Numdiff-12: Halle, 14-18 September, 2009
Lecture: Iterative Splitting Methods: Solver for Differential Equations

Jürgen Geiser
Humboldt Universität zu Berlin, D- Berlin, Germany

September 24, 2009
Outline

- Introduction: Iterative splitting method

Outline

- Introduction: Iterative splitting method
- Analysis: Consistency and Stability
Outline

- Introduction: Iterative splitting method
- Analysis: Consistency and Stability
- Extension to nonlinear operators
Outline

- Introduction: Iterative splitting method
- Analysis: Consistency and Stability
- Extension to nonlinear operators
- Improved initialization as acceleration of iterative schemes
Outline

- Introduction: Iterative splitting method
- Analysis: Consistency and Stability
- Extension to nonlinear operators
- Improved initialization as acceleration of iterative schemes
- Numerical Experiments
Decomposition methods are used to decouple differential equations into simpler and faster solvable equation parts.
Introduction

- Decomposition methods are used to decouple differential equations into simpler and faster solvable equation parts.

- Iterative operator splitting methods are developed to gain higher order decomposition schemes by using an iterative algorithms.
Decomposition methods are used to decouple differential equations into simpler and faster solvable equation parts.

Iterative operator splitting methods are developed to gain higher order decomposition schemes by using an iterative algorithm.

Extension to time- and nonlinear problems are possible.
Introduction

- Decomposition methods are used to decouple differential equations into simpler and faster solvable equation parts.
- Iterative operator splitting methods are developed to gain higher order decomposition schemes by using an iterative algorithms.
- Extension to time- and nonlinear problems are possible.
- Problems in the initialization can be done by weighting factors.
Motivation of solving a PDE with Splitting methods

1.) Solving a PDE by Semi-discretisation in Space, we consider to solve unbounded operators, e.g. Eulerian Formulation (spatial grid):

\[ \partial_t u = -\partial_x u + \partial_{xx} u, \quad (x, y, t) \in \Omega \times [0, T] \]  

(1)

with boundary and initial conditions.

Spatial discretization, e.g. FEM, FV method leads to:

\[ \partial_t u_h = A_h u_h + B_h u_h, \]  

(2)

where \( A_h \) and \( B_h \) are operator with norms approx. \( 1/h \) and \( 1/h^2 \)
2.) Solving a PDE by Time-integrator, e.g. Exponential integrators, we skip the problem of unbounded operators to the exp-formulation (and can bound the operators):

\[ \partial_t u = -\partial_x u + \partial_{xx} u, \quad (x, y, t) \in \Omega \times [0, T] \] (3)

with boundary and initial conditions, is formulated as:

\[ A + B = -\partial_x + \partial_{xx}, \] (4)

and solution:

\[ u(x, t) = \exp((A + B)t)u(x, 0), \] (5)

Here the motivation of the exp. formulation is to decouple into simpler products, like in the iterative splitting scheme to solve parts of the full equation.
Problem Definition for ODE systems

- Cauchy Problem:

\[
\frac{\partial c(t)}{\partial t} = Ac(t) + Bc(t), \text{ with } t \in [0, T], \quad c(0) = c_0, \quad (6)
\]

- Initial function \(c_0\)
Problem Definition for ODE systems

• Cauchy Problem:

$$\frac{\partial c(t)}{\partial t} = Ac(t) + Bc(t), \text{ with } t \in [0, T], \ c(0) = c_0, \ (6)$$

• Initial function $c_0$

• $A$ and $B$ are assumed to be bounded linear operators in the Banach-space $X$ with $A, B : X \rightarrow X$
Operator Splitting Method

Well-known operators splitting schemes:
1. Sequential operator-splitting: A-B splitting

\[
\frac{\partial c^*(t)}{\partial t} = A c^*(t) \text{ with } t \in [t^n, t^{n+1}], \quad c^*(t^n) = c_{sp}^n
\]

\[
\frac{\partial c^{**}(t)}{\partial t} = B c^{**}(t) \text{ with } t \in [t^n, t^{n+1}], \quad c^{**}(t^n) = c^*(t^{n+1})
\]

for \( n = 0, 1, ..., N - 1 \) whereby \( c_{sp}^n = c_0 \) is given from (6). The approximated split solution at the point \( t = t^{n+1} \) is defined as \( c_{sp}^{n+1} = c^{**}(t^{n+1}) \).

\[
\frac{\partial c^*(t)}{\partial t} = A c^*(t) \text{ with } t \in [t^n, t^{n+1/2}], \quad c^*(t^n) = c_{sp}^n, \tag{8}
\]
\[
\frac{\partial c^{**}(t)}{\partial t} = B c^{**}(t) \text{ with } t \in [t^n, t^{n+1/2}], \quad c^{**}(t^n) = c^*(t^{n+1/2}),
\]
\[
\frac{\partial c^{***}(t)}{\partial t} = A c^*(t) \text{ with } t \in [t^{n+1/2}, t^{n+1}], \quad c^{***}(t^{n+1/2}) = c^{**}(t^{n+1}),
\]

where \( t^{n+1/2} = t^n + 0.5\tau_n \), and the approximated split solution at the point \( t = t^{n+1} \) is defined as \( c_{sp}^{n+1} = c^{***}(t^{n+1}) \).
Operator Splitting Method

