Waveform Relaxation with Overlapping Splittings

Rolf Jeltsch, Bert Pohl

Research Report No. 91-02
February 1991

Seminar für Angewandte Mathematik
Eidgenössische Technische Hochschule
CH-8092 Zürich
Switzerland

Waveform Relaxation with Overlapping Splittings

Rolf Jeltsch, Bert Pohl
Seminar für Angewandte Mathematik
Eidgenössische Technische Hochschule
CH-8092 Zürich
Switzerland

Research Report No. 91-02 February 1991

Abstract

In this paper we present an extension of the wra for solving large systems of odes. The wra is well suited for parallel computation because it decomposes the solution space into several disjoint subspaces. Allowing the subspaces to overlap, i.e. dropping the assumption of disjointness, we obtain an extension of this algorithm. This new algorithm the so called msa is also well suited for parallel computation. As numerical examples demonstrate this overlapping of the subsystems heavily reduces the computation time.

Keywords: waveform relaxation, multi-splittings, parallel computation

Subject Classification: 65L05, 65F10, 65W05

1. Introduction

In the area of simulation of large electrical circuits the equations describing the circuit often yield a m dimensional nonlinear stiff initial value problem

$$x'(t) = f(t, x(t)) \qquad x(0) = x_0 \tag{1.1}$$

with $t \in [0,T]$, $x \in C^1([0,T]; \mathbb{R}^m)$, $f \in C([0,T], \mathbb{R}^m; \mathbb{R}^m)$, $x_0 \in \mathbb{R}^m$. If one has to simulate a Very Large Scale Integrated (VLSI) circuit the dimension of the problem can reach up to more than 10000 [WhSa87]. Very often the circuit consists of fast and slow components and the fast components force one to use implicit integration methods. Therefore one must solve a m dimensional nonlinear system of equations at each timepoint before advancing to the next timepoint.

In the beginning of the 1980's a new approach for solving these problems was developed at the Electronic Research Laboratory at Berkeley that circumvents these difficulties. In this approach called waveform relaxation algorithm the full system is decomposed into smaller subsystems which can be solved independently by different processors on a parallel computer. The subsystems are integrated over certain small time intervals so called windows. Inputs from other subsystems are taken from the previous iteration. The advantages of this method are obviously not only the possibility to use several processors in parallel and the smaller dimension of the subsystems but also to use different step sizes for different subsystems. Subsystems with slow components only can be integrated with larger step sizes than those containing fast components. A major drawback is the slow convergence of the iteration in case of a strong coupling between the subsystems. Moreover much more memory is needed to store all values of each component at each timepoint of the last iteration.

Multi-splitting methods were first introduced by O'Leary and White in [OLWh85] for solving large linear system of equations on a parallel computer. This idea was extended to nonlinear problems by White in [Whit86]. In this paper we adapt the ideas of O'Leary and White and study the use of multi-splitting methods for solving large systems of ordinary differential equations. It turns out that with a special set of parameters one recovers the waveform relaxation algorithm. We restrict ourselves to linear problems; this means that we are dealing only with linear m-dimensional initial value problems

$$x'(t) + Ax(t) = f(t) \qquad x(0) = x_0 \tag{1.2}$$

with $t \in [0,T]$, $A \in \mathbb{R}^{m \times m}$, $f \in C([0,T]; \mathbb{R}^m)$, $x_0 \in \mathbb{R}^m$, $x \in C^1([0,T]; \mathbb{R}^m)$.

In the second section of this paper we briefly review the usual formulation of the waveform relaxation algorithm. The third section contains the presentation of the multi-splitting algorithm a first analysis of which is given in the fourth section. In the fifth section we discuss a practical implementation of the multi-splitting algorithm. In the last part we present some numerical examples which demonstrate the faster convergence of the multi-splitting algorithm when overlapping splittings are used.

Let us introduce some notation. In order to avoid too many indices, we refer to the element in the i-th row and j-th column of a matrix C by $C(i,j)$. The matrix I_s indicates the $s \times s$ identitymatrix.

