Chapter 3
Algorithmic Part

Abstract In this chapter, we discuss the algorithmic parts with respect to the different methods we applied in the application part.

3.1 Introduction

In the following, we discuss different methods based on iterative and additive ideas to decompose scale-dependent equations.

Based on the different scale-dependent operators of the equations, we deal with the ideas of decomposing into simpler and faster computable equations.

Basic idea is that to decompose the operator with respect to their spatial and time scales into different scale-dependent operators, e.g. we decompose the operator

$$A = A_{macro} + A_{micro}, \tag{3.1}$$

where the operators are given as

- A_{macro} (macroscopic operator) has larger in order entries, and then
- A_{micro} (microscopic operator) has smaller in order entries,

while $|A_{micro,ij}| \leq \varepsilon \ |A_{macro,ij}|, \ \forall i, j \in I, 0 < \varepsilon \ll 1$, i.e. we decompose the different scales of two operators.

To solve the evolution equation,

$$\frac{\partial c}{\partial t} = A_{macro}(c)c + A_{micro}(c)c, \tag{3.2}$$

where $c(0) = c_0$ is the initial condition and we assume that the semi-discretized operator A has included the boundary conditions.

Two different solver ideas are discussed:

- Iterative Scheme: Based on iterative cycles, we solve the underlying decomposed equations based on the successive approximation or fixpoint scheme.
- Additive Scheme: Based on decomposing into tridiagonal matrices, we solve sequentially simpler equations.

J. Geiser, *Multicomponent and Multiscale Systems*,
DOI 10.1007/978-3-319-15117-5_3

3.2 Iterative Methods

This model is reformulated by the semi-discretization of the spatial operators to the following Cauchy problem, while c is now in the following vectorial function:

$$\frac{\partial c(t)}{\partial t} = Ac(t) + f(t) \tag{3.3}$$

$$= A_1 c(t) + A_2 c(t) + f(t), \text{ with } t \in [0, T],\ c(0) = c_0, \tag{3.4}$$

where the initial function c_0 is given. A_1 and A_2 are assumed to be bounded, constant, linear operators in an appropriate Banach space $\mathbf{X}$ with $A_1, A_2 : \mathbf{X} \to \mathbf{X}$ with an appropriate vector and matrix norm $||\cdot||$.

In the following, we deal with the following definition of the stiff operators, see also [1, 2].

Definition 3.1 We consider the stiffness in the following sense: A_1 is supposed to be stiff and A_2 non-stiff. Stiffness means that τA_1 is huge in norm for the range of step size τ, see [3]. Here, the step size represents a splitting step size. So we assume

$$||\tau A_1|| \gg 1,\ \ ||\tau A_2|| = O(\tau). \tag{3.5}$$

For the notation of the eigenvalues, we have

$$Re(\tau\lambda) \ll -1,\ \ |\tau\mu| = O(\tau), \tag{3.6}$$

where λ is a stiff eigenvalue of A_1 and μ a non-stiff eigenvalue of A_2.

In the next subsection, we present the iterative schemes.

3.2.1 Iterative Schemes

We then consider the following forms of the iterative splitting schemes to solve the linear model equation:
1. Iterative splitting with respect to a diagonal matrix part (Jacobi Scheme):

$$\frac{\partial c_i(t)}{\partial t} = A_1 c_i(t) + A_2 c_{i-1}(t) + f(t), \text{ with } c_i(t^n) = c^n \tag{3.7}$$

$$\frac{\partial c_{i+1}(t)}{\partial t} = A_1 c_{i-2}(t) + A_2 c_{i+1}(t) + f(t), \text{ with } c_{i+1}(t^n) = c^n, \tag{3.8}$$

$$i = 1, 3, \ldots, 2m + 1,$$

$c_0(t) = 0$ and $c_{-1}(t) = 0$, after each iterative step we update $i = i + 1$.

2. Iterative splitting with respect to a full matrix part (Gauss–Seidel Scheme):

$$\frac{\partial c_i(t)}{\partial t} = A_1 c_i(t) + A_2 c_{i-1}(t) + f(t), \text{ with } c_i(t^n) = c^n \tag{3.9}$$

$$\frac{\partial c_{i+1}(t)}{\partial t} = A_1 c_i(t) + A_2 c_{i+1}(t) + f(t), \text{ with } c_{i+1}(t^n) = c^n, \tag{3.10}$$

$$i = 1, 3, \ldots, 2m+1,$$

3. Unsymmetrical weighted iterative splitting (JOR, Jacobian Overrelaxation Scheme):

$$\frac{\partial c_i(t)}{\partial t} = \frac{1}{\omega} A_1 c_i(t) + A_2 c_{i-1} + \left(1 - \frac{1}{\omega}\right) A_1 c_{i-2}(t) + f(t), \tag{3.11}$$

$$\text{with } c_i(t^n) = c^n$$

$$\frac{\partial c_{i+1}(t)}{\partial t} = A_1 c_{i-2}(t) + \frac{1}{\omega} A_2 c_{i+1}(t) + \left(1 - \frac{1}{\omega}\right) A_2 c_{i-1}(t) + f(t), \tag{3.12}$$

$$\text{with } c_{i+1}(t^n) = c^n,$$

$$i = 1, 3, \ldots, 2m+1,$$

where $\omega \in (0, 1]$.

4. Symmetrical weighted iterative splitting (SOR: Successive Overrelaxation Scheme):

$$\frac{\partial c_i(t)}{\partial t} = \frac{1}{\omega} A_1 c_i(t) + A_2 c_{i-1} + \left(1 - \frac{1}{\omega}\right) A_1 c_{i-2}(t) + f(t), \text{ with } c_i(t^n) = c^n \tag{3.13}$$

$$\frac{\partial c_{i+1}(t)}{\partial t} = A_1 c_i(t) + \frac{1}{\omega} A_2 c_{i+1}(t) + \left(1 - \frac{1}{\omega}\right) A_2 c_{i-1}(t) + f(t), \text{ with } c_{i+1}(t^n) = c^n,$$

$$i = 1, 3, \ldots, 2m+1, \tag{3.14}$$

where $\omega \in (0, 1]$.

Remark 3.1 For all schemes, we assume that the operator A_1 has a large time scale and A_2 has a small time scale. In addition, the initialization is given as $c_0(t) = 0$, $c_{-1}(t) = 0$, while 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 $c^{n+1} = c_{2m+1}(t^{n+1})$, with $n = 1, \ldots, N-1$ and the final time is given as $T = N\,\tau$.

3.2.2 Reformulation to Waveform Relaxation Scheme

In the following, we reformulate in the notation of the waveform relaxation scheme. We obtain the following schemes, see also [4, 5]:

$$\frac{dU_{\tilde{i}}}{dt} = \mathscr{P}U_{\tilde{i}} + \mathscr{Q}U_{\tilde{i}-1} + F, \tag{3.15}$$

$$U_{\tilde{i}}(t^n) = U(t^n), \tag{3.16}$$

$$\tilde{i} = 1, 2, \ldots, m, \tag{3.17}$$

where $U_{\tilde{i}-1} = (c_{i-2}, c_{i-1})^t$, $U_{\tilde{i}} = (c_i, c_{i+1})^t$ and the initialization $U_0(t) = (0, 0)^t$ is given with the zero vectors. Furthermore, we define that $\mathscr{P}$ and $\mathscr{Q}$ are the diagonal and outerdiagonal matrices of the underlying splitting methods given in Sect. 3.2.1 and $\mathscr{A} = \mathscr{P} + \mathscr{Q}$ is the full matrix.

