag{27}$$

$$\frac{\partial c_{i,L}(t)}{\partial t} = A_l c_{i,L}(t) \; + \; B_l c_{i-1}(t), \text{ with } \; c_{i,L}(t^n) = c^n, \tag{28}$$

$$c_i(t) = \sum_{l=1}^{L} E_l c_{i,l}(t), \text{and } E = \sum_{l=1}^{L} E_l, \tag{29}$$

where we assume for the initialisation of the first step $c_0(t) = 0$. E is the identity matrix and E_l are diagonal matrices with positive entries. A_l and B_l are given in the Equations (20) and (21). Furthermore, $c^n = c(t^n)$ is the known split approximation at the time-level $t = t^n$, which is computed in the previous iterative process. The split approximation at the time-level $t = t^{n+1}$ is defined as $c^{n+1} = c_i(t^{n+1})$, which is computed in the current iterative process.

The solutions are given as:

$$c_{i,1}(t) = \exp(A_1 \; (t - t^n))c^n + \int_{t^n}^{t} \exp(A_1 \; (t - s)) \; B_1 \; c_{i-1}(s) \; ds, \tag{30}$$

$$c_{i,2}(t) = \exp(A_2 \; (t - t^n))c^n + \int_{t^n}^{t} \exp(A_2 \; (t - s)) \; B_2 \; c_{i-1}(s) \; ds, \tag{31}$$

$$\ldots, \tag{32}$$

$$c_{i,L}(t) = \exp(A_L \; (t - t^n))c^n + \int_{t^n}^{t} \exp(A_L \; (t - s)) \; B_L \; c_{i-1}(s) \; ds, \tag{33}$$

$$c_i(t) = \sum_{l=1}^{L} E_l c_{i,l}(t), \text{and } E = \sum_{l=1}^{L} E_l, \tag{34}$$

where $t \in [t^n, t^{n+1}]$.

The stopping criterion is given as:

$$||c_i - c_{i-1}|| \leq err. \tag{35}$$

The integrals can be solved by higher order integration-rules, see Remark 1.

In the classical algorithm, we have a synchronisation point, so that the next iterative step can only start if all processors have submitted the results, see Equation (34). Such a hard point allows for applying a simpler stopping criterion, which is given in Equation (35).

The bottleneck of the synchronous algorithm is that the finished processors could not go on with the next iterative step and the computational time is wasted. Therefore, we explain the ideas of the asynchronous algorithms, which are used to apply the parallel splitting method in an asynchronous version.

3.3. Asynchronous Algorithm

The idea of an asynchronous algorithm is that each processor works independently and has access to a common memory. If one processor needs an update of a solution of another processor, then he can read the solution in the common memory. The processors are independent and the convergence is given with the weighted norm, see [3].

An asynchronous algorithm is defined, as follows:

Definition 1. *For $i \in \mathbb{N}$, let $\{I(i)\}_{i \in \mathbb{N}}$ be the subset indicating which components are computed at the i-th iteration. Additionally, we have an iteration count $s_l(i) \in \mathbb{N}_0$ (prior to i), which indicates the iteration when the l-th component was computed in processor l.*

For $i \in \mathbb{N}$, we have $I(i) \in \{1, \ldots, L\}$, where L is the number of processors and $(s_1(i), \ldots, s_L(i)) \in \mathbb{N}_0^L$, such that:

$$s_l(i) \leq i - 1, \text{ for } l \in \{1, \ldots, L\},\ i \in \mathbb{N}, \tag{36}$$

$$\lim_{i \to \infty} s_l(i) = \infty, \text{ for } l \in \{1, \ldots, L\}, \tag{37}$$

$$|\{i \in \mathbb{N} : l \in I(i)\}| = \infty, \text{ for } l \in \{1, \ldots, L\}. \tag{38}$$

Subsequently, we define the asynchronous algorithm:

$$x_l^i = \begin{cases} x_l^{i-1}, & \text{for } l \notin I(i), \\ H_l(x_1^{s_1(i)}, \ldots, x_L^{s_L(i)}), & \text{for } l \in I(i), \end{cases} \tag{39}$$

where $\mathbf{x}^0 = (x_1^0, \ldots, x_L^0)^t$ are the initialisations and H_l is the solver function of the l-th component, see [3].

In the following, the convergence of the asynchronous algorithm is presented with respect to the weighted norm, see [3].

Definition 2. *We assume that each component space $\mathcal{E}_i$ has a normed linear space $(\mathcal{E}_i, ||\cdot||_i)$. We can define an appropriate norm for the parallel methods, which can be given as the weighted maximum norm:*

$$||x||_w = \max_{i=1}^{m} \frac{||x_i||_i}{w_i} \tag{40}$$

where the vector $w = (w_1, \ldots, w_m)^t$ is positive in each component $w_i > 0$ for all $i = 1, \ldots, m$.

Theorem 2. *We assume that we have a fixed point $\mathbf{x}^* \in \mathcal{E} = \mathcal{E}_1 \times \ldots \times \mathcal{E}_L$ with $\mathbf{H}^i(\mathbf{x}^*) = \mathbf{x}^*$ for all i. Further, we assume that there exists $\gamma \in [0, 1)$ and a positive vector $w \in \mathbb{R}^m$, such that:*

$$||\mathbf{H}^i(\mathbf{x}) - \mathbf{x}^*||_w \leq \gamma\ ||\mathbf{x} - \mathbf{x}^*||_w. \tag{41}$$

Based on the assumptions, we conclude that the asynchronous iterates x^k converge to x^*, which is the unique fixed point of $\mathbf{H}^i = (H_1^i, \ldots, H_l^i)^t$.

Proof. The proof is given in [3]. □

3.4. Parallel Splitting: Modern Version (Asynchronous Version)

We have to apply the following asynchronous algorithm, which is given in Definitions 1 and 2 and with the convergence results in Theorem 2. We assume to deal with L processors and $i = 1, \ldots, I$ are the iterative steps, while I is the maximum number of iterative steps.

We deal with the parallel splitting in the synchronous version as:

$$\frac{\partial c_{s_1(i+1)}(t)}{\partial t} = A_1 c_{s_1(i+1)}(t) \; + \; B_1 c_i(t), \text{ with } \; c_{s_1(i+1)}(t^n) = c^n, \tag{42}$$

$$\frac{\partial c_{s_2(i+1)}(t)}{\partial t} = A_2 c_{s_2(i+1)}(t) \; + \; B_2 c_i(t), \text{ with } \; c_{s_L(i+1)}(t^n) = c^n, \tag{43}$$

$$\ldots \tag{44}$$

$$\frac{\partial c_{s_L(i+1)}(t)}{\partial t} = A_L c_{s_L(i+1)}(t) \; + \; B_l c_i(t), \text{ with } \; c_{s_l(i+1)}(t^n) = c^n, \tag{45}$$

$$c_{i+1,l}(t) = \begin{cases} x_l^i, & \text{for } l \notin I(i+1), \\ x_l^{s_l(i+1)}, & \text{for } l \in I(i+1), \end{cases} \tag{46}$$

$$c_{i+1}(t) = \sum_{l=1}^{L} E_l c_{i+1,l}(t), \text{ and } E = \sum_{l=1}^{L} E_l, \tag{47}$$

where we assume for the initialisation of the first step $c_0(t) = 0$. E is the identity matrix and E_l are diagonal matrices with positive entries. A_l and B_l are given in Equations (20) and (21). Further, c^n is the known split approximation at time $t = t^n$, which is computed in the previous iterative process. The split approximation at time $t = t^{n+1}$ is given as $c^{n+1} = c_i(t^{n+1})$, which is computed in the current iterative process.

The solutions are given as:

$$c_{s_1(i+1)} = \exp(A_1 \; (t - t^n))c^n + \int_{t^n}^{t} \exp(A_1 \; (t - s)) \; B_1 \; c_i(s) \; ds, \tag{48}$$

$$c_{s_2(i+1)} = \exp(A_2 \; (t - t^n))c^n + \int_{t^n}^{t} \exp(A_2 \; (t - s)) \; B_2 \; c_i(s) \; ds, \tag{49}$$

$$\ldots, \tag{50}$$

$$c_{s_L(i+1)}(t) = \exp(A_L \; (t - t^n))c^n + \int_{t^n}^{t} \exp(A_L \; (t - s)) \; B_L \; c_i(s) \; ds, \tag{51}$$

$$c_{i+1,l}(t) = \begin{cases} x_l^i, & \text{for } l \notin I(i+1), \\ x_l^{s_l(i+1)}, & \text{for } l \in I(i+1), \end{cases} \tag{52}$$

$$c_{i+1}(t) = \sum_{l=1}^{L} E_l c_{i+1,l}(t), \text{ and } E = \sum_{l=1}^{L} E_l, \tag{53}$$

where $t \in [t^n, t^{n+1}]$.

The stopping criterion is given as:

$$||c_{i+1} - c_i||_w \leq \max_{l=1}^{L} \frac{||c_{i+1,l} - c_{i,l}||}{w_l} \leq err, \tag{54}$$

where $w = (w_1, \ldots, w_L)^t$ and we have the maximum-norm with $w_l = 1$ for all $l = 1, \ldots, L$.

The integrals can be solved by higher order integration-rules, see Remark 1.

In the modern algorithm, we do not have a synchronisation point, so that each processor can work independently. For the stopping criterion, we apply a weighted norm, which means that all of the single results of the processors have to be lower than the given error-bound, see the stopping criterion in Equation (54).

4. Theoretical Results

In the following, we deal with the m-dimensional initial value problem in the non-homogeneous form, also see the homogeneous form in Equation (10):

$$c'(t) = Ac(t) + f(t),\ c(0) = c_0, \tag{55}$$

where A is conveniently decomposed in two operators $A = M + N$, and f is the right hand side.

Further, we deal in the following with the proof-ideas that are related to Waveform-relaxation methods, see [37,45].

The initial value problem (55) is solved with the multisplitting Waveform-relaxation method, which is given as:

$$c'_{i+1}(t) = Mc_{i+1}(t) + Nc_i(t) + f(t),\ c(0) = c_0, \tag{56}$$

where A is given in Equation (10). Further, $c_0(t) = c_0$ is the starting condition.

For the multisplitting approach, we have the following Definition:

Definition 3. *Let $L \geq 1$ be the number of splittings, and A, A_l, B_l, E_l real-valued $m \times m$ matrices. We say that (A_l, B_l, E_l) for $l = 1, \ldots, L$ is a multisplitting triple if:*

- *$A = A_l + B_l$ and $B_l = \sum_{k=1, k\neq l}^{L} A_k$ with $l = 1, \ldots, L$,*
- *The matrices E_l are non-negative diagonal matrices and satisfy: $\sum_{l=1}^{L} E_l = I$,*
  where I is the identity matrix, and
- *$s_l(i+1) \leq i+1$ indicates the iteration, where the l-th component is computed prior to $i+1$.*

- The multisplitting approach that is based on the Waveform-relaxation in the classical version is given by:

$$c'_{l,i+1}(t) = A_l c_{l,i+1}(t) + B_l c_i + f(t),\ c_{l,i+1}(0) = c_0, \tag{57}$$

$$c_{i+1}(t) = \sum_{l=1}^{L} E_l c_{l,i+1}(t). \tag{58}$$

- The multisplitting approach based on the Waveform-relaxation in the modern version is given by:

$$c'_{s_l(i+1)}(t) = A_l c_{s_l(i+1)}(t) + B_l c_i + f(t),\ c_{s_l(i+1)}(0) = c_0, \tag{59}$$

$$c_{i+1}(t) = \sum_{l=1}^{L} E_l c_{s_l(i+1)}(t). \tag{60}$$

4.1. Stability Analysis

We deal with the following system:

$$c_{l,i+1}(t) = K_l c_i(t) + \phi_l(t), \tag{61}$$

where we have

$$K_l c(t) = \int_0^t k_l(t-s)\ c(s) ds,\ \text{for } l = 1, \ldots, L, \tag{62}$$

$$\phi_l(t) = \exp(tA_l)c_0 + \int_0^t \exp((t-s)A_l) f(s)\ ds,\ \text{for } l = 1, \ldots, L. \tag{63}$$

where $k_l(t) = \exp(tA_l)\ B_l$ for $l = 1, \ldots, L$, and we apply the multisplitting notation (58), with:

$$Kc(t) = \sum_{l=1}^{L} E_l K_l c(t), \tag{64}$$

$$\phi(t) = \sum_{l=1}^{L} E_l \phi_l(t), \tag{65}$$

where $k(t) = \sum_{l=1}^{L} E_l k_l(t)$ and we obtain the standard Waveform-relaxation method as:

$$c_{i+1}(t) = Kc_i(t) + \phi(t). \tag{66}$$

We can rewrite into an recursive notation and without loss of generality, we assume $f(t) = 0$, then we obtain:

$$c_{i+1}(t) = \sum_{l=1}^{L} E_l \int_0^t \exp((t-s)A_l) B_l c_i(s) ds + \sum_{l=1}^{L} E_l \exp(tA_l) c_0. \tag{67}$$

Given a well-conditioned system of eigenvectors,we can consider the eigenvalues $\lambda_{1,l}$ of A_l and $\lambda_{2,l}$ of B_l instead of the operators A_l, B_l themselves, for $l = 1, \ldots, L$. For the matrices E_l we have the eigenvalues λ_{E_l} with $0 \leq \lambda_{E_l} \leq 1$ and $\sum_{l=1}^{L} \lambda_{E_l} = 1$.

We can rewrite into the eigenvalue-notation and obtain:

$$c_{i+1}(t) = \sum_{l=1}^{L} \lambda_{E_l} \int_0^t \exp((t-s)\lambda_{A_l}) \lambda_{B_l} c_i(s) ds + \sum_{l=1}^{L} \lambda_{E_l} \exp(t\lambda_{A_l}) c_0. \tag{68}$$

We assume that all of the initial values $c_i(t^n) = c_{approx}(t^n)$ with $i = 0, 1, 2, \ldots$, are as $||c_{approx}(t_n) - c_n|| \leq O(\tau^I)$ where I is the order, following the ideas in the iterative splitting approach [46].

Further, we also assume that the pairs $\lambda_{1,l} \neq \lambda_{2,l}$ for $l = 1, \ldots, L$, otherwise we do not consider the iterative splitting approach, while the time-scales are equal, see [34].

In the following, we apply the $A(\alpha)$-stability.

4.1.1. $A(\alpha)$-Stability

We define $z_{1,l} = \tau\lambda_{1,l}$ and $z_{2,l} = \tau\lambda_{2,l}$ with $\tau = t^{n+1} - t^n$, $l = 1, \ldots, L$.

We have the following proposition 1:

Proposition 1. *Starting with $c(t^n) = c_n$ and a time-step $\tau = t^{n+1} - t^n$, we obtain:*

$$c_i(t^{n+1}) \quad = \quad \sum_{i=0}^{i} S_j(z_{1,l}, z_{2,l}, \tau)\ c_n, \tag{69}$$

where S_i is the stability function of the scheme. The S_i are given as:

$$S_0(z_{1,l}, z_{2,l}, t) = \sum_{l=1}^{L} \lambda_{E_l} \exp(t\lambda_{1,l}) c_n. \tag{70}$$

$$S_1(z_{1,l}, z_{2,l}, t) = \sum_{l=1}^{L} \lambda_{E_l} \int_0^t \exp((t-s)\lambda_{1,l}) \lambda_{2,l} \sum_{l_1=1}^{L} \lambda_{E_{l_1}} \exp(s\lambda_{1,l_1}) c_n\ ds,$$

$$S_1(z_{1,l}, z_{2,l}, t) = \sum_{l=1}^{L} \lambda_{E_l} \int_0^t \exp((t-s)\lambda_{1,l}) \lambda_{2,l} S_0(z_{1,l}, z_{2,l}, s) c_n\ ds. \tag{71}$$

and

$$S_i(z_{1,l}, z_{2,l}, t) = \tag{72}$$
$$= \sum_{l=1}^{L} \lambda_{E_l} \int_0^t \exp((t-s)\lambda_{1,l})\lambda_{2,l} \left(\sum_{l_1=1}^{L} \lambda_{E_{l_1}} \int_0^s \exp((s-s_1)\lambda_{1,l_1})\lambda_{1,l_1} \cdot \right.$$
$$\left. \cdot \left(\dots (\sum_{l_i=1}^{L} \lambda_{E_{l_1}} \int_0^{s_{i-1}} \exp((s_{i-1}-s_i)\lambda_{1,l_i})\lambda_{2,l_i} c_n \, ds_i) \dots ds_1 \right) ds \right),$$
$$S_i(z_{1,l}, z_{2,l}, t) = \sum_{l=1}^{L} \lambda_{E_l} \int_0^t \exp((t-s)\lambda_{1,l})\lambda_{2,l} S_{i-1}(z_{1,l}, z_{2,l}, s) c_n \, ds. \tag{73}$$

with i-iterations.

Proof. We apply the complete induction.

We start with $i = 0$ and obtain:

$$c_0(t^{n+1}) = \sum_{i=0}^{0} S_j(z_{1,l}, z_{2,l}, \tau) \; c_n, \tag{74}$$
$$c_0(t^{n+1}) = \sum_{l=1}^{L} \lambda_{E_l} \exp((t\lambda_{A_l})c_n. \tag{75}$$

We apply the induction step $i \to i+1$, while we apply Equation (68):

$$c_{i+1}(t^{n+1}) = S_0(z_1, z_2, \tau)c_n + \int_0^t \sum_{l=1}^{L} \lambda_{E_l} \exp((t-s)\lambda_{1,l}) \; \lambda_{2,l} c_i(s) ds, \tag{76}$$
$$c_{i+1}(t^{n+1}) = S_0(z_1, z_2, \tau)c_n + \int_0^t \sum_{l=1}^{L} \lambda_{E_l} \exp((t-s)\lambda_{1,l}) \; \lambda_{2,l} \sum_{j=0}^{i} S_j(z_1, z_2, s) c_n ds, \tag{77}$$

we apply the Equation (71) and obtain:

$$c_{i+1}(t^{n+1}) = S_0(z_1, z_2, \tau)c_n + \sum_{j=1}^{i+1} S_j(z_1, z_2, s) c_n ds, \tag{78}$$
$$c_{i+1}(t^{n+1}) = \sum_{j=0}^{i+1} S_j(z_1, z_2, s) c_n ds, \tag{79}$$

and we obtain the results. □

Let us consider the $A(\alpha)$-stability that is given by the following eigenvalues in a wedge:

$$\mathcal{W} = \{\zeta \in \mathbb{C} : |\arg(\zeta) \leq \alpha\}.$$

For the A-stability we have $|S_m(z_1, z_2)| \leq 1$ whenever $z_1 \in \mathcal{W}_{\pi/2}$. This means that we have the stiff operator, where we assume that z_2 is the non-stiff operator.

The stability of the two iterations is given in the following theorem with respect to A and $A(\alpha)$-stability.

Theorem 3. *We have the following stability for the iterative operator splitting scheme (71):*

For the stability function S_i, where i is the iterative step, we have the following A-stability

$$\max_{z_1 \in W_\alpha, z_2 \leq 0} |S_i(z_1, z_2)| \leq 1 \; , \; \forall \; \alpha \in [0, \pi/2], \tag{80}$$

with $\omega \in [0,1]$, the initialization is given as $c_{-1} = 0$ and the initial conditions are $c_i(t^n) = c_n$.

Proof. We consider the $z_1 \to -\infty$, while $z_2 \leq 0$ is bounded as a nonstiff operator.

We apply the complete induction.

We start with $i = 0$:

$$|S_0(-\infty, z_2, t)| = |\sum_{l=1}^{L} \lambda_{E_l} \exp(t\lambda_{1,l})| \leq 1, \tag{81}$$

with the assumption of the nonstiff operators $\lambda_{2,l} \leq 0$ with $l = 1, \ldots, L$.

