Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2011, Article ID 193781, 27 pages
doi:10.1155/2011/193781

Research Article

Computing Exponential for Iterative Splitting Methods: Algorithms and Applications

Jürgen Geiser

Humboldt-Univeristät zu Berlin, 10099 Berlin, Germany

Correspondence should be addressed to Jürgen Geiser, geiser@mathematik.hu-berlin.de

Received 29 November 2010; Accepted 27 January 2011

Academic Editor: Shuyu Sun

Iterative splitting methods have a huge amount to compute matrix exponential. Here, the acceleration and recovering of higher-order schemes can be achieved. From a theoretical point of view, iterative splitting methods are at least alternating Picards fix-point iteration schemes. For practical applications, it is important to compute very fast matrix exponentials. In this paper, we concentrate on developing fast algorithms to solve the iterative splitting scheme. First, we reformulate the iterative splitting scheme into an integral notation of matrix exponential. In this notation, we consider fast approximation schemes to the integral formulations, also known as ϕ-functions. Second, the error analysis is explained and applied to the integral formulations. The novelty is to compute cheaply the decoupled exp-matrices and apply only cheap matrix-vector multiplications for the higher-order terms. In general, we discuss an elegant way of embedding recently survey on methods for computing matrix exponential with respect to iterative splitting schemes. We present numerical benchmark examples, that compared standard splitting schemes with the higher-order iterative schemes. A real-life application in contaminant transport as a two phase model is discussed and the fast computations of the operator splitting method is explained.

1. Introduction

We are motivated to solving multiple phase problems that arose of transport problem in porous media. In the last years, the interest in numerical simulations with multiphase problems, that can be used to model potential damage events has significantly increased in the area of final repositories for chemical or radioactive waste.

Here, the modeling of the underlying porous media, which is the geosphere, spooled with water, is important, and we apply mobile and immobile pores as a two-phase problem in the media. With such a model, we achieve more realistic scenarios of the transported species, see [1]. The transport is structured in a convection and a diffusion portion. The convection portion is determined by the velocity vector $\mathbf{v}$ and indicates the spatial direction of the concentrations with the velocity. The diffusion portion is the spatial diffusion process of the concentrations, see [1].

The sorption allegorizes the exchange between the solute (mobile) pollutant and at the surface sorbed (immobile) pollutants and appears in the temporal term as well as in the reaction term. The reaction is reversible. The equilibrium-sorption therefore can be declared as a coefficient in the specific terms.

We concentrate on simplified models and taken into account all real conditions for achieving statements of such realistic simulations of potential damage event. Here we have the delicate problem of coupled partial and ordinary differential equations and their underlying multiscale problems. While we deal with splitting methods and decomposing such scale problems, we can overcome such scaling problems, see [2].

Moreover the computational part is important, while we dealing with large matrices and standard splitting schemes are expensive with respect to compute the exponential matrices. We solve the computational problem with a novel iterative splitting scheme, that concentrate on developing fast algorithms to solve an integral formulation of matrix exponentials. In such a scheme, we consider fast approximation schemes to the integral formulations, also known as ϕ-functions, see [3]. The novelty is to compute cheaply the decoupled exp matrices and apply only cheap matrix-vector multiplications for the higher-order terms. In general, we discuss an elegant way of embedding recent survey on methods for computing matrix exponential with respect to iterative splitting schemes.

In the following, we describe our model problem. The model equation for the multiple phase equations are given as coupled partial and ordinary differential equations:

$$\begin{aligned}\phi\partial_t c_i + \nabla\cdot\mathbf{F}_i &= g(-c_i + c_{i,\mathrm{im}}) + k_\alpha(-c_i + c_{i,\mathrm{ad}}) \\ &\quad -\lambda_{i,i}\phi c_i + \sum_{k=k(i)}\lambda_{i,k}\phi c_k + \widetilde{Q}_i, \quad \text{in } \Omega\times[0,t],\end{aligned} \tag{1.1}$$

$$\mathbf{F}_i = \mathbf{v}c_i - D^{e(i)}\nabla c_i + \eta\mathbf{E}c_i, \tag{1.2}$$

$$\begin{aligned}\phi\partial_t c_{i,\mathrm{im}} &= g(c_i - c_{i,\mathrm{im}}) + k_\alpha(c_{i,\mathrm{im,ad}} - c_{i,\mathrm{im}}) \\ &\quad -\lambda_{i,i}\phi c_{i,\mathrm{im}} + \sum_{k=k(i)}\lambda_{i,k}\phi c_{k,\mathrm{im}} + \widetilde{Q}_{i,\mathrm{im}}, \quad \text{in } \Omega\times[0,t],\end{aligned} \tag{1.3}$$

$$\phi\partial_t c_{i,\mathrm{ad}} = k_\alpha(c_i - c_{i,\mathrm{ad}}) - \lambda_{i,i}\phi c_{i,\mathrm{ad}} + \sum_{k=k(i)}\lambda_{i,k}\phi c_{k,\mathrm{ad}} + \widetilde{Q}_{i,\mathrm{ad}}, \quad \text{in } \Omega\times[0,t], \tag{1.4}$$

$$\begin{aligned}\phi\partial_t c_{i,\mathrm{im,ad}} &= k_\alpha(c_{i,\mathrm{im}} - c_{i,\mathrm{im,ad}}) - \lambda_{i,i}\phi c_{i,\mathrm{im,ad}} \\ &\quad + \sum_{k=k(i)}\lambda_{i,k}\phi c_{k,\mathrm{im,ad}} + \widetilde{Q}_{i,\mathrm{im,ad}}, \quad \text{in } \Omega\times[0,t],\end{aligned} \tag{1.5}$$

$$c_i(\mathbf{x},t) = c_{i,0}(\mathbf{x}), \quad c_{i,\mathrm{ad}}(\mathbf{x},t) = 0, \quad c_{i,\mathrm{im}}(\mathbf{x},t) = 0, \quad c_{i,\mathrm{im,ad}}(\mathbf{x},t) = 0, \quad \text{on } \Omega, \tag{1.6}$$

$$c_i(\mathbf{x},t) = c_{i,1}(\mathbf{x},t), \quad c_{i,\mathrm{ad}}(\mathbf{x},t) = 0, \quad c_{i,\mathrm{im}}(\mathbf{x},t) = 0, \quad c_{i,\mathrm{im,ad}}(\mathbf{x},t) = 0, \quad \text{on } \partial\Omega\times[0,t], \tag{1.7}$$

where the initial value is given as $c_{i,0}$ and we assume Dirichlet boundary conditions with the function $c_{i,1}(x,t)$ sufficiently smooth, all other initial and boundary conditions of the other phases zero.

ϕ:	effective porosity (−),
c_i:	concentration of the ith gaseous species in the plasma chamber phase (mol/cm^3),
$c_{i,\mathrm{im}}$:	concentration of the ith gaseous species in the immobile zones of the plasma chamber phase (mol/cm^3),
$c_{i,\mathrm{ad}}$:	concentration of the ith gaseous species in the mobile adsorbed zones of the plasma chamber phase (mol/cm^3),
$c_{i,\mathrm{im,ad}}$:	concentration of the ith gaseous species in the immobile adsorbed zones of the plasma chamber phase (mol/cm^3),
$\mathbf{v}$:	velocity through the chamber and porous substrate [4] (cm/nsec),
$D^{e(i)}$:	element-specific diffusions-dispersions tensor (cm^2/nsec),
$\lambda_{i,i}$:	decay constant of the ith species (1/nsec),
$\tilde{Q}_i$:	source term of the ith species (mol/(cm^{3}nsec)),
g:	exchange rate between the mobile and immobile concentration (1/nsec),
k_α:	exchange rate between the mobile and adsorbed concentration or immobile and immobile adsorbed concentration (kinetic controlled sorption) (1/nsec),
$\mathbf{E}$:	stationary electric field in the apparatus (V/cm),
η:	the mobility rate in the electric field, see [5] (cm^2/nsec).

with $i = 1, \ldots, M$ and M denotes the number of components.

The parameters in (1.1) are further described, see also [1].

The effective porosity is denoted by ϕ and declares the portion of the porosities of the aquifer, that is filled with plasma, and we assume a nearly fluid phase. The transport term is indicated by the Darcy velocity $\mathbf{v}$, that presents the flow-direction and the absolute value of the plasma flux. The velocity field is divergence-free. The decay constant of the ith species is denoted by λ_i. Thereby does $k(i)$ denote the indices of the other species.

In this paper we concentrate on solving linear evolution equations, such as the differential equation,

$$\partial_t u = Au, \qquad u(0) = u_0, \tag{1.8}$$

where A can be an unbounded operator. We assume to achieve large-scale differential equation, which are delicate to solve with standard solvers.

Our main focus will be to consider and contrast higher-order algorithms derived from standard schemes as for example Strang-splitting schemes.

We propose iterative splitting schemes as a solver scheme which is simple to implement and parallelisible.

A rewriting to integral formulation, allows to reduce the computation to numerical approximation schemes. While standard schemes has the disadvantage to compute commutators to achieve higher-order schemes, we could speed up our schemes by recursive algorithms. Iterative schemes can be seen as Successive approximations, which are based on recursive integral formulations in which an iterative method is enforce the time dependency.