We embed the iterative splitting schemes in the following waveform relaxation schemes:

(1) Jacobian:

$$\mathscr{P} = \begin{pmatrix} A_1 & 0 \\ 0 & A_2 \end{pmatrix}, \mathscr{Q} = \begin{pmatrix} 0 & A_2 \\ A_1 & 0 \end{pmatrix}, F = \begin{pmatrix} f \\ f \end{pmatrix}. \tag{3.18}$$

(2) Gauss–Seidel:

$$\mathscr{P} = \begin{pmatrix} A_1 & 0 \\ A_1 & A_2 \end{pmatrix}, \mathscr{Q} = \begin{pmatrix} 0 & A_2 \\ 0 & 0 \end{pmatrix}, F = \begin{pmatrix} f \\ f \end{pmatrix}. \tag{3.19}$$

(3) JOR:

$$\mathscr{P} = \frac{1}{\omega}\begin{pmatrix} A_1 & 0 \\ 0 & A_2 \end{pmatrix}, \mathscr{Q} = \left(1 - \frac{1}{\omega}\right)\begin{pmatrix} A_1 & 0 \\ 0 & A_2 \end{pmatrix} + \begin{pmatrix} 0 & A_2 \\ A_1 & 0 \end{pmatrix}, F = \begin{pmatrix} f \\ f \end{pmatrix}. \tag{3.20}$$

(4) SOR:

$$\begin{aligned} \mathscr{P} &= \frac{1}{\omega}\begin{pmatrix} A_1 & 0 \\ 0 & A_2 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ A_1 & 0 \end{pmatrix}, \\ \mathscr{Q} &= \left(1 - \frac{1}{\omega}\right)\begin{pmatrix} A_1 & 0 \\ 0 & A_2 \end{pmatrix} + \begin{pmatrix} 0 & A_2 \\ 0 & 0 \end{pmatrix}, F = \begin{pmatrix} f \\ f \end{pmatrix}. \end{aligned} \tag{3.21}$$

Remark 3.2 We have also extended the application to non-autonomous differential equations, by adding the right-hand side term.

3.3 Additive Methods

The idea of the additive methods is to decompose in an additive manner the different operators of the differential equation. We concentrate on solving such semi-discretized linear evolution equations and the notation for such a differential equation is

$$B\partial_t\, u = Au,\ u(0) = u_0, \tag{3.22}$$

where A and B can be unbounded operators. We obtain large-scale differential equation, which are delicate to solve with standard solvers.

The evolution equation (3.22) is solved with the following underlying splitting schemes:

- Additive Splitting schemes and
- Iterative Splitting schemes.

3.3.1 Additive Splitting Schemes

We deal with the following equation:

$$\sum_{\beta=1}^{p} B_{\alpha\beta}\partial_t\, u_\beta = \sum_{\beta=1}^{p} A_{\alpha\beta}u_\beta + f_\alpha,\ \alpha = 1, 2, \ldots, p, \tag{3.23}$$

$$u_\alpha(0) = u_{\alpha,0},\ \alpha = 1, 2, \ldots, p. \tag{3.24}$$

Furthermore, we assume that A and B are self-adjoint.

We apply the discretization with the schemes of weights and obtain

$$B\frac{u^{n+1} - u^n}{\tau} - A\left(\sigma u^{n+1} + (1-\sigma)u^n\right) = f\left(\sigma t^{n+1} + (1-\sigma)t^n\right), \tag{3.25}$$

By the transition to a new time level, we require

$$(B - A\sigma\tau)u^{n+1} = \phi^n, \tag{3.26}$$

while $\phi^n = (1-\sigma)\tau Au^n + Bu^n + f(\sigma t^{n+1} + (1-\sigma)t^n)$.

With the idea of splitting this into two problems, the original problem can be transformed to

$$\sum_{\beta=1}^{p}(B_{\alpha\beta} - A_{\alpha\beta}\sigma\tau)u_\beta^{n+1} = \phi_\alpha^n, \alpha = 1, 2, \ldots, p, \tag{3.27}$$

where $(\tilde{B} - A\sigma\tau) = (B - A_1\sigma\tau)B^{-1}(B - A_2\sigma\tau)$ and $\sigma \in (0, 1)$.

We have to solve the following pair of linear equations:

$$(B - A_1\sigma\tau)\psi^n = \phi^n, \tag{3.28}$$

$$(B - A_2\sigma\tau)u^{n+1} = \psi^n. \tag{3.29}$$

By a change to a sequence of simpler problems, we have

$$\left(B_{\alpha\alpha} - \frac{1}{2}A_{\alpha\alpha}\sigma\tau\right)u_\beta^{n+1/2} = \tilde{\psi}_\alpha^n, \alpha = 1, 2, \ldots, p, \tag{3.30}$$

$$\left(B_{\alpha\alpha} - \frac{1}{2}A_{\alpha\alpha}\sigma\tau\right)u_\beta^{n+1} = \hat{\psi}_\alpha^n, \alpha = 1, 2, \ldots, p. \tag{3.31}$$

Here, we have the benefit of needing to invert only the diagonal parts of the matrices and use the idea of solving the triangular splitting of the operator $A = A_1 + A_2$.

The second-order algorithm is given as a two-step method, see Algorithm 3.1.

Algorithm 3.1 (1) Compute
$\tilde{\psi}^{n+1} = (\tilde{\psi}_1^{n+1}, \ldots, \tilde{\psi}_p^{n+1})^T$ with $\phi^n = (\phi_1^{n+1}, \ldots, \phi_p^{n+1})^T$

$$\tilde{\psi}_1^{n+1} = \left(I - \frac{1}{2}A_{11}B_{11}^{-1}\sigma\tau\right)^{-1}\phi_1^n \tag{3.32}$$

$$\tilde{\psi}_2^{n+1} = \left(I - \frac{1}{2}A_{11}B_{11}^{-1}\sigma\tau\right)^{-1}\left(\phi_2^n + A_{21}\sigma\tau B_{11}^{-1}\tilde{\psi}_1^{n+1}\right) \tag{3.33}$$

$$\ldots \tag{3.34}$$

$$\tilde{\psi}_p^{n+1} = \left(I - \frac{1}{2}A_{pp}B_{pp}^{-1}\sigma\tau\right)^{-1}\left(\phi_p^n + \sum_{i=1}^{p-1}A_{pi}\sigma\tau B_{ii}^{-1}\tilde{\psi}_i^{n+1}\right), \tag{3.35}$$

while $\phi^n = (1 - \sigma)\tau Au^n + Bu^n + f(\sigma t^{n+1} + (1 - \sigma)t^n)$.

(2) Compute $u^{n+1} = (u_1^{n+1}, \ldots, u_p^{n+1})^T$ with $\tilde{\psi}^{n+1} = (\tilde{\psi}_1^{n+1}, \ldots, \tilde{\psi}_p^{n+1})^T$

$$u_p^{n+1} = \left(B_{pp} - \frac{1}{2}A_{pp}\sigma\tau\right)^{-1}\tilde{\psi}_p^n \tag{3.36}$$

$$u_{p-1}^{n+1} = \left(B_{p-1p-1} - \frac{1}{2}A_{p-1p-1}\sigma\tau\right)^{-1}\left(\tilde{\psi}_{p-1}^n + A_{p-1p}\sigma\tau u_p^{n+1}\right) \tag{3.37}$$

$$\ldots \tag{3.38}$$

$$u_1^{n+1} = \left(B_{11} - \frac{1}{2}A_{11}\sigma\tau\right)^{-1}\left(\tilde{\psi}_1^n + \sum_{i=2}^{p}A_{1i}\sigma\tau u_i^{n+1}\right). \tag{3.39}$$

Theorem 3.2 *If we choose $\sigma \geq \frac{1}{2}$, then the splitting scheme (3.30) and (3.31) is absolutely stable in an appropriate Hilbert space.*

Proof The outline of the proof is given in [6].

Example 3.1 We have $2n \times 2n$ matrices.

The algorithm is as follows:

(1) Compute $\tilde{\psi}^{n+1} = (\tilde{\psi}_1^{n+1}, \tilde{\psi}_2^{n+1})^T$ with $\phi^n = (\phi_1^n, \phi_2^n)^T$

$$\tilde{\psi}_1^{n+1} = \left(I - \frac{1}{2}A_{11}B_{11}^{-1}\sigma\tau\right)^{-1}\phi_1^n \tag{3.40}$$

$$\tilde{\psi}_2^{n+1} = \left(I - \frac{1}{2}A_{22}B_{22}^{-1}\sigma\tau\right)^{-1}\left(\phi_2^n + A_{21}\sigma\tau B_{11}^{-1}\tilde{\psi}_1^{n+1}\right), \tag{3.41}$$

while $\phi^n = (1-\sigma)\tau Au^n + Bu^n + f(\sigma t^{n+1} + (1-\sigma)t^n)$.