Further we also have $i = 1$:

$$|S_1(-\infty, z_1, t)| = |\sum_{l=1}^{L} \lambda_{E_l} \int_0^t \exp((t-s)\lambda_{1,l})\lambda_{2,l} S_0(z_{1,l}, z_{2,l}, s)\ ds| \tag{82}$$

$$\leq |\sum_{l=1}^{L} \lambda_{E_l}\ t\ \exp(t\lambda_{1,l})\lambda_{2,l}|S_0(z_{1,l}, z_{2,l}, t)| = 0 \leq 1. \tag{83}$$

Subsequently, we apply the induction step for $i \to i+1$:

$$|S_{i+1}(z_1, z_2, t)| = ||S_0(z_1, z_2, \tau) + \int_0^t \sum_{l=1}^{L} \lambda_{E_l} \exp((t-s)\lambda_{1,l})\ \lambda_{2,l} \sum_{j=0}^{i} S_j(z_1, z_2, s) ds|| \tag{84}$$

$$\leq ||S_0(z_1, z_2, \tau)|| + ||\sum_{l=1}^{L} \lambda_{E_l}\ t\ \exp(t\lambda_{1,l})\ \lambda_{2,l} \sum_{j=0}^{i} S_j(z_1, z_2, t)|| = 0 \leq 1, \tag{85}$$

where we applied $||\sum_{j=0}^{i} S_j(z_1, z_2, t)|| = 0$.

Afterwards, we obtain our results, also see the ideas of the stability proofs in [47]. □

4.2. Convergence Analysis

In order to apply the multisplitting Waveform-relaxation method (57) and (58), we write the solutions of the individual Equation (57) as:

$$c_{l,i+1}(t) = K_l c_i(t) + \phi_l(t), \tag{86}$$

where we have

$$K_l c(t) = \int_0^t k_l(t-s)\ c(s) ds, \text{ for } l = 1, \ldots, L, \tag{87}$$

$$\phi_l(t) = \exp(tA_l)c_0 + \int_0^t \exp((t-s)A_l) f(s)\ ds, \text{ for } l = 1, \ldots, L. \tag{88}$$

where $k_l(t) = \exp(tA_l)B_l$ for $l = 1, \ldots, L$.

Further, we apply the multisplitting notation (58) and obtain the summations:

$$Kc(t) = \sum_{l=1}^{L} E_l K_l c(t), \tag{89}$$

$$\phi(t) = \sum_{l=1}^{L} E_l \phi_l(t), \tag{90}$$

where $k(t) = \sum_{l=1}^{L} E_l k_l(t)$ and we obtain the standard Waveform-relaxation method as:

$$c_{i+1}(t) = Kc_i(t) + \phi(t). \tag{91}$$

We assume that Lemma 1 is fulfilled, see also [45].

Lemma 1. *The following items are equivalent:*

- $c(t)$ *is a solution of the initial value problem (55).*
- $c(t)$ *is a solution of each multisplitted equation* $c(t) = K_l c(t) + \phi_l(t),\ c(0) = c_0,\ l = 1, \ldots, L.$
- $c(t)$ *is the solution of the fixed point equation* $c(t) = Kc(t) + \phi_l(t).$

We define $||c||_T = \max_{t\in[0,T]} |c(t)|$ as maximum norm and we also denote by $||\cdot||$ the matrix norm induced by the vector norm $|\cdot|$.

Based on these assumptions, in the following we derive the errors and convergence results.

In Theorem 4, we derive the error of the i-th approximation, see also [45].

Theorem 4. *There exists a constant* $C := \sum_{l=1}^{L} C_l$*, which is given to estimate the kernel k of the multisplitting waveform-relaxation operator, such that we obtain* $||k||_T = C$*. Subsequently, the error of the i-th approximation of the classical multisplitting WR method (57) and (58) is given by*

$$||c^i - c||_T \leq \frac{(CT)^i}{i!}(\exp(CT)||\phi||_T + ||c_0||_T). \tag{92}$$

Proof. We have given

$$c_i(t) = Kc_{i-1}(t) + \phi(t), \tag{93}$$

We apply the Lemma 1 and follow with the iterative approach:

$$c_i(t) = K^i c_0(t) + \sum_{j=0}^{i-1} K^i \phi(t), \tag{94}$$

where is $K^i u(t)$ is the i-times convolution

$K^i u(t) = \int_0^t k(t-s_i)(\int_0^{s_i} k(t-s_i-s_{i-1})(\ldots \int_0^{t-\sum_{j=0}^{i} s_{i-j}} k(t-\sum_{j=0}^{i} s_{i-j})u(s_1)d_{s_1}\ldots)ds_{i-1})ds_i.$

Further, we have $||u||_T = \max_{t\in[0,t]} |u(t)|$, where $|\cdot|$ is an appropriate Banach-Norm.

There exists:

$$||k_l||_T \leq C_l, \text{ for } l = 1, \ldots, L, \tag{95}$$

and we have

$$|k(t)| \leq ||k||_T = ||\sum_{l=1}^{L} E_l k_l|| \leq \sum_{l=1}^{L} C_l = C. \tag{96}$$

We apply the estimation of the Waveform-relaxation, see [8], and obtain:

$$||K^i||_T \leq \frac{(CT)^i}{i!}, \tag{97}$$

where we have $\lim_{i\to\infty} ||K^i||_T \to 0$.

The error estimate is then given as

$$||c - c^i||_T = ||(\lim_{j\to\infty} K^j c_0 - \sum_{j=0}^{\infty} K^j \phi(t)) - (K^i c_0 - \sum_{j=0}^{i-1} K^j \phi(t))||$$
$$\leq ||\sum_{j=i}^{\infty} K^j \phi(t) + K^i c_0|| \leq ||K^i||_T (||\sum_{j=0}^{\infty} ||K^j||_T \, ||\phi(t)||_T + ||c_0||_T). \tag{98}$$

We apply

$$\sum_{j=0}^{\infty} ||K^j||_T = ||K^0||_T + ||K^1||_T + \ldots + ||K^\infty||_T \leq \exp(CT). \tag{99}$$

Subsequently, we obtain the estimation

$$||c - c^i||_T =\leq \frac{(CT)^i}{i!}(\exp(CT)||\phi||_T + ||c_0||_T). \tag{100}$$

□

We have the following new convergence Theorem 5 based on the extension of the classical convergence Theorem version 4.

Theorem 5. *There exists a constant $C := \sum_{l=1}^{L} C_l$, which is given to estimate the kernel k of the multisplitting waveform-relaxation operator, such that we obtain $||k||_T = C$. Subsequently, the convergence of the modern multisplittting WR method (59) and (60) is given by*

$$||c^{i_{\min}} - c||_T \leq \frac{(C\ T)^{i_{\min}}}{i_{\min}!}||c^0 - c||_T, \tag{101}$$

where $i_{\min} = \min_{l=1}^{L} s_l(i)$, where $s_l(i) \leq i$ are the retarded iterations of the l-th processor.

Proof. We start with the estimation of the i-th iteration:

$$||c^{i_{min}} - c||_T \leq ||\sum_{l=1} E_l K_l (c^{s_l(i)} - c)||_T \leq \tag{102}$$
$$\leq ||\sum_{l=1} E_l K_l (c^{i_{min}-1} - c)||_T \leq ||K||_T \, ||(c^{i_{min}-1} - c)||_T. \tag{103}$$

Afterwards, we have the recursion:

$$||c^{i_{min}} - c||_T \leq ||K^{i_{min}}||_T ||(c^0 - c)||_T. \tag{104}$$

where we apply $||K^{i_{min}}||_T \leq \frac{(C\ T)^{i_{min}}}{i_{min}!}$, based on the idea in [8], and we obtain:

$$||c^{i_{min}} - c||_T \leq \frac{(C\ T)^{i_{min}}}{i_{min}!} \, ||(c^0 - c)||_T, \tag{105}$$

where $i_{\min} = \min_{l=1}^{L} s_l(i)$. □

Remark 3. *For the parallel error, we obtain order $\mathcal{O}(\tau^m)$ if we assume that all processors have at least m iterative cycles, while, for the serial error, we have order $\mathcal{O}(\tau^{mL})$. Thus, in the serial version, we have to apply mL iterative steps in sum to obtain the result, while in the parallel version, we only apply m iterative steps, while L processors share the computation to solve the L sub-equations. Furthermore, we can assume that the sub-equations are faster to solve, because the sub-operators are much smaller and simpler to handle. Subsequently,*

we have $t_{sub} \leq \frac{t_{full}}{L}$, *where* t_{sub} *is the time to solve a sub-problem and* t_{full} *the time to solve the full problem. Therefore, we have a benefit in the parallel distribution and obtain faster the higher order* $\mathcal{O}(\tau^{mL})$ *than with the serial version.*

Remark 4. *While the classical version of the parallel splitting method has order* $\mathcal{O}(\tau^m)$, *see the Theorem 4, the modern version of the parallel splitting method can improve the order partially to higher order. We obtain* $\mathcal{O}(\tau^{m_{partial}})$ *with* $m \leq m_{partial} \leq mL_{fast}$, *when we split into slow and fast convergent processors and assume that the faster results of the fast convergent processors are sufficient for the update. We can define a new* $c_i = \sum_{l \in \text{Fast}} \tilde{E}_l c_{i,l}$ *with* $E = \sum_{l \in \text{Fast}} \tilde{E}_l$ *and the set* Fast *is given with the fast convergent processors, see also the ideas of [37,48]. Therefore, we circumvent the slow processors and, later, we redistribute the decomposition of the matrices to gain a more balanced load of the processors.*

5. Numerical Examples

In what follows, we deal with different numerical examples, which are motivated by real-life applications in fluid-flow and phase-field problems. We verify and test our theoretical results of the novel parallel iterative splitting approaches.

We deal with:

- Only time-dependent linear problem: we apply ordinary differential equation to verify the theoretical results.
- Time and space dependent linear problem: we apply a diffusion equation with different spatial dependent operators and test the application to partial differential equations.
- Time and space dependent nonlinear problem: we apply a mixed diffusion-convection with Burgers' equation to test and verify the application to nonlinear problems.
- Time and space dependent fractional problem: we apply a fractional diffusion equation with different spatial dependent operators and, furthermore, we test the application to fractional differential equations.

5.1. First Example: Matrix Problem

In the first test example, we consider the following matrix equation,

$$u'(t) = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix} u, \quad u(0) = u_0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \tag{106}$$

whose exact solution is

$$\exp(\begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix} t) = \begin{pmatrix} \exp(t) & 2t\ \exp(t) \\ 0 & \exp(t) \end{pmatrix}. \tag{107}$$

We split the matrix as:

- Two operator approach

$$A + B = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.3 & 1 \\ 0 & 0.3 \end{bmatrix} + \begin{bmatrix} 0.7 & 1 \\ 0 & 0.7 \end{bmatrix} \tag{108}$$

- Multiple operator approach

$$A_1 + B_1 = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.1 & 1.0 \\ 0 & 0.1 \end{bmatrix} + \begin{bmatrix} 0.9 & 1.0 \\ 0 & 0.9 \end{bmatrix} \tag{109}$$

$$A_2 + B_2 = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.5 & 0.1 \\ 0 & 0.5 \end{bmatrix} + \begin{bmatrix} 0.5 & 1.9 \\ 0 & 0.5 \end{bmatrix} \tag{110}$$

where the E_1 and E_2 are given as:

$$E_1 = \begin{bmatrix} 0.9 & 0 \\ 0 & 0.1 \end{bmatrix}, \; E_2 = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.9 \end{bmatrix}. \tag{111}$$

We include Tables 1 and 2 that correspond to multi-splitting iterative approach classical and modern version with the above partitions and using different discretizations in $[0, 1]$ of step h allowing for a maximum of 10 iterations and a tolerance of 10^{-3}. We can see in the results the relative and absolute errors for each component of the solution and the average iterations performed in order to reach the tolerance.

Figure 1 shows the influence of the tolerance value in the error of the modern version of the algorithm. Each group of bars represents the error for the different step sizes indicated in the legend. It can be seen that the error decreases with the step h if the tolerance is small enough. This is the case, except for $tol = 10^{-2}$, where higher errors appear for smaller steps. Looking at the bars that correspond to the same step and different tolerances, it can be observed that the error for a given temporal step h reaches a minimum as the tolerance decreases and, beyond this point, the errors stabilize or even increase slightly for smaller tolerances.

Table 1. Multisplitting classic version.

h	max e_1	max e_2	rel e_1	rel e_2	it 1	it 2
0.1	0.008308	0.00040332	0.0010188	0.00014837	4	4
0.05	0.00043184	2.906×10^{-5}	5.2955×10^{-5}	1.069×10^{-5}	3	3
0.025	0.00010623	7.2291×10^{-6}	1.3027×10^{-5}	2.6595×10^{-6}	3	3
0.0125	2.6335×10^{-5}	1.8026×10^{-6}	3.2294×10^{-6}	6.6316×10^{-7}	3	3
0.00625	6.5553×10^{-6}	4.5007×10^{-7}	8.0385×10^{-7}	1.6557×10^{-7}	3	3

Table 2. Multisplitting Modern version.

h	max e_1	max e_2	rel e_1	rel e_2	it 1	it 2
0.1	0.0081586	0.0003996	0.0010005	0.000147	4	4
0.05	0.00030027	3.1292×10^{-5}	0.00012447	1.1512×10^{-5}	3.35	3
0.025	0.00027348	1.3123×10^{-5}	3.3536×10^{-5}	4.8277×10^{-6}	3	3
0.0125	6.8274×10^{-5}	3.281×10^{-6}	8.3723×10^{-6}	1.207×10^{-6}	3	3
0.00625	1.7056×10^{-5}	8.2026×10^{-7}	2.0915×10^{-6}	3.0176×10^{-7}	3	3

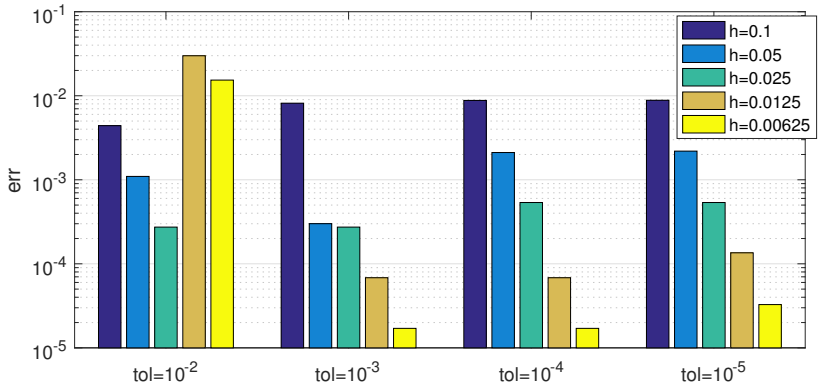

Figure 1. Matrix problem. Precision of the modern version of the multi-splitting iterative approach in terms of the time step h and the tolerance.

Remark 5. *We applied the multisplitting method with the classical (synchronous) and modern (chaotic) approach. We receive the same accuracy of the numerical results, which means that the methods are equally accurate. We obtain some more benefits of the modern method if we apply large time-steps, such that the solution of one sub-problem can be achieved faster and benefit the solution of the second sub-problem. For such small computational unbalances, the modern approach is more efficient.*

5.2. Second Example: Diffusion Problem

We deal with the following diffusion problem:

$$u'(\mathbf{x},t) = \Delta u(\mathbf{x},t),\ (\mathbf{x},t) \in \partial\Omega \times [0,T], \tag{112}$$

$$u(\mathbf{x},0) = \sin x\ \sin y\ \sin z,\ \mathbf{x} \in \Omega, \tag{113}$$

$$u(\mathbf{x},t) = 0,\ (\mathbf{x},t) \in \partial\Omega \times [0,T], \tag{114}$$

where we have the analytical solution $u_{an}(\mathbf{x},t) = \exp(-3t)\ \sin x\ \sin y\ \sin z$, with $\mathbf{x} = (x,y,z)^t$ and $\Omega = [-\pi,\pi] \times [-\pi,\pi] \times [-\pi,\pi]$.

In operator notation, we write:

$$A = A_1 + A_2 + A_3, \tag{115}$$

where $A_1 = \frac{\partial^2}{\partial x^2}$, $A_2 = \frac{\partial^2}{\partial y^2}$, $A_3 = \frac{\partial^2}{\partial z^2}$ and we assume that the zero-boundary conditions (Dirichlet boundary conditions) are fulfilled.

The problem is discretized by using a four-dimensional (4-D) mesh in $\Omega \times [0,T]$. Denote by $u_{i,j,k,t}$ the approximated value of the solution at node (x_i, y_j, z_k, t) for a given t. For the time-integration, we apply the integral formulation, see Equations (15)–(18).

For the spatial discretization, we test a second order scheme:

$$\frac{\partial^2}{\partial x^2} u_{i,j,k,t} = \frac{u_{i+1,j,k,t} - 2u_{i,j,k,t} + u_{i-1,j,k,t}}{\Delta x^2}, \tag{116}$$

and a fourth order scheme:

$$\frac{\partial^2}{\partial x^2} u_{i,j,k,t} = \frac{-u_{i+2,j,k,t} + 16u_{i+1,j,k,t} - 30u_{i,j,k,t} + 16u_{i-1,j,k,t} - u_{i-2,j,k,t}}{12\Delta x^2}, \tag{117}$$

where we have the analogous operators for the y and z derivatives.

We compute the solution $u(\Delta x, h)$ obtained using spatial and temporal steps $\Delta x = \Delta y = \Delta z$ and h, respectively, in order to establish the convergence of the algorithms. We use different measures to estimate the convergence. On one hand, we can compare the outcome of the method $u(\Delta x, h)$ with the exact solution u_{ana} for every point of the mesh, which shows the convergence of the method. On the other hand, we can compare $u(\Delta x, h)$ with the result that was obtained halving the time steps, $h/2$, at the points shared by the corresponding meshes. This allows for analyzing how the results depend on these steps.

Denote by $e_{i,j,k}(\Delta x, h)$ the difference between the results at a mesh point in the final time T, (x_i, y_j, z_k, T), obtained using time steps h and $h/2$, and by $\delta_{i,j,k}(\Delta x, h)$ the difference with the analytical solution at the same point. In the tables, we will denote the maximum errors by

$$e_{\max} = \max_{i,j,k} |e_{i,j,k}(\Delta x, h)|, \tag{118}$$

and

$$\delta_{\max} = \max_{i,j,k} |\delta_{i,j,k}(\Delta x, h)|, \tag{119}$$

and the mean errors by

$$e_{\text{mean}} = \frac{1}{N} \sum_{i,j,k} |e_{i,j,k}(\Delta x, h)|, \tag{120}$$

and

$$\delta_{\text{mean}} = \frac{1}{N} \sum_{i,j,k} |\delta_{i,j,k}(\Delta x, h)|, \tag{121}$$

where N is the number of spatial nodes at time T.

In the following, we discuss different decompositions of the multi-operator splitting approach:

- Directional decomposition: We decompose into the different directions:

$$A_1 = \frac{\partial^2}{\partial x^2}, \; A_2 = \frac{\partial^2}{\partial y^2}, \; A_3 = \frac{\partial^2}{\partial z^2}. \tag{122}$$

  Here, we have the benefit of decomposing the different directions.

  The drawback is related to the unbalanced decomposition, where the matrices have different sparse entries. Therefore, the exponential matrices of the operators are different in their sparse behaviour and the error can not be optimally reduced.

  We can reduce the unbalanced problem, if we deal with the idea to use $\Delta t \approx \Delta x$, see [38]. Subsequently, we obtain at least a second order scheme (related to the spatial discretization).

  We compare our sequential and parallel iterative splitting methods with standard ones such as the sequential operator splitting [49] and the Strang–Marchuk splitting [50]. We apply the splitting algorithms with directional decomposition in $[0, 1]$. The splitting is iterated until a tolerance of 10^{-8} or a maximum of 10 iterations is reached. The values that are shown in the tables correspond to maximum or mean values at $T = 1$. Tables 3 and 4 present the results obtained using different number of temporal steps and the second and fourth order schemes for the spatial discretization, respectively.