The paper is outlined as follows. In Section 2, we present the underlying splitting methods. The algorithms for the exponentials are given in Section 3. Numerical verifications are given in Section 4. In Section 5, we briefly summarize our results.

2. Iterative Splitting Method

The following algorithm is based on the iteration with fixed-splitting discretization step-size τ, namely, on the time interval $[t^n, t^{n+1}]$ we solve the following subproblems consecutively for $i = 0, 2, \ldots, 2m$. (cf. [6, 7]):

$$\begin{aligned} \frac{\partial c_i(t)}{\partial t} &= Ac_i(t) + Bc_{i-1}(t), \quad \text{with } c_i(t^n) = c^n,\ c_0(t^n) = c^n,\ c_{-1} = 0.0, \\ \frac{\partial c_{i+1}(t)}{\partial t} &= Ac_i(t) + Bc_{i+1}(t), \quad \text{with } c_{i+1}(t^n) = c^n, \end{aligned} \tag{2.1}$$

where c^n is the known split approximation at the time level $t = t^n$. The split approximation at the time level $t = t^{n+1}$ is defined as $c^{n+1} = c_{2m+1}(t^{n+1})$, (clearly, the function $c_{i+1}(t)$ depends on the interval $[t^n, t^{n+1}]$, too, but, for the sake of simplicity, in our notation we omit the dependence on n).

In the following we will analyze the convergence and the rate of convergence of the method (2.1) for m tends to infinity for the linear operators $A, B : \mathbf{X} \to \mathbf{X}$, where we assume that these operators and their sum are generators of the C_0 semigroups. We emphasize that these operators are not necessarily bounded, so the convergence is examined in a general Banach space setting.

The novelty of the convergence results are the reformulation in integral-notation. Based on this, we can assume to have bounded integral operators which can be estimated and given in a recursive form. Such formulations are known in the work of [8, 9] and estimations of the kernel part with the exponential operators are sufficient to estimate the recursive formulations.

2.1. Error Analysis

We present the results of the consistency of our iterative method. We assume for the system of operator the generator of a C_0 semigroup based on their underlying graph norms.

Theorem 2.1. *Let one consider the abstract Cauchy problem in a Hilbert space* $\mathbf{X}$

$$\begin{gathered} \partial_t c(x,t) = Ac(x,t) + Bc(x,t), \quad 0 < t \le T,\ x \in \Omega, \\ c(x,0) = c_0(x), \quad x \in \Omega, \\ c(x,t) = c_1(x,t), \quad x \in \partial\Omega \times [0,T], \end{gathered} \tag{2.2}$$

where $A, B : D(\mathbf{X}) \rightarrow \mathbf{X}$ are given linear operators which are generators of the C_0-semigroup and $c_0 \in \mathbf{X}$ is a given element. We assume A, B are unbounded. Further, we assume the estimations of the unbounded operator B with sufficient smooth initial conditions (see [8]):

$$\left\| B \exp((A+B)\tau) u_0 \right\| \leq \kappa. \tag{2.3}$$

Further we assume the estimation of the partial integration of the unbounded operator B (see [8]):

$$\left\| B \int_0^\tau \exp(Bs) s ds \right\| \leq \tau C. \tag{2.4}$$

Then, we can bound our iterative operator splitting method as

$$\left\| (S_i - \exp((A+B)\tau)) \right\| \leq C\tau^i, \tag{2.5}$$

where S_i is the approximated solution for the ith iterative step and C is a constant that can be chosen uniformly on bounded time intervals.

Proof. Let us consider the iteration (2.1) on the subinterval $[t^n, t^{n+1}]$.

For the first iterations, we have

$$\partial_t c_1(t) = A c_1(t), \quad t \in \left(t^n, t^{n+1}\right], \tag{2.6}$$

and for the second iteration we have:

$$\partial_t c_2(t) = A c_1(t) + B c_2(t), \quad t \in \left(t^n, t^{n+1}\right]. \tag{2.7}$$

In general, we have:

(i) for the odd iterations, $i = 2m + 1$ for $m = 0, 1, 2, \ldots$

$$\partial_t c_i(t) = A c_i(t) + B c_{i-1}(t), \quad t \in \left(t^n, t^{n+1}\right], \tag{2.8}$$

where for $c_0(t) \equiv 0$,

(ii) for the even iterations, $i = 2m$ for $m = 1, 2, \ldots$

$$\partial_t c_i(t) = A c_{i-1}(t) + B c_i(t), \quad t \in \left(t^n, t^{n+1}\right]. \tag{2.9}$$

We have the following solutions for the iterative scheme: the solutions for the first two equations are given by the variation of constants:

$$\begin{aligned} c_1(t) &= \exp\left(A\left(t^{n+1}-t\right)\right)c(t^n), \quad t \in \left(t^n, t^{n+1}\right], \\ c_2(t) &= \exp(Bt)c(t^n) + \int_{t^n}^{t^{n+1}} \exp\left(B\left(t^{n+1}-s\right)\right)Ac_1(s)ds, \quad t \in \left(t^n, t^{n+1}\right]. \end{aligned} \tag{2.10}$$

For the recurrence relations with even and odd iterations, we have the solutions: for the odd iterations: $i = 2m+1$, for $m = 0, 1, 2, \ldots$,

$$c_i(t) = \exp(A(t-t^n))c(t^n) + \int_{t^n}^{t} \exp\left(\left(t^{n+1}-s\right)A\right)Bc_{i-1}(s)ds, \quad t \in \left(t^n, t^{n+1}\right]. \tag{2.11}$$

For the even iterations, $i = 2m$, for $m = 1, 2, \ldots$

$$c_i(t) = \exp(B(t-t^n))c(t^n) + \int_{t^n}^{t} \exp\left(\left(t^{n+1}-s\right)B\right)Ac_{i-1}(s)ds, \quad t \in \left(t^n, t^{n+1}\right]. \tag{2.12}$$

The consistency is given as the following.

For e_1, we have

$$\begin{aligned} c_1(\tau) &= \exp(A\tau)c(t^n), \\ c(\tau) &= \exp((A+B)\tau)c(t^n) = \exp(A\tau)c(t^n) \\ &\quad + \int_{t^n}^{t^{n+1}} \exp\left(A\left(t^{n+1}-s\right)\right)B\exp(s(A+B))c(t^n)ds. \end{aligned} \tag{2.13}$$

We obtain

$$\|e_1\| = \|c - c_1\| \le \left\|\exp((A+B)\tau)c(t^n) - \exp(A\tau)c(t^n)\right\| \le C_1\tau c(t^n). \tag{2.14}$$

For e_2 we have (alternating)

$$\begin{aligned} c_2(\tau) &= \exp(B\tau)c(t^n) + \int_{t^n}^{t^{n+1}} \exp\left(B\left(t^{n+1}-s\right)\right)A\exp(sA)c(t^n)ds, \\ c(\tau) &= \exp(B\tau)c(t^n) + \int_{t^n}^{t^{n+1}} \exp\left(B\left(t^{n+1}-s\right)\right)A\exp(sA)c(t^n)ds \\ &\quad + \int_{t^n}^{t^{n+1}} \exp(Bs)A\int_{t^n}^{t^{n+1}-s} \exp\left(A\left(t^{n+1}-s-\rho\right)\right)B\exp(\rho(A+B))c(t^n)d\rho ds. \end{aligned} \tag{2.15}$$

We obtain

$$\|e_2\| \leq \left\|\exp((A+B)\tau)c(t^n) - c_2\right\| \leq C_2\tau^2 c(t^n). \tag{2.16}$$

For odd and even iterations, the recursive proof is given in the following. For the odd iterations (only iterations over A), $i = 2m+1$ for $m = 0, 1, 2, \ldots$, for e_i, we have:

$$\begin{aligned}
c_i(\tau) &= \exp(A\tau)c(t^n) \\
&\quad + \int_{t^n}^{t^{n+1}} \exp\Big(A\Big(t^{n+1} - s\Big)\Big) B \exp(sA) c(t^n)\, ds \\
&\quad + \int_{t^n}^{t^{n+1}} \exp\Big(A\Big(t^{n+1} - s_1\Big)\Big) B \int_{t^n}^{t^{n+1}-s_1} \exp\Big(\Big(t^{n+1} - s_1 - s_2\Big)A\Big) B \exp(s_2 A) c(t^n)\, ds_2\, ds_1 \\
&\quad + \cdots \\
&\quad + \int_{t^n}^{t^{n+1}} \exp\Big(A\Big(t^{n+1} - s_1\Big)\Big) B \int_{t^n}^{t^{n+1}-s_1} \exp\Big(\Big(t^{n+1} - s_1 - s_2\Big)A\Big) B \exp(s_2 A) c(t^n)\, ds_2\, ds_1 + \cdots \\
&\quad + \int_{t^n}^{t^{n+1}} \exp\Big(A\Big(t^{n+1} - s_1\Big)\Big) B \cdots \int_{t^n}^{t^{n+1} - \sum_{j=1}^{i-2} s_j} \exp\left(\left(t^{n+1} - \sum_{j=1}^{i-1} s_j\right) A\right) B \\
&\qquad \times \exp\,(s_i A) c(t^n)\, ds_{i-1} \cdots ds_2\, ds_1, \\
c(\tau) &= \exp(A\tau)c(t^n) \\
&\quad + \int_{t^n}^{t^{n+1}} \exp\Big(A\Big(t^{n+1} - s\Big)\Big) B \exp(sA) c(t^n)\, ds \\
&\quad + \int_{t^n}^{t^{n+1}} \exp\Big(A\Big(t^{n+1} - s_1\Big)\Big) B \int_{t^n}^{t^{n+1}-s_1} \exp\Big(\Big(t^{n+1} - s_1 - s_2\Big)A\Big) B \exp(s_2 A) c(t^n)\, ds_2\, ds_1 \\
&\quad + \cdots \\
&\quad + \int_{t^n}^{t^{n+1}} \exp\Big(A\Big(t^{n+1} - s_1\Big)\Big) B \int_{t^n}^{t^{n+1}-s_1} \exp\Big(\Big(t^{n+1} - s_1 - s_2\Big)A\Big) B \exp(s_2 A) c(t^n)\, ds_2\, ds_1 + \cdots \\
&\quad + \int_{t^n}^{t^{n+1}} \exp\Big(A\Big(t^{n+1} - s_1\Big)\Big) B \cdots \int_{t^n}^{t^{n+1} - \sum_{j=1}^{i-2} s_j} \exp\left(\left(t^{n+1} - \sum_{j=1}^{i} s_j\right) A\right) B \\
&\qquad \times \exp\,(s_i A) c(t^n)\, ds_i \cdots ds_2\, ds_1.
\end{aligned} \tag{2.17}$$