(2) Compute $u^{n+1} = (u_1^{n+1}, u_2^{n+1})^T$ with $\tilde{\psi}^{n+1} = (\tilde{\psi}_1^{n+1}, \tilde{\psi}_2^{n+1})^T$

$$u_2^{n+1} = \left(B_{22} - \frac{1}{2}A_{22}\sigma\tau\right)^{-1}\tilde{\psi}_2^n \tag{3.42}$$

$$u_1^{n+1} = \left(B_{11} - \frac{1}{2}A_{11}\sigma\tau\right)^{-1}\left(\tilde{\psi}_1^n + A_{12}\sigma\tau u_2^{n+1}\right), \tag{3.43}$$

3.3.2 Higher Order Additive Splitting Method

The drawback of the standard additive splitting method is the restriction to a second-order scheme.

To overcome this limitation, an extension can be made in the direction of the higher order Crank–Nicolson scheme, see [7].

The higher order Crank–Nicolson method can be derived as follows (see also [7]):

$$\begin{aligned} u(t^{n+1}) &= u(t^n) + h\frac{du}{dt}(t^n) \\ &\quad + \frac{h^2}{2!}\frac{d^2u}{dt^2}(t^n) + \frac{h^3}{3!}\frac{d^3u}{dt^3}(t^n) + \frac{h^4}{4!}\frac{d^4u}{dt^4}(t^n)\ldots, \end{aligned} \tag{3.44}$$

$$\begin{aligned} u(t^n) &= u(t^{n+1}) - h\frac{du}{dt}(t^{n+1}) \\ &\quad + \frac{h^2}{2!}\frac{d^2u}{dt^2}(t^{n+1}) - \frac{h^3}{3!}\frac{d^3u}{dt^3}(t^{n+1}) + \frac{h^4}{4!}\frac{d^4u}{dt^4}(t^{n+1})\ldots, \end{aligned} \tag{3.45}$$

subtracting the two equations and applying it to Eq. (5.534) in the form $\partial_t u = B^{-1}Au = \tilde{A}u$,

$$\begin{aligned} u(t^{n+1}) - u(t^n) &= \frac{h}{2}\left(\tilde{A}u(t^{n+1}) + \tilde{A}u(t^n)\right) \\ &+ \frac{h^2}{2\,2!}\left(-\tilde{A}^2u(t^{n+1}) + \tilde{A}^2u(t^n)\right) \\ &+ \frac{h^3}{2\,3!}\left(\tilde{A}^3u(t^{n+1}) + \tilde{A}^3u(t^n)\right)\ldots, \end{aligned} \tag{3.46}$$

we obtain

$$\left(I - \frac{h}{2}\tilde{A} + \frac{h^2}{2\,2!}\tilde{A}^2\right)u(t^{n+1}) = \left(I + \frac{h}{2}\tilde{A} + \frac{h^2}{2\,2!}\tilde{A}^2\right)u(t^n), \tag{3.47}$$

which is a third-order scheme.

The same can be obtained by the fractional step scheme

$$\begin{aligned} &\left(I - \sigma h\tilde{A} + \frac{\sigma\, h^2}{2!}\tilde{A}^2\right)u(t^{n+1}) \\ &= \left(I + (1-\sigma)\, h\tilde{A} + \frac{(1-\sigma)\, h^2}{2!}\tilde{A}^2\right)u(t^n), \end{aligned} \tag{3.48}$$

which is a third-order scheme for $\sigma = \frac{1}{2}$.

There is a decomposition idea based on a splitting into tridiagonal matrices.

The higher order additive splitting algorithm is given in the following scheme:

$$\begin{aligned} &\left(I - \sigma h\tilde{A} + \sigma\frac{h^2}{2!}\tilde{A}^2\right) \\ &= \left(I - \sigma h\tilde{A}_1 + \sigma\frac{h^2}{2!}\tilde{A}_1^2\right) \\ &\cdot\left(I - \sigma h\tilde{A}_2 + \sigma\frac{h^2}{2!}\tilde{A}_2^2\right) + \sigma\frac{h^2}{2!}[\tilde{A}_2, \tilde{A}_1] + O(h^3), \end{aligned} \tag{3.49}$$

where $\tilde{A} = B^{-1}A$ and $A = A_1 + A_2$ where $A_1 = A_2^t$.
The commutator is $[A_2, A_1] = A_2A_1 - A_1A_2$.

By the transition to a new time level, we require

$$\left(I - \sigma h\tilde{A}_1 + \sigma\frac{h^2}{2!}\tilde{A}_1^2\right)\left(I - \sigma h\tilde{A}_2 + \sigma\frac{h^2}{2!}\tilde{A}_2^2\right)u(t^{n+1}) = \phi^n, \tag{3.50}$$

where $\phi^n = \left(I + (1-\sigma)\, h\tilde{A} + (1-\sigma)\frac{h^2}{2!}\tilde{A}^2 - \sigma\frac{h^2}{2!}[\tilde{A}_2, \tilde{A}_1]\right)u(t^n)$.

We have to solve the following pair of linear equations:

$$\left(I - \sigma h \tilde{A}_1 + \sigma \frac{h^2}{2!} \tilde{A}_1^2\right) \psi^{n+1} = \phi^n, \tag{3.51}$$

$$\left(I - \sigma h \tilde{A}_2 + \sigma \frac{h^2}{2!} \tilde{A}_2^2\right) u^{n+1} = \psi^{n+1}, \tag{3.52}$$

where the ψ^{n+1} are the intermediate solutions of the scheme.

The third-order algorithm is given as a three-step method, presented in the following Algorithm 3.3.

Algorithm 3.3 (1) Compute $\psi^{n+1} = (\psi_1^{n+1}, \ldots, \psi_p^{n+1})^T$ with $\phi^n = (\phi_1^{n+1}, \ldots, \phi_p^{n+1})^T$

$$\psi_1^{n+1} = \left(I - \frac{1}{2} A_{11} \sigma h + \left(\frac{1}{2} A_{11}\right)^2 \sigma \frac{h^2}{2!}\right)^{-1} \phi_1^n \tag{3.53}$$

$$\psi_2^{n+1} = \left(I - \frac{1}{2} A_{22} \sigma h + \left(\frac{1}{2} A_{22}\right)^2 \sigma \frac{h^2}{2!}\right)^{-1} \left(\phi_2^n + \left(A_{21} \sigma h - \{A_1 A_1\}_{21}\, \sigma \frac{h^2}{2!}\right) \psi_1^{n+1}\right) \tag{3.54}$$

$$\ldots \tag{3.55}$$

$$\psi_p^{n+1} = \left(I - \frac{1}{2} A_{pp} \sigma h + \left(\frac{1}{2} A_{pp}\right)^2 \sigma \frac{h^2}{2!}\right)^{-1} \left(\phi_p^n + \sum_{i=1}^{p-1} \left(A_{pi} \sigma h - \{A_1 A_1\}_{pi}\, \sigma \frac{h^2}{2!}\right) \psi_i^{n+1}\right), \tag{3.56}$$

while $\phi^n = \left(I + (1-\sigma)\, hA + (1-\sigma) \frac{h^2}{2!} A^2 - \sigma \frac{h^2}{2!} [A_2, A_1]\right) u(t^n)$ and the matrix multiplication $\{A_1 A_1\}_{ij} = \sum_{k=1}^{p} A_{1,ik} A_{1,kj}$, where p is the rank of the matrix A_1 and $A_{1,ij}$ is the i, jth element of the matrix A_1.

(2) Compute $u^{n+1} = (u_1^{n+1}, \ldots, u_p^{n+1})^T$ with $\psi^{n+1} = (\psi_1^{n+1}, \ldots, \psi_p^{n+1})^T$

$$u_p^{n+1} = \left(I - \frac{1}{2} A_{pp} \sigma h + \left(\frac{1}{2} A_{pp}\right)^2 \sigma \frac{h^2}{2!}\right)^{-1} \psi_p^n \tag{3.57}$$

$$u_{p-1}^{n+1} = \left(I - \frac{1}{2} A_{p-1p-1} \sigma h + \left(\frac{1}{2} A_{p-1p-1}\right)^2 \sigma \frac{h^2}{2!}\right)^{-1} \cdot \left(\psi_{p-1}^n + \left(A_{p-1p} \sigma h - \{A_2 A_2\}_{p-1p}\, \sigma \frac{h^2}{2!}\right) u_p^{n+1}\right) \tag{3.58}$$

$$\ldots \tag{3.59}$$

$$u_1^{n+1} = \left(I - \frac{1}{2} A_{11} \sigma \frac{h}{2} + \left(\frac{1}{2} A_{11}\right)^2 \sigma \frac{h^2}{2!}\right)^{-1} \cdot \left(\psi_1^n + \sum_{i=2}^{p} \left(A_{1i} \sigma h - \{A_2 A_2\}_{1i}\, \sigma \frac{h^2}{2!}\right) u_i^{n+1}\right), \tag{3.60}$$

and the matrix multiplication $\{A_2 A_2\}_{ij} = \sum_{k=1}^{p} A_{2,ik} A_{2,kj}$, where p is the rank of the matrix A_2 and $A_{2,ij}$ is the i, jth element of the matrix A_2.