  The standard methods, sequential operator, and Strang–Marchuk splitting give almost the same results independently of the number of time steps, because of the linearity of the equation. The e error estimates are negligible, whereas the δ error estimates are independent of the time step, reflecting the spatial discretization error.

Table 3. Diffusion problem. Results of the directional decomposition using the second order scheme for the spatial discretization, with 10 spatial subintervals and different number of temporal steps.

Splitting Algorithm	Time Steps	$e_{\max}$	e_{mean}	$\delta_{\max}$	δ_{mean}	Average Iterations
Sequential Operator	20	6.3838×10^{-16}	8.4685×10^{-17}	3.2022×10^{-2}	4.0127×10^{-3}	1.0000
	40	2.1580×10^{-15}	2.1350×10^{-16}	3.2022×10^{-2}	4.0127×10^{-3}	1.0000
	80	1.6133×10^{-15}	2.3525×10^{-16}	3.2022×10^{-2}	4.0127×10^{-3}	1.0000
	160	3.6013×10^{-15}	3.2583×10^{-16}	3.2022×10^{-2}	4.0127×10^{-3}	1.0000
Strang-Marchuk	20	9.7838×10^{-16}	6.7970×10^{-17}	3.2022×10^{-2}	4.0127×10^{-3}	1.6667
	40	2.4702×10^{-15}	1.5928×10^{-16}	3.2022×10^{-2}	4.0127×10^{-3}	1.6667
	80	6.6128×10^{-15}	5.9749×10^{-16}	3.2022×10^{-2}	4.0127×10^{-3}	1.6667
	160	1.4572×10^{-14}	9.5031×10^{-16}	3.2022×10^{-2}	4.0127×10^{-3}	1.6667
Serial Iterative	20	4.7546×10^{-3}	1.6356×10^{-4}	2.0304×10^{-2}	2.1113×10^{-3}	3.0000
	40	2.2831×10^{-3}	7.5764×10^{-5}	2.0135×10^{-2}	2.0213×10^{-3}	3.0000
	80	1.1013×10^{-3}	3.6053×10^{-5}	2.0059×10^{-2}	1.9958×10^{-3}	2.0688
	160	5.3905×10^{-4}	1.7569×10^{-5}	2.0024×10^{-2}	1.9835×10^{-3}	2.0031
Classical Parallel	20	6.1563×10^{-4}	7.1759×10^{-5}	2.0942×10^{-2}	2.2622×10^{-3}	7.0000
	40	3.1759×10^{-4}	3.7914×10^{-5}	2.1295×10^{-2}	2.3328×10^{-3}	5.0750
	80	1.6070×10^{-4}	1.9398×10^{-5}	2.1479×10^{-2}	2.3701×10^{-3}	4.8688
	160	8.0751×10^{-5}	9.7999×10^{-6}	2.1574×10^{-2}	2.3891×10^{-3}	4.0031
Modern Parallel	20	5.4241×10^{-4}	8.2138×10^{-5}	2.0487×10^{-2}	2.2348×10^{-3}	7.9750
	40	3.0696×10^{-4}	4.5356×10^{-5}	2.1029×10^{-2}	2.3158×10^{-3}	6.0000
	80	1.6316×10^{-4}	2.3798×10^{-5}	2.1336×10^{-2}	2.3605×10^{-3}	4.9938
	160	8.4106×10^{-5}	1.2186×10^{-5}	2.1499×10^{-2}	2.3841×10^{-3}	4.0031

Table 4. Diffusion problem. Results of the directional decomposition using the fourth order scheme for the spatial discretization, with 10 subintervals in each dimension and different number of temporal steps.

Splitting Algorithm	Time Steps	$e_{\max}$	e_{mean}	$\delta_{\max}$	δ_{mean}	Average Iterations
Sequential Operator	20	2.8449×10^{-16}	2.3056×10^{-17}	2.6024×10^{-4}	4.7017×10^{-5}	1.0000
	40	2.9143×10^{-16}	2.6939×10^{-17}	2.6024×10^{-4}	4.7017×10^{-5}	1.0000
	80	1.2490×10^{-16}	1.5822×10^{-17}	2.6024×10^{-4}	4.7017×10^{-5}	1.0000
	160	7.9103×10^{-16}	1.2310×10^{-16}	2.6024×10^{-4}	4.7017×10^{-5}	1.0000
Strang-Marchuk	20	1.6653×10^{-16}	1.4661×10^{-17}	2.6024×10^{-4}	4.7017×10^{-5}	1.6667
	40	3.2613×10^{-16}	3.9698×10^{-17}	2.6024×10^{-4}	4.7017×10^{-5}	1.6667
	80	3.1919×10^{-16}	2.0470×10^{-17}	2.6024×10^{-4}	4.7017×10^{-5}	1.6667
	160	1.5127×10^{-15}	2.1817×10^{-16}	2.6024×10^{-4}	4.7017×10^{-5}	1.6667
Serial Iterative	20	5.3739×10^{-5}	1.0912×10^{-5}	3.1645×10^{-4}	6.1276×10^{-5}	3.0000
	40	1.3425×10^{-5}	2.7263×10^{-6}	2.7126×10^{-4}	5.0429×10^{-5}	3.0000
	80	3.3557×10^{-6}	6.8147×10^{-7}	2.6299×10^{-4}	4.7851×10^{-5}	2.0063
	160	8.3886×10^{-7}	1.7036×10^{-7}	2.6093×10^{-4}	4.7225×10^{-5}	2.0031
Classical Parallel	20	5.3681×10^{-5}	1.0912×10^{-5}	3.1644×10^{-4}	6.1276×10^{-5}	6.0250
	40	1.3412×10^{-5}	2.7263×10^{-6}	2.7125×10^{-4}	5.0429×10^{-5}	5.0125
	80	3.3528×10^{-6}	6.8152×10^{-7}	2.6299×10^{-4}	4.7851×10^{-5}	4.0062
	160	8.3793×10^{-7}	1.7032×10^{-7}	2.6093×10^{-4}	4.7225×10^{-5}	4.0031
Modern Parallel	20	5.3684×10^{-5}	1.0912×10^{-5}	3.1644×10^{-4}	6.1276×10^{-5}	6.0250
	40	1.3412×10^{-5}	2.7263×10^{-6}	2.7125×10^{-4}	5.0429×10^{-5}	5.0125
	80	3.3528×10^{-6}	6.8152×10^{-7}	2.6299×10^{-4}	4.7851×10^{-5}	4.0062
	160	8.3794×10^{-7}	1.7032×10^{-7}	2.6093×10^{-4}	4.7225×10^{-5}	4.0031

Tables 3 and 4 show that the estimated errors $e_{\max}$ and e_{mean} of the iterative splitting methods are proportional to h for the second order scheme of discretization of the directional derivatives, whereas they are proportional to h^2 for the fourth order scheme. The mean differences with the analytical solution, δ_{mean}, of the iterative splitting methods tend to those of the non iterative ones as h decreases. The δ error estimates are more than one order of magnitude better for the fourth order scheme than for the second order scheme. Figures 2 and 3 depict these results.

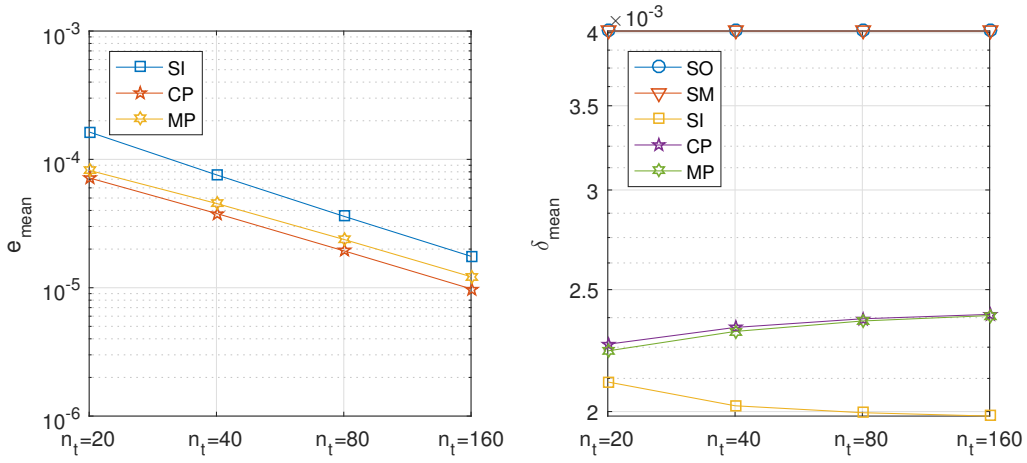

Figure 2. Diffusion problem, directional decomposition. Precision of the proposed methods: sequential iterative SI, classical parallel CP, and modern parallel MP, as compared with the classical ones: sequential operator SO and Strang–Marchuk SM, with second order approximations for the spatial derivatives for different number of time steps.

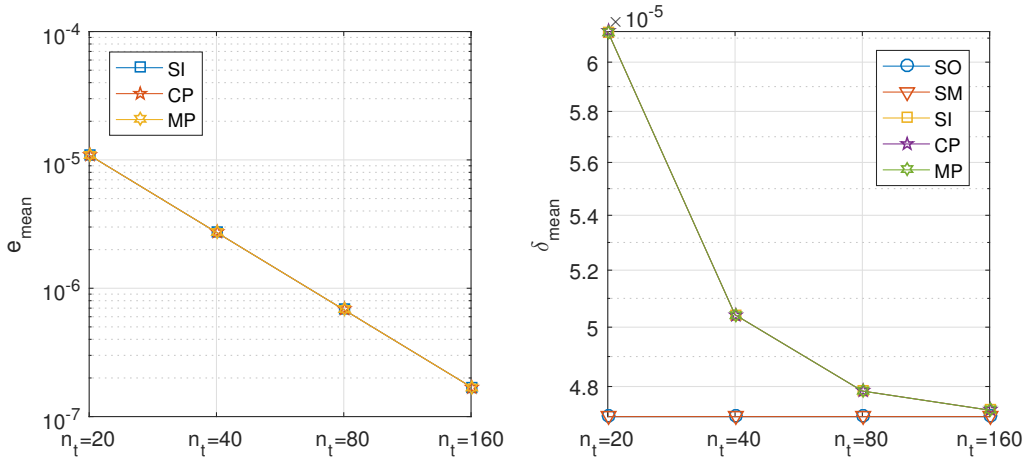

Figure 3. Diffusion problem, directional decomposition. Precision of the proposed methods: sequential iterative SI, classical parallel CP, and modern parallel MP, compared with the classical ones: sequential operator SO and Strang–Marchuk SM, with fourth order approximations for the spatial derivatives for different number of time steps.

Now, we analyze the influence of the number of spatial intervals on the convergence properties of the iterative splitting algorithms. The spatial step is reduced simultaneously in the three dimensions, in order to keep the increments equal, because the performance of the method is better in this case. The time step is small enough to ensure the convergence in the case of the smaller spatial subinterval and is the same in all of the cases to allow the comparison depending only on the spatial step.

For the second order scheme (see Table 5), the e error estimates are proportional to the number of spatial subintervals in each dimension, whereas the δ error estimates decrease, indicating a better approximation to the analytical solution as the spatial step decreases.The parallel methods obtain slightly less approximation to the analytical result and they require more iterations than the serial iterative method. The modern parallel method needs more iterations per step to converge than the classical parallel method.

For the fourth order scheme (see Table 6), the e error estimates are unaffected by the number of spatial subintervals, so that the convergence is maintained. The δ error estimates are proportional

to a power of the spatial step of degree between 3 and 5, greatly improving the approximation of the second order scheme.

Table 5. Diffusion problem. Results of the directional decomposition using the second order scheme for the spatial discretization, with 320 temporal steps and different number of spatial subintervals.

Splitting Algorithm	Space Intervals	e_{max}	e_{mean}	δ_{max}	δ_{mean}	Average Iterations
Sequential Operator	10	1.8437×10^{-14}	2.3626×10^{-15}	3.2022×10^{-2}	4.0127×10^{-3}	1.0000
	20	2.2100×10^{-14}	4.3596×10^{-15}	1.8587×10^{-2}	2.7497×10^{-3}	1.0000
	40	3.7588×10^{-14}	6.9991×10^{-15}	9.7462×10^{-3}	1.5908×10^{-3}	1.0000
Strang-Marchuk	10	2.8838×10^{-14}	2.2970×10^{-15}	3.2022×10^{-2}	4.0127×10^{-3}	1.6667
	20	3.7408×10^{-14}	3.9795×10^{-15}	1.8587×10^{-2}	2.7497×10^{-3}	1.6667
	40	6.2721×10^{-14}	8.1078×10^{-15}	9.7462×10^{-3}	1.5908×10^{-3}	1.6667
Serial Iterative	10	2.6647×10^{-4}	8.6700×10^{-6}	2.0006×10^{-2}	1.9774×10^{-3}	2.0000
	20	4.4399×10^{-4}	9.5320×10^{-6}	1.1625×10^{-2}	1.1212×10^{-3}	2.0000
	40	7.4728×10^{-4}	9.9603×10^{-6}	5.8701×10^{-3}	6.0205×10^{-4}	2.0078
Classical Parallel	10	4.0467×10^{-5}	4.9240×10^{-6}	2.1621×10^{-2}	2.3988×10^{-3}	4.0000
	20	8.2970×10^{-5}	1.1843×10^{-5}	1.2119×10^{-2}	1.7252×10^{-3}	4.0016
	40	1.5751×10^{-4}	2.4059×10^{-5}	6.1534×10^{-3}	9.8751×10^{-4}	4.4063
Modern Parallel	10	4.2699×10^{-5}	6.1656×10^{-6}	2.1583×10^{-2}	2.3962×10^{-3}	4.0000
	20	9.7992×10^{-5}	1.5103×10^{-5}	1.2050×10^{-2}	1.7183×10^{-3}	4.9828
	40	1.7725×10^{-4}	2.9509×10^{-5}	6.0466×10^{-3}	9.7394×10^{-4}	6.9922

Table 6. Diffusion problem, directional decomposition. Results using the fourth order scheme for the spatial discretization, with 320 temporal steps and different number of subintervals in each dimension.

Splitting Algorithm	Space Intervals	e_{max}	e_{mean}	δ_{max}	δ_{mean}	Average Iterations
Sequential Operator	10	9.6451×10^{-16}	1.5113×10^{-16}	2.6024×10^{-4}	4.7017×10^{-5}	1.0000
	20	1.0686×10^{-15}	1.0900×10^{-16}	1.2606×10^{-5}	3.0953×10^{-6}	1.0000
	40	3.9899×10^{-15}	8.3332×10^{-16}	4.1987×10^{-7}	1.1088×10^{-7}	1.0000
Strang-Marchuk	10	1.7972×10^{-15}	2.6813×10^{-16}	2.6024×10^{-4}	4.7017×10^{-5}	1.6667
	20	1.5821×10^{-15}	1.8215×10^{-16}	1.2606×10^{-5}	3.0953×10^{-6}	1.6667
	40	1.3531×10^{-15}	2.0462×10^{-16}	4.1987×10^{-7}	1.1088×10^{-7}	1.6667
Serial Iterative	10	2.0971×10^{-7}	4.2590×10^{-8}	2.6041×10^{-4}	4.7069×10^{-5}	2.0000
	20	2.4313×10^{-7}	5.2851×10^{-8}	1.2777×10^{-5}	3.1437×10^{-6}	2.0000
	40	2.4310×10^{-7}	5.7886×10^{-8}	4.1609×10^{-7}	1.1596×10^{-7}	2.0000
Classical Parallel	10	2.0952×10^{-7}	4.2587×10^{-8}	2.6041×10^{-4}	4.7069×10^{-5}	4.0000
	20	2.4312×10^{-7}	5.2848×10^{-8}	1.2777×10^{-5}	3.1437×10^{-6}	4.0000
	40	2.4309×10^{-7}	5.7883×10^{-8}	4.1609×10^{-7}	1.1596×10^{-7}	4.0000
Modern Parallel	10	2.0952×10^{-7}	4.2587×10^{-8}	2.6041×10^{-4}	4.7069×10^{-5}	4.0000
	20	2.4312×10^{-7}	5.2848×10^{-8}	1.2777×10^{-5}	3.1437×10^{-6}	4.0000
	40	2.4309×10^{-7}	5.7883×10^{-8}	4.1609×10^{-7}	1.1596×10^{-7}	4.0000

Figures 4 and 5 present the results that were obtained by the analyzed splitting methods with different number of spatial steps, using the second and the fourth order schemes for the spatial discretization, respectively.

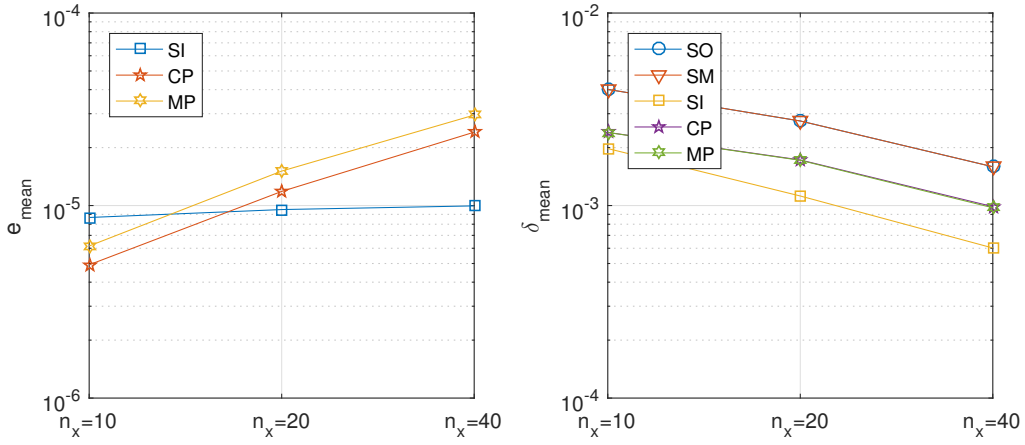

Figure 4. Diffusion problem, directional decomposition. Precision of the proposed methods: sequential iterative SI, classical parallel CP, and modern parallel MP, compared with the classical ones: sequential operator SO and Strang–Marchuk SM, with second order approximations for the spatial derivatives with 320 time steps and different number of spatial steps.

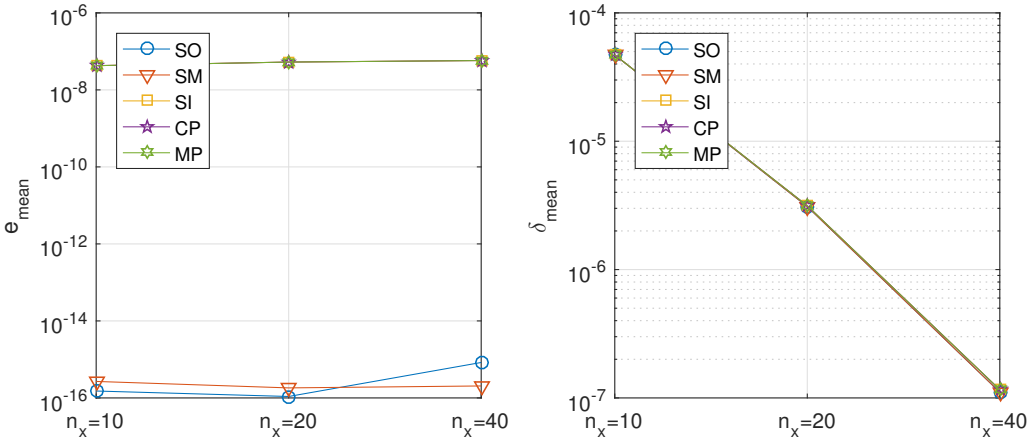

Figure 5. Diffusion problem, directional decomposition. Precision of the proposed methods: sequential iterative SI, classical parallel CP, and modern parallel MP, compared with the classical ones: sequential operator SO and Strang–Marchuk SM, with fourth order approximations for the spatial derivatives for different number of spatial steps.