We obtain

$$\|e_i\| \le \left\|\exp((A+B)\tau)c(t^n) - c_i\right\| \le C\tau^i c(t^n), \tag{2.18}$$

where i is the number of iterative steps.

The same idea can be applied to the even iterative scheme and also for alternating A and B. □

Remark 2.2. The same idea can be applied to $A = \nabla D \nabla \;\; B = -\mathbf{v}\cdot\nabla$, so that one operator is less unbounded, but we reduce the convergence order

$$\begin{gathered}\|e_1\| = K\|B\|\tau^{\alpha_1}\|e_0\| + \mathcal{O}\left(\tau^{1+\alpha_1}\right), \\ \text{and hence,} \\ \|e_2\| = K\|B\|\|e_0\|\tau^{1+\alpha_1+\alpha_2} + \mathcal{O}\left(\tau^{1+\alpha_1+\alpha}\right),\end{gathered} \tag{2.19}$$

where $0 \le \alpha_1, \alpha_2 < 1$.

Remark 2.3. If we assume the consistency of $\mathcal{O}(\tau^m)$ for the initial value $e_1(t^n)$ and $e_2(t^n)$, we can redo the proof and obtain at least a global error of the splitting methods of $\mathcal{O}(\tau^{m-1})$.

2.2. Splitting Method to Couple Mobile, Immobile, and Adsorbed Parts

The motivation of the splitting method are based on the following observations.

(i) The mobile phase is semidiscretized with fast finite volume methods and can be stored into a stiffness-matrix. We achieve large time steps, if we consider implicit Runge-Kutta methods of lower order (e.g., implicit Euler) as a time discretization method.

(ii) The immobile, adsorbed and immobile-adsorbed phases are purely ordinary differential equations and each of them is cheap to solve with explicit Runge-Kutta schemes.

(iii) The ODEs can be seen as perturbations and can be solved all explicit in a fast iterative scheme.

For the full equation we consider the following matrix notation:

$$\begin{gathered}\partial_t \mathbf{c} = A_1\mathbf{c} + A_2\mathbf{c} + B_1(\mathbf{c} - \mathbf{c}_{\text{im}}) + B_2(\mathbf{c} - \mathbf{c}_{\text{ad}}) + \mathbf{Q}, \\ \partial_t \mathbf{c}_{\text{im}} = A_2\mathbf{c}_{\text{im}} + B_1(\mathbf{c}_{\text{im}} - \mathbf{c}) + B_2(\mathbf{c}_{\text{im}} - \mathbf{c}_{\text{im,ad}}) + \mathbf{Q}_{\text{im}}, \\ \partial_t \mathbf{c}_{\text{ad}} = A_2\mathbf{c}_{\text{ad}} + B_2(\mathbf{c}_{\text{ad}} - \mathbf{c}) + \mathbf{Q}_{\text{ad}}, \\ \partial_t \mathbf{c}_{\text{im,ad}} = A_2\mathbf{c}_{\text{im,ad}} + B_2(\mathbf{c}_{\text{im,ad}} - \mathbf{c}_{\text{im}}) + \mathbf{Q}_{\text{im,ad}},\end{gathered} \tag{2.20}$$

where $\mathbf{c} = (c_1, \ldots, c_m)^T$ is the spatial discretized concentration in the mobile phase, see (1.1), $\mathbf{c}_{\text{im}} = (c_{1,\text{im}}, \ldots, c_{m,\text{im}})^T$ is the concentration in the immobile phase, the some also for the other phase concentrations. A_1 is the stiffness matrix of (1.1), A_2 is the reaction matrix of the right-hand side of (1.1), B_1 and B_2 are diagonal matrices with the exchange of the immobile and kinetic parameters, see (1.4) and (1.5).

Further, $\mathbf{Q}, \ldots, \mathbf{Q}_{\text{im,ad}}$ are the spatial discretized sources vectors.

Now, we have the following ordinary differential equation:

$$\partial_t \mathbf{C} = \begin{pmatrix} A_1 + A_2 + B_1 + B_2 & -B_1 & -B_2 & 0 \\ -B_1 & A_2 + B_1 + B_2 & 0 & -B_2 \\ -B_2 & 0 & A_2 + B_2 & 0 \\ 0 & -B_2 & 0 & A_2 + B_2 \end{pmatrix} \mathbf{C} + \tilde{\mathbf{Q}}, \tag{2.21}$$

where $\mathbf{C} = (\mathbf{c}, \mathbf{c}_{\text{im}}, \mathbf{c}_{\text{ad}}, \mathbf{c}_{\text{im,ad}})^T$ and the right-hand side is given as $\tilde{\mathbf{Q}} = (\mathbf{Q}, \mathbf{Q}_{\text{im}}, \mathbf{Q}_{\text{ad}}, \mathbf{Q}_{\text{im,ad}})^T$.

For such an equation, we apply the decomposition of the matrices:

$$\begin{gathered} \partial_t \mathbf{C} = \tilde{A}\mathbf{C} + \tilde{\mathbf{Q}}, \\ \partial_t \mathbf{C} = \tilde{A}_1\mathbf{C} + \tilde{A}_2\mathbf{C} + \tilde{\mathbf{Q}}, \end{gathered} \tag{2.22}$$

where

$$\tilde{A}_1 = \begin{pmatrix} A_1 + A_2 & 0 & 0 & 0 \\ 0 & A_2 & 0 & 0 \\ 0 & 0 & A_2 & 0 \\ 0 & 0 & 0 & A_2 \end{pmatrix}, \qquad \tilde{A}_2 = \begin{pmatrix} B_1 + B_2 & -B_1 & -B_2 & 0 \\ -B_1 & B_1 + B_2 & 0 & -B_2 \\ -B_2 & 0 & B_2 & 0 \\ 0 & -B_2 & 0 & B_2 \end{pmatrix}. \tag{2.23}$$

The equation system is numerically solved by an iterative scheme.

Algorithm 2.4. We divide our time interval $[0, T]$ into subintervals $[t^n, t^{n+1}]$, where $n = 0, 1, \ldots N$, $t^0 = 0$, and $t^N = T$.

We start with $n = 0$.

(1) The initial conditions are given with $\mathbf{C}_0(t^{n+1}) = \mathbf{C}(t^n)$. We start with $k = 0$.

(2) Compute the fix-point iteration scheme given as

$$\partial_t \mathbf{C}^k = \tilde{A}_1\mathbf{C}^k + \tilde{A}_2\mathbf{C}^{k-1} + \tilde{\mathbf{Q}}, \tag{2.24}$$

where k is the iteration index, see [10]. For the time integration, we apply Runge-Kutta methods as ODE solvers, see [11, 12].

(3) The stop criterion for the time interval $[t^n, t^{n+1}]$ is given as

$$\left\| \mathbf{C}^k\left(t^{n+1}\right) - \mathbf{C}^{k-1}\left(t^{n+1}\right) \right\| \leq \text{err}, \tag{2.25}$$

where $\| \cdot \|$ is the maximum norm over all components of the solution vector. err is a given error bound, for example, err $= 10^{-4}$.

If (2.25) is fulfilled, we have the result

$$\mathbf{C}\left(t^{n+1}\right) = \mathbf{C}^k\left(t^{n+1}\right). \tag{2.26}$$

If $n = N$ then we stop and are done.

If (2.25) is not fulfilled, we do $k = k + 1$ and go to (2).

The error analysis of the schemes are given in the following Theorem.

Theorem 2.5. *Let $A, B \in \mathcal{L}(\mathbf{X})$ be given linear bounded operators in a Banach space $\mathcal{L}(\mathbf{X})$. We consider the abstract Cauchy problem:*

$$\begin{gathered} \partial_t \mathbf{C}(t) = \widetilde{A}\mathbf{C}(t) + \widetilde{B}\mathbf{C}(t), \quad t_n \leq t \leq t_{n+1}, \\ \mathbf{C}(t_n) = \mathbf{C}_n, \quad \textit{for } n = 1, \ldots, N, \end{gathered} \tag{2.27}$$

where $t_1 = 0$ and the final time is $t_N = T \in \mathbb{R}^+$. Then problem (2.27) *has a unique solution. For a finite steps with time size $\tau_n = t^{n+1} - t^n$, the iteration* (2.24) *for $k = 1, 2, \ldots, q$ is consistent with an order of consistency $\mathcal{O}(\tau_n^q)$.*

Proof. The outline of the proof is given in [2]. □

In the next section we describe the computation of the integral formulation with exp-functions.