Theorem 3.4 *If we choose $\sigma \geq \frac{1}{2}$, then the splitting scheme (3.51) and (3.52) is absolutely stable in an appropriate Hilbert space.*

Proof The outline of the proof is given in [6].

3.3.3 Iterative Splitting Method

The following algorithm is based on an iteration with a 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. [8, 9]):

$$\frac{\partial c_i(t)}{\partial t} = A_1 c_i(t) + A_2 c_{i-1}(t), \text{ with } c_i(t^n) = c^n \tag{3.61}$$

$$\text{and } c_0(t^n) = c^n, \quad c_{-1} = 0.0,$$

$$\frac{\partial c_{i+1}(t)}{\partial t} = A_1 c_i(t) + A_2 c_{i+1}(t), \tag{3.62}$$

$$\text{with } c_{i+1}(t^n) = c^n,$$

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 analyse the convergence and the rate of convergence of the method (3.61) and (3.62) as m tends to infinity for the linear operators $A_1, A_2 : \mathbf{X} \to \mathbf{X}$, where we assume that these operators and their sum are generators of C_0 semi-groups. 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 that we have bounded integral operators which can be estimated and given in a recursive form. Such formulations are known in the work of [10, 11], and estimations of the kernel part with the exponential operators are sufficient to estimate the recursive formulations.

3.4 Parallelization

The parallelization is important to accelerate the solver methods.

We distinguish between three different parallelization areas:

- Parallelization in Time, e.g. Parareal method: Decomposition of large time intervals to smaller time intervals
- Parallelization in Operators, e.g. Parallel operator splitting method
- Parallelization in Space, e.g. Schwartz waveform relaxation, Domain decomposition algorithms

The application of the different parallel methods are discussed in the following:

- Time parallelization: The large time interval is decomposed into smaller time intervals (time decomposition). The full equations can be handled in one processor, such that the memory effect is not too important. But the duration of the full time interval is very large such that it will take too long for one processor. Therefore, we decompose it to smaller time intervals and parallelize the large time interval, i.e. each time slot can be handled independently by one processor, see [12].
- Operator splitting methods or parallelization of the different operators: The problem is based on storing the full operator of the differential equation in one processor (this was the motivation of the earliest splitting schemes [13]). Therefore, we decompose the full operator into simpler operators and distribute the simpler operators, which can be stored into one processor, to various processors. The operators are coupled via the operator splitting scheme and can be computed in parallel. Such ideas allow to deal with modular coupling of program codes, e.g. different specialized codes for an E- and B-field (e.g. Maxwell equation) and a transport field (e.g. particle code), which can be computed on different PC clusters.

Example 3.2 Reduction of the computational time via time parallelization.

We assume to have an effective parallel algorithm with about 20–50 %, see [12].

Therefore, we reduce the computational time for one processor, for example, of 48 [h], with 128 processors and an efficiency of about 20 % to 2–3 [h].

3.4.1 Time Parallelization: Parareal Algorithm as an Iterative Solver

The original algorithm was introduced by [14]. The idea is to partition the time domain $\Omega_N = [0, T]$, which is large, into N time subdomains:

$$\Omega_n \quad [T_{n-1}, T_n], n = 1, \ldots, N, \tag{3.63}$$

furthermore, we define the following solvers:

- coarse solver (coarse propagator): $G(T_n, T_{n-1}, x)$ and
- fine solver (fine propagator): $F(T_n, T_{n-1}, x)$

with both, we can approximate the underlying differential equation:

$$U'(t) = f(t, U(t)), U(T_{n-1}) = x. \tag{3.64}$$

Here, we assume the following:

1. The coarse integrator is computationally much faster, i.e. a lower order scheme, than the fine integrator.
2. The fine integrator is much more accurate, i.e. a higher order scheme, and much more time consuming, and therefore we need the benefit of parallelization.

We have the following steps:

- In the first iteration, we use the coarse integrator in a serial fashion to provide initial conditions to each time slice Ω_n:

$$U_n^1 = G(T_n, T_{n-1}, U_{n-1}^1), \quad n = 1, 2, \ldots, N.$$

- In the second step, we use the fine propagator and integrate independently (i.e. in parallel) N initial value problems $F(T_n, T_{n-1}, U_{n-1}^k)$ $(n = 1, 2, \ldots, N)$, yielding new approximations for the initial conditions on the following time slices.
- In each iteration k, the corrections are then again quickly propagated using the coarse integrator:

$$U_n^{k+1} = F(T_n, T_{n-1}, U_{n-1}^k) + G(T_n, T_{n-1}, U_{n-1}^{k+1}) - G(T_n, T_{n-1}, U_{n-1}^k), \tag{3.65}$$

Example 3.3 We deal with a differential equation,

$$U' = AU + BU, U(0) = u(0), \tag{3.66}$$

with two operators A and B.

We assume to have F as a fine integrator and choose the iterative splitting method as a more accurate propagator. Further, we assume to have G as a coarse integrator and choose the A–B splitting scheme as a lower order accurate propagator, see the example in [15].

The method can be compared be the so-called *Multiple Shooting Method*, see [16]. While we repeat each time slot with an improved approximation and if the error is small enough, we go on to the next time intervals, see also Fig. 3.1.

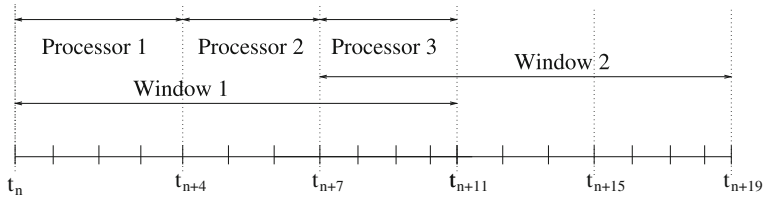

Fig. 3.1 Parallelization with Parareal, windowing of the parallel process

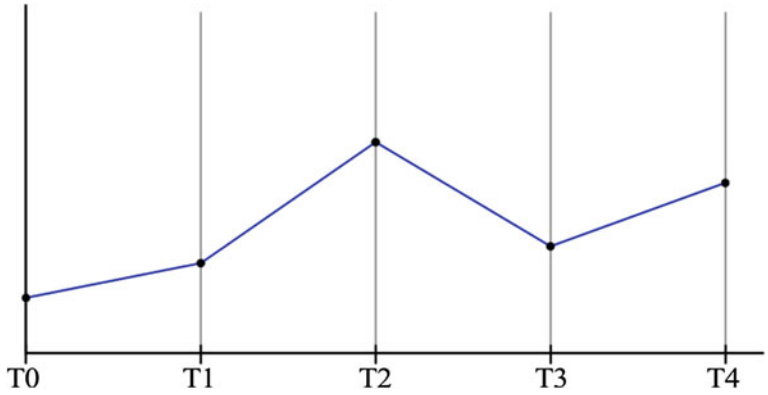

Fig. 3.2 First initialization step in the algorithm

In the following, we discuss the different steps.

Step 1:

Coarse computation of one processor of the full interval with a fast-and low-order solver method, e.g. forward Euler scheme, see Fig. 3.2.

We propagate in the coarse method with

$$U_0^1 = u_0, \tag{3.67}$$

$$U_n^1 = G(T_n, T_{n-1}, U_{n-1}^1), n = 1, \ldots, N. \tag{3.68}$$

Step 2:

The next step is a fine propagator with n-processors, for each smaller time interval. The methods for the smaller time intervals are of higher order and expensive in time, see Fig. 3.3.