Iterative operators splitting schemes as alternative schemes to obtain higher order schemes:

3. Iterative splitting with respect to one operator

\[
\frac{\partial c_i(t)}{\partial t} = Ac_i(t) + Bc_{i-1}(t),
\]

with \( c_i(t^n) = c^n \), \( i = 0, 1, \ldots, m \)

where \( c_0(t^n) = c^n \), \( c_{-1} = 0 \) and \( c^n \) 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 \( c^{n+1} = c_{m+1}(t^{n+1}) \).
4. Iterative splitting with respect to alternating operators

\[
\frac{\partial c_i(t)}{\partial t} = Ac_i(t) + Bc_{i-1}(t), \quad \text{with} \quad c_i(t^n) = c^n, \quad (10)
\]

\[
\frac{\partial c_{i+1}(t)}{\partial t} = Ac_i(t) + Bc_{i+1}(t), \quad \text{with} \quad c_{i+1}(t^n) = c^n, \quad (11)
\]

\[i = 0, 1, \ldots, m,\]

where \(c_0(t^n) = c^n\), \(c_{-1} = 0\) and \(c^n\) 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 \(c^{n+1} = c_{m+1}(t^{n+1})\).
Analysis of the iterative method for bounded and unbounded operators

- Higher order results can be obtained with bounded operators
Analysis of the iterative method for bounded and unbounded operators

- Higher order results can be obtained with bounded operators
- Order reduction problem with stiff operators, overcome with balance of time and spatial scale time-steps
Analysis of the iterative method for bounded and unbounded operators

- Higher order results can be obtained with bounded operators
- Order reduction problem with stiff operators, overcome with balance of time and spatial scale time-steps
- Extension to unbounded operators based on exponential formulation to shift unbounded operators to exponents (apply semigroup theory)
Practical Part:

\[
\frac{dc(t)}{dt} = c(t), \quad t \in (0, T],
\]
\[
c(0) = 1,
\]

where the exact solution is given \( u(t) = \exp(t) \).

We decompose as

\[
\frac{dc_i(t)}{dt} = 0.5c_i(t) + 0.5c_{i-1}, \quad t \in (0, T],
\]
\[
c_i(0) = 1,
\]
\[
\frac{dc_{i+1}(t)}{dt} = 0.5c_i(t) + 0.5c_{i+1}, \quad t \in (0, T],
\]
\[
c_{i+1}(0) = 1,
\]

where \( c_0 = 1 \) and \( i = 1, 3, \ldots, 2m - 1 \).
One can do the exact iterations as:

\[
c_1(t) = 2 \exp\left(\frac{t}{2}\right) - 1
\]

\[
c_2(t) = t \exp\left(\frac{t}{2}\right) + 1
\]

\[
c_3(t) = \left(2 + \frac{t^2}{4}\right) \exp\left(\frac{t}{2}\right) - 1
\]

with local errors: \(err_1 = \mathcal{O}(t^2)\), \(err_2 = \mathcal{O}(t^2)\), \(err_3 = \mathcal{O}(t^4)\), etc. Important are the initial estimate, otherwise the order is lowered.
Theoretical Part: Consistency to unbounded operators (exp-formulation):

**Theorem**

Let us consider the abstract Cauchy problem in a Banach space $\mathbf{X}$

$$\partial_t c(t) = Ac(t) + Bc(t), \quad 0 < t \leq T$$

$$c(0) = c_0$$

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 bounded.
Further, we assume following bounds of the operators with sufficient smooth initial conditions:

\[ \| B^\alpha \exp((A + B)\tau) \| \leq \kappa, \quad (22) \]

and

\[ \| \exp(A\tau)B^{1-\alpha} \| \leq \tilde{\kappa}, \quad (23) \]

where \( \alpha \in (0, 1) \) and \( \kappa, \tilde{\kappa} \in \mathbb{R}^+ \), same argumentation also to operator \( A \).

Then, we can bound our iterative operator splitting method as:

\[ \|(S_i - \exp((A + B)\tau))\| \leq C\tau^i, \quad (24) \]

where \( S_i \) is the approximated solution for the \( i \)-th iterative step and \( C \) is a constant that can be chosen uniformly on bounded time intervals.
Proof.

Let us consider the iterative splitting scheme on the sub-interval 
$[t^n, t^{n+1}]$.

For the first iterations we have:

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

and for the second iteration we have:

$$\partial_t c_2(t) = A c_1(t) + B c_2(t), \quad t \in (t^n, t^{n+1}],$$
In general, we have:

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

$$\partial_t c_i(t) = A c_i(t) + B c_{i-1}(t), \quad t \in (t^n, t^{n+1}], \quad (27)$$

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

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

$$\partial_t c_i(t) = A c_{i-1}(t) + B c_i(t), \quad t \in (t^n, t^{n+1}]. \quad (28)$$
We have the following solutions for the iterative scheme: the solutions for the first two equations are given by the variation of constants:

\[ c_1(t) = \exp(A(t - t^n))c(t^n), \quad t \in (t^n, t^{n+1}], \quad (29) \]

\[ c_2(t) = \exp(B(t - t^n))c(t^n) + \int_{t^n}^{t} \exp(B(t^{n+1} - s))Ac_1(s)\,ds, \quad t \in (t^n, t^{n+1}]. \quad (30) \]
For the recurrence relations with even and odd iterations, we obtain:

for odd iterations: \(i = 2m + 1\),
\[ c_i(t) = \exp(A(t - t^n))c(t^n) + \int_{t^n}^{t} \exp(sA)Bc_{i-1}(t^{n+1} - s) \, ds, \]
\[ t \in (t^n, t^{n+1}]. \] \hspace{1cm} (31)