3. Exponentials to Compute Iterative Schemes

The theoretical ideas can be discussed in the following formulation:

$$\begin{gathered} D_A(B,t)^{[0]} = \exp(tB), \\ D_A(B,t)^{[k]} = k\int_0^t \exp((t-s)B)\, A D_A(B,s)^{[k-1]} ds, \end{gathered} \tag{3.1}$$

and the matrix formulation of our two-step scheme is given as

$$\tilde{A} = \begin{pmatrix} A & 0 & \cdots & \cdots \\ A & B & 0 & \cdots \\ 0 & B & A & \cdots \\ \vdots & \ddots & \ddots & \ddots \\ 0 & \cdots & B & A \end{pmatrix}, \tag{3.2}$$

the computation of the exp-Matrix can be expressed as:

$$\exp\left(\tilde{A}t\right) := \begin{pmatrix} \exp(At) & 0 & \cdots & \cdots \\ D_A(B) & \exp(Bt) & 0 & \cdots \\ \dfrac{D_B(A)^{[2]}}{2!} & \dfrac{D_B(A)^{[1]}}{1!} & \exp(At) & \cdots \\ \vdots & \ddots & \ddots & \ddots \\ \dfrac{D_B(A)^{[n]}}{n!} & \cdots & \dfrac{D_B(A)^{[1]}}{1!} & \exp(At) \end{pmatrix}. \tag{3.3}$$

Here, we have to compute the right-hand side as time dependent term, means we evaluate $\exp(At)$ and $\exp(Bt)$ as a Taylor expansion. Then the integral formulation can be done with polynomials and we derive the expansion of such integral operators with the ϕ-function.

The ϕ-functions are defined as the integration of the exp-functions. We can analytically derive ϕ-functions with respect to the integral-formulation of a matrix exp-function.

In the following we reduce to a approximation of the fixed right-hand side (means we assume $D_A(B,s)^{[k-1]} \approx D_A(B,0)^{[k-01]}$).

Later we also follow with more extended schemes.

Computing ϕ-Functions:

$$\begin{aligned} \phi_0(x) &= e^x, \\ \phi_{k+1}(x) &= \frac{\phi_k(x) - 1/k!}{x}, \quad k \geq 0. \end{aligned} \tag{3.4}$$

So the matrix formulation of our scheme is given as

$$Y(t) = \exp\left(\tilde{A}t\right)Y(0), \tag{3.5}$$

where $\widetilde{A}$ is given as,

$$\widetilde{A} = \begin{pmatrix} A & 0 & \cdots & \cdots \\ A & B & 0 & \cdots \\ 0 & A & B & \cdots \\ \vdots & \ddots & \ddots & \ddots \\ 0 & \cdots & B & A \end{pmatrix}, \tag{3.6}$$

the computation of the exp-Matrix can be expressed in a first-order scheme and with the assumption of commutations is given as:

$$\exp\left(\widetilde{A}t\right) := \begin{pmatrix} \exp(At) & 0 & \cdots & \cdots & \cdots \\ \phi_1(Bt)A\ t & \exp(Bt) & 0 & \cdots & \cdots \\ \phi_2(At)B(B+A)t^2 & \phi_1(At)Bt & \exp(At) & \cdots & \cdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \phi_i(At)B(A+B)^{i-1}\ t^i & \cdots & \phi_1(At)Bt & & \exp(At) \end{pmatrix}, \tag{3.7}$$

where we assume $u_i(s) = u(t^n)$, mean we approximate the right-hand side term.

For higher-orders we should also include the full derivations of $c_1, c_2, \ldots$.

3.1. Derivation of the Formula

Consider the equation

$$\dot{x} = Ax + a, \quad x(0) = 0, \tag{3.8}$$

the solution of this equation in terms of the ϕ_k, $k = 0, 1$, is given as

$$x = t\phi_1(tA)a. \tag{3.9}$$

Similarly, the equation

$$\dot{x} = Ax + bt, \quad x(0) = 0, \tag{3.10}$$

has a solution of the form

$$x = t^2\phi_2(tA)b. \tag{3.11}$$

In general, the solution of the equation

$$\dot{x} = Ax + a + bt + c\frac{t^2}{2!} + \cdots, \quad x(0) = 0, \tag{3.12}$$

admits the form

$$x = t\phi_1(tA)a + t^2\phi_2(tA)b + t^3\phi_3(tA)c + \cdots. \tag{3.13}$$

In this section, we use the formula (3.21) for iterative schemes given as

$$\begin{aligned}\dot{u}_1 &= Au_1,\\ \dot{u}_2 &= Au_1 + Bu_2,\\ \dot{u}_3 &= Au_3 + Bu_2,\\ \dot{u}_4 &= \cdots.\end{aligned} \tag{3.14}$$

The solution of the first iteration given by

$$u_1 = e^{At}u_0. \tag{3.15}$$

Inserting this into (3.26) and expanding e^{At} up to the order 1, we have 2nd order approximation of the exact solution which has a form

$$u_2 = e^{Bt}u_0 + \phi_1(tA)Atu_0. \tag{3.16}$$

Similarly, inserting (3.16) into (3.27) and expanding $\phi_1(tA)$, we have a third order approximation of the exact solution

$$u_3 = e^{Bt}u_0 + \phi_1(tB)Btu_0 + \phi_2(tB)B(B+A)t^2u_0. \tag{3.17}$$

In general for $i = 0, 2, 4, \ldots$, we have the $p = i + 1$-th order approximation of the exact solution in terms of the ϕ_i function as follows:

$$u_i = e^{At}u_0 + \phi_1(tA)Atu_0 + \phi_2(tA)A(B+A)t^2u_0 + \cdots + \phi_i(tA)A(A+B)^{i-1}u_0. \tag{3.18}$$

For $i = 1, 3, 5, \ldots$, we have

$$u_i = e^{Bt}u_0 + \phi_1(tB)B\ tu_0 + \phi_2(tB)B(B+A)t^2u_0 + \cdots + \phi_i(tB)B(B+A)^{i-1}t^iu_0. \tag{3.19}$$

3.2. Algorithms

In the following, we discuss the one- and two-side algorithms.

3.3. Two-Side Scheme

For the implementation of the integral formulation, we have to deal with parallel ideas, which means we select independent parts of the formulation.

Step 1. Determine the order of the method by fixed iteration number.

Step 2. Consider the time interval $[t_0, T]$, divide it into N subintervals so that time step is $h = (T - t_0)/N$.

Step 3. On each subinterval, $[t_n, t_n + h]$, $n = 0, 1, \ldots, N$, use the algorithm by considering initial conditions for each step as $u(t_0) = u_0$, $u_i(t_n) = u_{i-1}(t_n) = u(t_n)$,

$$\begin{aligned}
u_2(t_n + h) &= (\phi_0(Bt) + \phi_1(Bt)At)u(t_n) \\
u_3(t_n + h) &= \left(\phi_0(At) + \phi_1(At)Bt + \phi_2(At)B(B + A)\ t^2\right)u(t_n) \\
&\ \ \vdots \\
u_{2i}(t_n + h) &= \left(\phi_0(Bt) + \phi_1(Bt)At + \cdots + \phi_{2i-1}(Bt)A(A + B)^{2i-1}\ t^{2i}\right)u_{2i-1}(t_n) \\
u_{2i+1}(t_n + h) &= \left(\phi_0(At) + \phi_1(At)Bt + \cdots + \phi_{2i}(At)B(B + A)^{2i}t^{2i+1}\right)u_{2i}(t_n).
\end{aligned} \tag{3.20}$$

Step 4. $u_i(t_n + h) \rightarrow u(t_n + h)$.

Step 5. Repeat this procedure for next interval until the desired time T is reached.

3.4. One-Side Scheme (Alternative Notation with Commutators)

For the one-side scheme, we taken into account of the following commutator relation.

Definition 3.1. The following relation is given:

$$\exp(-tA)B\exp(tA) = \left[B, \exp(tA)\right], \tag{3.21}$$

where $[\cdot, \cdot]$ is the matrix commutator.

The integration is given as

$$\int_0^t \exp(-sA)B\exp(sA)ds = \left[B, \phi_1(tA)\right], \tag{3.22}$$

the representation is given in [13].