We propagate with the fine propagator, while the initial conditions are given for each subdomain Ω_n:

$$U_{fine,n}^1 = F\left(T_n, T_{n-1}, U_{n-1}^1\right), n = 1, \ldots, N. \tag{3.69}$$

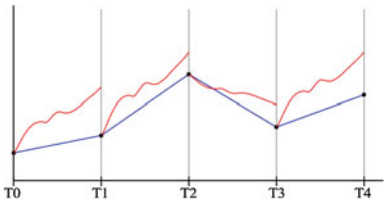

Fig. 3.3 Second step of a fine propagator step done in parallel

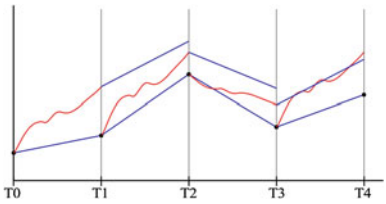

Fig. 3.4 The third step is the corrector step, which coupled the coarse and fine step and go on with the initialization of the next timeframe

Step 3:

The next step is the corrector step to couple the coarse and fine steps together (coupling process). One processor computes the corrections between each time interval. Such time intervals which fit of the accuracy are finished and we step forward. The other intervals are computed via the first step and so on, see Fig. 3.4.

We apply the improved initial guess and propagate coarsely in a correction:

$$U_n^{k+1} = F\left(T_n, T_{n-1}, U_{n-1}^k\right) + G\left(T_n, T_{n-1}, U_{n-1}^{k+1}\right) - G\left(T_n, T_{n-1}, U_{n-1}^k\right). \quad (3.70)$$

If the error in the time slot is sufficient small, we shift the window to the next time slot and start with the step 1, till we are done. Otherwise, we go on with the next iterative step $k = k + 1$.

3.4.2 Operator Parallelization: Operator Splitting Method

We deal with large operators in a differential equation, which is given as

$$\frac{dc(t)}{dt} = A_{\text{full}} c(t), \quad \text{for } t \in (t^n, T), \quad (3.71)$$

$$\frac{dc(t)}{dt} = \sum_{i=1}^{m} A_i \; c(t), \quad \text{for } t \in (t^n, T), \quad (3.72)$$

$$c(0) = c_0, \; \text{Initial-conditions}, \quad (3.73)$$

where we have the time intervals $t_1, t_2, \ldots, t_N$.

We assume that the full operator A_{full} is partitioned into different smaller operators A_j, $j = 1, \ldots, m$. Furthermore, we have an appropriate Banach space with a vector and induced matrix norm $||\cdot||$, where $c \in \mathbf{X}$ and also the operators are given in $A_j \in \mathbf{X}^2$ for $j = 1, \ldots, m$.

We also assume that the operators include the boundary conditions and are derived of semi-discretizations, e.g. Finite Difference or Finite Element Methods. Based on their problems, they might have different physical behaviours, e.g. diffusion, convection or reaction operators, if we deal with a fluid flow problem, see [17].

We deal with the following problems:

- The full operator A_{full} cannot be stored into one processor, and therefore we have to partition the problem to A_j, $j = 1, \ldots, m$ smaller operators.
- The physical problem allows to deal with different program codes, e.g. we have a code for the diffusion problem, a code for the reaction problem and so on. We only like to couple such problems via the splitting approach.

3.4.3 Sequential Operator Splitting Method

Such a scheme can be applied to couple the different operators to the full operator equations (3.71). We deal with a successive computation of each operator in each time slot and couple via the initial conditions of each step, see [18].

We solve m subproblem sequentially on the subintervals $[t^n, t^{n+1}]$, where $n = 0, 1, \ldots, N-1$, $t^0 = 0$ and $t^N = T$.

The subproblems are given in the following and coupled via the initial conditions:

$$\frac{\partial c_1(t)}{\partial t} = A_1 c_1(t), \quad \text{with } c_1(t^n) = c(t^n), \tag{3.74}$$

$$\frac{\partial c_2(t)}{\partial t} = A_2 c_2(t), \quad \text{with } c_2(t^n) = c_1(t^{n+1}), \tag{3.75}$$

$$\vdots \tag{3.76}$$

$$\frac{\partial c_m(t)}{\partial t} = A_m c_m(t), \quad \text{with } c_m(t^n) = c_{m-1}(t^{n+1}), \tag{3.77}$$

for $n = 0, 1, \ldots, N-1$ and $\tau = t^{n+1} - t^n$, where $c(t^{n+1}) = c_m(t^{m+1})$ is the approximated solution at the time point t^{n+1}.

The local splitting error of the sequential scheme is $\mathcal{O}(\tau^2)$ and the global splitting error of the sequential scheme is $\mathcal{O}(\tau)$, if we assume non-commutable operators. Otherwise, we are exact.

Here, we have to wait for the next initial condition, while we are dependent on the result of the previous step, such that the scheme is only interested to couple different codes, see [18].

3.4.4 Parallel Operator Splitting Method: Version 1

The following first parallel operator splitting method is also called splitting-up method, see [19, 20].

We also deal with m sub-problems, which can be solved independently, i.e. parallel, while the initial conditions are given at time point t^n for each sub-problem and independent of other sub-problems, see [21].

We have the subintervals $[t^n, t^{n+1}]$, where $n = 0, 1, \ldots, N-1$, $t^0 = 0$ and $t^N = T$. We deal with m parallel sub-problem given as follows:

$$\frac{\partial c_1(t)}{\partial t} = A_1 c_1(t), \quad \text{with } c_1(t^n) = c(t^n), \tag{3.78}$$

$$\frac{\partial c_2(t)}{\partial t} = A_2 c_2(t), \quad \text{with } c_2(t^n) = c(t^n), \tag{3.79}$$

$$\vdots \tag{3.80}$$

$$\frac{\partial c_m(t)}{\partial t} = A_m c_m(t), \quad \text{with } c_m(t^n) = c(t^n), \tag{3.81}$$

and result in one additive step that couples the independent sub-steps:

$$c(t^{n+1}) = c(t^n) + \sum_{i=1}^{m} \Big(c_i(t^{n+1}) - c(t^n) \Big),$$
$$n = 1, 2, \ldots, N, \quad \text{where } c(0) = c_0.$$

The local splitting error of the parallel scheme is $\mathcal{O}(\tau)$ if we deal with non-commutable operators. Otherwise, we are exact.

Based on the low-order scheme, we introduce in the following a second-order scheme, which can also be applied in parallel, see [21].

3.4.5 Parallel Operator-Splitting Method: Version 2

The following second parallel operator-splitting method is also weighted sequential splitting method, see [21].

We obtain a second-order scheme, like the Strang splitting scheme, see [13], while we apply sequential splitting in both directions, i.e., $A_1 \to A_2 \to \cdots, \to A_m$ and $A_m \to A_{m-1} \to \cdots, \to A_1$.

We also deal with two sequential splitting problems, which can be handled parallel, while each splitting problem has m sub-problems. These sub-problems are dependent and are done sequentially, see [21].

We have the two independent m sequential problems in the subintervals $[t^n, t^{n+1}]$, where $n = 0, 1, \ldots, N-1$, $t^0 = 0$ and $t^N = T$.

We deal with the first m sequential sub-problem given as

$$\frac{\partial c_1(t)}{\partial t} = A_1 c_1(t), \quad \text{with } c_1(t^n) = c(t^n), \tag{3.82}$$

$$\frac{\partial c_2(t)}{\partial t} = A_2 c_2(t), \quad \text{with } c_2(t^n) = c_1(t^{n+1}), \tag{3.83}$$

$$\vdots \tag{3.84}$$

$$\frac{\partial c_m(t)}{\partial t} = A_m c_m(t), \quad \text{with } c_m(t^n) = c_{m-1}(t^{n+1}), \tag{3.85}$$

and the second m sequential sub-problem given as

$$\frac{\partial v_1(t)}{\partial t} = A_m v_1(t), \quad \text{with } v_1(t^n) = c(t^n), \tag{3.86}$$

$$\frac{\partial v_2(t)}{\partial t} = A_{m-1} v_2(t), \quad \text{with } v_2(t^n) = v_1(t^{n+1}), \tag{3.87}$$

$$\vdots \tag{3.88}$$

$$\frac{\partial v_m(t)}{\partial t} = A_1 v_m(t), \quad \text{with } v_m(t^n) = v_{m-1}(t^{n+1}). \tag{3.89}$$

We result in one additive step that couples the independent sub-problems:

$$c(t^{n+1}) = \frac{1}{2} c_m(t^{n+1}) + \frac{1}{2} v_m(t^{n+1}).$$

The local splitting error of the parallel scheme is $\mathcal{O}(\tau^2)$, and the global splitting error of the parallel scheme is $\mathcal{O}(\tau)$, if we deal with non-commutable operators. Otherwise, we are exact.