For even iterations: \(i = 2m\),
\[ c_i(t) = \exp(B(t - t^n))c(t^n) + \int_{t^n}^{t} \exp(sB)Ac_{i-1}(t^{n+1} - s) \, ds, \]
\[ t \in (t^n, t^{n+1}]. \] \hspace{1cm} (32)
The consistency is given as:

For $e_1$ we have:

$$c_1(t^{n+1}) = \exp(A\tau)c(t^n), \quad (33)$$

$$c(t^{n+1}) = \exp((A + B)\tau)c(t^n) = \exp(A\tau)c(t^n) \quad (34)$$

$$+ \int_{t^n}^{t^{n+1}} \exp(As)B \exp((t^{n+1} - s)(A + B))c(t^n) \, ds.$$  

We obtain:

$$\|e_1\| = \|c - c_1\| \leq \|\exp((A + B)\tau)c(t^n) - c_1(t^{n+1})\| (35)$$

$$\leq C_1 \tau c(t^n).$$
For $e_2$ we have:

$$c_2(t^{n+1}) = \exp(B\tau)c(t^n)$$

$$+ \int_{t^n}^{t^{n+1}} \exp(Bs)A\exp((t^{n+1} - s)A)c(t^n) \, ds,$$  \hspace{1cm} (36)

$$c(t^{n+1}) = \exp(B\tau)c(t^n) + \int_{t^n}^{t^{n+1}} \exp(Bs)A\exp((t^{n+1} - s)A)c(t^n) \, ds$$

$$+ \int_{t^n}^{t^{n+1}} \exp(Bs)A$$

$$\int_{t^n}^{t^{n+1} - s} \exp(A\rho)B\exp((t^{n+1} - s - \rho)(A + B))c(t^n) \, d\rho \, ds.$$  \hspace{1cm} (37)
We obtain:

\[
\|e_2\| \leq \|\exp((A + B)\tau)c(t^n) - c_2\| \leq C_2\tau^2 c(t^n).
\]
For odd and even iterations, the recursive proof is given in the following:

for odd iterations:  \( i = 2m + 1 \)

for \( m = 0, 1, 2, \ldots \),

for \( e_i \) we have:

\[
\begin{align*}
    c_i(t^{n+1}) &= \exp(A)\tau c(t^n) \\
    &+ \int_{t^n}^{t^{n+1}} \exp(As)B \exp((t^{n+1} - s)B)c(t^n) \, ds \\
    &+ \int_{t^n}^{t^{n+1}} \exp(As_1)B \int_{t^n}^{t^{n+1} - s_1} \exp(s_2 B)A \exp((t^{n+1} - s_1 - s_2)A)c(t^n) \, ds_2 \, ds_1 \\
    &+ \ldots + \int_{t^n}^{t^{n+1}} \exp(As_1)B \ldots \\
    &\ldots \int_{t^n}^{t^{n+1} - \sum_{j=1}^{i-1} s_j} \exp(Bs_i)A \exp((t^{n+1} - \sum_{j=1}^{i-1} s_j)A)c(t^n) \, ds_i \ldots \, ds_1,
\end{align*}
\]
\[ c(t^{n+1}) = \exp(A \tau) c(t^n) \]

\[ + \int_{t^n}^{t^{n+1}} \exp(As)B \exp((t^{n+1} - s)B) c(t^n) \, ds \]

\[ + \int_{t^n}^{t^{n+1}} \exp(As_1)B \int_{t^n}^{t^{n+1} - s_1} \exp(s_2B)A \exp((t^{n+1} - s_1 - s_2)A) c(t^n) \, ds_2 \, ds_1 \]

\[ + \ldots + \int_{t^n}^{t^{n+1}} \exp(As_1)B \ldots \]

\[ \ldots \int_{t^n}^{t^{n+1} - \sum_{j=1}^{i} s_j} \exp(Bs_i)A \exp((t^{n+1} - \sum_{j=1}^{i} s_j)(A + B)) c(t^n) \, ds_{i+1} \ldots ds_1, \]

(41)
We obtain:

\[ ||e_i|| \leq ||\exp((A + B)\tau)c(t^n) - c_i|| \leq C\tau^i c(t^n), \] (42)

The same idea can be applied to the even iterative scheme.
Extension to nonlinear differential equations:
The iterative operator-splitting method is used as a fixed-point scheme or with embedded Newton’s method to linearize the nonlinear operators.
We concentrate again on nonlinear differential equations of the form

$$\frac{du}{dt} = A(u(t))u(t) + B(u(t))u(t), \quad \text{with } u(t^n) = u^n, \quad (43)$$

where $A(u), B(u)$ are matrices with nonlinear entries and densely defined, where we assume that the entries involve the spatial derivatives of $c$. 
Iterative operator-splitting method as fixed-point scheme