Further, we have the recursive integration

$$\left[B, \phi_i(tA)t^i\right] = \int_0^t \left[B, \phi_{i-1}(sA)\right] ds, \tag{3.23}$$

where ϕ_i given defined as

$$\begin{aligned} \phi_0(At) &= \exp(At), \\ \phi_i(At) &= \int_0^t \phi_{i-1}(As) ds, \\ \phi_i(At) &= \frac{\phi_{i-1}(At) - I\left(t^{i-1}/(i-1)!\right)}{A}. \end{aligned} \tag{3.24}$$

The iterative scheme with the equations

$$\dot{u}_1 = Au_1, \tag{3.25}$$

$$\dot{u}_2 = Au_2 + Bu_1, \tag{3.26}$$

$$\dot{u}_3 = Au_3 + Bu_2, \tag{3.27}$$

$$\dot{u}_4 = \cdots, \tag{3.28}$$

is solved as

$$\begin{aligned} c_1(t) &= \exp(At)c(t^{\mathrm{n}}), \\ c_2(t) &= c_1(t) + c_1(t) \int_0^t \left[B, \exp(sA)\right] ds, \\ c_2(t) &= c_1(t) + c_1(t)\left[B, \phi_1(tA)\right], \\ c_3(t) &= c_2(t) + c_1(t) \int_0^t \left[B, \exp(sA), B, \phi_1(sA)\right] ds, \\ c_3(t) &= c_2(t) + c_1(t)\left(\left[B, \exp(tA), B, \phi_2(tA)\right] + \left[B, A\exp(tA), B, \phi_3(tA)\right]\right) + O\left(t^3\right), \\ &\cdots \end{aligned} \tag{3.29}$$

The recursion is given as

$$c_i(t) = c_{i-1}(t) + c_1(t) \int_0^t \left[B, \exp(sA)\right] \int_0^{s_1} \left[B, \exp(s_1 A)\right] \int_0^{s_2} \cdots \int_0^{s_{i-2}} \left[B, \exp(s_{i-1}A)\right] ds_{i-1} \cdots ds_1 dt. \tag{3.30}$$

Remark 3.2. For the novel notation, we have embedded the commutator to the computational scheme. For such a scheme we could save to compute additional the commutators.

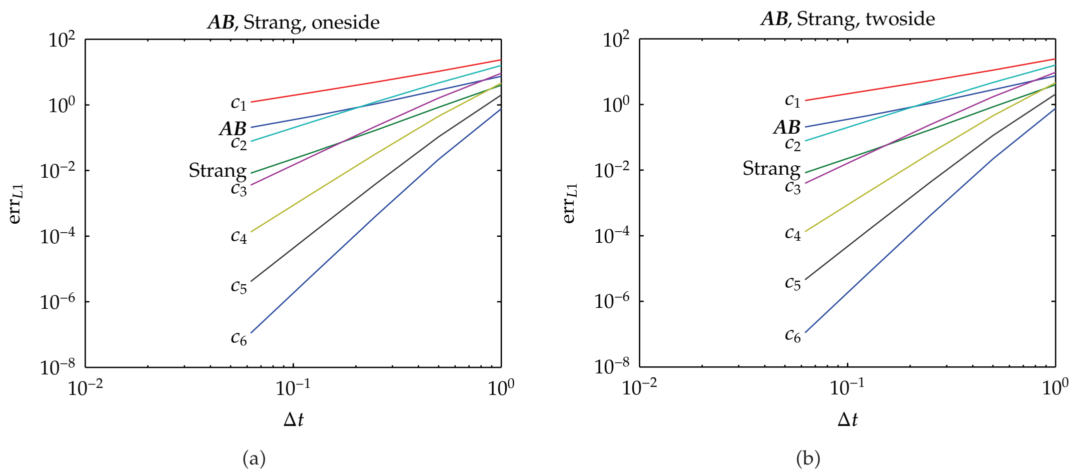


Figure 1: Numerical errors of the standard splitting scheme (a) and the iterative schemes (b) with $1, \ldots, 6$ iterative steps.

4. Numerical Examples

In the following, we deal with numerical example to verify the theoretical results.

4.1. First Example: Matrix Problem

For another example, consider the matrix equation

$$u'(t) = \begin{bmatrix} 1 & 2 \\ 3 & 0 \end{bmatrix} u, \qquad u(0) = u_0 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \tag{4.1}$$

the exact solution is

$$u(t) = \frac{2\left(e^{3t} - e^{-2t}\right)}{5}. \tag{4.2}$$

We split the matrix as

$$\begin{bmatrix} 1 & 2 \\ 3 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 2 & 0 \end{bmatrix}. \tag{4.3}$$

The Figure 1 presents the numerical errors between the exact and the numerical solution.

The Figure 2 presents the CPU time of the standard and the iterative splitting schemes.

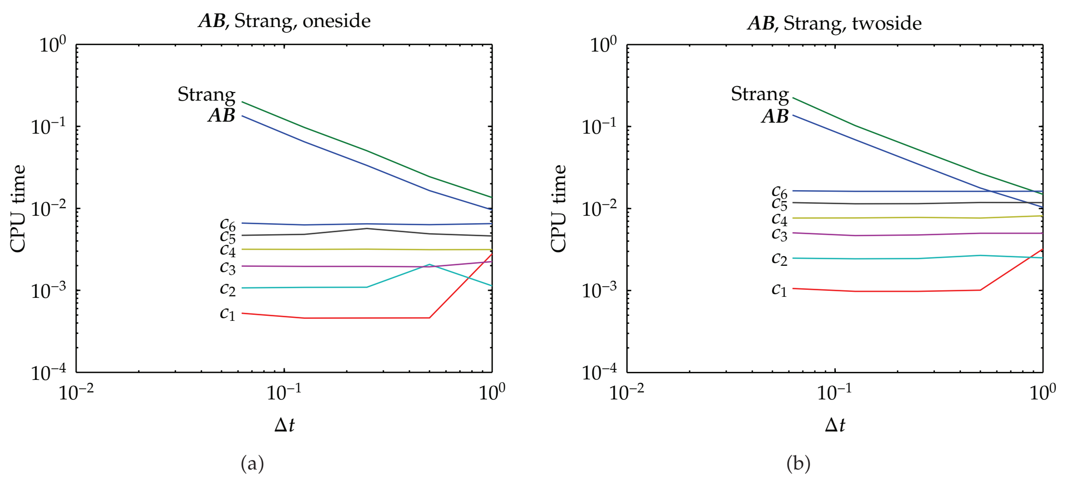


Figure 2: CPU time of the standard splitting scheme (a) and the iterative schemes (b) with $1, \ldots, 6$ iterative steps.

4.2. Second Experiment: 10×10 *Matrix*

We deal in the first with an ODE and separate the complex operator in two simpler operators.

We deal with the 10×10 ODE system:

$$\partial_t u_1 = -\lambda_{1,1} u_1 + \lambda_{2,1} u_2 + \cdots + \lambda_{10,1} u_{10}, \tag{4.4}$$

$$\partial_t u_2 = \lambda_{1,2} u_1 - \lambda_{2,2}(t) u_2 + \cdots + \lambda_{10,2} u_{10}, \tag{4.5}$$

$$\vdots \tag{4.6}$$

$$\partial_t u_{10} = \lambda_{1,10} u_1 + \lambda_{2,10}(t) u_2 + \cdots - \lambda_{10,10} u_{10}, \tag{4.7}$$

$$u_1(0) = u_{1,0}, \ldots,\ u_{10}(0) = u_{10,0} \quad \text{(initial conditions)}, \tag{4.8}$$

where $\lambda_1(t) \in \mathbb{R}^+$ and $\lambda_2(t) \in \mathbb{R}^+$ are the decay factors and $u_{1,0}, \ldots, u_{10,0} \in \mathbb{R}^+$. We have the time interval $t \in [0, T]$.

We rewrite (4.4) in operator notation, we concentrate us to the following equations:

$$\partial_t u = A(t)u + B(t)u, \tag{4.9}$$

where $u_1(0) = u_{10} = 1.0$, $u_2(0) = u_{20} = 1.0$ are the initial conditions, where we have $\lambda_1(t) = t$ and $\lambda_2(t) = t^2$.

The operators are splitted in the following way, while operator A, see (4.10), covers the upper diagonal entries and operator B, see (4.11), covers the lower diagonal entries of the ODE system (4.4)–(4.7). Such a decomposition allows to reduce the computation of the matrix exponential to a upper or lower matrix and speedup the solver process

$$A = \begin{pmatrix} -0.01 & 0.01 & 0 & \cdots & & & & & & \\ 0.01 & -0.01 & 0 & \cdots & & & & & & \\ 0.01 & 0.01 & -0.02 & 0 & \cdots & & & & & \\ 0.01 & 0.01 & 0.01 & -0.03 & 0 & \cdots & & & & \\ \vdots & & & & & & & & & \\ 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & -0.08 & 0 \\ 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0 & -0.08 \end{pmatrix}, \tag{4.10}$$

$$B = \begin{pmatrix} -0.08 & 0 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \\ 0 & -0.08 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \\ \vdots & & & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.02 & 0.01 & 0.01 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.01 & 0.01 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.01 & -0.01 \end{pmatrix}. \tag{4.11}$$

The Figure 3 present the numerical errors between the exact and the numerical solution. Here we have the best approximation to the exact solution with a sixth-order iterative scheme (6 iterative steps), but no additionally more computational time for finer time step.

The Figure 4 present the CPU time of the standard and the iterative splitting schemes. Compared to standard splitting schemes, the iterative schemes are cheaper to compute and higher in accuracy. We can also choose smaller time steps to obtain more accurate results and use nearly the same computational resource.

Remark 4.1. The computational results show the benefit of the iterative schemes. We save computational time and achieve higher-order accuracy. The one-side and two-side schemes have the same results.