2. The Waveform Relaxation Algorithm

For linear initial value problems (1.2) the waveform relaxation algorithm is based on a splitting of the matrix A into $A = M - N$ which yields

$$x'(t) + Mx(t) = Nx(t) + f(t) \qquad x(0) = x_0. \tag{2.1}$$

This is written as an iteration where the right hand side is taken as an input to the iteration

$$x'_{n+1}(t) + Mx_{n+1}(t) = Nx_n(t) + f(t) \qquad x_{n+1}(0) = x_0. \tag{2.2}$$

The starting function $x_0(t)$ is chosen as constant initial values $x_0(t) \equiv x_0$. In the case of Block-Gauss-Jacobi the matrix M is chosen to be block diagonal and in the case of Block-Gauss-Seidel M has block lower triangular structure. With a block diagonal M the algorithm is well suited for parallel computers. Suppose

$$M = \begin{pmatrix} D_1 & & & \\ & D_2 & & 0 \\ & & \ddots & \\ & 0 & & D_r \end{pmatrix}$$

where we assume that each D_j is a real $m_j \times m_j$ matrix. By this the problem is naturally decomposed into r subsystems. On a parallel computer each subsystem is now assigned to a processor and the solution of each subsystem is computed in parallel. This is done not only for one timepoint but over the whole domain of integration[1] $[0,T]$. After each subsystem has been solved the index of the iteration is increased and another iteration is performed until convergence has occurred.

We observe that the sum of the dimensions of the subsystems always equals the dimension of the underlying problem. In the next section we will show that it is possible to allow the sum of the dimensions of the subsystems to exceed the dimension of the original problem.

3. The Multi-splitting Algorithm

In 1985 O'Leary and White presented a method for solving $Ax = b$ on a parallel computer by splitting the matrix A not just once but several times into $A = M_l -$

[1]It has turned out that it is more advantageous to break the domain of integration into time blocks so called windows $[0,T_1],[T_1,T_2],\ldots,[T_s,T]$ and to perform the integration only on one window. After convergence is reached on that particular window one proceeds to the next window.

$N_l l = 1, \ldots, L$. Frommer and Mayer demonstrated in [FrMa90] that it can have computational advantages to split A more than once and to overlap the subsystems. We need the following definition:

Definition 3.1:
Let $L \geq 1$ be a fixed integer which will be called **the number of splittings**. Let A, M_l, N_l, E_l be real $m \times m$ matrices. The set of ordered triples $(M_l, N_l, E_l) l = 1, \ldots, L$ is called a **multi-splitting of** A if

(i)

$$A = M_l - N_l \quad l = 1, \ldots, L \tag{3.1}$$

(ii) The matrices E_l are diagonal matrices and satisfy the **consistency condition**

$$\sum_{l=1}^{L} E_l = I_m. \tag{3.2}$$

Using (3.1) we can rewrite (1.2) as

$$x'(t) + M_l x(t) \;=\; N_l x(t) + f(t) \qquad x(0) = x_0 \quad l = 1, \ldots, L$$

As in the waveform relaxation algorithm we will take the right hand side as input and the left hand side as unknown. Again this is solved in an iterative way

$$y'_{l,n}(t) + M_l y_{l,n}(t) = N_l x_n(t) + f_l(t), \qquad y_{l,n}(0) = x_0 l = 1, \ldots, L \tag{3.3}$$

After having solved each subsystem we compute a new approximation to the solution of (1.2) by

$$x_{n+1}(t) = \sum_{l=1}^{L} E_l y_{l,n}(t) \tag{3.4}$$

Definition 3.2:
For any $l \in \{1, \ldots, L\}$ we will refer to

$$y'_l(t) + M_l y_l(t) = N_l x(t) + f(t) \quad y_l(0) = x_0$$

as a **subsystem** of (1.2).

Remark 3.3:
If we take a closer look at (3.3) we see that there is no interaction between two different subsystems. Therefore we can solve these subsystems on different processors in parallel.