We split our nonlinear differential equation (43) by applying

\[
\frac{du_i(t)}{dt} = A(u_{i-1}(t))u_i(t) + B(u_{i-1}(t))u_{i-1}(t), \quad (44)
\]

with \( u_i(t^n) = c^n \),

\[
\frac{du_{i+1}(t)}{dt} = A(u_{i-1}(t))u_i(t) + B(u_{i-1}(t))u_{i+1}(t), \quad (45)
\]

with \( u_{i+1}(t^n) = c^n \),

where the time step is \( \tau = t^{n+1} - t^n \). The iterations are \( i = 1, 3, \ldots, 2m + 1 \). \( u_0(t) = c_n \) is the starting solution, where we assume that the solution \( c^{n+1}_n \) is near \( c^n \), or \( u_0(t) = 0 \). So we have to solve the local fixed-point problem. \( c^n \) is the known split approximation at time level \( t = t^n \).
Operator-splitting method with embedded Jacobian Newton iterative method

The Newton’s method is used to solve the nonlinear parts of the iterative operator-splitting method, see the linearization techniques in [Kelley2003], Karlsson1997. We apply the iterative operator-splitting method and obtain:

\[ F_1(u_i) = \partial_t u_i - A(u_i)u_i - B(u_{i-1})u_{i-1} = 0, \]
with \( u_i(t^n) = c^n \),

\[ F_2(u_{i+1}) = \partial_t u_{i+1} - A(u_i)u_i - B(u_{i+1})u_{i+1} = 0, \]
with \( u_{i+1}(t^n) = c^n \),

where the time step is \( \tau = t^{n+1} - t^n \). The iterations are \( i = 1, 3, \ldots, 2m + 1 \). \( c_0(t) = 0 \) is the starting solution and \( c^n \) is the known split approximation at time level \( t = t^n \). The results of the methods are \( c(t^{n+1}) = u_{2m+2}(t^{n+1}) \).
The splitting method with embedded Newton’s method is given as

\[ u^{(k+1)}_i = u^{(k)}_i - D(F_1(u^{(k)}_i))^{-1}(\partial_t u^{(k)}_i - A(u^{(k)}_i)u^{(k)}_i - B(u^{(k)}_{i-1})u^{(k)}_{i-1}), \]

with \( D(F_1(u^{(k)}_i)) = -(A(u^{(k)}_i) + \frac{\partial A(u^{(k)}_i)}{\partial u^{(k)}_i} u^{(k)}_i) \),

and \( k = 0, 1, 2, \ldots, K \), with \( u_i(t^n) = c^n \),

\[ u^{(l+1)}_i = u^{(l)}_{i+1} - D(F_2(u^{(l)}_{i+1}))^{-1}(\partial_t u^{(l)}_{i+1} - A(u^{(k)}_i)u^{(k)}_i - B(u^{(k)}_{i+1})u^{(k)}_{i+1}), \]

with \( D(F_2(u^{(l)}_{i+1})) = -(B(u^{(l)}_{i+1}) + \frac{\partial B(u^{(l)}_{i+1})}{\partial u^{(l)}_{i+1}} u^{(l)}_{i+1}) \),

and \( l = 0, 1, 2, \ldots, L \), with \( u_{i+1}(t^n) = c^n \).
Consistency and Stability Proofs:
1.) Bounded case: First linearisation, then application of the linear theory.
2.) Unbounded Case: Linearisation and exponential formulation.
Improvement of initialization and stability problems

- Standard splitting methods have the problem to be less effective in the rate of the convergence and CPU times.
Improvement of initialization and stability problems

- Standard splitting methods have the problem to be less effective in the rate of the convergence and CPU times

- Here we propose the followings to overcome these difficulties:
Improvement of initialization and stability problems

- Standard splitting methods have the problem to be less effective in the rate of the convergence and CPU times
- Here we propose the followings to overcome these difficulties:
  - Improve the starting conditions via Zassenhaus product formula,
Improvement of initialization and stability problems

- Standard splitting methods have the problem to be less effective in the rate of the convergence and CPU times.
- Here we propose the followings to overcome these difficulties:
  - Improve the starting conditions via Zassenhaus product formula,
  - Accelerate the subproblems via *Weighted Polynomials*,

Jürgen Geiser
Humboldt Universität zu Berlin, D- Berlin, Germany
Improvement of initialization and stability problems

- Standard splitting methods have the problem to be less effective in the rate of the convergence and CPU times.

- Here we propose the followings to overcome these difficulties:
  - Improve the starting conditions via Zassenhaus product formula,
  - Accelerate the subproblems via *Weighted Polynomials*,
  - Extend operator Splitting methods via Zassenhaus product formula.
Improved Splitting Methods

The standard exponential splitting methods are based on the following decomposition idea, (see Ostermann and Hansen 08, also Chin 06, Yoshida 1991, etc.):

$$\exp((A + B)t) = \pi_{i=1}^{j} \exp(a_{i}At) \exp(b_{i}Bt) + O(t^{j+1}).$$
Improved Splitting Methods