4.3. Third Example: Commutator Problem

We assume to have a large norm of the commutator $[A, B]$ and deal with the matrix equation

$$u'(t) = \begin{bmatrix} 10 & 1 \\ 1 & 10 \end{bmatrix} u, \qquad u(0) = u_0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \tag{4.12}$$

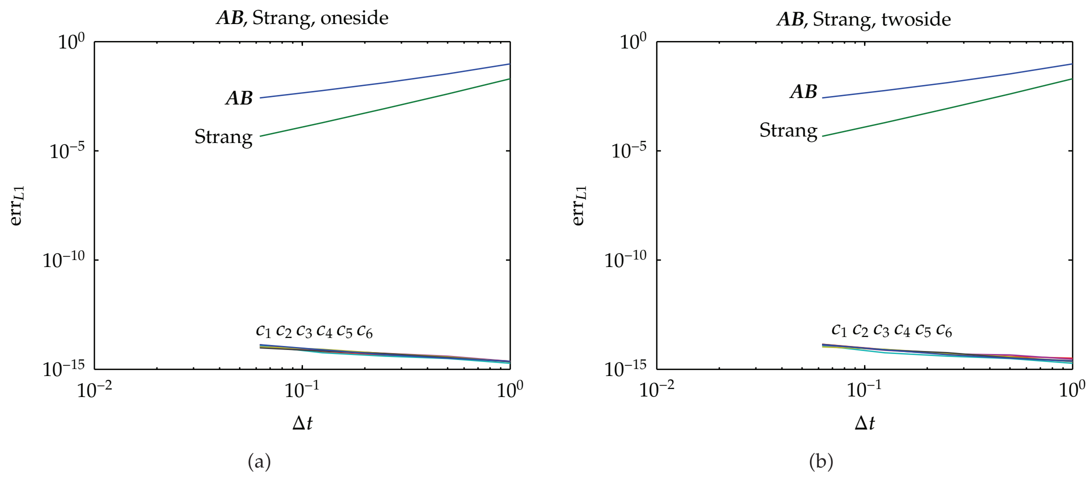


Figure 3: Numerical errors of the standard splitting scheme (a) and the iterative schemes (b) with $1, \ldots, 6$ iterative steps.

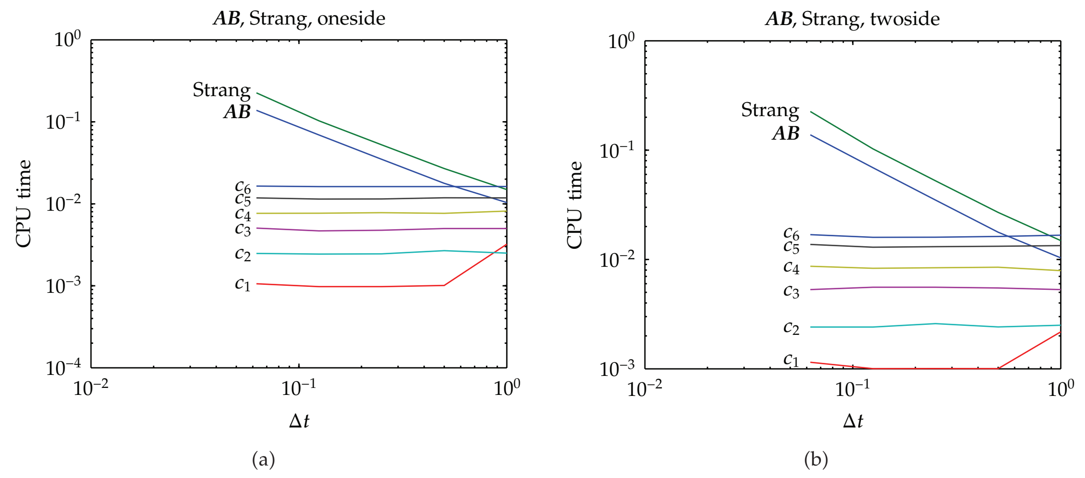


Figure 4: CPU time of the standard splitting scheme (a) and the iterative schemes (b) with $1, \ldots, 6$ iterative steps.

In the following, we discuss different splitting ideas.

Version 1

We split the matrix as

$$\begin{bmatrix} 10 & 1 \\ 1 & 10 \end{bmatrix} = \begin{bmatrix} 9 & 0 \\ 1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 1 \\ 0 & 9 \end{bmatrix}, \tag{4.13}$$

while $\|[A, B]\| \geq \max\{\|A\|, \|B\|\}$.

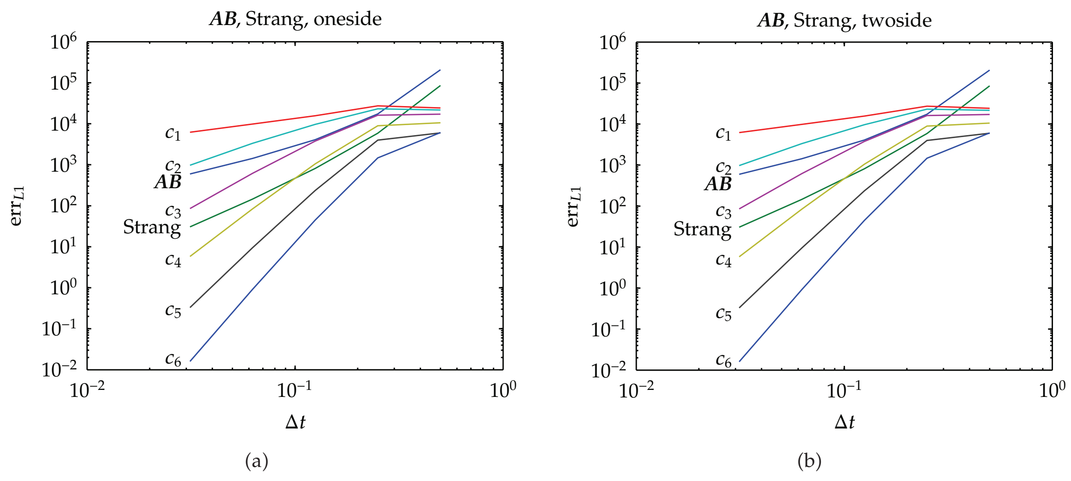


Figure 5: Numerical errors of the standard splitting scheme (a) and the iterative schemes (b) with $1, \ldots, 6$ iterative steps.

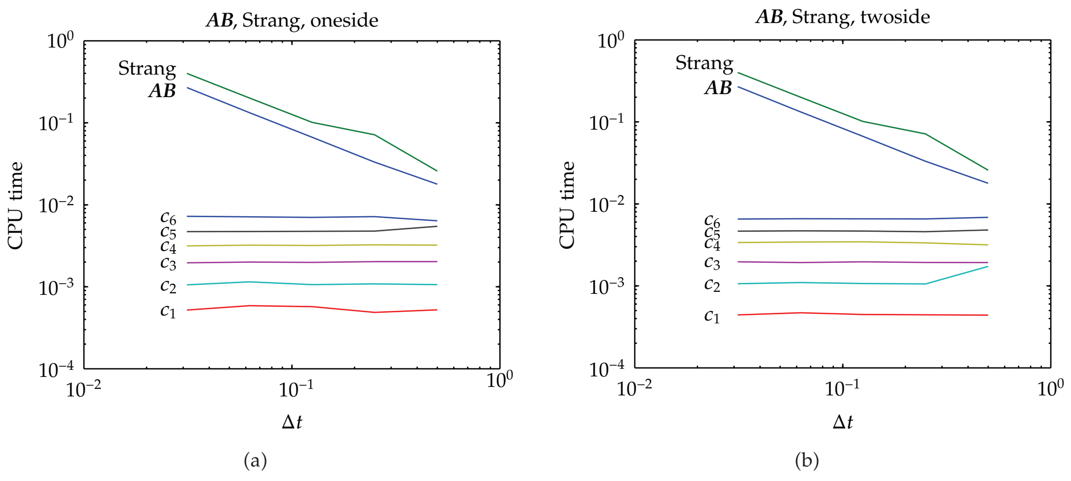


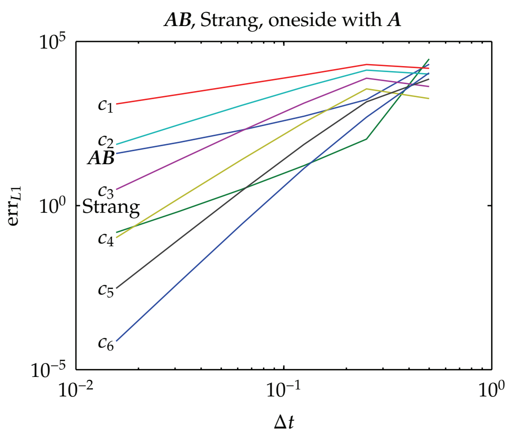
Figure 6: CPU time of the standard splitting scheme (a) and the iterative schemes (b) with $1, \ldots, 6$ iterative steps.

Here we have to deal with highly noncommutative operators and the computational speedup is given in the iterative schemes, while the commutator is not needed to obtain more accurate results, while the standard splitting schemes deal with such commutators for the error reduction.