Remark 3.4:
From (3.4) we see that we do not need to compute those components i of $y_{l,n}(t)$ where $E_l(i,i) = 0$. Therefore in practical implementations we will not compute

these components at all. We will return to this aspect in a discussion of a practical implementation in section 5.
We conclude this section by describing the **Multi-splitting Algorithm** in an algorithmic way :

```
initialize
    $x_0(t) \equiv x_0 \ \forall t \in [0,T]$
    n:=0
repeat
    solve for $l = 1, \ldots, L$ { in parallel }
    $y'_{l,n}(t) + M_l y_{l,n}(t) = N_l x_n(t) + f(t) \quad y_{l,n}(0) = x_0$
    $x_{n+1}(t) := \sum_{l=1}^{L} E_l y_{l,n}(t)$
until stopping criterion
```

4. The Analysis of the Multi-splitting Algorithm

In [Neva89] Nevanlinna analyzed the waveform relaxation algorithm. In this section we will adapt the means presented in that paper to analyze the multi-splitting algorithm.
First we introduce the following notation

$$k_l(t) = \exp(-tM_l)N_l \quad \text{for } l = 1, \ldots, L. \tag{4.1}$$

$k_l(t)$ is the kernel of the linear integral operator defined by

$$K_l u(t) = \int_0^t k_l(t-s)u(s)\,ds \quad \text{for } l = 1, \ldots, L. \tag{4.2}$$

If we denote

$$\varphi_l(t) = \exp(-tM_l)x_0 + \int_0^t \exp((s-t)M_l)f(s)\,ds \quad \text{for } l = 1, \ldots, L \tag{4.3}$$

we can write the solution $y_l(t)$ of a subsystem (3.3) using (4.1)-(4.3) as

$$y_{l,n}(t) = K_l x_n(t) + \varphi_l(t)$$

Having computed the solution of each subsystem we have to weight the solutions by the E_l matrices. Then we sum the weighted solutions over all subsystems to get a new approximation to the solution of (1.2). Therefore the following notation will be used frequently

$$\begin{aligned}
\tilde{k}(t) &= \sum_{l=1}^{L} E_l k_l(t) \\
\tilde{\mathcal{K}}\, u(t) &= \sum_{l=1}^{L} E_l K_l u(t) \\
\tilde{\varphi}(t) &= \sum_{l=1}^{L} E_l \varphi_l(t)
\end{aligned}$$

.

Using this notation the next iteration can be written as

$$x_{n+1}(t) = \tilde{\mathcal{K}}\, x_n(t) + \tilde{\varphi}(t) \tag{4.4}$$

The following lemma is obvious and is given without proof.

Lemma 4.1:
$x(t)$ is the exact solution of (1.2) if and only if $x(t)$ is also a solution of each subsystem

$$x(t) = K_l x(t) + \varphi_l(t) \quad \text{for } l = 1, \ldots, L.$$

We shall make use of Lemma 4.1 in the following

Lemma 4.2:
$x(t)$ is the solution of (1.2) if and only if $x(t)$ is the solution of the fixpoint equation $x(t) = \tilde{\mathcal{K}}\, x(t) + \tilde{\varphi}(t)$.
Proof:
Let x(t) be the solution of (1.2). We get for $l = 1, \ldots, L$:

$$\begin{aligned} E_l x(t) &= E_l K_l x(t) + E_l \varphi_l(t) \\ \Rightarrow \quad \sum_{l=1}^{L} E_l x(t) &= \sum_{l=1}^{L} E_l K_l x(t) + \sum_{l=1}^{L} E_l \varphi_l(t) \\ \Rightarrow \quad x(t) &= \tilde{\mathcal{K}}\, x(t) + \tilde{\varphi}(t). \end{aligned}$$

The other direction of the lemma will be shown later.