The standard exponential splitting methods are based on the following decomposition idea, (see Ostermann and Hansen 08, also Chin 06, Yoshida 1991, etc.):

$$\exp((A + B)t) = \pi_{i=1}^{i} \exp(a_i At) \exp(b_i Bt) + O(t^{i+1}).$$

Novel method based on Zassenhaus product formula (see Scholz et al. 2008):

$$\exp((A + B)t) = \exp(At) \exp(Bt) \pi_{j=2}^{m} \exp(C_j t^j) + O(t^{m+1}).$$

where $C_j$ is a function of Lie brackets with $A$ and $B$. 
Improved A-B Splitting Method

Theorem

We solve the initial value problem (7) and (8). We assume bounded and constant operators A and B. The consistency error of the A-B splitting is $O(\tau_n)$, then we can improve the error of the A-B splitting scheme to $O(\tau_n^p)$, $p > 1$ by improving the starting conditions $c_0$ as

$$\tilde{c}_0 = (\pi_{j=2}^p \exp(C_j \tau_n^j))c_0$$

where $C_j$ is called as Zassenhaus exponents, thus local splitting error of A-B splitting method can be read as follows

$$\rho_n = (\exp(\tau_n(A + B)) - \exp(\tau_nB)\exp(\tau_nA)(\pi_{j=2}^p \exp(C_j \tau_n^j)))c_0$$

$$= C_T \tau_n^{p+1} + O(\tau_n^{p+2}) \quad (46)$$
Improved Iterative Splitting Method

Theorem

We solve the initial value problem (7) and (8). We assume bounded and constant operators A and B.

The consistency error of the iterative splitting is $O(\tau_n^i)$, then we can improve the error of the iterative splitting scheme to $O(\tau_n^{i+p})$, $p > 1$ by improving the starting conditions $c_0$ as

$$\tilde{c}_0 = (\pi_{j=2}^p \exp(\tilde{C}_j \tau_n^j))c_0$$

where $\tilde{C}_j$ is called as Zassenhaus exponents. thus local splitting error can be improved as

$$\rho_n = (\exp(\tau_n(A + B)) - S_i \pi_{j=2}^p \exp(\tilde{C}_j \tau_n^j))c_0$$

$$= C_T \tau_n^{i+p+1} + O(\tau_n^{i+p+2})$$ (47)
Example for Iterative splitting scheme

Theorem

There exists a Weighted Polynomial so that the order of the accuracy of iterative splitting with alternating operators can be increased up to $O(t^3)$. 
Proof.

We give the proof by construction in the following steps:

- Step 1: Start the initiation as $c_0 = 0$
- Step 2: Accelerate the $c_1$ as $\tilde{c}_1 = w_1 c_1 = (I + Wt) c_1$
- Step 3: Compute $c_2$ with the $\tilde{c}_1$
- Step 4: Expand $\exp(Bt)$ up to $O(t^3)$,
- Step 5: Finally compare this with exact solution up to $O(t^3)$ to find the commutator and $w_1$ as follows.
Example for bounded operators: The exact solution of the problem is given by,

\[ c_{\text{exact}} = e^{(A+B)t} \]

\[ = (I + (A + B)t + (\frac{A^2}{2} + \frac{AB}{2} + \frac{BA}{2} + \frac{B^2}{2})t^2)c_0 + O(t^3) \]

and the error can be found by subtracting the Equation (48) from the new \( c_2 \) and we have

\[ |c_{\text{exact}} - c_2| \leq (\frac{AB}{2} - \frac{AW}{2})O(t^2) + O(t^3), \]
From this expression if $W = B$, the order of the accuracy of iterative splitting with respect to alternating operators can be increased up to $O(t^3)$, thus we can find the *Weighted Polynomial* as follows:

$$w_1 = I + Bt.$$  \hspace{1cm} (50)

Therefore,

$$|c_{\text{exact}} - c(i=2)| \leq C O(t^3).$$ \hspace{1cm} (51)

where $C$ is the function of Commutators.
We deal with the following linear ordinary differential equation:

\[
\frac{\partial u(t)}{\partial t} = \begin{pmatrix} -\lambda_1 & \lambda_2 \\ \lambda_1 & -\lambda_2 \end{pmatrix} u(t), \quad t \in [0, T],
\]

\[u(0) = u_0,\]

where the initial condition \(u_0 = (1, 1)\) is given on the interval \([0, T]\) and analytical solution.
We split our linear operator into two operators by setting

\[
\frac{\partial u(t)}{\partial t} = \begin{pmatrix} -\lambda_1 & 0 \\ \lambda_1 & 0 \end{pmatrix} u(t) + \begin{pmatrix} 0 & \lambda_2 \\ 0 & -\lambda_2 \end{pmatrix} u(t).
\] (54)

We choose \(\lambda_1 = 0.25\) and \(\lambda_2 = 0.5\) on the interval \([0, 1]\), i.e. with \(T = 1\).
For our time integration method, we assume a time interval $[0, T]$ and divide it in $n$ intervals with the length $\tau_n = \frac{T}{n}$. We can improve our results by using smaller time steps and more iteration steps.

For the initialization of our iterative method, for $i = 1$, we use $u_0(0) = (0, 0)^t$.