In the next subsection, we present an iterative splitting scheme, which deals with a parallelization of the exp-operators.

3.4.6 Iterative Splitting Scheme

The iterative splitting scheme is based on a relaxation idea, see [18].

Here, we deal with exp-operators, which can also be applied independently. Based on the idea to relax only to the so-called dominant operators, see [18], we only apply multiplications via the non-dominant operators, see [22].

We assume that we are partitioned into two operators and deal with the following algorithm. The time interval is given as $[t^n, t^{n+1}]$ and we solve the following sub-problems with the iterative steps $i = 1, 2, \ldots, I$:

$$\frac{dc_i(t)}{dt} = Ac_i(t) \; + \; Bc_{i-1}(t), \; \text{with} \; c_i(t^n) = c_{\text{sp}}^n, \tag{3.90}$$

where $c_0(t)$ is an initialization for the iterative scheme, e.g. $c_0(t) = 0$.

The iterative schemes are solved in the following manner:

$$c_1(t) = \exp(At)c(t^n), \tag{3.91}$$

$$c_2(t) = c_1(t) + c_1(t) \int_0^t [B, \exp(sA)]ds, \tag{3.92}$$

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

Based on the $\exp(At)$ operators, we can decouple into A and B dependent terms, see [22].

Remark 3.3 The iterative splitting schemes have the benefit of their modularization, i.e. we could add relaxed operators to the scheme. A drawback is the strong coupling in each iterative step, which means that the parallelization is more delicate, and compare also the waveform relaxation methods with Jacobian or Gauss–Seidel Schemes, see [4].

3.4.7 Spatial Parallelization Techniques

Domain Decomposition

Traditional domain decomposition schemes, e.g. Schwarz waveform relaxation schemes, motivate with a different idea to decompose the domains into subdomains, such that the operator is only defined in subdomains.

Example 3.4 We start to decompose the operator A into operators defined at each subdomain. We have $\overline{\Omega} = \overline{\Omega_1} \cup \overline{\Omega_2}$; here, we obtain an *artificial* boundary with $\Omega_1 \cap \Omega_2$, which is not considered in an iterative operator splitting scheme.

The main advantage of such decomposition is the two decoupled independent equations on each domain:

$$A|_{\Omega_1} u|_{\Omega_1} = f|_{\Omega_1} \tag{3.93}$$

$$A|_{\Omega_2} u|_{\Omega_2} = f|_{\Omega_2} \tag{3.94}$$

where we assume that the boundary condition at the boundary $\partial\Omega \cap \Omega_1$ is included in A_1 and the boundary condition $\partial\Omega \cap \Omega_2$ is included in A_2.

To couple the two separate equations, we have to apply waveform relaxation methods with the *artificial* boundary condition, which can be given as

$$A|_{\Omega_1} u_i|_{\Omega_1} = f|_{\Omega_1} \tag{3.95}$$

$$B|_{\Omega_1 \cap \Omega_2} u_i|_{\Omega_1 \cap \Omega_2} = B|_{\Omega_2 \cap \Omega_1} \hat{u}_{i-1}|_{\Omega_1 \cap \Omega_2} \tag{3.96}$$

$$A|_{\Omega_2} u_i|_{\Omega_2} = f|_{\Omega_2} \tag{3.97}$$

$$B|_{\Omega_2 \cap \Omega_1} \hat{u}_i|_{\Omega_1 \cap \Omega_2} = B|_{\Omega_1 \cap \Omega_2} u_i|_{\Omega_1 \cap \Omega_2}, \tag{3.98}$$

where $i = 1, 2, \ldots, I$ and we start from an initial guess of $u_0|_{\Omega_2}$.

Here, we iterate via the two decoupled equations and achieve $u_i|_{\Omega_1 \cap \Omega_2} = \hat{u}_i|_{\Omega_1 \cap \Omega_2}$ for i sufficient large.

At least we have to *double* the variables at the *artificial* boundary.

3.4.7.1 Domain Decomposition Methods: Discussion

The motivation of domain decomposition methods arose to the fact of decomposing into smaller and simpler calculatable domains. We want to apply a standard solver code for each domain, based on the same model equations, see [23].

We can classify the following techniques:

- Non-iterative methods, e.g. FETI methods, Mortar element methods, [24, 25],
- Iterative methods, e.g. Schwarz waveform relaxation methods, see [26].

3.4.7.2 Iterative Method: Schwarz Waveform Relaxation Method

The Schwarz waveform relaxation method deals with the idea to iterate over the decoupled domains. The model equation, the decomposition methods and the underlying software codes are discussed in the manuscript [27].

We deal with the following equations as

$$-\frac{\partial^2 u}{\partial x^2} + \eta u = f, \text{ in } \Omega = [0,1], \tag{3.99}$$

$$u(0) = g_g, \; u(1) = g_d, \tag{3.100}$$

and then separate them into the following equations and apply the iterative steps:

$$-\frac{\partial^2 u_1^{n+1}}{\partial x^2} + \eta u_1^{n+1} = f, \text{ in } \Omega_1 = [0,\beta], \tag{3.101}$$

$$u_1^{n+1}(0) = g_g,$$

$$u_1^{n+1}(\beta) = u_2^n(\beta),$$

$$-\frac{\partial^2 u_2^{n+1}}{\partial x^2} + \eta u_2^{n+1} = f, \text{ in } \Omega_2 = [\alpha,1], \tag{3.102}$$

$$u_2^{n+1}(0) = g_d,$$

$$u_2^{n+1}(\alpha) = u_1^{n+1}(\alpha).$$

We can deal with the different ideas to partition the domains, e.g. overlapping or non-overlapping, see Fig. 3.5.

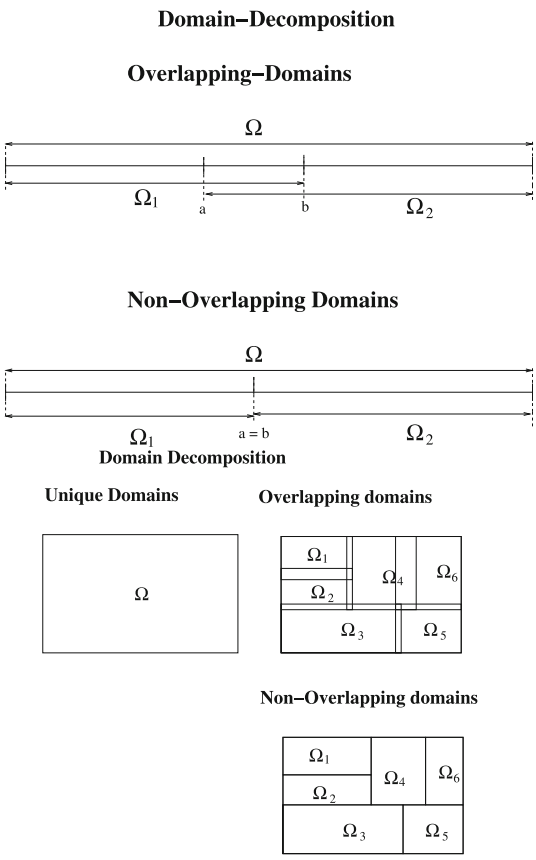

Fig. 3.5 Domain decomposition with respect to overlapping and non-overlapping domains

Further, we discretize our decomposed equation and we deal with the following discretized equations as

$$-\frac{u_{j+1}-2u_j+u_{j-1}}{h^2}+\eta u_j=f_j,\ 1\le j\le J, \tag{3.103}$$

and then decompose into

$$-\frac{(u_1^{n+1})_{j+1}-2(u_1^{n+1})_j+(u_1^{n+1})_{j-1}}{h^2}+\eta(u_1^{n+1})_j=f_j,$$
$$1\le j\le b-1,\ (u_1^{n+1})_b=(u_2^n)_b, \tag{3.104}$$

$$-\frac{(u_2^{n+1})_{j+1}-2(u_2^{n+1})_j+(u_2^{n+1})_{j-1}}{h^2}+\eta(u_2^{n+1})_j=f_j,$$
$$a+1\le j\le J,\ (u_2^{n+1})_a=(u_1^{n+1})_a. \tag{3.105}$$