■

Disregarding convergence one can verify by substitution that

$$x(t) = \sum_{i=0}^{\infty} \tilde{\mathcal{K}}^{\,i} \tilde{\varphi}(t). \tag{4.5}$$

is a formal solution of the fixpoint equation. Using (4.4) inductively yields:

Lemma 4.3:
For $n \geq 1$ we obtain

$$x_n(t) = \tilde{\mathcal{K}}^{\,n} x_0(t) + \sum_{i=0}^{n-1} \tilde{\mathcal{K}}^{\,i} \tilde{\varphi}(t). \tag{4.6}$$

We observe that the convergence of the iteration depends only on the behaviour of the linear operator $\tilde{\mathcal{K}}^{\,n}$.

Before we show that the fixpoint equation has unique solution, we take a closer look at the operator $\tilde{\mathcal{K}}^{\,n}$.

$$\tilde{\mathcal{K}}^{\,n} u(t) = \sum_{l_1=1}^{L} \sum_{l_2=1}^{L} \ldots \sum_{l_n=1}^{L} E_{l_1} K_{l_1} E_{l_2} K_{l_2} \ldots E_{l_n} K_{l_n} u(t).$$

The next lemma shows that $\tilde{\mathcal{K}}^{\,n}$ can be regarded as an n-fold convolution.

Lemma 4.4:
Let $\tilde{k}^{i*}$ with $i \in \mathbb{N}$ be recursively defined by

$$\begin{aligned} \tilde{k}^{(i+1)*} &= \tilde{k} * \tilde{k}^{i*} \\ \tilde{k} * u(t) &= \int_0^t \tilde{k}(t-s)u(s)ds \end{aligned}$$

Then

$$\tilde{\mathcal{K}}^n u(t) = \int_0^t \tilde{k}^{n*}(t-s)u(s)ds. \tag{4.7}$$

Proof:
The proof is done by induction with respect to n.
For $n = 1$ we get

$$\tilde{\mathcal{K}}\, u(t) = \sum_{l=1}^{L} E_l \int_0^t k_l(t-s)u(s)\; ds = \int_0^t \tilde{k}(t-s)u(s)\, ds$$

Assuming that (4.7) holds for some n we get for $n+1$

$$\begin{aligned} \tilde{\mathcal{K}}^{n+1} u(t) &= \tilde{\mathcal{K}}\,(\tilde{\mathcal{K}}^n u(t)) \\ &= \tilde{\mathcal{K}}\,(\int_0^t \tilde{k}^{n*}(t-s)\, u(s)\, ds) \\ &= \int_0^t \tilde{k}(t-s) \int_0^s \tilde{k}^{n*}(s-v)\, u(v)\, dv\, ds \\ &= \int_0^t \int_v^t \tilde{k}(t-s)\, \tilde{k}^{n*}(s-v)\, ds\, u(v)\, dv \end{aligned}$$

Substituting now $w = s - v$ yields

$$\begin{aligned} \tilde{\mathcal{K}}^{n+1} u(t) &= \int_0^t \int_0^{t-v} \tilde{k}((t-v)-w)\, \tilde{k}^{n*}(w)\, dw\, u(v)\, dv \\ &= \int_0^t \tilde{k} * \tilde{k}^{n*}(t-v)\, u(v)\, dv \\ &= \int_0^t \tilde{k}^{(n+1)*}(t-v)u(v)dv \end{aligned}$$

■

If we want to show convergence of the series (4.5) we need that the operator $\tilde{\mathcal{K}}$ is a contraction operator.

Let us introduce the following norm:

$$\|u\|_T = \max_{t\in[0,T]} |u(t)| \tag{4.8}$$

where $|\cdot|$ denotes any vector norm. By $\|\cdot\|_T$ we also denote the induced matrix or operator norm.