From the examples, one can see that the order increases by each iteration step.
<table>
<thead>
<tr>
<th>Iteration steps $i$</th>
<th>Number of splitting partitions $n$</th>
<th>$\text{err}_1$</th>
<th>$\text{err}_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>4.5321e-002</td>
<td>4.5321e-002</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>3.9664e-003</td>
<td>3.9664e-003</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>3.9204e-004</td>
<td>3.9204e-004</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>7.6766e-003</td>
<td>7.6766e-003</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>6.6385e-005</td>
<td>6.6385e-005</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>6.5369e-007</td>
<td>6.5369e-007</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4.6126e-004</td>
<td>4.6126e-004</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>4.1321e-007</td>
<td>4.1321e-007</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>4.0839e-010</td>
<td>4.0839e-010</td>
</tr>
</tbody>
</table>

**Table:** Numerical results for iterative splitting method and fourth-order Gauß-RK method.
Linear ODE with Stiff Parameters

We deal with the same equation as in the first example, now choosing \( \lambda_1 = 1 \) and \( \lambda_2 = 10^4 \) on the interval \([0,1]\).

We therefore have the operators:

\[
A = \begin{pmatrix} -1 & 0 \\ 1 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 0 & 10^4 \\ 0 & -10^4 \end{pmatrix}.
\]
The discretization of the linear ordinary differential equation is done with the BDF3 method. Our numerical results are presented in Table 2. For the stiff problem, we choose more iteration steps and time partitions and show the error between the analytical and numerical solution in the supremum norm, i.e.

$$\text{err}_k = |u_{k,\text{exact}} - u_{k,\text{num}}| \text{ with } k = 1, 2.$$
<table>
<thead>
<tr>
<th>Iteration steps $i$</th>
<th>Number of splitting partitions $n$</th>
<th>$\text{err}_1$</th>
<th>$\text{err}_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1</td>
<td>3.4434e-001</td>
<td>3.4434e-001</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>3.0907e-004</td>
<td>3.0907e-004</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>2.2600e-006</td>
<td>2.2600e-006</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>1.5397e-011</td>
<td>1.5397e-011</td>
</tr>
</tbody>
</table>

**Table:** Numerical results for the stiff example with iterative operator splitting method and BDF3 method.
For the stiff case, we obtain improved results with more than 5 iteration steps. Because of the inexact starting function, the accuracy must be improved by more iteration steps. Higher-order time discretization methods, such as BDF3 method and iterative operator splitting method, accelerate the solving process.
Test example: Convection-Reaction Model

We consider the following test problem:

\[ u_t + au_x - bu = 0 \]  \hspace{1cm} \text{(55)}

where \((x, t) \in [0, 1] \times [0, 1]\) with exact solution \(u(x, t) = e^x e^{(b-a)t}\)

where \(a = 1\), \(b = 1\) and initial conditions, boundary conditions are taken from exact solution.
For the spatial discretization, we apply a second order finite difference method:

\[ u_x \approx \frac{1}{\Delta x} \left[ -\frac{3}{2} \quad 2 \quad -\frac{1}{2} \right] \]

The semi-discretized system is solved by iterative splitting method and improved iterative method with initialization by Zassenhaus formula.
Comparison of errors with Iterative Splitting and Midpoint for solutions without weight, with one term weight and with two term weight are given in table (49):

<table>
<thead>
<tr>
<th></th>
<th>$err_{L_\infty}$</th>
<th>$err_{L_1}$</th>
<th>CPU Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterative Method</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Without w</td>
<td>0.1119</td>
<td>0.1756</td>
<td>0.073314</td>
</tr>
<tr>
<td>With one w</td>
<td>0.0769</td>
<td>0.1112</td>
<td>0.079872</td>
</tr>
<tr>
<td>With two w</td>
<td>0.0766</td>
<td>0.1107</td>
<td>0.080240</td>
</tr>
</tbody>
</table>

**Table:** Comparison of errors with Iterative splitting and Midpoint for $\Delta x = 0.05\,\Delta t = 0.01$
Comparison of errors with Iterative Splitting and Midpoint for solutions without weight, with one term weight and with two term weight are given in table (52):

<table>
<thead>
<tr>
<th></th>
<th>$err_{L_{\infty}}$</th>
<th>$err_{L_{1}}$</th>
<th>CPU Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterative Method</td>
<td>Without w</td>
<td>0.3721</td>
<td>0.3016</td>
</tr>
<tr>
<td></td>
<td>With one w</td>
<td>0.0693</td>
<td>0.0619</td>
</tr>
<tr>
<td></td>
<td>With two w</td>
<td>0.0612</td>
<td>0.0403</td>
</tr>
</tbody>
</table>

Table: Comparison of errors with Iterative splitting and Midpoint for $\Delta x = 0.2$ $\Delta t = 0.1$
Comparison of errors with Iterative Splitting and 4th Order Runge-Kutta Method for solutions without weight, with one term weight and with two term weight are given in table (51):

<table>
<thead>
<tr>
<th>Iterative Method</th>
<th>$err_{L_{\infty}}$</th>
<th>$err_{L_1}$</th>
<th>CPU Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without w</td>
<td>0.2197</td>
<td>0.1831</td>
<td>0.029454</td>
</tr>
<tr>
<td>With one w</td>
<td>0.0858</td>
<td>0.0804</td>
<td>0.030352</td>
</tr>
<tr>
<td>With two w</td>
<td>0.0631</td>
<td>0.0625</td>
<td>0.030736</td>
</tr>
</tbody>
</table>