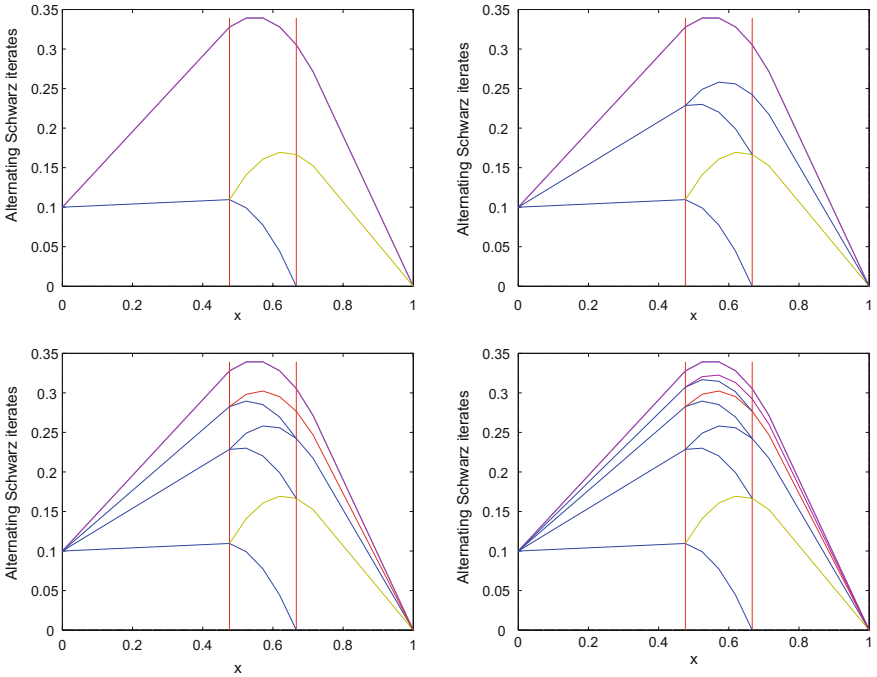

Fig. 3.6 We start with the initialization (0th iteration, *upper left* figure), then we have the first iteration (*upper right* figure), then we conclude with the second iteration (*lower left* figure) and at least the third iteration (*lower right* figure) of the Schwartz waveform relaxation method

Then, we have the following Schwartz waveform iterative steps, where the exact solution of the decomposed equation is also in the different iterative solutions. The iterative steps are given in Fig. 3.6.

Remark 3.4 Some ideas and motivations for using non-iterative or iterative domain decomposition methods are discussed below:

1. Iterative Method:
    a. Benefit: Simple to implement, and an inversion of a matrix is not necessary,
    b. Drawback: We deal with an iteration method, i.e. we need relaxation-steps to obtain the *correct* solution (additional time).
2. Non-iterative Method:
    a. Benefit: We obtain a direct solution, we have only one step.
    b. Drawback: Often delicate solver methods for the solutions are needed, e.g., Schur-complement methods for the coupled matrices (to solve such systems, it is necessary to apply iterative solvers).

Remark 3.5 The benefits and drawbacks of the overlapping or non-overlapping decomposition methods are discussed below. Here are some ideas:

1. Overlapping Domain Decomposition:
   a. Benefit: Stronger coupling of the equation-parts, i.e. we achieve more stable methods, which converge faster.
   b. Drawback: Higher computational amount and more delicate to parallelize.
2. Non-overlapping Domain Decomposition:
   a. Benefit: Simpler to parallelize (it is stronger decoupled).
   b. Drawback: Solver amount is higher, while we need additional iterative steps to couple the equation parts. Often, we have a slower convergence of the method.

In the following Example 3.5, we explain an application of the Schwarz waveform relaxation method.

Example 3.5 **Application to Convection-Diffusion-Reaction Equations**

The example is given in the author's paper [28]. Here, we conclude with some ideas and aspects of the decomposition method.

We consider the convection-diffusion-reaction equation, given by

$$u_t = Du_{xx} - vu_x - \lambda u, \tag{3.106}$$

defined on the domain $\Omega \times T$, where $\Omega = [0, L]$ and $T = [T_0, T_f]$, with the following boundary and initial conditions:

$$u(0,t) = f_1(t), \quad u(L,t) = f_2(t), \quad u(x,T_0) = u_0.$$

To solve the model problem using overlapping Schwarz waveform relaxation method, we subdivide the domain Ω into two overlapping subdomains $\Omega_1 = [0, L_2]$ and $\Omega_2 = [L_1, L]$, where $L_1 < L_2$ and $\Omega_1 \bigcap \Omega_2 = [L_1, L_2]$ are the overlapping regions for Ω_1 and Ω_2, respectively.

To start the waveform relaxation algorithm, we consider first the solution of the model problem (3.106) over Ω_1 and Ω_2 as follows:

$$\begin{aligned} v_t &= Dv_{xx} - vv_x - \lambda v \text{ over } \Omega_1, \quad t \in [T_0, T_f] \\ v(0,t) &= f_1(t), \quad t \in [T_0, T_f] \\ v(L_2,t) &= w(L_2,t), \quad t \in [T_0, T_f] \\ v(x,T_0) &= u_0 \quad x \in \Omega_1, \end{aligned} \tag{3.107}$$

$$\begin{aligned} w_t &= Dw_{xx} - vw_x - \lambda w \text{ over } \Omega_2, \quad t \in [T_0, T_f] \\ w(L_1,t) &= v(L_1,t), \quad t \in [T_0, T_f] \\ w(L,t) &= f_2(t), \quad t \in [T_0, T_f] \\ w(x,T_0) &= u_0 \quad x \in \Omega_2, \end{aligned} \tag{3.108}$$

where $v(x,t) = u(x,t)|_{\Omega_1}$ and $w(x,t) = u(x,t)|_{\Omega_2}$.

Then the Schwarz waveform relaxation is given by

$$\begin{aligned} v_t^{k+1} &= Dv_{xx}^{k+1} - \nu v_x^{k+1} - \lambda v^{k+1} \text{ over } \Omega_1, \quad t \in [T_0, T_f] \\ v^{k+1}(0,t) &= f_1(t), \quad t \in [T_0, T_f] \\ v^{k+1}(L_2,t) &= w^k(L_2,t), \quad t \in [T_0, T_f] \\ v^{k+1}(x,T_0) &= u_0 \quad x \in \Omega_1, \end{aligned} \tag{3.109}$$

$$\begin{aligned} w_t^{k+1} &= Dw_{xx}^{k+1} - \nu w_x^{k+1} - \lambda w^{k+1} \text{ over } \Omega_2, \quad t \in [T_0, T_f] \\ w^{k+1}(L_1,t) &= v^k(L_1,t), \quad t \in [T_0, T_f] \\ w^{k+1}(L,t) &= f_2(t), \quad t \in [T_0, T_f] \\ w^{k+1}(x,T_0) &= u_0 \quad x \in \Omega_2. \end{aligned} \tag{3.110}$$

We are interested in estimating the decay of the error of the solution over the overlapping subdomains obtained with the overlapping Schwarz waveform relaxation method over long time interval.