Since $[0,T]$ is a finite interval we can suppose that for the kernel k_l the following bound holds:

$$\|k_l\|_T \leq C_l \quad \text{for } l = 1,\dots,L \tag{4.9}$$

We define

$$C = \sum_{l=1}^{L} C_l \quad \text{and assume} \quad \|E_l\|_T \leq E \quad \text{for } l = 1,\dots,L. \tag{4.10}$$

Using (4.9) and (4.10) we get the following bound:

$$|\tilde{k}(t)| \leq \|\tilde{k}\|_T \leq CE =: \bar{C}. \tag{4.11}$$

Before estimating $\tilde{\mathcal{K}}^n$ we derive a bound for its kernel $\tilde{k}^{n*}$:

Lemma 4.5:
For the kernel of the iteration operator $\tilde{\mathcal{K}}^{n*}$ we have

$$|\tilde{k}^{n*}(t)| \leq \bar{C} \int_0^t |\tilde{k}^{(n-1)*}(s)|\, ds$$

and

$$|\tilde{k}^{n*}(t)| \leq \bar{C} \frac{(\bar{C}t)^{n-1}}{(n-1)!}. \tag{4.12}$$

Proof:

$$\begin{aligned} |\tilde{k}^{n*}(t)| &= |\tilde{k} * \tilde{k}^{(n-1)*}(t)| = |\int_0^t \tilde{k}(t-s)\tilde{k}^{(n-1)*}(s)\, ds| \\ &\leq \int_0^t |\tilde{k}(t-s)|\, |\tilde{k}^{(n-1)*}(s)|\, ds \\ &\leq \int_0^t \|\tilde{k}\|_T\, |\tilde{k}^{(n-1)*}(s)|\, ds \leq \bar{C} \int_0^t |\tilde{k}^{(n-1)*}(s)|\, ds \end{aligned}$$

¿From this (4.12) follows by induction. ■

Now we derive a bound for the iteration operator $\tilde{\mathcal{K}}^n$ itself. Using Lemma 4.4 and Lemma 4.5 we get:

$$\|\tilde{\mathcal{K}}^n\|_T \leq \frac{(\bar{C}T)^n}{n!} \tag{4.13}$$

since

$$\begin{aligned} \|\tilde{\mathcal{K}}^n\|_T = \sup_{\|u\|_T=1} \|\tilde{\mathcal{K}}^n u\|_T &= \sup_{\|u\|_T=1} \sup_{t\in[0,T]} |\tilde{\mathcal{K}}^n u(t)| \\ &= \sup_{\|u\|_T=1} \sup_{t\in[0,T]} |\int_0^t \tilde{k}^{n*}(t-s)u(s)ds| \end{aligned}$$

$$
\begin{aligned}
&\leq \sup_{\|u\|_T=1} \sup_{t\in[0,T]} \left|\int_0^t \tilde{k}^{n*}(t-s)\|u\|_T ds\right| \\
&= \sup_{t\in[0,T]} \int_0^t |\tilde{k}^{n*}(t-s)| ds \\
&= \sup_{t\in[0,T]} \int_0^t |\tilde{k}^{n*}(v)| dv \\
&\leq \int_0^T \bar{C}^n \frac{v^{n-1}}{(n-1)!} dv \\
&\leq \frac{(\bar{C}T)^n}{n!}
\end{aligned}
$$

■

We return to the fixpoint equation and we will show that it has a unique solution. Since $\|\tilde{\mathcal{K}}\|_T \leq \bar{C}T$ we can find an interval $[0, T_0]$ with $T_0 < {}^1\!/_{\bar{C}}$. Therefore the operator $\tilde{\mathcal{K}}$ is a contraction operator on that particular interval. It is easily seen that with (4.13) we also have convergence of the series (4.5). We now complete the proof of Lemma 4.2.

Proof: (Continuation of Lemma 4.2)
We have that

$$x(t) = \sum_{i=0}^{\infty} \tilde{\mathcal{K}}^i \tilde{\varphi}(t)$$

is a solution of the fixpoint equation. Since $\tilde{\mathcal{K}}$ is a contraction operator on sufficiently small intervals the solution of the fixpoint equation is unique. Moreover we already know that if $x(t)$ is a solution of (1.2) then it is also a solution of the fixpoint equation. Combining this results completes the proof. ■

Remark 4.6:
This approach of splitting the domain of integration $[0, T]$ into subintervals $[0, T_0], [T_0, T_1], \dots, [T_s, T]$ so called windows is used in implementations to accelerate the rate of convergence. The iteration is performed only on one particular window $[T_{i-1}, T_i]$ and one proceeds to the next interval $[T_i, T_{i+1}]$ after convergence has occurred on $[T_{i-1}, T_i]$.