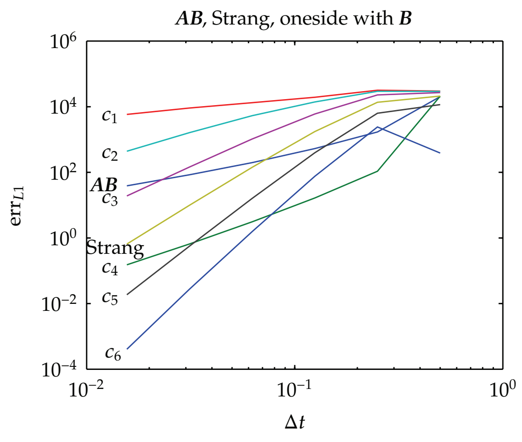
The Figure 5 present the numerical errors between the exact and the numerical solution.

The Figure 6 present the CPU time of the standard and the iterative splitting schemes.

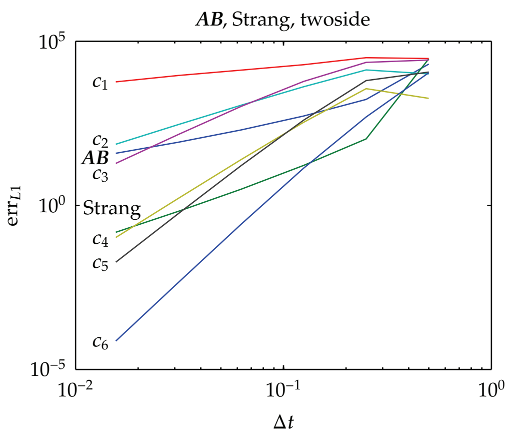
Further for the one-side,we obtain more improved results for the following splitting.



(a)



(b)



(c)

Figure 7: Numerical errors of the standard splitting scheme (a) and the iterative schemes based on one-side to operator A or B (b) and two-side scheme (c).

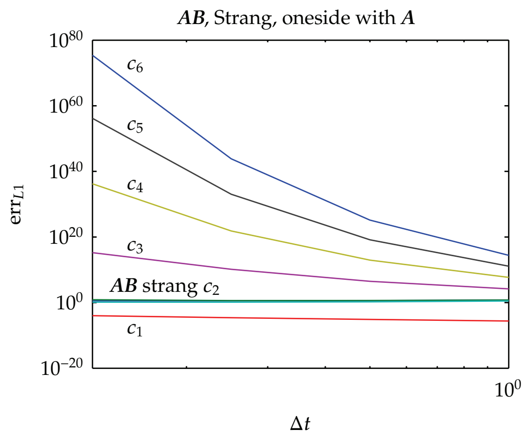
Version 2

In this version, we have

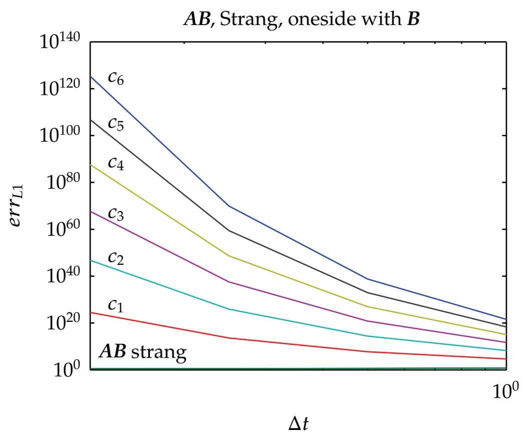
$$\begin{bmatrix} 10 & 1 \\ 1 & 10 \end{bmatrix} = \begin{bmatrix} 9 & 0 \\ 1 & 9 \end{bmatrix} + \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}. \tag{4.14}$$

The Figure 7 presents the numerical errors between the exact and the numerical solution.

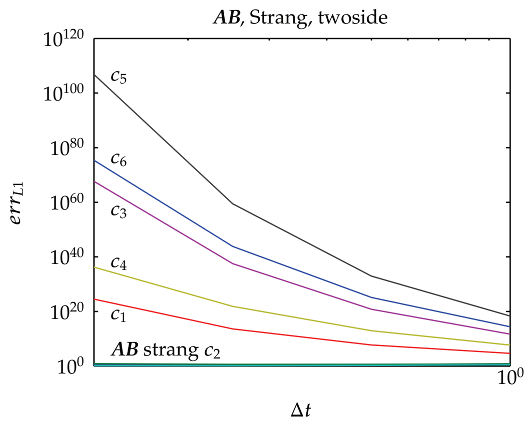
A more delicate problem is given for the stiff matrices.



(a)



(b)



(c)

Figure 8: Numerical errors of the one-side splitting scheme with A (a), the one-side splitting scheme with B (b) and the iterative schemes with $1, \ldots, 6$ iterative steps (c).

Version 3

In this version, we have

$$\begin{bmatrix} 10^4 & 1 \\ 1 & 10^4 \end{bmatrix} = \begin{bmatrix} 10^4 - 1 & 0 \\ 1 & 10^4 - 1 \end{bmatrix} + \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}. \tag{4.15}$$

The Figure 8 present the numerical errors between the exact and the numerical solution.

The Figure 9 present the CPU time of the standard and the iterative splitting schemes.

Remark 4.2. The iterative schemes with fast computations of the exponential matrices have a speedup. The constant CPU time of the iterative schemes shows that it benefit instead of the expensive standard schemes. Also for stiff problems with multi iterative steps, we reach the same results of the standard A-B or Strang-Splitting schemes, while we are independent of the commutator estimation.

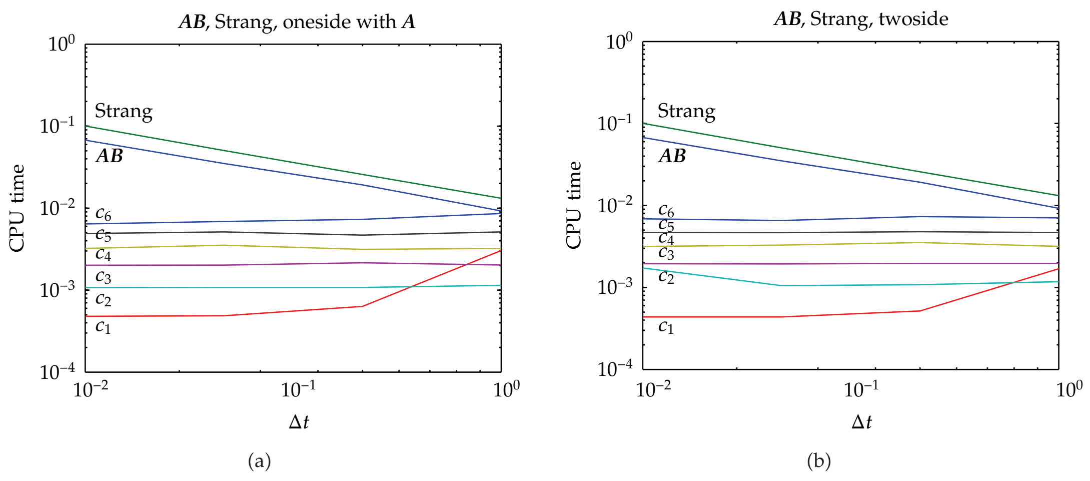


Figure 9: CPU time of the one-side splitting scheme with A (a) and the iterative schemes with $1, \ldots, 6$ iterative steps (b).

4.4. Two-Phase Example

The next example is a simplified real-life problem for a multiphase transport-reaction equation. We deal with mobile and immobile pores in the porous media, such simulations are given for waste scenarios.

We concentrate on the computational benefits of a fast computation of the iterative scheme, given with matrix exponentials.

The equation is given as:

$$
\begin{gathered}
\partial_t c_1 + \nabla \cdot \mathbf{F} c_1 = g(-c_1 + c_{1,\mathrm{im}}) - \lambda_1 c_1, \quad \text{in } \Omega \times [0,t], \\
\partial_t c_2 + \nabla \cdot \mathbf{F} c_2 = g(-c_2 + c_{2,\mathrm{im}}) + \lambda_1 c_1 - \lambda_2 c_2, \quad \text{in } \Omega \times [0,t], \\
\mathbf{F} = \mathbf{v} - D\nabla, \\
\partial_t c_{1,\mathrm{im}} = g(c_1 - c_{1,\mathrm{im}}) - \lambda_1 c_{1,\mathrm{im}}, \quad \text{in } \Omega \times [0,t], \\
\partial_t c_{2,\mathrm{im}} = g(c_2 - c_{2,\mathrm{im}}) + \lambda_1 c_{1,\mathrm{im}} - \lambda_2 c_{2,\mathrm{im}}, \quad \text{in } \Omega \times [0,t], \\
c_1(\mathbf{x},t) = c_{1,0}(\mathbf{x}), \quad c_2(\mathbf{x},t) = c_{2,0}(\mathbf{x}), \quad \text{on } \Omega, \\
c_1(\mathbf{x},t) = c_{1,1}(\mathbf{x},t), \quad c_2(\mathbf{x},t) = c_{2,1}(\mathbf{x},t), \quad \text{on } \partial\Omega \times [0,t], \\
c_{1,\mathrm{im}}(\mathbf{x},t) = 0, \quad c_{2,\mathrm{im}}(\mathbf{x},t) = 0, \quad \text{on } \Omega, \\
c_{1,\mathrm{im}}(\mathbf{x},t) = 0, \quad c_{2,\mathrm{im}}(\mathbf{x},t) = 0, \quad \text{on } \partial\Omega \times [0,t].
\end{gathered}
\tag{4.16}
$$

In the following, we deal with the semidiscretized equation given with the matrices:

$$\partial_t \mathbf{C} = \begin{pmatrix} A - \Lambda_1 - G & 0 & G & 0 \\ \Lambda_1 & A - \Lambda_2 - G & 0 & G \\ G & 0 & -\Lambda_1 - G & 0 \\ 0 & G & \Lambda_1 & -\Lambda_2 - G \end{pmatrix} \mathbf{C}, \tag{4.17}$$

where $\mathbf{C} = (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_{1\text{im}}, \mathbf{c}_{2\text{im}})^T$, while $\mathbf{c}_1 = (\mathbf{c}_{1,1}, \ldots, \mathbf{c}_{1,I})$ is the solution of the first species in the mobile phase in each spatial discretization point $(i = 1, \ldots, I)$, the same is also for the other solution vectors.