- Balanced decomposition: We decompose into:

$$A_1 = \frac{1}{3}A,\ A_2 = \frac{1}{3}A,\ A_3 = \frac{1}{3}A. \tag{123}$$

Here, we have the benefit of equal load balances of the matrices, such that the exp-matrices have the same sparse structure. The results of the splitting algorithms with balanced decomposition are shown in Tables 7 and 8 for the second and fourth order schemes for the spatial discretization, and in Figures 6 and 7, respectively. The numerical behaviour is similar to that of the directional decomposition.

Table 7. Diffusion problem. Results of the balanced decomposition using the second order scheme for the spatial discretization, with 10 spatial subintervals and different number of temporal steps.

Splitting Algorithm	Time Steps	$e_{\max}$	e_{mean}	$\delta_{\max}$	δ_{mean}	Average Iterations
Sequential Operator	20	4.2674×10^{-15}	4.9711×10^{-16}	4.3817×10^{-3}	8.9251×10^{-4}	1.0000
	40	7.5459×10^{-15}	9.4849×10^{-16}	4.3817×10^{-3}	8.9251×10^{-4}	1.0000
	80	7.9486×10^{-15}	1.0959×10^{-15}	4.3817×10^{-3}	8.9251×10^{-4}	1.0000
	160	3.1839×10^{-14}	4.4734×10^{-15}	4.3817×10^{-3}	8.9251×10^{-4}	1.0000
Strang-Marchuk	20	3.9690×10^{-15}	3.6874×10^{-16}	4.3817×10^{-3}	8.9251×10^{-4}	1.6667
	40	7.9000×10^{-15}	1.2019×10^{-15}	4.3817×10^{-3}	8.9251×10^{-4}	1.6667
	80	2.3551×10^{-14}	3.5211×10^{-15}	4.3817×10^{-3}	8.9251×10^{-4}	1.6667
	160	8.0304×10^{-14}	1.1365×10^{-14}	4.3817×10^{-3}	8.9251×10^{-4}	1.6667
Serial Iterative	20	5.3493×10^{-5}	1.0896×10^{-5}	4.3104×10^{-3}	8.7798×10^{-4}	3.0000
	40	1.3365×10^{-5}	2.7223×10^{-6}	4.3639×10^{-3}	8.8888×10^{-4}	3.0000
	80	3.3408×10^{-6}	6.8047×10^{-7}	4.3773×10^{-3}	8.9160×10^{-4}	2.0063
	160	8.3516×10^{-7}	1.7011×10^{-7}	4.3806×10^{-3}	8.9228×10^{-4}	2.0031
Classical Parallel	20	5.3493×10^{-5}	1.0896×10^{-5}	4.3104×10^{-3}	8.7798×10^{-4}	6.0000
	40	1.3365×10^{-5}	2.7223×10^{-6}	4.3639×10^{-3}	8.8888×10^{-4}	5.0125
	80	3.3410×10^{-6}	6.8052×10^{-7}	4.3773×10^{-3}	8.9160×10^{-4}	4.0062
	160	8.3499×10^{-7}	1.7008×10^{-7}	4.3806×10^{-3}	8.9228×10^{-4}	4.0031
Modern Parallel	20	5.3493×10^{-5}	1.0896×10^{-5}	4.3104×10^{-3}	8.7798×10^{-4}	6.0000
	40	1.3365×10^{-5}	2.7223×10^{-6}	4.3639×10^{-3}	8.8888×10^{-4}	5.0125
	80	3.3410×10^{-6}	6.8052×10^{-7}	4.3773×10^{-3}	8.9160×10^{-4}	4.0062
	160	8.3499×10^{-7}	1.7008×10^{-7}	4.3806×10^{-3}	8.9228×10^{-4}	4.0031

Table 8. Diffusion problem. Results of the balanced decomposition using the fourth order scheme for the spatial discretization, with 10 spatial subintervals and different number of temporal steps.

Splitting Algorithm	Time Steps	$e_{\max}$	e_{mean}	$\delta_{\max}$	δ_{mean}	Average Iterations
Sequential Operator	20	1.1796×10^{-15}	1.0685×10^{-16}	2.6024×10^{-4}	4.7017×10^{-5}	1.0000
	40	2.9143×10^{-15}	3.5778×10^{-16}	2.6024×10^{-4}	4.7017×10^{-5}	1.0000
	80	4.3715×10^{-15}	5.4513×10^{-16}	2.6024×10^{-4}	4.7017×10^{-5}	1.0000
	160	1.3073×10^{-14}	1.7961×10^{-15}	2.6024×10^{-4}	4.7017×10^{-5}	1.0000
Strang-Marchuk	20	8.4655×10^{-16}	6.8614×10^{-17}	2.6024×10^{-4}	4.7017×10^{-5}	1.6667
	40	1.9776×10^{-15}	2.8005×10^{-16}	2.6024×10^{-4}	4.7017×10^{-5}	1.6667
	80	3.6429×10^{-15}	2.9301×10^{-16}	2.6024×10^{-4}	4.7017×10^{-5}	1.6667
	160	4.2882×10^{-15}	5.7185×10^{-16}	2.6024×10^{-4}	4.7017×10^{-5}	1.6667
Serial Iterative	20	5.3718×10^{-5}	1.0912×10^{-5}	3.1641×10^{-4}	6.1276×10^{-5}	3.0000
	40	1.3421×10^{-5}	2.7263×10^{-6}	2.7124×10^{-4}	5.0429×10^{-5}	3.0000
	80	3.3547×10^{-6}	6.8147×10^{-7}	2.6299×10^{-4}	4.7851×10^{-5}	2.0063
	160	8.3865×10^{-7}	1.7036×10^{-7}	2.6093×10^{-4}	4.7225×10^{-5}	2.0031
Classical Parallel	20	5.3718×10^{-5}	1.0912×10^{-5}	3.1641×10^{-4}	6.1276×10^{-5}	6.0250
	40	1.3421×10^{-5}	2.7263×10^{-6}	2.7124×10^{-4}	5.0429×10^{-5}	5.0125
	80	3.3550×10^{-6}	6.8152×10^{-7}	2.6299×10^{-4}	4.7851×10^{-5}	4.0062
	160	8.3847×10^{-7}	1.7032×10^{-7}	2.6093×10^{-4}	4.7225×10^{-5}	4.0031
Modern Parallel	20	5.3718×10^{-5}	1.0912×10^{-5}	3.1641×10^{-4}	6.1276×10^{-5}	6.0250
	40	1.3421×10^{-5}	2.7263×10^{-6}	2.7124×10^{-4}	5.0429×10^{-5}	5.0125
	80	3.3550×10^{-6}	6.8152×10^{-7}	2.6299×10^{-4}	4.7851×10^{-5}	4.0062
	160	8.3847×10^{-7}	1.7032×10^{-7}	2.6093×10^{-4}	4.7225×10^{-5}	4.0031

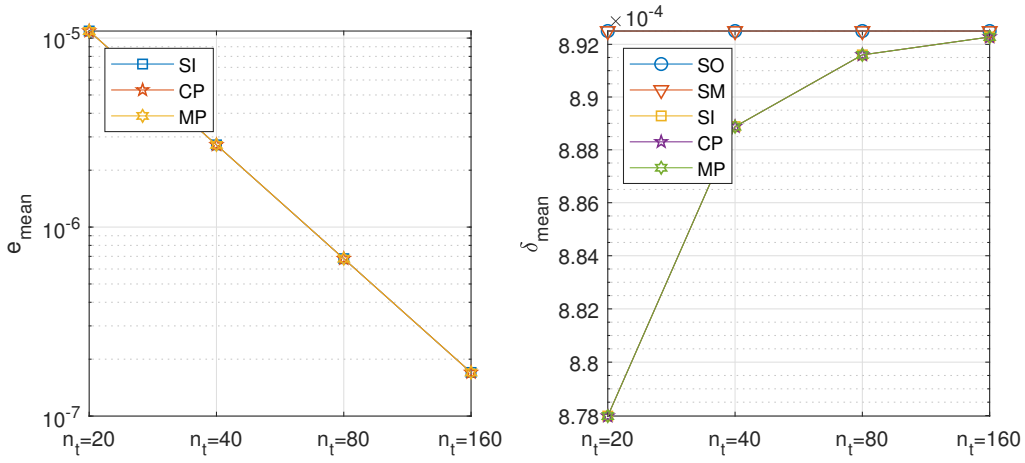

Figure 6. Diffusion problem, balanced decomposition. Precision of the proposed methods: sequential iterative SI, classical parallel CP, and modern parallel MP, compared with the classical ones: sequential operator SO and Strang–Marchuk SM, with second order approximations for the spatial derivatives for different number of time steps.

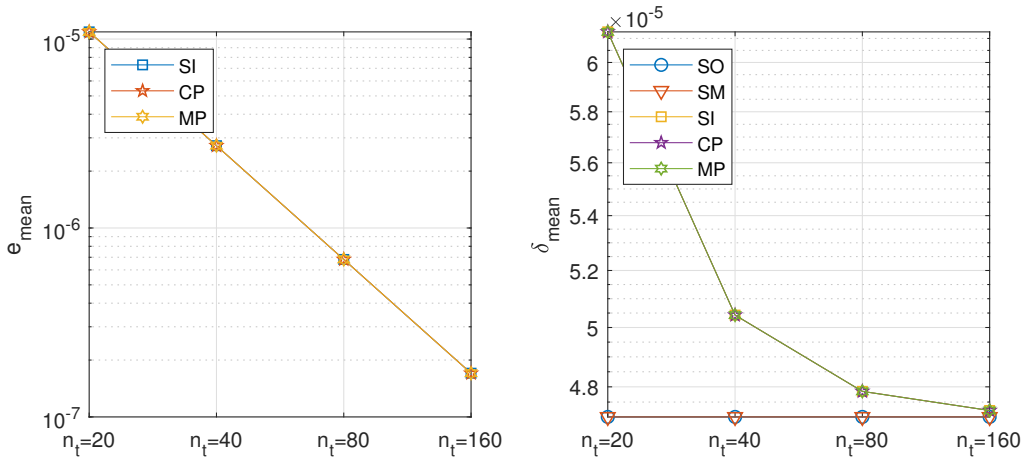

Figure 7. Diffusion problem, balanced decomposition. Precision of the proposed methods: sequential iterative SI, classical parallel CP, and modern parallel MP, compared with the classical ones: sequential operator SO and Strang–Marchuk SM, with fourth order approximations for the spatial derivatives for different number of time steps.

- Mixed decomposition: We decompose into:

$$A_1 = (1-\epsilon)\frac{\partial^2}{\partial x^2} + \frac{\epsilon}{2}(\frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}),\ A_2 = (1-\epsilon)\frac{\partial^2}{\partial y^2} + \frac{\epsilon}{2}(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2}), \tag{124}$$

$$A_3 = (1-\epsilon)\frac{\partial^2}{\partial z^2} + \frac{\epsilon}{2}(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}), \tag{125}$$

where $\epsilon = [0, 2/3]$. For $\epsilon = 0$, we have the directional decomposition, while, for $\epsilon = 2/3$, we have the balanced decomposition.

Remark 6. *Based on the balanced decomposition with $\epsilon = 2/3$, we do not have problems with the splitting approaches and obtain optimal results. For the pure unbalanced decomposition, which means $\epsilon = 0$, we decompose into different directions. Here, we have to restrict us to the exact second order approach, which is $\Delta t \approx \Delta x$.*

Remark 7. *We obtain the benefit of the classical and modern parallel iterative splitting method that is based on larger time-steps and more iterative steps. In such an optimal version, we are much faster than the serial version and also the result is more accurate. For very fine time-steps, we do not see an improvement in the accuracy, but we see a benefit in the computational time; the parallel versions are faster.*

5.3. Third Example: Mixed Convection-Diffusion and Burgers' Equation

We consider a partial differential equation, which is a two-dimesnional (2D) example of a mixed convection-diffusion and Burgers' equation. Such mixed PDEs are used to model fluid flow problems in traffic or population dynamics, see [15,16,51]. For testing the numerical methods, we consider a Burgers' equation, where we can find an analytical solution. The model problem is:

$$\begin{aligned} \partial_t u = & -1/2u\partial_x u - 1/2u\partial_y u - 1/2\partial_x u - 1/2\partial_y u \\ & + \mu(\partial_{xx}u + \partial_{yy}u) + f(x,y,t),\ (x,y,t) \in \Omega \times [0,T], \end{aligned} \tag{126}$$

$$u(x,y,0) = u_{\text{ana}}(x,y,0),\ (x,y) \in \Omega, \tag{127}$$

$$u(x,y,t) = u_{\text{ana}}(x,y,t) \text{ on } \partial\Omega \times [0,T], \tag{128}$$

where $\Omega = [0,1] \times [0,1]$, $T = 1.25$, and μ is the viscosity.

The analytical solution is given by

$$u_{\text{ana}}(x,y,t) = (1 + \exp(\frac{x+y-t}{2\mu}))^{-1} + \exp(\frac{x+y-t}{2\mu}), \tag{129}$$

where we compute $f(x,y,t)$ accordingly.

As in the previous example, denote, by $u(\Delta,h)$, the numerical solution obtained using spatial subintervals of amplitude $\Delta = \Delta x = \Delta y$, time steps h and a tolerance of $tol = 10^{-6}$, allowing a maximum of 40 iterations. On one hand, we will compare the numerical solution with the exact one u_{ana} for every point of the mesh, which shows the convergence of the method. On the other hand, we will compare $u(\Delta,h)$ with the result obtained halving the time steps, $h/2$, at the points that are shared by the corresponding meshes. Denote by $e_{i,j,k}(\Delta,h)$ the difference between the results obtained using two different time steps, h and $h/2$, at a common mesh point (x_i, y_j, t_k), and by $\delta_{i,j,k}(\Delta,h)$ the difference with the analytical solution at the same point. In the tables, we will use the error estimates given by

$$e_{\max} = \max_{i,j,k} |e_{i,j,k}(\Delta,h)|, \tag{130}$$

and

$$\delta_{\max} = \max_{i,j,k} |\delta_{i,j,k}(\Delta,h)|, \tag{131}$$

and the mean errors by

$$e_{\text{mean}} = \frac{1}{N} \sum_{i,j,k} |e_{i,j,k}(\Delta,h)|, \tag{132}$$

and

$$\delta_{\text{mean}} = \frac{1}{N} \sum_{i,j,k} |\delta_{i,j,k}(\Delta,h)|, \tag{133}$$

where N is the total number of nodes (x_i, y_j, t_k).

We measure the time consumed by processor l in each iteration m, in $[t_k, t_{k+1}]$, $tp_{k,l,m}$ in order to obtain the temporal cost of the parallel schemes. In the classical parallel scheme, the processors synchronize at each iteration, so the cost for this time interval is $tp_k = \sum_m \max_{l=1,2} tp_{k,l,m}$, whereas, in the modern parallel scheme, the processors iterate independently in $[t_k, t_{k+1}]$ performing m_l, $l = 1, 2$ iterations until convergence, thus the cost is $tp_k = \max_{l=1,2} \sum_m tp_{k,l,m}$. The final cost is obtained adding the results of all time subintervals.

In the following, we discuss different decompositions of the multi-operator splitting approach:

- Directional decomposition

  We decompose into the different directions (x and y direction):

$$A(u)u = -1/2u\partial_x u - 1/2\partial_x u + \mu\partial_{xx} u, \tag{134}$$

$$B(u)u = -1/2u\partial_y u - 1/2\partial_y u + \mu\partial_{yy} u + f(x, y, t). \tag{135}$$

Let us first analyze the influence of parameter μ on the convergence of the algorithms.

Tables 9 and 10 show the error estimations of the considered algorithms for different values of μ using the second and the fourth order discretization scheme, respectively, taking 10 subintervals in each spatial dimension and 640 time steps. The results for the different algorithms are similar. As it could be expected, higher viscosity values give lower error estimates.

Table 9. Mixed convection-diffusion and Burgers' equation. Results of the directional decomposition using the second order scheme for the spatial discretization, with 10 spatial subintervals, 640 temporal intervals and different values of μ.

Splitting Algorithm	μ	$e_{\max}$	e_{mean}	$\delta_{\max}$	δ_{mean}	Average Iterations	Time
Sequential Operator	0.25	5.8112×10^{-1}	8.0638×10^{-3}	9.3476×10^{-1}	1.1794×10^{-2}	1.0000	10.9421
	0.5	4.5508×10^{-3}	2.1796×10^{-4}	7.9207×10^{-3}	5.7747×10^{-4}	1.0000	11.8920
	1	2.8491×10^{-4}	1.0611×10^{-4}	6.0982×10^{-4}	2.3120×10^{-4}	1.0000	11.2869
	2	1.3365×10^{-4}	5.5167×10^{-5}	2.7611×10^{-4}	1.1307×10^{-4}	1.0000	11.8224
Strang-Marchuk	0.25	1.1244×10^{-1}	8.2317×10^{-4}	6.5305×10^{-2}	7.8549×10^{-3}	1.5000	16.2900
	0.5	1.8260×10^{-3}	3.3699×10^{-4}	4.8191×10^{-3}	1.0202×10^{-3}	1.5000	16.4826
	1	4.5546×10^{-4}	1.4794×10^{-4}	9.5796×10^{-4}	3.1493×10^{-4}	1.5000	17.5409
	2	1.4546×10^{-4}	6.5449×10^{-5}	2.9273×10^{-4}	1.3371×10^{-4}	1.5000	16.8117
Serial Iterative	0.25	1.3150×10^{-2}	1.2382×10^{-3}	2.2903×10^{-1}	1.0361×10^{-2}	2.1500	22.3302
	0.5	2.0471×10^{-3}	3.7850×10^{-4}	5.3038×10^{-3}	1.1031×10^{-3}	2.0031	21.5187
	1	4.6720×10^{-4}	1.5130×10^{-4}	9.7092×10^{-4}	3.2097×10^{-4}	2.5313	27.0622
	2	1.5460×10^{-4}	6.6664×10^{-5}	3.0698×10^{-4}	1.3545×10^{-4}	3.0141	34.6163
Classical Parallel	0.25	1.2263×10^{-2}	1.2385×10^{-3}	2.2913×10^{-1}	1.0361×10^{-2}	3.2531	17.8904
	0.5	2.0418×10^{-3}	3.7840×10^{-4}	5.2987×10^{-3}	1.1030×10^{-3}	3.9859	22.5435
	1	4.5870×10^{-4}	1.5117×10^{-4}	9.6710×10^{-4}	3.2087×10^{-4}	4.0078	24.1887
	2	1.4252×10^{-4}	6.6562×10^{-5}	2.9093×10^{-4}	1.3539×10^{-4}	5.0328	32.4515
Modern Parallel	0.25	1.2348×10^{-2}	1.2388×10^{-3}	2.2923×10^{-1}	1.0361×10^{-2}	3.7172	25.2081
	0.5	2.0354×10^{-3}	3.7825×10^{-4}	5.2929×10^{-3}	1.1029×10^{-3}	4.0062	24.5462
	1	4.5939×10^{-4}	1.5113×10^{-4}	9.6862×10^{-4}	3.2087×10^{-4}	5.0109	30.8930
	2	1.4410×10^{-4}	6.6516×10^{-5}	2.9168×10^{-4}	1.3524×10^{-4}	10.0828	63.1537

Table 10. Mixed convection-diffusion and Burgers' equation. Results of the directional decomposition using the fourth order scheme for the spatial discretization, with 10 spatial subintervals, 640 temporal intervals and different values of μ.