**Table:** Comparison of errors with Iterative splitting and 4th Order Runge-Kutta Method for $\Delta x = 0.2$ $\Delta t = 0.1$
Comparison of errors with Iterative Splitting and 4th Order Runge-Kutta Method for solutions without weight, with one term weight and with two term weight are given in table (52):

<table>
<thead>
<tr>
<th></th>
<th>$err_{L_\infty}$</th>
<th>$err_{L_1}$</th>
<th>CPU Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterative Method</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Without w</td>
<td>0.1561</td>
<td>0.2587</td>
<td>0.038844</td>
</tr>
<tr>
<td>With one w</td>
<td>0.0906</td>
<td>0.1408</td>
<td>0.040069</td>
</tr>
<tr>
<td>With two w</td>
<td>0.0900</td>
<td>0.1392</td>
<td>0.040208</td>
</tr>
</tbody>
</table>

Table: Comparison of errors with Iterative splitting and 4th Order Runge-Kutta Method for $\Delta x = 0.1 \Delta t = 0.02$
For hyperbolic problem, the iterative operator splitting scheme is considered with improved initialization. While the numerical error is reduced twice and more, the amount of computational time is nearly the same. Also stiff problems can be damped by the improved initialization. Here we have the benefit of the weighted method, that accelerates the initialisation process at the beginning of the method.
Test example: Bernoulli Equation \((n=3)\)

We deal with the Bernoulli equation and explain the splitting ideas. Our equation is given as:

\[
\frac{dy}{dx} = \lambda_1 y + \lambda_2 y^n.
\]  

We have the analytical solution:

\[
y = \sqrt{\frac{\exp(-2t)}{-\exp(-2t) + 2}},
\]

where \(\lambda_1 = -1, \lambda_2 = -1\).
Here we decouple with respect to the linear and nonlinear term: We choose the time interval \([0,1]\).

The operators for our splitting methods are given as
\[ Au = \lambda_1 u \quad \text{and} \quad B(u) = \lambda_2 u^2(t^n)u, \]
where we linearize the term \(B(u) = \lambda_2 u^2(t^n)u\).

The approximation error is calculated by the maximum error and given as
\[ \text{Max-error} = \max_{i,j} \| u_{\text{exact}}(T) - u_{\text{approx}}(T) \|. \]
Assembling of the linearizations
We have implemented different linearization techniques to accelerate the solving process.
For all methods we have $t \in (t^n, t^{n+1})$ with $t^{n+1} = t^n + k \cdot h$, where $h$ is the length of a time step per interval and $k$ is the number of time steps per interval.
Here we deal with $k = 1000, 100, 10$ as number of time steps per interval.
1.) time linearization:
\[ u^m_i(t^n + h \cdot (j+1)) \approx u^{m-1}_i(t^n + h \cdot j) \cdot u \] \text{ and } j = 0, 1, \ldots k - 1

2.) Fixpoint linearization:
\[ u^m_i(t^n + h \cdot (j+1)) \approx u^{m-1}_{i-1}(t^n + h \cdot j) \cdot u \] \text{ and } j = 0, 1, \ldots k - 1

3.) Fixpoint linearization:
\[ u^m_i(t^n + h \cdot (j+1)) \approx u^{m-1}_{i-1}(t^n + h \cdot (j+1)) \cdot u \] \text{ and } j = 0, 1, \ldots k - 1

4.) Trapezoidal linearization:
\[ u^m_i(t^n + h \cdot (j+1)) \approx \frac{u^{m-1}_i(t^n + h \cdot j) + u^{m-1}_{i-1}(t^n + h \cdot (j+1))}{2} \cdot u \] \text{ and } j = 0, 1, \ldots k - 1
We have compared the first 4 linearization schemes with different techniques for \( n = 3 \). We present the absolute error to the exact solution at the time point \( t = 1 \).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>1.)</th>
<th>2.)</th>
<th>3.)</th>
<th>4.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>1000</td>
<td>9.0918e-2</td>
<td>1.5120e-1</td>
<td>1.5081e-1</td>
<td>1.1983e-1</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>100</td>
<td>3.5313e-3</td>
<td>6.5402e-3</td>
<td>6.4061e-3</td>
<td>4.9586e-3</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>10</td>
<td>4.0558e-4</td>
<td>6.5628e-4</td>
<td>5.3055e-4</td>
<td>4.6807e-4</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1000</td>
<td>1.2493e-2</td>
<td>6.5628e-4</td>
<td>2.2099e-2</td>
<td>1.7378e-2</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>100</td>
<td>1.5572e-4</td>
<td>3.4215e-4</td>
<td>3.3263e-4</td>
<td>2.4289e-4</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>10</td>
<td>1.8933e-6</td>
<td>3.3227e-6</td>
<td>2.4834e-6</td>
<td>2.1881e-6</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1000</td>
<td>6.2378e-3</td>
<td>1.7962e-2</td>
<td>1.7332e-2</td>
<td>1.0961e-2</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>100</td>
<td>9.9185e-5</td>
<td>1.2001e-4</td>
<td>7.9856e-5</td>
<td>1.0473e-5</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>10</td>
<td>9.0573e-5</td>
<td>9.0986e-5</td>
<td>9.0857e-5</td>
<td>1.1228e-7</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1000</td>
<td>6.3537e-4</td>
<td>2.0411e-3</td>
<td>1.8416e-3</td>
<td>1.1300e-3</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>100</td>
<td>7.1710e-6</td>
<td>8.1401e-6</td>
<td>6.8926e-6</td>
<td>2.2160e-7</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>10</td>
<td>4.9658e-7</td>
<td>4.9905e-7</td>
<td>8.3813e-7</td>
<td>1.6957e-7</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1000</td>
<td>4.1674e-4</td>
<td>1.3140e-3</td>
<td>5.6985e-4</td>
<td>4.5031e-4</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>100</td>
<td>9.5261e-5</td>
<td>1.0045e-4</td>
<td>1.0032e-4</td>
<td>1.0809e-7</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>10</td>
<td>9.0569e-5</td>
<td>9.0972e-5</td>
<td>9.0879e-5</td>
<td>1.0153e-7</td>
</tr>
</tbody>
</table>