If we now subtract (4.6) from (4.5) we have a first form of the error of the n-th approximation:

$$x(t) - x_n(t) = \sum_{i=n}^{\infty} \tilde{\mathcal{K}}^i \tilde{\varphi}(t) - \tilde{\mathcal{K}}^n x_0(t).$$

Using this result we can prove the following lemma:

Lemma 4.7:
The error of the n-th approximation is bounded by

$$\|x - x_n\|_T \leq \frac{(\bar{C}T)^n}{n!} \Big(\exp(\bar{C}T)\|\tilde{\varphi}\|_T + \|x_0\|_T\Big) \tag{4.14}$$

Proof: Using (4.5), (4.6) and (4.13) we have

$$\begin{aligned}
\|x - x_n\|_T &\leq \max_{t\in[0,T]} \sum_{i=n}^{\infty} |\tilde{\mathcal{K}}^i \tilde{\varphi}(t) - \tilde{\mathcal{K}}^n x_0(t)| \\
&\leq \sum_{i=n}^{\infty} \|\tilde{\mathcal{K}}^i\|_T \, \|\tilde{\varphi}\|_T + \|\tilde{\mathcal{K}}^n\|_T \, \|x_0\|_T \\
&\leq \sum_{i=n}^{\infty} \frac{(\bar{C}T)^i}{i!} \|\tilde{\varphi}\|_T + \frac{(\bar{C}T)^n}{n!} \|x_0\|_T \\
&\leq \frac{(\bar{C}T)^n}{n!} \left(\sum_{i=0}^{\infty} \frac{(\bar{C}T)^i i!}{(i+n)!} \|\tilde{\varphi}\|_T + \|x_0\|_T \right)
\end{aligned}$$

Using the inequality

$$\frac{a!}{(a+b)!} \leq \frac{1}{b!}$$

completes the proof. ■

¿From Lemma 4.7 we see that we can achieve convergence of the multi-splitting algorithm on a fixed interval $[0, T]$. If T is large however, the convergence can get very slow. It is possible to circumvent this difficulty by introducing an exponentially weighted norm, in which convergence can be achieved on $[0, \infty)$. We will discuss this different approach in future work.

5. A Multi-Splitting based on Overlapping Block Decomposition

In this section we discuss a practical implementation of the multi-splitting algorithm which is based on an overlapping block decomposition of the matrix A. In section 2 we have seen that we do not need to compute the i-th component of $y_{l,n}$ whenever $E_l(i,i) = 0$. We give a formalism to eliminate all these unused components. This results in a reduction of the dimensions of the subsystems.

We need the following definition:

Definition 5.1:
Let $S = \{1, \ldots, m\}$ which will be referred to as the **set of the components of problem (1.2)**.

First we choose L subsets $S_1, S_2, \ldots, S_L$ of S satisfying the condition

$$\bigcup_{l=1}^{L} S_l = S$$

We get immediately that if m_l denotes the number of elements of S_l for $l = 1, \ldots, L$ then

$$\sum_{l=1}^{L} m_l \geq m$$

Furthermore we have equality if the $S_1, \dots, S_L$ are disjoint.

Definition 5.2:
If there exists at least one pair of indices $i \neq j$ with $i, j \in \{1, \dots, L\}$ so that $S_i \cap S_j \neq \emptyset$ then we will call a multi-splitting an **overlapping multi-splitting** otherwise it will be named a **disjoint multi-splitting**.

Remark 5.3:
We use a block decomposition which means if $i \in S_l$ and $i + r \in S_l$ then $j \in S_l$ with $i \leq j \leq i + r$. We overlap only adjoining subsystems where moreover we only encounter the case that one component is in at most two subsystems.