Splitting Algorithm	μ	$e_{\max}$	e_{mean}	$\delta_{\max}$	δ_{mean}	Average Iterations	Time(s)
Sequential Operator	0.25	5.8896×10^{-1}	7.9743×10^{-3}	1.1575	1.5807×10^{-2}	1.0000	9.7930
	0.5	4.5804×10^{-3}	2.1804×10^{-4}	8.9458×10^{-3}	4.3546×10^{-4}	1.0000	9.7708
	1	2.8621×10^{-4}	1.0646×10^{-4}	5.8378×10^{-4}	2.1488×10^{-4}	1.0000	10.0130
	2	1.3465×10^{-4}	5.5478×10^{-5}	2.7645×10^{-4}	1.1254×10^{-4}	1.0000	10.7038
Strang-Marchuk	0.25	1.7993×10^{-2}	6.0015×10^{-4}	5.4593×10^{-2}	1.2833×10^{-3}	1.5000	14.9624
	0.5	1.8507×10^{-3}	3.3834×10^{-4}	3.7379×10^{-3}	6.7850×10^{-4}	1.5000	14.9328
	1	4.5736×10^{-4}	1.4837×10^{-4}	9.4192×10^{-4}	2.9859×10^{-4}	1.5000	15.6433
	2	1.4761×10^{-4}	6.5866×10^{-5}	2.9398×10^{-4}	1.3333×10^{-4}	1.5000	15.4479
Serial Iterative	0.25	1.5854×10^{-2}	1.2976×10^{-3}	3.1234×10^{-2}	2.5814×10^{-3}	2.1922	18.4870
	0.5	2.0512×10^{-3}	3.7922×10^{-4}	4.1861×10^{-3}	7.6029×10^{-4}	2.0031	17.0251
	1	4.6948×10^{-4}	1.5139×10^{-4}	9.5395×10^{-4}	3.0419×10^{-4}	2.1734	18.9489
	2	3.5872×10^{-4}	6.7393×10^{-5}	3.5875×10^{-4}	1.3508×10^{-4}	3.0781	28.1390
Classical Parallel	0.25	1.6499×10^{-2}	1.2986×10^{-3}	3.1931×10^{-2}	2.5824×10^{-3}	3.2641	15.7949
	0.5	2.0442×10^{-3}	3.7916×10^{-4}	4.1796×10^{-3}	7.6023×10^{-4}	4.0000	18.9540
	1	4.6044×10^{-4}	1.5136×10^{-4}	9.5063×10^{-4}	3.0417×10^{-4}	4.0094	20.2018
	2	4.9841×10^{-4}	6.7084×10^{-5}	4.9845×10^{-4}	1.3493×10^{-4}	5.8922	30.5905
Modern Parallel	0.25	1.7037×10^{-2}	1.2990×10^{-3}	3.2501×10^{-2}	2.5830×10^{-3}	4.0906	20.3866
	0.5	2.0357×10^{-3}	3.7908×10^{-4}	4.1717×10^{-3}	7.6017×10^{-4}	4.0062	20.6630
	1	4.6092×10^{-4}	1.5129×10^{-4}	9.5182×10^{-4}	3.0411×10^{-4}	5.0813	26.9929
	2	2.4848×10^{-2}	8.3803×10^{-5}	2.4848×10^{-2}	1.5144×10^{-4}	40.0000	221.3149

Figure 8 shows the dependence on μ of the values of e_{mean} for the different algorithms, using second and fourth order approximations for the spatial derivatives. In contrast with the former example, the use of higher order approximations for the spatial derivatives produces only a slight improvement in the error estimations, in both cases being of the same order with respect to the time step.

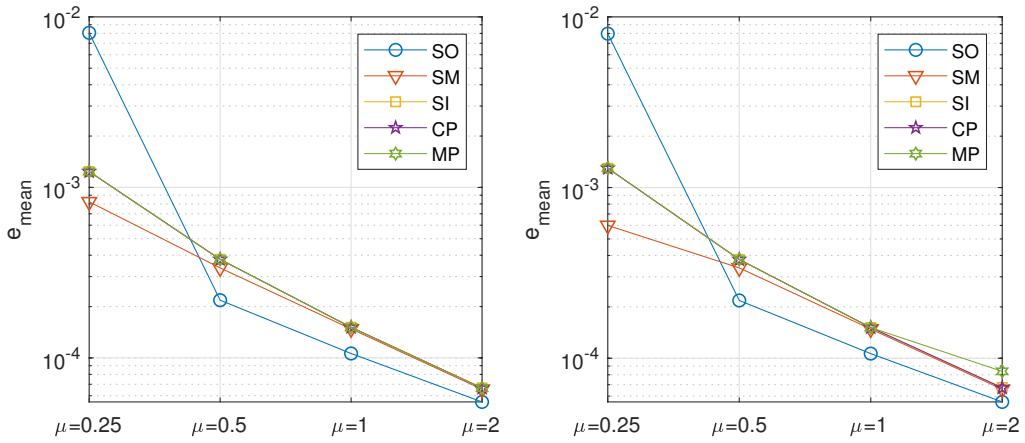

Figure 8. Mixed convection-diffusion and Burgers' equation, directional decomposition. Precision of the proposed methods: sequential iterative SI, classical parallel CP, and modern parallel MP, compared with the classical ones: sequential operator SO and Strang–Marchuk SM, for different viscosities μ. Left: order 2 approximation and right: order 4 approximation for the discretization of the spatial derivatives

As we can observe, the mean error is approximately proportional to the inverse of the viscosity, μ, and it is not much affected by the order of approximation of the spatial derivatives. The number of iterations and the execution time are not affected by the viscosity changes, except for the case of $\mu = 2$ in the modern parallel algorithm, where the maximum number of iterations is reached in each step, also consuming more execution time. In what follows, we will take the viscosity parameter $\mu = 0.5$ and use second order approximations for the spatial derivatives.

We will now analyze the influence of the number of time steps on the convergence of the algorithms.

Table 11 shows that the estimated error is roughly proportional to the time step, although, in the end, the differences δ with the analytical solution decrease more slowly, due to the discretization error. All of the considered methods have errors of the same order. The sequential operator splitting method presents higher estimates than the iterative schemes for the maximum error, but lower estimates of the mean error, as seen in Figure 9.

Table 11. Mixed convection-diffusion and Burgers' equation. Results of the directional decomposition with 10 spatial subintervals and different number of temporal steps.

Splitting Algorithm	Time Steps	e_{max}	e_{mean}	δ_{max}	δ_{mean}	Average Iterations	Time
Sequential Operator	160	2.1627×10^{-2}	8.8698×10^{-4}	3.9223×10^{-2}	1.8270×10^{-3}	1.0000	1.8802
	320	9.6835×10^{-3}	4.3790×10^{-4}	1.7596×10^{-2}	9.7195×10^{-4}	1.0000	3.7410
	640	4.5508×10^{-3}	2.1796×10^{-4}	7.9207×10^{-3}	5.7747×10^{-4}	1.0000	7.5102
	1280	2.2024×10^{-3}	1.0880×10^{-4}	3.3878×10^{-3}	4.1692×10^{-4}	1.0000	14.3002
Strang-Marchuk	160	6.7325×10^{-3}	1.2989×10^{-3}	1.5103×10^{-2}	2.9795×10^{-3}	1.5000	2.7755
	320	3.5515×10^{-3}	6.6569×10^{-4}	8.3706×10^{-3}	1.6847×10^{-3}	1.5000	5.5802
	640	1.8260×10^{-3}	3.3699×10^{-4}	4.8191×10^{-3}	1.0202×10^{-3}	1.5000	11.0716
	1280	9.2592×10^{-4}	1.6954×10^{-4}	2.9931×10^{-3}	6.8365×10^{-4}	1.5000	22.0197
Serial Iterative	160	8.0785×10^{-3}	1.4732×10^{-3}	1.6809×10^{-2}	3.3199×10^{-3}	5.0375	9.8504
	320	3.8934×10^{-3}	7.4966×10^{-4}	9.1969×10^{-3}	1.8514×10^{-3}	3.0062	11.2158
	640	2.0471×10^{-3}	3.7850×10^{-4}	5.3038×10^{-3}	1.1031×10^{-3}	2.0031	14.7620
	1280	1.0522×10^{-3}	1.9023×10^{-4}	3.2567×10^{-3}	7.2500×10^{-4}	2.0008	28.9387
Classical Parallel	160	6.9233×10^{-3}	1.4687×10^{-3}	1.6080×10^{-2}	3.3149×10^{-3}	10.0938	10.1398
	320	3.8634×10^{-3}	7.4911×10^{-4}	9.1617×10^{-3}	1.8507×10^{-3}	5.0125	9.7186
	640	2.0418×10^{-3}	3.7840×10^{-4}	5.2987×10^{-3}	1.1030×10^{-3}	3.9859	15.7576
	1280	1.0525×10^{-3}	1.9024×10^{-4}	3.2570×10^{-3}	7.2501×10^{-4}	3.0016	22.4570
Modern Parallel	160	6.7471×10^{-3}	1.4653×10^{-3}	1.5863×10^{-2}	3.3107×10^{-3}	20.0562	21.0610
	320	3.8313×10^{-3}	7.4855×10^{-4}	9.1238×10^{-3}	1.8501×10^{-3}	6.0188	12.0327
	640	2.0354×10^{-3}	3.7825×10^{-4}	5.2929×10^{-3}	1.1029×10^{-3}	4.0062	16.4724
	1280	1.0530×10^{-3}	1.9026×10^{-4}	3.2575×10^{-3}	7.2504×10^{-4}	3.0023	23.0908

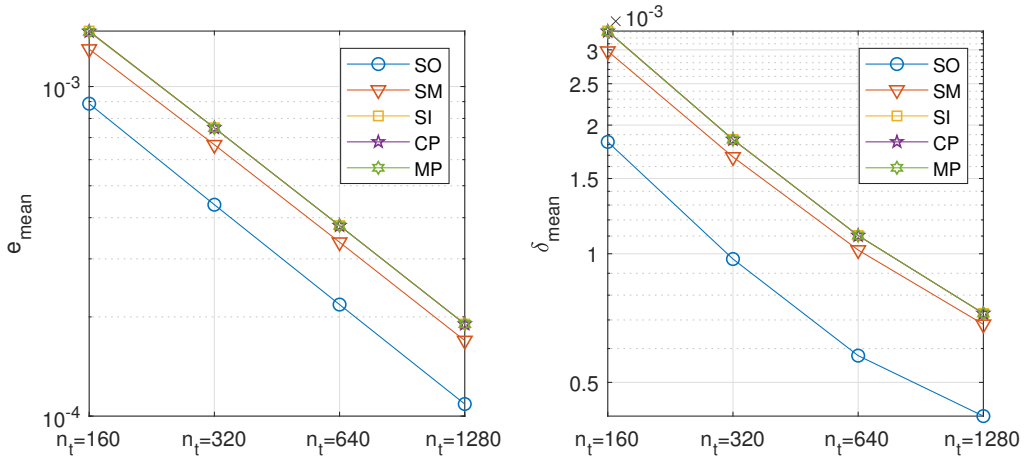

Figure 9. Mixed convection-diffusion and Burgers' equation, directional decomposition. Precision of the proposed methods: sequential iterative SI, classical parallel CP, and modern parallel MP, compared with the classical ones: sequential operator SO and Strang–Marchuk SM, for different number of time steps.

Now, we analyze the dependence on the number of spatial subintervals. Table 12 displays the errors obtained varying the number of space subintervals, considering 1280 time steps, $\mu = 0.5$, tolerance 10^{-6} and maximum number of iterations 40.

Table 12. Mixed convection-diffusion and Burgers' equation. Results of the directional decomposition using the second order scheme for the spatial discretization, with 1280 time subintervals and different number of spatial steps.

Splitting Algorithm	Space Steps	e_{max}	e_{mean}	δ_{max}	δ_{mean}	Average Iterations	Time(s)
Sequential operator	5	1.9600×10^{-2}	1.0751×10^{-4}	3.8363×10^{-2}	1.1866×10^{-3}	1.0000	2.5386
	10	5.4143×10^{-2}	2.7859×10^{-4}	1.0181×10^{-1}	6.0435×10^{-4}	1.0000	7.0674
	20	1.5522×10^{-1}	2.4284×10^{-3}	2.4936×10^{-1}	3.3760×10^{-3}	1.0000	19.1647
Strang-Marchuk	5	1.9504×10^{-2}	1.1148×10^{-4}	3.8228×10^{-2}	1.2082×10^{-3}	1.5000	3.7953
	10	5.3092×10^{-2}	2.2084×10^{-4}	1.0038×10^{-1}	5.9598×10^{-4}	1.5000	9.3821
	20	1.4010×10^{-1}	1.5951×10^{-3}	2.2965×10^{-1}	2.2729×10^{-3}	1.5000	24.3018
Serial Iterative	5	1.5415×10^{-4}	1.5346×10^{-5}	6.7133×10^{-3}	1.0434×10^{-3}	2.0008	6.7669
	10	1.2967×10^{-3}	1.3550×10^{-4}	2.2505×10^{-3}	4.2699×10^{-4}	2.0008	8.3770
	20	3.0299×10^{-2}	1.6372×10^{-3}	3.9211×10^{-2}	2.1837×10^{-3}	2.4992	27.8058
Classical Parallel	5	1.5192×10^{-4}	9.4243×10^{-6}	6.7124×10^{-3}	1.0434×10^{-3}	3.0008	4.2199
	10	1.0957×10^{-3}	1.0800×10^{-4}	2.2581×10^{-3}	3.5285×10^{-4}	3.0016	10.2357
	20	1.8765×10^{-2}	1.0091×10^{-3}	2.5443×10^{-2}	1.4653×10^{-3}	4.0047	34.6686
Modern Parallel	5	1.5292×10^{-4}	9.4575×10^{-6}	6.7112×10^{-3}	1.0434×10^{-3}	3.0008	4.8821
	10	1.1220×10^{-3}	1.1151×10^{-4}	2.2857×10^{-3}	3.5492×10^{-4}	3.0023	8.1929
	20	2.0590×10^{-2}	1.1583×10^{-3}	2.7563×10^{-2}	1.6374×10^{-3}	7.0039	51.4063

Figure 10 shows, on the left, that the convergence properties degrade with the number of spatial steps for a fixed time step and, on the right, that the best approximations are obtained for 10 subintervals in each spatial dimension. The error increment for more space intervals can be due to the fact that the condition number of the exponential matrix utilized in order to solve the differential system increases fast with the number of spatial intervals, making the solution less reliable.

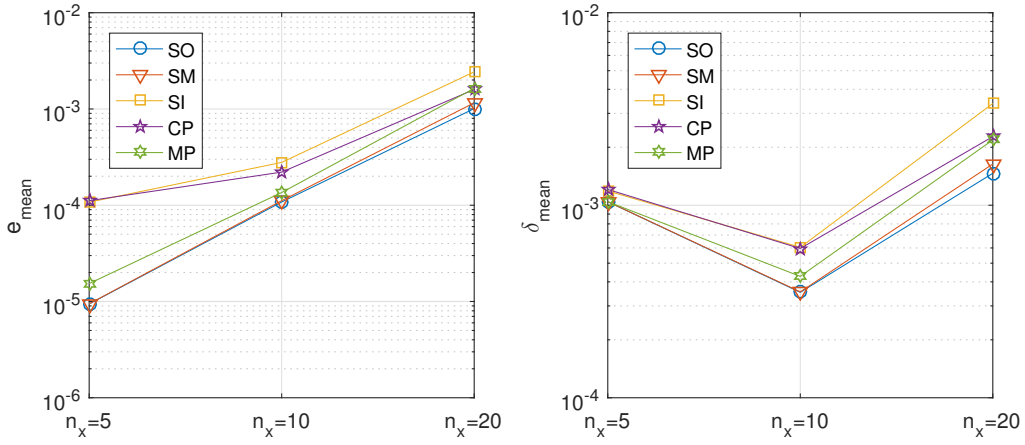

Figure 10. Mixed convection-diffusion and Burgers' equation, directional decomposition. Precision of the proposed methods: sequential iterative SI, classical parallel CP, and modern parallel MP, compared with the classical ones: sequential operator SO and Strang–Marchuk SM, for different number of spatial steps in the directional decomposition.

- Convection and diffusion decomposition

Here, we decompose to an explicit part, which is the convection, and into an implicit part, which is the diffusion.

$$A(u)u = -1/2u(\partial_x u + \partial_y u) - 1/2(\partial_x u + \partial_y u), \tag{136}$$

$$Bu = \frac{1}{2}\mu(\partial_{xx} u + \partial_{yy} u) + f(x, y, t). \tag{137}$$

Table 13 analyzes the convergence of the different splitting methods for the convection and diffusion decomposition varying the time step. The errors are proportional to the time step, as shown in the

log-log diagrams in Figure 11. The iterative methods have slightly better accuracy than the reference methods, particularly better than the sequential operator splitting method. The modern parallel version does not converge for 160 time steps.

Table 13. Mixed convection-diffusion and Burgers' equation. Results for the convection diffusion decomposition using the second order scheme for the spatial discretization with 10 spatial subintervals and different number of temporal steps.

Splitting Algorithm	Time Steps	e_{max}	e_{mean}	δ_{max}	δ_{mean}	Average Iterations	Time(s)
Sequential Operator	160	5.5916×10^{-2}	3.9170×10^{-3}	1.1155×10^{-1}	7.5286×10^{-3}	1.0000	2.2217
	320	2.8229×10^{-2}	1.9781×10^{-3}	5.5638×10^{-2}	3.6224×10^{-3}	1.0000	4.1601
	640	1.4194×10^{-2}	9.9398×10^{-4}	2.7412×10^{-2}	1.6466×10^{-3}	1.0000	8.1867
	1280	7.1179×10^{-3}	4.9822×10^{-4}	1.3220×10^{-2}	6.5429×10^{-4}	1.0000	16.1663
Strang-Marchuk	160	1.0750×10^{-2}	1.8276×10^{-3}	2.2543×10^{-2}	3.9088×10^{-3}	1.5000	3.2557
	320	5.3570×10^{-3}	8.8281×10^{-4}	1.1836×10^{-2}	2.0865×10^{-3}	1.5000	6.1221
	640	2.6780×10^{-3}	4.3354×10^{-4}	6.4794×10^{-3}	1.2052×10^{-3}	1.5000	12.2784
	1280	1.3392×10^{-3}	2.1479×10^{-4}	3.8023×10^{-3}	7.7212×10^{-4}	1.5000	23.9329
Serial Iterative	160	1.1984×10^{-2}	1.3870×10^{-3}	2.2076×10^{-2}	3.0691×10^{-3}	3.7250	7.0300
	320	4.8721×10^{-3}	6.7652×10^{-4}	1.0137×10^{-2}	1.6865×10^{-3}	3.0062	10.7644
	640	2.1637×10^{-3}	3.3484×10^{-4}	5.2782×10^{-3}	1.0112×10^{-3}	2.0031	14.1231
	1280	1.0151×10^{-3}	1.6667×10^{-4}	3.1179×10^{-3}	6.7675×10^{-4}	2.0008	29.3916
Classical Parallel	160	9.7517×10^{-3}	1.4278×10^{-3}	1.9648×10^{-2}	3.1946×10^{-3}	10.1000	11.5662
	320	4.5454×10^{-3}	7.1452×10^{-4}	9.9089×10^{-3}	1.7712×10^{-3}	5.0125	10.6269
	640	2.1597×10^{-3}	3.5723×10^{-4}	5.3681×10^{-3}	1.0580×10^{-3}	4.0031	17.3560
	1280	1.0506×10^{-3}	1.7865×10^{-4}	3.2100×10^{-3}	7.0114×10^{-4}	3.0016	25.8503
Modern Parallel	320	4.5794×10^{-3}	7.1442×10^{-4}	9.9479×10^{-3}	1.7711×10^{-3}	9.6687	20.3683
	640	2.1612×10^{-3}	3.5716×10^{-4}	5.3734×10^{-3}	1.0579×10^{-3}	4.8344	20.8872
	1280	1.0537×10^{-3}	1.7866×10^{-4}	3.2137×10^{-3}	7.0116×10^{-4}	3.0047	26.5525

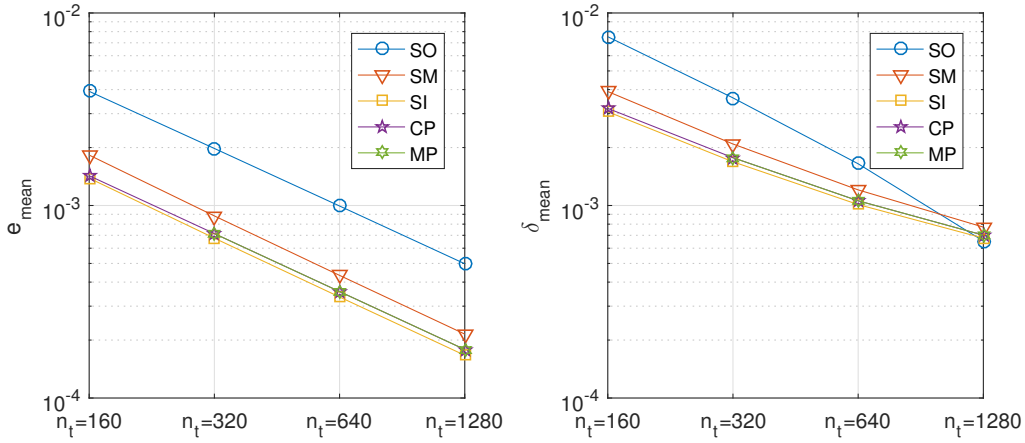

Figure 11. Mixed convection-diffusion and Burgers' equation, convection-diffusion decomposition. Precision of the proposed methods: sequential iterative SI, classical parallel CP, and modern parallel MP, compared with the classical ones: sequential operator SO and Strang–Marchuk SM, for different number of time steps.