We have the following two operators for the splitting method

$$\begin{aligned} A = \frac{D}{\Delta x^2} \cdot &\begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{pmatrix} \\ + \frac{v}{\Delta x} \cdot &\begin{pmatrix} 1 & & & & \\ -1 & 1 & & & \\ & \ddots & \ddots & & \\ & & -1 & 1 & \\ & & & -1 & 1 \end{pmatrix} \in \mathbb{R}^{I \times I}, \end{aligned} \tag{4.18}$$

where I is the number of spatial points.

$$\Lambda_1 = \begin{pmatrix} \lambda_1 & 0 & & & \\ 0 & \lambda_1 & 0 & & \\ & \ddots & \ddots & \ddots & \\ & & 0 & \lambda_1 & 0 \\ & & & 0 & \lambda_1 \end{pmatrix} \in \mathbb{R}^{I \times I},$$

$$\Lambda_2 = \begin{pmatrix} \lambda_2 & 0 & & & \\ 0 & \lambda_2 & 0 & & \\ & \ddots & \ddots & \ddots & \\ & & 0 & \lambda_2 & 0 \\ & & & 0 & \lambda_2 \end{pmatrix} \in \mathbb{R}^{I\times I},$$

$$G = \begin{pmatrix} g & 0 & & & \\ 0 & g & 0 & & \\ & \ddots & \ddots & \ddots & \\ & & 0 & g & 0 \\ & & & 0 & g \end{pmatrix} \in \mathbb{R}^{I\times I}. \tag{4.19}$$

We decouple into the following matrices:

$$A_1 = \begin{pmatrix} A & 0 & 0 & 0 \\ 0 & A & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \in \mathbb{R}^{4I\times 4I},$$

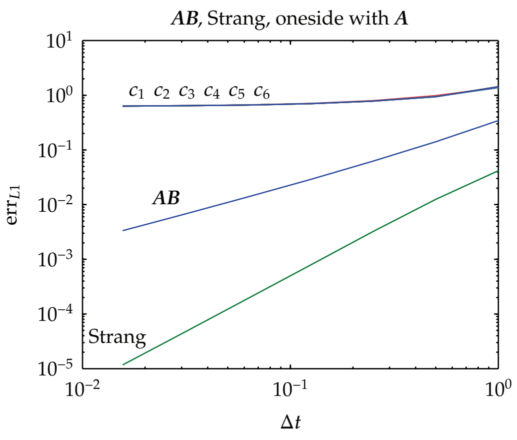
$$\widetilde{A}_2 = \begin{pmatrix} -\Lambda_1 & 0 & 0 & 0 \\ \Lambda_1 & -\Lambda_2 & 0 & 0 \\ 0 & 0 & -\Lambda_1 & 0 \\ 0 & 0 & \Lambda_1 & -\Lambda_2 \end{pmatrix} \in \mathbb{R}^{4I\times 4I}, \tag{4.20}$$

$$\widetilde{A}_3 = \begin{pmatrix} -G & 0 & G & 0 \\ 0 & -G & 0 & G \\ G & 0 & -G & 0 \\ 0 & G & 0 & -G \end{pmatrix} \in \mathbb{R}^{4I\times 4I}.$$

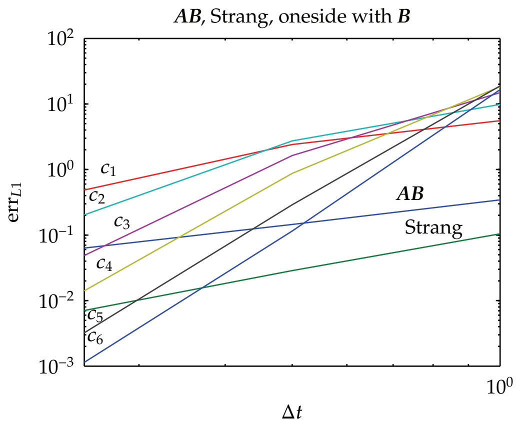
For the operator A_1 and $A_2 = \widetilde{A}_2 + \widetilde{A}_3$, we apply the iterative splitting method.

Based on the decomposition, operator A_1 is only tridiagonal and operator A_2 is block diagonal. Such matrix structure reduce the computation of the exponential operators.

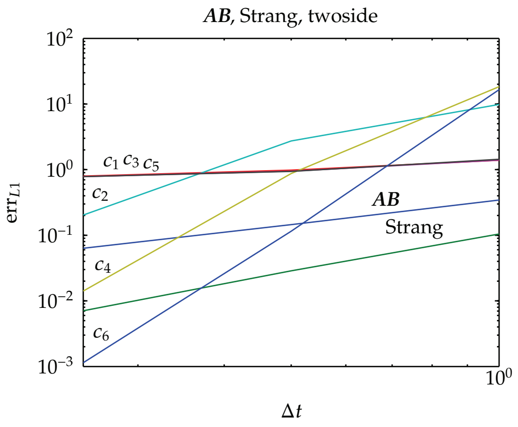
The Figure 10 present the numerical errors between the exact and the numerical solution. Here we obtain optimal results for one-side iterative schemes on operator B, means we iterate with respect to B and use A as right-hand side.



(a)



(b)



(c)

Figure 10: Numerical errors of the one-side splitting scheme with A (a), the one-side splitting scheme with B (b) and the iterative schemes with $1, \ldots, 6$ iterative steps (c).

Remark 4.3. For all iterative schemes, we can reach faster results as for the The iterative schemes with fast computations of the exponential matrices standard schemes. With 4-5 iterative steps we obtain more accurate results as we did for the expensive standard schemes. With one-side iterative schemes we reach the best convergence results.

5. Conclusions and Discussions

In this work, we have presented a very economical and practical method by successive approximations. Here the idea to decouple the expensive computation of matrix exponential via iterative splitting schemes has the benefit of less computational time. While the error analysis present stable methods and higher-order schemes, the applications show the speedup with the iterative schemes. In, future, we concentrate on matrix dependent scheme, based on iterative splitting algorithms.

References

[1] J. Geiser, *Discretisation methods for systems of convective-diffusive dispersive-reactive equations and applications*, Ph.D. thesis, Universität Heidelberg, Heidelberg, Germany, 2004.

[2] J. Geiser, “Decomposition methods for partial differential equations: theory and applications in multiphysics problems,” in *Numerical Analysis and Scientific Computing Series*, Magoules and Lai, Eds., CRC Press, Boca Raton, Fla, USA, 2009.

[3] M. Hochbruck and A. Ostermann, “Explicit exponential Runge-Kutta methods for semilinear parabolic problems,” *SIAM Journal on Numerical Analysis*, vol. 43, no. 3, pp. 1069–1090, 2005.

[4] H. Rouch, “MOCVD research reactor simulation,” in *Proceedings of the COMSOL Users Conference*, Paris, France, 2006.

[5] L. Rudniak, “Numerical simulation of chemical vapour deposition process in electric field,” *Computers and Chemical Engineering*, vol. 22, no. 1, supplement, pp. S755–S758, 1998.

[6] R. Glowinski, “Numerical methods for fluids,” in *Handbook of Numerical Analysis*, P. G. Ciarlet and J. Lions, Eds., vol. 9, North-Holland, Amsterdam, The Netherlands, 2003.

[7] J. F. Kanney, C. T. Miller, and C. T. Kelley, “Convergence of iterative split-operator approaches for approximating nonlinear reactive problems,” *Advances in Water Resources*, vol. 26, no. 3, pp. 247–261, 2003.

[8] E. Hansen and A. Ostermann, “Exponential splitting for unbounded operators,” *Mathematics of Computation*, vol. 78, no. 267, pp. 1485–1496, 2009.

[9] T. Jahnke and C. Lubich, “Error bounds for exponential operator splittings,” *BIT Numerical Mathematics*, vol. 40, no. 4, pp. 735–744, 2000.

[10] I. Faragó and J. Geiser, “Iterative operator-splitting methods for linear problems,” *International Journal of Computational Science and Engineering*, vol. 3, no. 4, pp. 255–263, 2007.

[11] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving ordinary differential equations. I*, vol. 8 of *Springer Series in Computational Mathematics*, Springer, Berlin, 1987, Nonstiff problem.

[12] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations. II*, vol. 14 of *Springer Series in Computational Mathematics*, Springer, Berlin, Germany, 2nd edition, 1996.

[13] W. Magnus, “On the exponential solution of differential equations for a linear operator,” *Communications on Pure and Applied Mathematics*, vol. 7, pp. 649–673, 1954.