For simplicity we assume that the number of overlapping components between two adjoining subsystems is constant.

Definition 5.4:
By **overlap** k we mean that

$$k = |S_l \cap S_{l+1}| \qquad \text{for } l = 1, \dots, L-1.$$

Next we define the elements of the E_l matrices by setting those diagonal elements $E_l(i,i)$ nonzero, where the according index i is an element of S_l. Furthermore we require that the consistency condition (3.2) is satisfied:

$$E_l(i,j) = \begin{cases} 0 & \text{if } i \neq j \\ 0 & \text{if } i = j \text{ and } i \notin S_l \\ \neq 0 & \text{if } i = j \text{ and } i \in S_l \end{cases}$$

Now we know which components i of the l-th subsystem will be weighted by a factor $E_l(i,i) \neq 0$ and which will be dropped by multiplying it with $E_l(i,i) = 0$. By computing values for components that will be thrown away afterwards not only computing time is wasted but also more memory is used. Therefore we will compute a component i of subsystem l only if $E_l(i,i) \neq 0$ or equivalently if $i \in S_l$. Using the subsets $S_1, S_2, \dots, S_L$ we can define L projection matrices $P_1, P_2, \dots, P_L$ in the following way

$$P_l(i,j) = \begin{cases} 0 & \text{if } i \neq j \\ 0 & \text{if } i = j \text{ and } i \notin S_l \\ 1 & \text{if } i = j \text{ and } i \in S_l \end{cases}$$

where $i, j \in \{1, \dots, m\}$ and $l \in \{1, \dots, L\}$ i.e. only those diagonal elements of P_l are nonzero where the according index is an element of S_l.

By using these projection matrices we project problem (1.2) into L different subspaces. We solve now the projected problems in each subspace but we use components from outside the particular subspace as an input.

Remark 5.5:
In a disjoint multi-splitting the matrices P_l and E_l coincide.

Lemma 5.6:
The following relations hold:

- (i) $P_l E_l = E_l P_l = E_l$ for $l = 1, \ldots, L$
- (ii) If a disjoint multi-splitting is used then $E_i E_j = E_j E_i = 0$

for $i \neq j$.

Recalling the algorithm we see that the iterate x_{n+1} is computed by

$$x_{n+1}(t) = \sum_{l=1}^{L} E_l y_{l,n}(t) = \sum_{l=1}^{L} E_l P_l y_{l,n}(t)$$

where Lemma 5.6 (i) is used. This means that in the l-th subsystem we only have to compute $P_l y_{l,n}$. In order to have only the components i with $i \in S_l$ as unknowns in the l-th subsystem in a practical implementation of the algorithm not the matrix A but the projected matrix $P_l A$ is split. This saves computing time and much less memory is needed. Therefore we use from now on

$$P_l A = M_l - N_l \quad l = 1, \ldots, L$$

Remark 5.7:
It is easy to see that the case $L = 1$ yields exactly the waveform relaxation algorithm as presented in section 2. Moreover the Block Gauss Jacobi iteration in Example 2.1 can be expressed as a disjoint multi-splitting with $L = r$ and $M_l = P_l A P_l$.

Example 5.8:
We consider the 5-dimensional problem:

$$x'(t) + \begin{pmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{pmatrix} x(t) = \begin{pmatrix} -1 \\ 1 \\ 1 \\ 1 \\ -1 \end{pmatrix}, \quad x(0) = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

where $x(t) = (x_1(t), x_2(t), x_3(t), x_4(t), x_5(t))'$.

The splitting is given by the subsets of S:

Block Gauss Jacobi, $S_1 = \{1, 2\}$, $S_2 = \{3, 4, 5\}$:

$$M = \begin{pmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{pmatrix} \quad N = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Multi-Splitting, $S_1 = \{1, 2, 3, 4\}$, $S_2 = \{3, 4, 5\}$:

$$M_1 = \begin{pmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad N_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$M_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{pmatrix} \quad N_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$E_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & s_1 & 0 & 0 \\ 0 & 0 & 0 & s_2 & 0 \\ ,0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad E_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1-s_1 & 0 & 0 \\ 0 & 0 & 0 & 1-s_2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

with $0 \leq s_1, s_2 \leq 1$.