Table 14 analyzes the dependence on the spatial step. The behavior is similar to the case of the directional decomposition, presenting an increment of the estimated error with the number of spatial subintervals for a fixed time step. In the second order scheme, the δ-errors decrease with the space step. The temporal cost is relatively high in the case of 20 subintervals, due to the computational overhead for dealing with big matrices.

Table 14. Mixed convection-diffusion and Burgers' equation. Results for the convection diffusion decomposition using the second order scheme for the spatial discretization with 10 temporal steps and different number of spatial subintervals.

Splitting Algorithm	Spatial Intervals	e_{max}	e_{mean}	δ_{max}	δ_{mean}	Average Iterations	Time(s)
Sequential operator	5	4.1014×10^{-3}	3.1046×10^{-4}	3.0286×10^{-3}	4.4136×10^{-4}	1.0000	4.9225
	10	7.1179×10^{-3}	4.9822×10^{-4}	1.3220×10^{-2}	6.5429×10^{-4}	1.0000	13.9807
	20	9.5637×10^{-3}	6.3048×10^{-4}	1.9117×10^{-2}	1.1655×10^{-3}	1.0000	246.2290
Strang-Marchuk	5	1.0128×10^{-3}	1.3655×10^{-4}	8.9821×10^{-3}	1.3134×10^{-3}	1.5000	8.8805
	10	1.3392×10^{-3}	2.1479×10^{-4}	3.8023×10^{-3}	7.7212×10^{-4}	1.5000	25.2552
	20	1.5215×10^{-3}	2.6922×10^{-4}	3.1757×10^{-3}	6.3273×10^{-4}	1.5000	379.1765
Serial Iterative	5	5.6621×10^{-4}	1.0099×10^{-4}	8.0418×10^{-3}	1.2428×10^{-3}	2.0008	21.1443
	10	1.0151×10^{-3}	1.6667×10^{-4}	3.1179×10^{-3}	6.7675×10^{-4}	2.0008	49.0657
	20	1.5741×10^{-3}	2.1460×10^{-4}	3.1037×10^{-3}	5.2448×10^{-4}	2.0016	707.5677
Classical Parallel	5	6.4980×10^{-4}	1.1055×10^{-4}	8.2248×10^{-3}	1.2617×10^{-3}	3.0008	18.2641
	10	1.0506×10^{-3}	1.7865×10^{-4}	3.2100×10^{-3}	7.0114×10^{-4}	3.0016	43.1654
	20	1.4562×10^{-3}	2.2646×10^{-4}	2.9778×10^{-3}	5.4990×10^{-4}	4.0039	792.4449
Modern Parallel	5	6.4782×10^{-4}	1.1048×10^{-4}	8.2242×10^{-3}	1.2617×10^{-3}	3.0016	8.3758
	10	1.0537×10^{-3}	1.7866×10^{-4}	3.2137×10^{-3}	7.0116×10^{-4}	3.0047	21.2526
	20	1.4709×10^{-3}	2.2651×10^{-4}	2.9890×10^{-3}	5.4993×10^{-4}	6.0141	687.0917

Figure 12 compares the estimated mean errors e_{mean} and δ_{mean} of the different methods with the convection-diffusion decomposition and second order approximation of the spatial derivatives for different number of spatial steps. The sequential operator splitting has poorer convergence properties than the other methods, and its result differs more from the analytical solution as the number of spatial nodes increases.

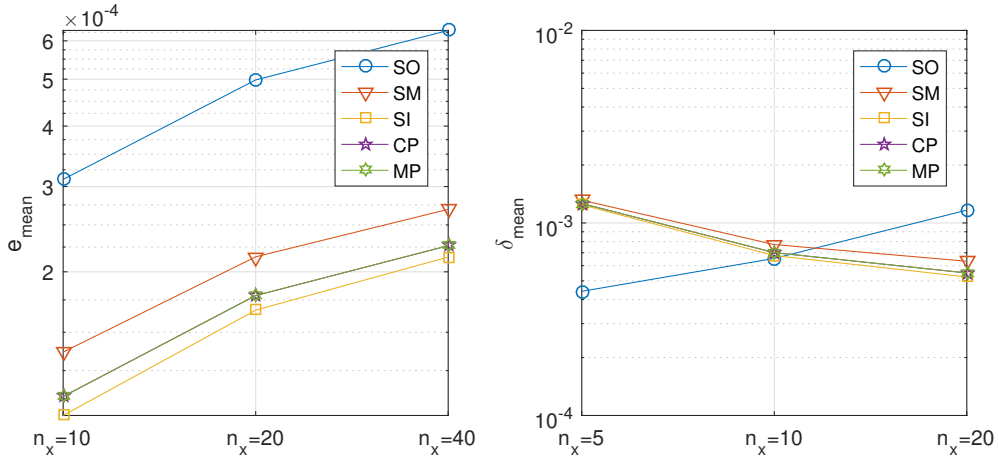

Figure 12. Mixed convection-diffusion and Burgers' equation, convection-diffusion decomposition. Precision of the proposed methods: sequential iterative SI, classical parallel CP, and modern parallel MP, compared with the classical ones: sequential operator SO and Strang–Marchuk SM, for different number of spatial steps.

- Balanced decomposition

We decompose into:

$$A(u)u = (1-\epsilon)\left(-1/2u\partial_x u - 1/2\partial_x u + \mu\partial_{xx}u\right) + \epsilon\left(-1/2u\partial_y u - 1/2\partial_y u + \mu\partial_{yy}u\right) + \epsilon f(x,y,t), \quad (138)$$
$$B(u)u = \epsilon\left(-1/2u\partial_x u - 1/2\partial_x u + \mu\partial_{xx}u\right) + (1-\epsilon)\left(-1/2u\partial_y u - 1/2\partial_y u + \mu\partial_{yy}u\right) + (1-\epsilon)f(x,y,t).$$

where ϵ is an arbitrary parameter that can be tuned in order to achieve the maximum efficiency.

We first examine the influence of parameter ϵ on the convergence of the different splitting schemes, see Table 15, The method has the same behavior for parameter values symmetric with respect to 0.5. The results are quite uniform for ϵ in the range $[-0.1, 1.1]$, whereas, for other parameter values, the

method may diverge. The classical parallel algorithm yields results for $\epsilon = 2$, whereas the serial and the modern parallel iterative methods fail for $\epsilon = 2$.

Table 15. Mixed convection-diffusion and Burgers' equation, balanced decomposition. Results for the balanced decomposition using the second order scheme for the spatial discretization with 640 temporal subintervals and different values of parameter ϵ.

Splitting Algorithm	ϵ	$e_{\max}$	e_{mean}	$\delta_{\max}$	δ_{mean}	Average Iterations	Time(s)
Sequential operator	0.5	4.5184×10^{-3}	2.1168×10^{-4}	7.8771×10^{-3}	5.6809×10^{-4}	1.0000	6.6720
	1	4.5508×10^{-3}	2.1796×10^{-4}	7.9207×10^{-3}	5.7747×10^{-4}	1.0000	7.5672
	2	4.8177×10^{-3}	2.7309×10^{-4}	8.2784×10^{-3}	6.6081×10^{-4}	1.0000	7.8492
Strang-Marchuk	0.5	1.8701×10^{-3}	3.3767×10^{-4}	4.8778×10^{-3}	1.0211×10^{-3}	1.0000	9.7979
	1	1.8260×10^{-3}	3.3699×10^{-4}	4.8191×10^{-3}	1.0202×10^{-3}	1.0000	12.8640
	2	1.4965×10^{-3}	3.3155×10^{-4}	4.3586×10^{-3}	1.0130×10^{-3}	1.0000	13.6113
Serial Iterative	0.5	2.0353×10^{-3}	3.7750×10^{-4}	5.2893×10^{-3}	1.1017×10^{-3}	2.8812	20.1360
	1	1.8680×10^{-3}	3.7128×10^{-4}	5.4439×10^{-3}	1.1173×10^{-3}	2.0031	20.5409
	2	8.3539×10^{29}	NaN	Inf	NaN	1.3125	10.5239
Classical Parallel	0.5	2.0337×10^{-3}	3.7747×10^{-4}	5.2887×10^{-3}	1.1017×10^{-3}	3.9859	15.0469
	1	1.9713×10^{-3}	3.7521×10^{-4}	5.2081×10^{-3}	1.0988×10^{-3}	3.9859	21.0260
	2	1.9719×10^{-3}	3.4606×10^{-4}	5.2186×10^{-3}	1.0635×10^{-3}	3.9875	19.8748
Modern Parallel	0.5	2.0337×10^{-3}	3.7747×10^{-4}	5.2887×10^{-3}	1.1017×10^{-3}	3.9859	15.9368
	1	1.9721×10^{-3}	3.7521×10^{-4}	5.2139×10^{-3}	1.0990×10^{-3}	4.0062	19.0337
	2	2.0550×10^{11}	NaN	2.0550×10^{11}	NaN	2.7375	12.0642

The dependence on the time and space steps of the algorithms with the convection-diffusion decomposition is similar to that of the previously analyzed decompositions. Table 16 compares the behavior of the considered methods with different decompositions.

Table 16. Mixed convection-diffusion and Burgers' equation with 10 spatial intervals and 640 temporal steps. Results of the different splitting methods with directional decomposition, D, convection-diffusion decomposition, CD, and ϵ-balanced decomposition, ϵ B.

Splitting Algorithm	Decomposition	e_{max}	e_{mean}	δ_{max}	δ_{mean}	Average Iterations	Time
Sequential operator	D	1.1412×10^{-1}	9.7183×10^{-4}	1.0181×10^{-1}	6.0435×10^{-4}	1.0000	3.4686
	CD	1.4194×10^{-2}	9.9398×10^{-4}	1.3220×10^{-2}	6.5429×10^{-4}	1.0000	12.5035
	ϵ D	4.5184×10^{-3}	2.1168×10^{-4}	3.3769×10^{-3}	4.1340×10^{-4}	1.0000	12.5689
Strang-Marchuk	D	1.0966×10^{-1}	6.9728×10^{-4}	1.0038×10^{-1}	5.9598×10^{-4}	1.0000	5.1649
	CD	2.6780×10^{-3}	4.3354×10^{-4}	3.8023×10^{-3}	7.7212×10^{-4}	1.0000	19.4012
	ϵ B	1.8701×10^{-3}	3.3767×10^{-4}	3.0078×10^{-3}	6.8387×10^{-4}	1.0000	20.3991
Serial Iterative	D	5.1944×10^{-3}	5.5729×10^{-4}	2.2505×10^{-3}	4.2699×10^{-4}	2.0008	8.1039
	CD	2.1637×10^{-3}	3.3484×10^{-4}	3.1179×10^{-3}	6.7675×10^{-4}	2.0008	29.1385
	ϵ B	2.0353×10^{-3}	3.7750×10^{-4}	3.2541×10^{-3}	7.2469×10^{-4}	2.0008	30.3495
Classical Parallel	D	4.0117×10^{-3}	4.0302×10^{-4}	2.2581×10^{-3}	3.5285×10^{-4}	3.0016	6.8251
	CD	2.1597×10^{-3}	3.5723×10^{-4}	3.2100×10^{-3}	7.0114×10^{-4}	3.0016	24.2979
	ϵ B	2.0337×10^{-3}	3.7747×10^{-4}	3.2550×10^{-3}	7.2471×10^{-4}	3.0016	24.4019
Modern Parallel	D	4.2148×10^{-3}	4.3094×10^{-4}	2.2857×10^{-3}	3.5492×10^{-4}	3.0023	6.3850
	CD	2.1612×10^{-3}	3.5716×10^{-4}	3.2137×10^{-3}	7.0116×10^{-4}	3.0047	24.3399
	ϵ B	2.0337×10^{-3}	3.7747×10^{-4}	3.2550×10^{-3}	7.2471×10^{-4}	3.0016	24.6888

Figure 13 shows that the non-iterative splitting methods give better results with the ϵ-balanced decomposition, whereas the iterative methods give similar results with all of the decompositions, being slightly better for the directional decomposition.

Figure 14 compares the temporal costs that are shown in Table 16. The results indicate that the directional decomposition is better than the convection-diffusion decomposition and the ϵ-balanced decomposition for all the considered splitting algorithms.

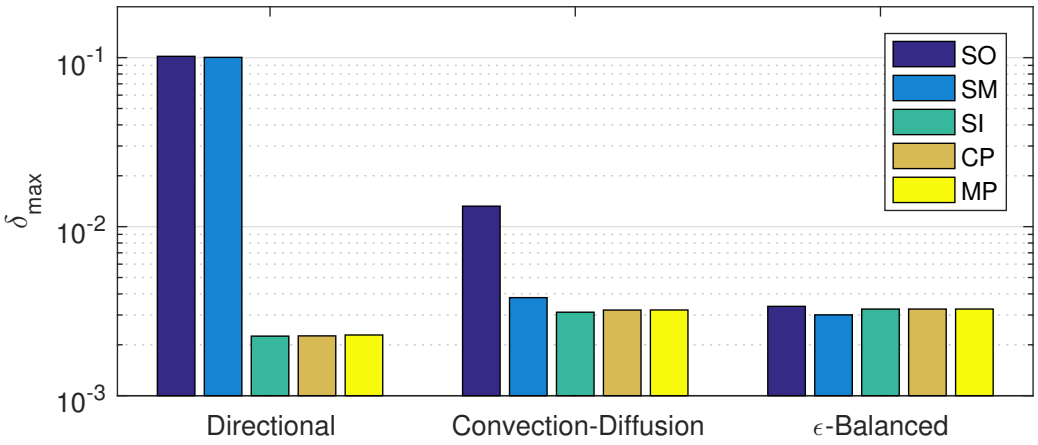

Figure 13. Mixed convection-diffusion and Burgers' equation with 10 spatial intervals and 1280 temporal steps. Precision of the proposed methods: sequential iterative SI, classical parallel CP, and modern parallel MP, compared with the classical ones: sequential operator SO and Strang–Marchuk SM, for different decomposition methods.

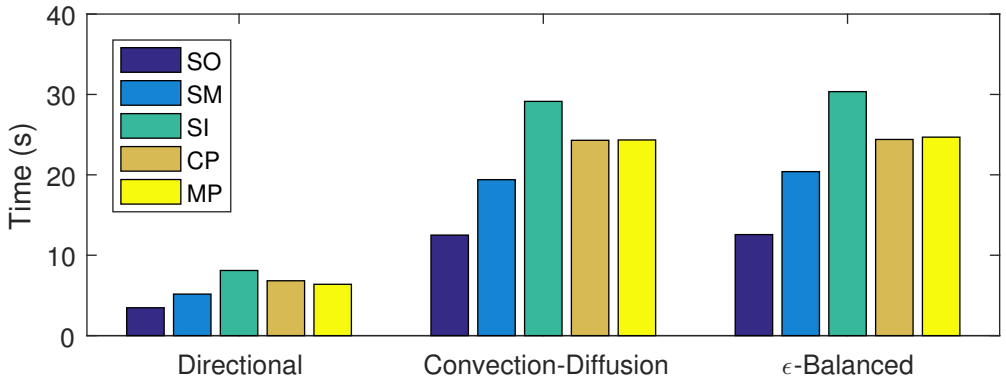

Figure 14. Mixed convection-diffusion and Burgers' equation with 10 spatial intervals and 640 time steps. Temporal cost in seconds of the proposed methods: sequential iterative SI, classical parallel CP, and modern parallel MP, compared with the classical ones: sequential operator SO and Strang–Marchuk SM, for different decomposition methods.

5.4. Fourth Example: Fractional Diffusion Problem

We deal with the following fractional diffusion problem, see also [27]:

$$u'(\mathbf{x},t) = d(\mathbf{x})\frac{\partial^{1.8}u(\mathbf{x},t)}{\partial x^{1.8}} + e(\mathbf{x})\frac{\partial^{1.6}u(\mathbf{x},t)}{\partial y^{1.6}} + q(\mathbf{x},t),\ (\mathbf{x},t) \in \Omega \times [0,T], \tag{139}$$

$$u(\mathbf{x},0) = x^3\ y^{3.6},\ \mathbf{x} \in \Omega, \tag{140}$$

$$u(\mathbf{x},t) = \exp(-t)\ x^3\ y^{3,6},\ (\mathbf{x},t) \in \partial\Omega \times [0,T], \tag{141}$$

where we have the analytical solution $u_{an}(\mathbf{x},t) = \exp(-t)\ \ x^3\ y^{3.6}$, with $\mathbf{x} = (x,y)^t$ and $\Omega = [0,1] \times [0,1]$ and $t \in [0,T]$ with $T = 1.0$. $q(\mathbf{x},t) = -(1+2xy)\exp(-t)\ \ x^3\ y^{3.6}$ and $d(\mathbf{x}) = \Gamma(2.2)x^{2.8}y/6$, $e(\mathbf{x}) = 2xy^{2.6}/\Gamma(4.6)$.

In operator notation, we write:

$$A = A_1 + A_2, \tag{142}$$

where $A_1 = d(\mathbf{x})\frac{\partial^\alpha}{\partial x^\alpha}$, $A_2 = e(\mathbf{x})\frac{\partial^\beta}{\partial y^\beta} + q(\mathbf{x},t)$ and we assume that the Dirichlet boundary conditions are embedded into the operators.

We apply the normalized Grünwald weights by:

$$g_{\alpha,k} = \frac{\Gamma(k-\alpha)}{\Gamma(-\alpha)\,\Gamma(k+1)} = (-1)^k \binom{\alpha}{k}, \tag{143}$$

for the right-shifted Grünwald formula, see [32].