\( n_{it} \) - Number of iterations

\( n_{int} \) - Number of intervals

\( n_{ts} \) - Number of time steps per interval
Test example: Burgers equation

We deal with a 2D example where we can derive an analytical solution.

\[ \partial_t u = -u \partial_x u - u \partial_y u + \mu(\partial_{xx} u + \partial_{yy} u) + f(x, y, t), \quad (x, y, t) \in \Omega \times [0, T] \]
\[ u(x, y, 0) = u_{\text{ana}}(x, y, 0), \quad (x, y) \in \Omega \]

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

where \( \Omega = [0, 1] \times [0, 1] \), \( T = 1.25 \), and \( \mu \) is the viscosity. The analytical solution is given as

\[ u_{\text{ana}}(x, y, t) = (1 + \exp\left(\frac{x + y - t}{2\mu}\right))^{-1}, \]

where \( f(x, y, t) = 0 \).
The operators are given as:
\(A(u)u = -u \partial_x u - u \partial_y u\), hence \(A(u) = -u \partial_x - u \partial_y\) (the nonlinear operator),
\(Bu = \mu(\partial_{xx} u + \partial_{yy} u) + f(x, y, t)\) (the linear operator).
We apply the nonlinear Algorithm 44 to the first equation and obtain
\(A(u_{i-1})u_i = -u_{i-1} \partial_x u_i - u_{i-1} \partial_y u_i\) and
\(Bu_{i-1} = \mu(\partial_{xx} + \partial_{yy})u_{i-1} + f\),
and we obtain linear operators, because \(u_{i-1}\) is known from the previous time step.
In the second equation we obtain by using Algorithm 45:
\(A(u_{i-1})u_i = -u_{i-1} \partial_x u_i - u_{i-1} \partial_y u_i\) and
\(Bu_{i+1} = \mu(\partial_{xx} + \partial_{yy})u_{i+1} + f\),
and we have also linear operators.
We have the following results, see Tables 7, for different steps in time and space and different viscosities.

<table>
<thead>
<tr>
<th>$\Delta x = \Delta y$</th>
<th>$\Delta t$</th>
<th>$\text{err}_{L_1}$</th>
<th>$\text{err}_{\text{max}}$</th>
<th>$\rho_{L_1}$</th>
<th>$\rho_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/10</td>
<td>1/10</td>
<td>0.0549</td>
<td>0.1867</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1/20</td>
<td>1/10</td>
<td>0.0468</td>
<td>0.1599</td>
<td>0.2303</td>
<td>0.2234</td>
</tr>
<tr>
<td>1/40</td>
<td>1/10</td>
<td>0.0418</td>
<td>0.1431</td>
<td>0.1630</td>
<td>0.1608</td>
</tr>
<tr>
<td>1/10</td>
<td>1/20</td>
<td>0.0447</td>
<td>0.1626</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1/20</td>
<td>1/20</td>
<td>0.0331</td>
<td>0.1215</td>
<td>0.4353</td>
<td>0.4210</td>
</tr>
<tr>
<td>1/40</td>
<td>1/20</td>
<td>0.0262</td>
<td>0.0943</td>
<td>0.3352</td>
<td>0.3645</td>
</tr>
<tr>
<td>1/10</td>
<td>1/40</td>
<td>0.0405</td>
<td>0.1551</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1/20</td>
<td>1/40</td>
<td>0.0265</td>
<td>0.1040</td>
<td>0.6108</td>
<td>0.5768</td>
</tr>
<tr>
<td>1/40</td>
<td>1/40</td>
<td>0.0181</td>
<td>0.0695</td>
<td>0.5517</td>
<td>0.5804</td>
</tr>
</tbody>
</table>

Table: Numerical results for the Burgers equation with viscosity $\mu = 0.05$, initial condition $u_0(t) = c_n$, and two iterations per time step.
Figure 2 presents the profile of the 2D nonlinear Burgers equation.

Figure: Burgers equation at initial time $t = 0.0$ for viscosity $\mu = 0.05$. 
Figure 2 presents the profile of the 2D nonlinear Burgers equation.

**Figure**: Burgers equation at end time $t = 1.25$ for viscosity $\mu = 0.05$. 
Future Works

Outview

1) Theory for time-dependent and nonlinear problems.
2) Splitting schemes for PDE with Perturbation of Boundary.
3) Framework to improved decomposition methods