Let us assume that $e^{k+1}(x,t) = u(x,t) - v^{k+1}(x,t)$ and $d^{k+1}(x,t) = u(x,t) - w^{k+1}(x,t)$ are the errors of (3.109) and (3.110) over Ω_1 and Ω_2, respectively. The corresponding differential equations satisfied by $e^{k+1}(x,t)$ and $d^{k+1}(x,t)$ are

$$\begin{aligned} e_t^{k+1} &= De_{xx}^{k+1} - \nu e_x^{k+1} - \lambda e^{k+1} \text{over } \Omega_1, \quad t \in [T_0, T_f] \\ e^{k+1}(0,t) &= 0, \quad t \in [T_0, T_f] \\ e^{k+1}(L_2,t) &= d^k(L_2,t), \quad t \in [T_0, T_f] \\ e^{k+1}(x,T_0) &= 0 \quad x \in \Omega_1, \end{aligned} \tag{3.111}$$

$$\begin{aligned} d_t^{k+1} &= Dd_{xx}^{k+1} - \nu d_x^{k+1} - \lambda d^{k+1} \text{over } \Omega_2, \quad t \in [T_0, T_f] \\ d^{k+1}(L_1,t) &= e^k(L_1,t), \quad t \in [T_0, T_f] \\ d^{k+1}(L,t) &= 0, \quad t \in [T_0, T_f] \\ d^{k+1}(x,T_0) &= 0, \quad x \in \Omega_2. \end{aligned} \tag{3.112}$$

We define for bounded functions $h(x,t) : \Omega \times [T_0, T_f] \to \mathbf{R}$ the norm

$$||h(.,.)||_\infty := \sup_{x \in \Omega, t \in [T_0, T_f]} |h(x,t)|.$$

The theory behind our error estimates is based on the positivity lemma by Pao (or the maximum principle theorem), which is introduced as follows.

Lemma 3.1 *Let $u \in C(\overline{\Omega}_T) \cap C^{1,2}(\Omega_T)$, where $\Omega_T = \Omega \times (0,T]$ and $\partial\Omega_T = \partial\Omega \times (0,T]$, be such that*

$$u_t - D\, u_{xx} + \nu\, u_x + c\, u \geq 0, \quad \textit{in } \Omega_T \tag{3.113}$$

$$\alpha_0\, \partial u \partial \nu + \beta_0\, u \geq 0, \quad \textit{on } \partial\Omega_T \tag{3.114}$$

$$u(x,0) \geq 0, \quad \textit{in } \Omega \tag{3.115}$$

where $\alpha_0 \geq 0$, $\beta_0 \geq 0$, $\alpha_0 + \beta_0 > 0$ *on* $\partial\Omega_T$, *and* $c \equiv c(x,t)$ *is a bounded function in* Ω_T, *Then* $u(x,t) \geq 0$ *in* Ω_T.

The convergence and error estimates of e^{k+1} and d^{k+1} given by (3.111) and (3.112), respectively, are presented in the following theorem.

Theorem 3.5 *Let* e^{k+1} *and* d^{k+1} *be the errors from the solution of the sub-problems (3.107) and (3.108) by Schwarz waveform relaxation over* Ω_1 *and* Ω_2, *respectively, then*

$$||e^{k+2}(L_1,t)||_\infty \leq \gamma ||e^k(L_1,t)||_\infty,$$

and

$$||d^{k+2}(L_2,t)||_\infty \leq \gamma ||d^k(L_1,t)||_\infty,$$

where

$$\gamma = \frac{\sinh(\beta L_1)}{\sinh(\beta L_2)} \frac{\sinh(\beta(L_2 - L))}{\sinh(\beta(L_1 - L))} < 1,$$

with $\beta = \frac{\sqrt{v^2 + 4D\lambda}}{2D}$.

Proof The proof is given in [28].

References

1. K. Dekker, J.G. Verwer, *Stability of Runge-Kutta Methods for Stiff Nonlinear Differential Equations* (North-Holland Elsevier Science Publishers, Amsterdam, 1984)
2. B. Sportisse, An analysis of operator splitting techniques in the stiff case. J. Comput. Phys. **161**, 140–168 (2000)
3. E. Hairer, G. Wanner, *Solving Ordinary Differential Equations II.*, SCM, vol. 14 (Springer, Heidelberg, 1996)
4. S. Vandewalle, *Parallel Multigrid Waveform Relaxation for Parabolic Problems* (B.G. Teubner Stuttgart, Teubner Skripten zur Numerik, 1993)
5. M.J. Gander, A.M. Stuart, Space-time continuous analysis of waveform relaxation for the heat equation. SIAM J. Sci. Comput. **19**(6), 2014–2031 (1998)
6. P. Vabishchevich, Additive schemes (splitting schemes) for some systems of evolutionary equations. Math. Comput. **83**(290), 2787–2797 (2014)
7. E.J. Davison, A high order Crank-Nicholson technique for solving differential equations. Comput. J. **10**(2), 195–197 (1967)
8. R. Glowinski, in *Numerical Methods for Fluids*. Handbook of Numerical Analysis, vol. IX, ed. by P.G. Ciarlet, J. Lions (North-Holland Elsevier, Amsterdam, 2003)
9. J. Kanney, C. Miller, C.T. Kelley, Convergence of iterative split-operator approaches for approximating nonlinear reactive transport problems. Adv. Water Res. **26**, 247–261 (2003)
10. E. Hansen, A. Ostermann, Exponential splitting for unbounded operators. Math. Comput. **78**, 1485–1496 (2009)
11. T. Jahnke, C. Lubich, Error bounds for exponential operator splittings. BIT Numer. Math. **40**(4), 735–745 (2000)
12. M.J. Gander, S. Vanderwalle, Analysis of the parareal time-parallel time-integration method. SIAM J. Sci. Comput. **29**(2), 556–578 (2007)

13. G. Strang, On the construction and comparison of difference schemes. SIAM J. Numer. Anal. **5**, 506–517 (1968)
14. J.L. Lions, Y. Maday, G. Turincini, A "parareal" in time discretization of PDEs. C. R. Acad. Sci. Paris Sér. I Math. **332**, 661–668 (2001)
15. J. Geiser, St. Guettel, Coupling methods for heat-transfer and heat-flow: operator splitting and the parareal algorithm. J. Math. Anal. Appl. **388**(2), 873–887 (2012)
16. H.G. Bock, K.J. Plitt, A multiple shooting algorithm for direct solution of optimal control problems, in *Proceedings 9th IFAC World Congress Budapest* (Pergamon Press, 1984), pp. 243–247
17. J. Geiser, in *Coupled Systems: Theory, Models and Applications in Engineering*. Numerical Analysis and Scientific Computing Series, ed. by F. Magoules, C.H. Lai (CRC Press, Boca Raton, 2014)
18. J. Geiser, in *Iterative Splitting Methods for Differential Equations*. Numerical Analysis and Scientific Computing Series, ed. by F. Magoules, C.H. Lai (Chapman & Hall/CRC, Boca Raton, 2011)
19. A.R. Gourlay, Proc. Conf. Univ. York, *Splitting methods for time-dependent partial differential equations*, The state of art in numerical analysis (Academic Press, London, 1976). Heslington 1997
20. T. Lu, P. Neittaanmaki, X.-C. Tai, A parallel splitting-up method for partial differential equations and its applications to Navier-Stokes equations. RAIRO Model. Math. Anal. Numer. **26**, 673–708 (1992)
21. P. Csomós, I. Faragó, A. Havasi, Weighted sequential splittings and their analysis. Comput. Math. Appl. **50**(7), 1017–1031 (2005)
22. J. Geiser, Computing exponential for iterative splitting methods. J. Appl. Math. Hindawi Publishing Corp., New York, Article ID 193781, 27 pp. (2011)
23. A. Quarteroni, A. Valli, *Domain Decomposition Methods for Partial Differential Equations*, Series: Numerical Mathematics and Scientific Computation (Clarendon Press, Oxford, 1999)
24. A. Toselli, O. Widlund, *Domain Decomposition Methods-Algorithms and Theory*. Springer Series in Computational Mathematics, vol. 34 (Springer, Berlin, 2004)
25. C. Farhat, J. Mandel, F.-X. Roux, Optimal convergence properties of the FETI domain decomposition method. Comput. Methods Appl. Mech. Eng. **115**, 365–385 (1994)
26. M.J. Gander, Y.-L. Jiang, R.-J. Li, Parareal Schwarz Waveform Relaxation methods. *Domain Decomposition Methods in Science and Engineering XX*. Lecture Notes in Computational Science and Engineering, vol. 91 (Springer, New York, 2013), pp. 451–458
27. M.J. Gander, L. Halpern, *Methodes de decomposition de domainess*. Lecture Notes, Section de Mathematiques, Universite de Geneve, Switzerland, 26, April 2012
28. D. Daoud, J. Geiser, Overlapping Schwarz wave form relaxation for the solution of coupled and decoupled system of convection diffusion reaction equation. Appl. Math. Comput. **190**(1), 946–964 (2007)