We apply:

$$d(x_i, y_j)\frac{\partial^\alpha u(x_i, y_j, t)}{\partial x^\alpha} \simeq \frac{d_{i,j}}{(\Delta x)^\alpha} \sum_{k=0}^{i+1} g_{\alpha,k}\; u_{i-k+1,j}, \tag{144}$$

$$e(x_i, y_j)\frac{\partial^\beta u(x_i, y_j, t)}{\partial y^\beta} \simeq \frac{e_{i,j}}{(\Delta y)^\beta} \sum_{k=0}^{j+1} g_{\beta,k}\; u_{i,j-k+1}. \tag{145}$$

In order to establish the convergence of the algorithms, we compute the solution $u(\Delta, h)$ obtained while using spatial and temporal steps $\Delta = \Delta x = \Delta y$ and h,, respectively. We use different measures to estimate the convergence. On one hand, we compare the outcome of the method $u(\Delta, h)$ with the exact solution u_{an} for every point of the mesh, which shows the convergence of the method. On the other hand, we can compare $u(\Delta, h)$ with the result obtained halving the time or space steps, $h/2$, at the final time $T = 1$. Denote, by $e_{i,j}(\Delta, h)$, the difference between the results at a mesh point $(x_i, y_j, 1)$, obtained using two different time steps, h and $h/2$, and by $\delta_{i,j}(\Delta, h)$ the difference with the analytical solution at the same point. In the tables, we will denote the maximum errors by

$$e_{\max} = \max_{i,j} |e_{i,j}(\Delta x, h)|, \tag{146}$$

and

$$\delta_{\max} = \max_{i,j} |\delta_{i,j}(\Delta x, h)|, \tag{147}$$

and the mean errors by

$$e_{\text{mean}} = \frac{1}{N}\sum_{i,j} |e_{i,j}(\Delta x, h)|, \tag{148}$$

and

$$\delta_{\text{mean}} = \frac{1}{N}\sum_{i,j} |\delta_{i,j}(\Delta x, h)|, \tag{149}$$

where N is the number of spatial nodes at time T.

In the following, we discuss different decompositions of the multi-operator splitting approach:

- Directional decomposition:

We decompose into the different directions:

$$A_1 = d(\mathbf{x})\frac{\partial^\alpha}{\partial x^\alpha}, \tag{150}$$

$$A_2 = e(\mathbf{x})\frac{\partial^\beta}{\partial y^\beta} + q(\mathbf{x}, t). \tag{151}$$

The directional decomposition allows obtaining the solution solving linear systems of size $n_x - 1$, where n_x is the number of spatial subintervals. Table 17 shows the results of the considered splitting algorithms for 20 spatial subintervals and different time steps, allowing for a maximum of three iterations and using tolerance 10^{-4}. Figure 15 shows, on the left, that the parallel iterative methods perform slightly better than the serial iterative method and, on the right, the approximation to the analytical solution for different number of time steps.

Table 17. Fractional diffusion equation. Results for the directional decomposition with 20 spatial subintervals and different number of temporal steps.

Splitting Algorithm	Time Steps	$e_{\max}$	e_{mean}	$\delta_{\max}$	δ_{mean}	Average Iterations	Time(s)
Sequential Operator	40	3.6909×10^{-3}	2.1918×10^{-4}	7.4079×10^{-3}	4.3302×10^{-4}	1.0000	0.0595
	80	1.9763×10^{-3}	1.1228×10^{-4}	3.7171×10^{-3}	2.4365×10^{-4}	1.0000	0.1169
	160	1.0222×10^{-3}	5.6858×10^{-5}	1.7407×10^{-3}	1.7337×10^{-4}	1.0000	0.2488
	320	5.1954×10^{-4}	2.8614×10^{-5}	8.8420×10^{-4}	1.5956×10^{-4}	1.0000	0.4207
Strang-Marchuk	40	6.1536×10^{-4}	7.5827×10^{-5}	1.0176×10^{-3}	1.8768×10^{-4}	1.5000	1.1779
	80	3.4782×10^{-4}	4.2854×10^{-5}	1.3636×10^{-3}	2.3792×10^{-4}	1.5000	0.1476
	160	1.7770×10^{-4}	2.1897×10^{-5}	1.1561×10^{-3}	2.0663×10^{-4}	1.5000	0.3029
	320	8.9844×10^{-5}	1.1077×10^{-5}	1.0605×10^{-3}	1.9393×10^{-4}	1.5000	0.6012
Serial Iterative	40	4.3494×10^{-2}	3.7787×10^{-3}	3.9615×10^{-2}	4.8916×10^{-3}	1.1625	1.1494
	80	2.6689×10^{-2}	2.2996×10^{-3}	7.8003×10^{-2}	8.5729×10^{-3}	3.0000	0.3391
	160	1.1493×10^{-2}	1.0086×10^{-3}	5.2376×10^{-2}	6.3074×10^{-3}	2.9625	0.6819
	320	5.3112×10^{-3}	4.7046×10^{-4}	4.3672×10^{-2}	5.3400×10^{-3}	1.9625	0.9298
Classical Parallel	40	5.3373×10^{-3}	4.4982×10^{-4}	3.9545×10^{-2}	4.8664×10^{-3}	1.7414	1.6636
	80	8.1666×10^{-3}	2.3243×10^{-4}	3.4634×10^{-2}	4.4640×10^{-3}	3.0000	0.3520
	160	1.2229×10^{-2}	6.1940×10^{-4}	3.5694×10^{-2}	4.4046×10^{-3}	2.9750	0.6967
	320	4.3711×10^{-3}	1.9453×10^{-4}	4.2061×10^{-2}	4.9619×10^{-3}	2.5344	1.2122
Modern Parallel	40	2.5500×10^{-3}	1.1787×10^{-4}	3.7503×10^{-2}	7.5934×10^{-3}	1.5680	1.5756
	80	2.7226×10^{-3}	7.5531×10^{-5}	3.8166×10^{-2}	7.6333×10^{-3}	3.0000	0.3567
	160	5.7234×10^{-4}	2.0251×10^{-5}	3.8201×10^{-2}	7.7009×10^{-3}	2.9937	0.7005
	320	1.4722×10^{-3}	1.2626×10^{-4}	3.8233×10^{-2}	7.7196×10^{-3}	2.5500	1.2045

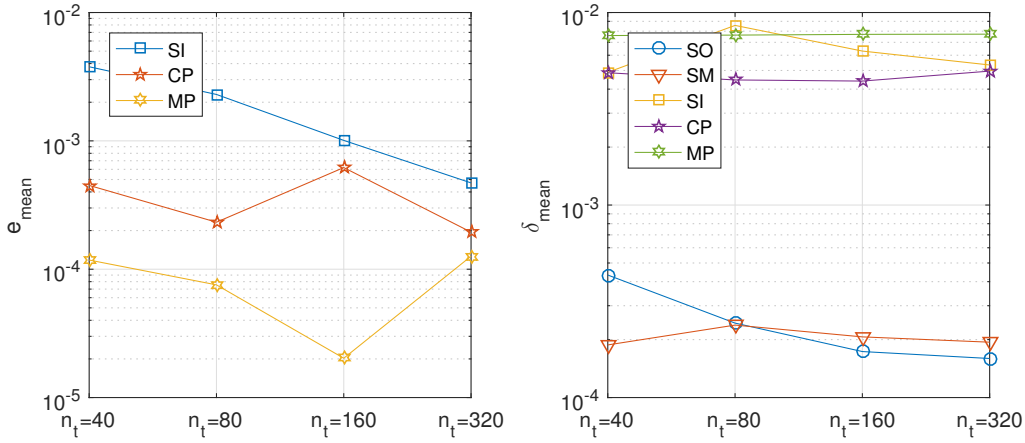

Figure 15. Fractional diffusion equation, directional decomposition. Precision of the proposed methods: sequential iterative SI, classical parallel CP, and modern parallel MP, compared with the classical ones: sequential operator SO and Strang–Marchuk SM, for different number of time steps.

Table 18 shows the results of the considered splitting algorithms for 320 time steps and different number of spatial subintervals, allowing for a maximum of three iterations and using tolerance 10^{-4}. Figure 16 presents graphically the results.

Table 18. Fractional diffusion equation. Results for the directional decomposition with 320 time steps and different number of spatial subintervals.

Splitting Algorithm	Spatial Intervals	$e_{\max}$	e_{mean}	$\delta_{\max}$	δ_{mean}	Average Iterations	Time(s)
Sequential Operator	10	3.1792×10^{-4}	2.4488×10^{-5}	1.7691×10^{-3}	3.5031×10^{-4}	1.0000	0.2665
	20	5.1954×10^{-4}	2.8614×10^{-5}	8.8420×10^{-4}	1.5956×10^{-4}	1.0000	0.4556
	40	6.7834×10^{-4}	3.0654×10^{-5}	1.2946×10^{-3}	8.9356×10^{-5}	1.0000	2.0992
Strang-Marchuk	10	6.7693×10^{-5}	9.4661×10^{-6}	1.9450×10^{-3}	3.9668×10^{-4}	1.5000	0.1661
	20	8.9844×10^{-5}	1.1077×10^{-5}	1.0605×10^{-3}	1.9393×10^{-4}	1.5000	0.5920
	40	1.2152×10^{-4}	1.1864×10^{-5}	5.9024×10^{-4}	1.0142×10^{-4}	1.5000	3.1666
Serial Iterative	10	9.7111×10^{-4}	1.0946×10^{-4}	2.1149×10^{-2}	2.6542×10^{-3}	1.7750	0.2555
	20	5.3112×10^{-3}	4.7046×10^{-4}	4.3672×10^{-2}	5.3400×10^{-3}	1.9625	0.9364
	40	2.7080×10^{-2}	2.2282×10^{-3}	8.9409×10^{-2}	1.0208×10^{-2}	1.9937	4.5270
Classical Parallel	10	7.3233×10^{-4}	5.9725×10^{-5}	2.0834×10^{-2}	2.6001×10^{-3}	1.8281	0.2669
	20	4.3711×10^{-3}	1.9453×10^{-4}	4.2061×10^{-2}	4.9619×10^{-3}	2.5344	1.1800
	40	4.3727×10^{-2}	1.6621×10^{-3}	4.8885×10^{-2}	5.8880×10^{-3}	2.9406	6.5080
Modern Parallel	10	4.3702×10^{-4}	5.3345×10^{-5}	4.1936×10^{-2}	9.1615×10^{-3}	2.0875	0.2990
	20	1.4722×10^{-3}	1.2626×10^{-4}	3.8233×10^{-2}	7.7196×10^{-3}	2.5500	1.2041
	40	4.5579×10^{-3}	1.1580×10^{-4}	3.5053×10^{-2}	6.3818×10^{-3}	2.9000	6.6945

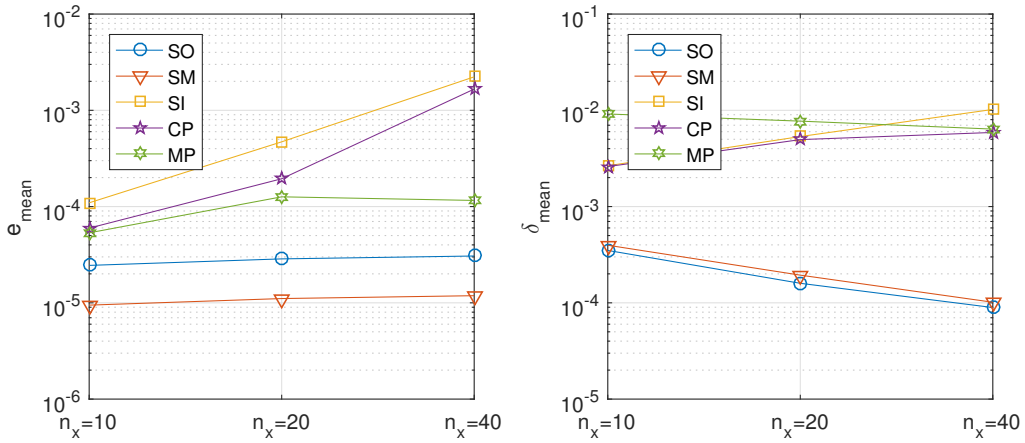

Figure 16. Fractional diffusion equation, directional decomposition. Precision of the proposed methods: sequential iterative SI, classical parallel CP, and modern parallel MP, compared with the classical ones: sequential operator SO and Strang–Marchuk SM, for 320 temporal steps and different number of spatial subintervals.

- Balanced decomposition:

We decompose into:

$$A_1 = \frac{1}{2}A, \; A_2 = \frac{1}{2}A. \tag{152}$$

Here, we have the benefit of equal load balances of the matrices, such that the exp-matrices have the same sparse structure.

Table 19 shows the results of the considered splitting algorithms for 20 spatial subintervals and different time steps, allowing a maximum of three iterations and using tolerance 10^{-4}. Figure 17 shows, on the left, that the parallel iterative methods perform slightly better than the serial iterative method and, on the right, the approximation to the analytical solution for different number of time steps.

Table 19. Fractional diffusion equation. Results for the balanced decomposition with 20 spatial subintervals and different number of temporal steps.

Splitting Algorithm	Time Steps	$e_{\max}$	e_{mean}	$\delta_{\max}$	δ_{mean}	Average Iterations	Time(s)
Sequential Operator	40	5.3158×10^{-4}	4.9157×10^{-5}	1.1895×10^{-3}	2.1594×10^{-4}	1.0000	0.9777
	80	3.1002×10^{-4}	2.5518×10^{-5}	1.0759×10^{-3}	1.8993×10^{-4}	1.0000	1.8580
	160	1.7340×10^{-4}	1.3021×10^{-5}	1.0190×10^{-3}	1.8313×10^{-4}	1.0000	3.6733
	320	9.2152×10^{-5}	6.5802×10^{-6}	9.9049×10^{-4}	1.8186×10^{-4}	1.0000	7.4484
Strang Marchuk	40	7.5805×10^{-4}	5.9623×10^{-5}	1.3763×10^{-3}	1.6935×10^{-4}	1.5000	1.4172
	80	4.2151×10^{-4}	3.0984×10^{-5}	8.6828×10^{-4}	1.5511×10^{-4}	1.5000	2.8931
	160	2.3309×10^{-4}	1.5810×10^{-5}	9.1488×10^{-4}	1.6222×10^{-4}	1.5000	5.6685
	320	1.2289×10^{-4}	7.9883×10^{-6}	9.3832×10^{-4}	1.7070×10^{-4}	1.5000	11.1808
Serial Iterative	40	4.9767×10^{-4}	6.4003×10^{-5}	1.3036×10^{-1}	7.6993×10^{-3}	3.0000	0.9013
	80	2.4927×10^{-4}	3.1592×10^{-5}	1.3045×10^{-1}	7.7224×10^{-3}	3.0000	1.5581
	160	1.2439×10^{-4}	1.6482×10^{-5}	1.3041×10^{-1}	7.7335×10^{-3}	2.5875	2.6810
	320	6.2430×10^{-5}	8.1657×10^{-6}	1.3047×10^{-1}	7.7399×10^{-3}	1.6688	3.5020
Classical Parallel	40	9.3505×10^{-4}	7.4913×10^{-5}	1.2923×10^{-1}	7.6842×10^{-3}	2.0000	1.8878
	80	7.2531×10^{-4}	5.4398×10^{-5}	1.3017×10^{-1}	7.7186×10^{-3}	2.0000	2.1625
	160	1.9158×10^{-4}	2.1530×10^{-5}	1.3089×10^{-1}	7.7522×10^{-3}	1.8062	3.8818
	320	1.0043×10^{-4}	6.3991×10^{-6}	1.3071×10^{-1}	7.7557×10^{-3}	1.3406	6.4414
Modern Parallel	40	4.9675×10^{-4}	6.3589×10^{-5}	1.3077×10^{-1}	7.7019×10^{-3}	3.0000	0.8175
	80	3.8832×10^{-4}	5.0606×10^{-5}	1.3053×10^{-1}	7.7228×10^{-3}	3.0000	1.7332
	160	1.9024×10^{-4}	2.1449×10^{-5}	1.3089×10^{-1}	7.7524×10^{-3}	2.6000	2.6822
	320	1.0046×10^{-4}	6.3750×10^{-6}	1.3071×10^{-1}	7.7557×10^{-3}	1.6719	3.4380

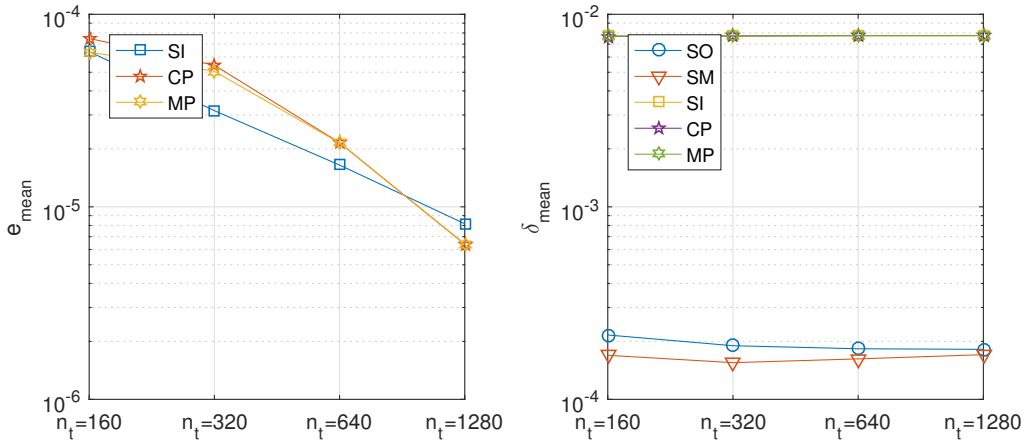

Figure 17. Fractional diffusion equation, balanced decomposition. Precision of the proposed methods: sequential iterative SI, classical parallel CP, and modern parallel MP, compared with the classical ones: sequential operator SO and Strang–Marchuk SM, for different number of time steps.

Table 20 shows the results of the considered splitting algorithms for 320 time steps and different number of spatial subintervals, allowing a maximum of three iterations and using tolerance 10^{-4}. Figure 18 graphically presents the results.

Table 20. Fractional diffusion equation. Results for the balanced decomposition with 320 time steps and different number of spatial subintervals.

Splitting Algorithm	Spatial Intervals	$e_{\max}$	e_{mean}	$\delta_{\max}$	δ_{mean}	Average Iterations	Time(s)
Sequential Operator	5	1.4662×10^{-5}	4.4384×10^{-6}	3.3195×10^{-3}	8.1165×10^{-4}	1.0000	0.1604
	10	6.0500×10^{-5}	5.6659×10^{-6}	1.8656×10^{-3}	3.8575×10^{-4}	1.0000	0.6763
	20	9.2152×10^{-5}	6.5802×10^{-6}	9.9049×10^{-4}	1.8186×10^{-4}	1.0000	14.9386
Strang Marchuk	5	3.9269×10^{-5}	4.6669×10^{-6}	3.2665×10^{-3}	8.0162×10^{-4}	1.5000	0.1949
	10	9.0735×10^{-5}	6.8613×10^{-6}	1.8124×10^{-3}	3.7471×10^{-4}	1.5000	0.8352
	20	1.2289×10^{-4}	7.9883×10^{-6}	9.3832×10^{-4}	1.7070×10^{-4}	1.5000	21.1158
Serial Iterative	5	4.4368×10^{-5}	7.8056×10^{-6}	3.0745×10^{-2}	3.9191×10^{-3}	1.2875	0.1144
	10	5.7553×10^{-5}	8.1817×10^{-6}	8.9020×10^{-2}	6.4615×10^{-3}	1.6063	0.4583
	20	6.2430×10^{-5}	8.1657×10^{-6}	1.3047×10^{-1}	7.7399×10^{-3}	1.6688	7.4995
Classical Parallel	5	1.4093×10^{-5}	3.0061×10^{-6}	3.0875×10^{-2}	3.9338×10^{-3}	1.1469	0.1123
	10	6.3414×10^{-5}	5.1056×10^{-6}	8.9191×10^{-2}	6.4759×10^{-3}	1.3062	0.4936
	20	1.0043×10^{-4}	6.3991×10^{-6}	1.3071×10^{-1}	7.7557×10^{-3}	1.3406	5.5945
Modern Parallel	5	1.4149×10^{-5}	3.0076×10^{-6}	3.0875×10^{-2}	3.9338×10^{-3}	1.2906	0.1332
	10	6.3456×10^{-5}	5.1059×10^{-6}	8.9192×10^{-2}	6.4759×10^{-3}	1.6094	0.4803
	20	1.0046×10^{-4}	6.3750×10^{-6}	1.3071×10^{-1}	7.7557×10^{-3}	1.6719	6.4903

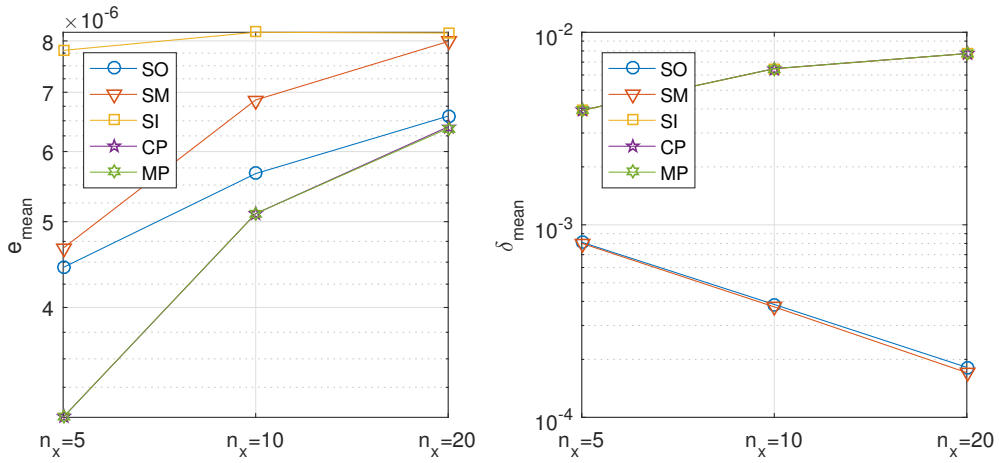

Figure 18. Fractional diffusion equation, balanced decomposition. Precision of the proposed methods: sequential iterative SI, classical parallel CP, and modern parallel MP, compared with the classical ones: sequential operator SO and Strang–Marchuk SM, for 320 temporal steps and different number of spatial subintervals.

- Mixed decomposition

We decompose into the different directions:

$$A_1 = (1-\epsilon)d(\mathbf{x})\frac{\partial^\alpha}{\partial x^\alpha} + \epsilon(e(\mathbf{x})\frac{\partial^\beta}{\partial y^\beta} + q(\mathbf{x},t)), \tag{153}$$

$$A_2 = \epsilon\, d(\mathbf{x})\frac{\partial^\alpha}{\partial x^\alpha} + (1-\epsilon)(e(\mathbf{x})\frac{\partial^\beta}{\partial y^\beta} + q(\mathbf{x},t)), \tag{154}$$

where $\epsilon = [0, 1/2]$. For $\epsilon = 0$, we have the directional decomposition, while, for $\epsilon = 1/2$, we have the balanced decomposition.

Table 21 shows the influence of ϵ on the convergence of the different splitting methods with mixed decomposition for the fractional diffusion problem using 20 subintervals in each spatial direction and 320 time steps. The same information can be seen in Figure 19.

Table 21. Fractional diffusion equation. Results for the mixed decomposition with 20 subintervals in each spatial direction, 320 time steps, and different values of ϵ.

Splitting Algorithm	ϵ	$e_{\max}$	e_{mean}	$\delta_{\max}$	δ_{mean}	Average Iterations	Time(s)
Sequential Operator	0.1	4.0169×10^{-4}	2.2303×10^{-5}	8.9352×10^{-4}	1.6047×10^{-4}	1.0000	11.7929
	0.2	2.9957×10^{-4}	1.7062×10^{-5}	9.1842×10^{-4}	1.6368×10^{-4}	1.0000	11.6719
	0.3	2.2243×10^{-4}	1.2715×10^{-5}	9.4435×10^{-4}	1.6863×10^{-4}	1.0000	11.4894
	0.4	1.5449×10^{-4}	9.1917×10^{-6}	9.6838×10^{-4}	1.7504×10^{-4}	1.0000	12.7896
Strang Marchuk	0.1	1.2924×10^{-4}	9.3575×10^{-6}	9.9183×10^{-4}	1.8011×10^{-4}	1.5000	22.0758
	0.2	1.0913×10^{-4}	7.1597×10^{-6}	9.7285×10^{-4}	1.7825×10^{-4}	1.5000	18.4355
	0.3	9.9531×10^{-5}	6.6577×10^{-6}	9.6026×10^{-4}	1.7604×10^{-4}	1.5000	18.5075
	0.4	1.0929×10^{-4}	7.1156×10^{-6}	9.5002×10^{-4}	1.7349×10^{-4}	1.5000	21.7024
Serial Iterative	0.1	8.5542×10^{-3}	1.5771×10^{-4}	1.9578×10^{-1}	1.3803×10^{-2}	1.8375	6.1359
	0.2	5.8233×10^{-3}	1.0769×10^{-4}	1.7232×10^{-1}	1.1791×10^{-2}	1.8062	6.9199
	0.3	3.4843×10^{-3}	6.5522×10^{-5}	1.5323×10^{-1}	1.0007×10^{-2}	1.7688	6.2609
	0.4	1.5374×10^{-3}	3.1802×10^{-5}	1.4154×10^{-1}	8.5870×10^{-3}	1.7188	5.8413
Classical Parallel	0.1	1.4317×10^{-3}	4.2833×10^{-5}	1.3293×10^{-1}	7.8025×10^{-3}	1.3375	4.7969
	0.2	8.3872×10^{-4}	2.5891×10^{-5}	1.3196×10^{-1}	7.7821×10^{-3}	1.3375	5.5666
	0.3	4.1720×10^{-4}	1.4159×10^{-5}	1.3126×10^{-1}	7.7674×10^{-3}	1.3375	5.7738
	0.4	1.6622×10^{-4}	8.0251×10^{-6}	1.3085×10^{-1}	7.7586×10^{-3}	1.3406	4.7526
Modern Parallel	0.1	6.0380×10^{-4}	2.1615×10^{-5}	1.3061×10^{-1}	7.7407×10^{-3}	3.0000	10.3976
	0.2	8.3108×10^{-4}	2.5046×10^{-5}	1.3055×10^{-1}	7.7404×10^{-3}	3.0000	10.1307
	0.3	1.1571×10^{-3}	2.8534×10^{-5}	1.3007×10^{-1}	7.7378×10^{-3}	2.9297	11.0316
	0.4	1.6138×10^{-4}	1.1869×10^{-5}	1.3084×10^{-1}	7.7536×10^{-3}	2.3078	7.8798

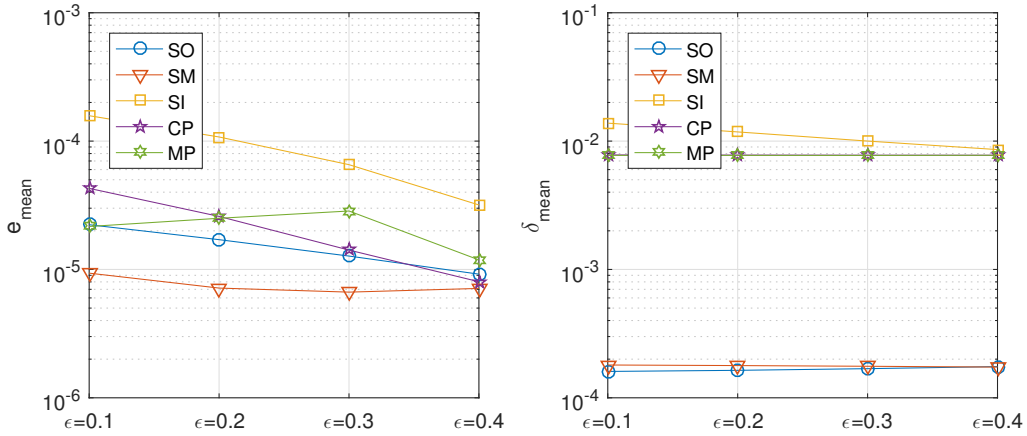

Figure 19. Fractional diffusion equation, mixed decomposition. Precision of the proposed methods: sequential iterative SI, classical parallel CP, and modern parallel MP, compared with the classical ones: sequential operator SO and Strang–Marchuk SM, for 20 temporal steps, 320 temporal steps, and different values of ϵ.

Remark 8. *For the fractional diffusion model, we could also obtain the same results as in the previous diffusion and Burgers' equation. We could stabilize the schemes with respect to $\epsilon \neq 0$ and obtain a good load balance of the matrices. Here, we could apply real-life problems with respect to the fractional differential equations, while we developped a stable novel parallel solver scheme.*

6. Conclusions

In the paper, we have discussed the extensions of the iterative splitting approaches to parallel solver methods. Such novel methods allow for reducing the computational time. We can achieve the same accuracy as in the serial version. The improvements are obtained with larger time-steps and additional iterative steps, where we could reduce the computational time with the parallel methods. The benefit is, of course, the balance to multiple processors with additional memories. Further, we could apply the resources to improve with additional steps the accuracy of the approximations. We circumvent the problem of the memory of the algorithm, see [38], which we have if we only apply a

serial method. Based on the parallel distribution, we have additional iterative steps for each processor and we distribute such a memory to all processors. For large scale numerical experiments, we could present the benefit of the parallel resources.

In our proposed iterative methods, we gain accuracy if we apply more iterative cycles, so that we have to devote additional computational time. We could reduce the computational cost more than with serial iterative methods due to the application of parallel ideas. On the other hand, it is important to optimize the parallel amount of work with additional adaptive and distributed ideas that improve the efficiency of the proposed parallel methods.

In summary, we optimize the reduction of computational time and the results accuracy with the help of parallel iterative splitting methods. In the future, we will consider more real-life problems and stochastic processes. Furthermore, we will extend the parallel iterative methods to more efficient adaptive schemes.

Author Contributions: The theory, the formal analysis and the methodology presented in this paper were developed by J.G. The software development and the numerical validation of the methods were done by J.L.H. and E.M. The paper was written by J.G., J.L.H. and E.M. and was corrected and edited by J.G., J.L.H. and E.M. The writing–review was done by J.G., J.L.H. and E.M. The supervision and project administration were done by E.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially supported by Ministerio de Economía y Competitividad, Spain, under grant PGC2018-095896-B-C21-C22 and German Academic Exchange Service grant number 91588469.

Acknowledgments: We acknowledge support by the DFG Open Access Publication Funds of the Ruhr-Universität of Bochum, Germany.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Farago, I.; Geiser, J. Iterative Operator-Splitting methods for Linear Problems. *Int. J. Comput. Sci. Eng.* **2007**, *3*, 255263.
2. Geiser, J. *Iterative Splitting Methods for Differential Equations*; Numerical Analysis and Scientific Computing Series; Taylor & Francis Group: Boca Raton, FL, USA; London, UK; New York, NY, USA, 2011.
3. Frommer, A.; Szyld, D.B. On asynchronous iterations. *J. Comput. Appl. Math.* **2000**, *123*, 201–216.
4. O'Leary, D.P.; White, R.E. Multi-splittings of matrices and parallel solution of linear systems. *Siam Algebr. Discret. Methods* **1985**, *6*, 630–640.
5. White, R.E. Parallel algorithms for nonlinear problems. *Siam Algebr. Discret. Methods* **1986**, *7*, 137–149 .
6. Geiser, J. Iterative Splitting Methods for Coulomb Collisions in Plasma Simulations. *arXiv* **2017**, arXiv:1706.06744.
7. Geiser, J. Picard's iterative method for nonlinear multicomponent transport equations. *Cogent Math.* **2016**, *3*, 1158510.
8. Miekkala, U.; Nevanlinna, O. Convergence of dynamic iteration methods for initial value problems. *SIAM J. Sci. Stat. Comput.* **1987**, *8*, 459–482.
9. Miekkala, U.; Nevanlinna, O. Iterative solution of systems of linear differential equations. *Acta Numer.* **1996**, *5*, 259–307.
10. Vandewalle, S. *Parallel Multigrid Waveform Relaxation for Parabolic Problems*; Teubner Skripten zur Numerik, B.G. Teubner Stuttgart; Springer: Berlin/Heidelberg, Germany, 1993.
11. Geiser, J. *Multicomponent and Multiscale Systems: Theory, Methods, and Applications in Engineering*; Springer: Berlin/Heidelberg, Germany, 2016.
12. Geiser, J. Multi-stage waveform Relaxation and Multisplitting Methods for Differential Algebraic Systems. *arXiv* **2016**, arXiv:1601.00495.
13. Geiser, J. Iterative operator-splitting methods for nonlinear differential equations and applications. *Numer. Methods Partial. Differ. Equ.* **2011**, *27*, 1026–1054.
14. He, D.; Pan, K.; Hu, H. A spatial fourth-order maximum principle preserving operator splitting scheme for the multi-dimensional fractional Allen-Cahn equation. *Appl. Numer. Math.* **2020**, *151*, 44–63.

15. Haberman, R. *Mathematical Models: Mechanical Vibrations, Population Dynamics, and Traffic Flow*; Society for Industrial and Applied Mathematics (SIAM): Philadelphia, PA, USA, 1998.
16. Orlandi, P. (Ed.) The Burgers equation. In *Fluid Flow Phenomena: A Numerical Toolkit*; Springer: Dordrecht, The Netherlands, 2000; pp. 40–50.
17. Ginoa, M.; Cerbelli, S.; Roman, H.E. Fractional diffusion equation and relaxation in complex viscoelastic materials. *Physica A* **1992**, *191*, 449–453.
18. Nigmatullin, R.R. The realization of the generalized transfer equation in a medium with fractal geometry. *Phys. Stat. Sol. B* **1986**, *133*, 425–430.
19. Allen, S.M.; Cahn, J.W. A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening. *Acta Metall.* **1979**, *27*, 1085–1095.
20. Yue, P.; Feng, J.; Liu, C.; Shen, J. Diffuse-interface simulations of drop coalescence and retraction in viscoelastic fluids. *J. Non-Newton. Fluid Mech.* **2005**, *129*, 163–176.
21. Sommacal, L.; Melchior, P.; Oustaloup, A.; Cabelguen, J.-M.; Ijspeert, A.J. Fractional Multi-models of the Frog Gastrocnemius Muscle. *J. Vib. Control.* **2008**, *14*, 1415–1430.
22. Moshrefi-Torbati, M.; Hammond, J.K. Physical and geometrical interpretation of fractional operators. *J. Frankl. Inst.* **1998**, *335*, 1077–1086.
23. Rami Ahmad El-Nabulsi. Fractional Dirac operators and deformed field theory on Clifford algebra. *Chaos Solitons Fractals* **2009**, *42*, 2614–2622.
24. Kilbas, A.A.; Srivastava, H.M.; Trujillo, J.J. Theory and Applications of Fractional Differential Equations. In *Mathematics Studies*, 1st ed.; Elesevier: North-Holland, The Netherlands, 2006; Volume 204.
25. Kanney, J.; Miller, C.; Kelley, C.T. Convergence of iterative split-operator approaches for approximating nonlinear reactive transport problems. *Adv. Water Resour.* **2003**, *26*, 247–261.
26. Geiser, J.; Hueso, J.L.; Martinez, E. Adaptive Iterative Splitting Methods for convection-diffusion-reaction equations. *Mathematics* **2020**, *8*, 302.
27. Meerschaert, M.M.; Scheffler, H.P.; Tadjeran, C. Finite difference methods for two-dimensional fractional dispersion equation. *J. Comput. Phys.* **2006**, *211*, 249–261.
28. Argyros, I.K.; Regmi, S. *Undergraduate Research at Cameron University on Iterative Procedures in Banach and Other Spaces*; Nova Science Publisher: New York, NY, USA, 2019.
29. Cresson, J.; Inizan, P. Irreversibility, Least Action Principle and Causality. Preprint, HAL, 2008. Available online: https://hal.archives-ouvertes.fr/hal-00348123v1 (accessed on 11 April 2020).
30. Cresson, J. Fractional embedding of differential operators and Lagrangian systems. *J. Math. Phys.* **2007**, *48*, 033504.
31. Gustafsson, B. *High Order Difference Methods for Time Dependent PDE*; Springer Series in Computational Mathematics; Springer: Berlin/Heidelberg, Germany, 2007; Volume 38.
32. Meerschaert, M.M.; Tadjeran, C. Finite difference approximations for fractional advection–dispersion flow equations. *J. Comput. Appl. Math.* **2003**, *172*, 65–77.
33. Geiser, J. Computing Exponential for Iterative Splitting Methods: Algorithms and Applications. *J. Appl. Math.* **2011**, *2011*, 193781.
34. Geiser, J. Iterative Operator-Splitting Methods with Higher Order Time-Integration Methods and Applications for Parabolic Partial Differential Equations. *J. Comput. Appl. Math.* **2008**, *217*, 227–242.
35. Ladics, T. Error analysis of waveform relaxation method for semi-linear partial differential equations. *J. Comput. Appl. Math.* **2015**, *285*, 15–31.
36. Kelley, C.T. *Iterative Methods for Linear and Nonlinear Equations*; SIAM Frontiers in Applied Mathematics, no. 16; Society for Industrial and Applied Mathematics (SIAM): Philadelphia, PA, USA, 1995.
37. Yuan, D.; Burrage, K. Convergence of the parallel chaotic waveform relaxation method for stiff systems. *J. Comput. Appl. Math.* **2003**, *151*, 201–213.
38. Ladics, T.; Farago, I. Generalizations and error analysis of the iterative operator splitting method. *Cent. Eur. J. Math.* **2013**, *11*, 1416–1428.
39. Moler, C.B.; Loan, C.F.V. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Rev.* **2003**, *45*, 3–49.
40. Najfeld, I.; Havel, T.F. Derivatives of the matrix exponential and their computation. *Adv. Appl. Math.* **1995**, *16*, 321–375.
41. Hochbruck, M.; Ostermann, A. Exponential integrators. *Acta Numer.* **2010**, *19*, 209–286.

42. Casas, F.; Iserles, A. Explicit Magnus expansions for nonlinear equations. *J. Phys. A Math. Gen.* **2006**, *39*, 5445–5461.
43. Magnus, W. On the exponential solution of differential equations for a linear operator. *Commun. Pure Appl. Math.* **1954**, *7*, 649–673.
44. Stoer, J.; Bulirsch, R. *Introduction to Numerical Analysis*; Texts in Applied Mathematics No.12; Springer: New York, NY, USA, 2002.
45. Jeltsch, R.; Pohl, B. Waveform Relaxation with Overlapping Splittings. *SIAM J. Sci.Comput.* **1995**, *16*, 40–49.
46. Farago, I. A Modified iterated operator-splitting method. *Appl. Math. Model.* **2008**, *32*, 1542–1551.
47. Hairer, E.; Lubich, C.; Wanner, G. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*; Springer: Berlin-Heidelberg, Germany; New York, NY, USA, 2002.
48. Li, J.; Jiang, Y.-L.; Miao, Z. A parareal approach of semi-linear parabolic equations based on general waveform relaxation. *Numer. Methods Partial. Differ. Equ.* **2019**, *35*, 2017–2034.
49. Trotter, H.F. On the product of semi-groups of operators. *Proc. Am. Math. Soc.* **1959**, *10*, 545–551.
50. Strang, G. On the construction and comparision of difference schemes. *SIAM J. Numer. Anal.* **1968**, *5*, 506–517.
51. Geiser, J. Operator-Splitting Methods in Respect of Eigenvalue Problems for Nonlinear Equations and Applications to Burgers Equations. *J. Comput. Appl. Math.* **2009**, *231*, 815–827.