6. Numerical Results

In this section we present some numerical experiments that have the same characteristics. The number of iterations necessary to satisfy a given accuracy is for the multi-splitting algorithm always lower than for the waveform relaxation algorithm. Throughout this section we will refer to the waveform relaxation algorithm as a disjoint multi-splitting or as a multi-splitting with overlap 0.

The problem:
We always use the same dimension of the test problem, the same number of timepoints and the same interval of integration. The dimension m is always 150, the number of timepoints is 150 and the interval of integration is $[0, T] = [0, 2]$. As the right hand side function $f(t)$ we have chosen: $f(t) = 1 - \sin(4t)$.
We vary the number of subsystems and the number of overlapping components.
The matrix A is defined as follows:

$$A = \begin{pmatrix} 2 & -\frac{1}{2} & -\frac{1}{4} & -\frac{1}{8} & -\frac{1}{16} & -\frac{1}{32} & 0 & & \cdots & & 0 & -\frac{1}{64} \\ -\frac{1}{2} & 2 & -\frac{1}{2} & -\frac{1}{4} & -\frac{1}{8} & -\frac{1}{16} & \ddots & 0 & & \cdots & & 0 \\ -\frac{1}{4} & -\frac{1}{2} & 2 & -\frac{1}{2} & -\frac{1}{4} & -\frac{1}{8} & \ddots & & & & & \\ -\frac{1}{8} & -\frac{1}{4} & -\frac{1}{2} & 2 & -\frac{1}{2} & -\frac{1}{4} & \ddots & & & & & \vdots \\ -\frac{1}{16} & -\frac{1}{8} & -\frac{1}{4} & -\frac{1}{2} & 2 & -\frac{1}{2} & \ddots & & & & & \\ -\frac{1}{32} & -\frac{1}{16} & -\frac{1}{8} & -\frac{1}{4} & -\frac{1}{2} & 2 & \ddots & & & & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & & & & \ddots & -\frac{1}{32} \\ & 0 & & & & & & & & & \ddots & -\frac{1}{16} \\ \vdots & & & & & & & & & & \ddots & -\frac{1}{8} \\ & \vdots & & & & & & & & & \ddots & -\frac{1}{4} \\ 0 & & & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & -\frac{1}{2} \\ -\frac{1}{64} & 0 & & \cdots & & 0 & -\frac{1}{32} & -\frac{1}{16} & -\frac{1}{8} & -\frac{1}{4} & -\frac{1}{2} & 2 \end{pmatrix}$$

As a stopping criterion we require that all components i of the solution satisfy

$$\sum_t |x_{n+1,i}(t) - x_{n,i}(t)| \leq 10^{-3}. \tag{6.1}$$

where t is a timepoint. The subsystems were solved with the semi-implicit mid-point rule [BaDe81] and as already mentioned constant stepsize $h = \frac{1}{75}$.

How do we split?

Since the structure of the matrix A is simple we have chosen the subsets $S_1, S_2, \ldots, S_L$ that each subset contains at least $\frac{m}{L}$ elements. If L is not a factor of m, i.e. $m = \alpha L + \beta$ with $\beta \neq 0$ we add to the last β subsets $S_{L-\beta+1}, \ldots, S_L$ one element. If we use overlap j we add to all but the last subsystem j elements. By this we achieve that the workload is distributed evenly among the different processors if a parallel computer is used.

This strategy is illustrated in Example 5.9 with $m = 5, L = 2$ and overlap 0 and overlap 2 respectively.

How do we choose the elements of the E_l matrices?

It seems natural to use as a first approach $E_l(i,i) = \frac{1}{2}$ whenever component i of the l-th subsystem is in two subsystems.

The number of iterations necessary to satisfy a given stopping criterion is as expected a monotone decreasing function of the number of overlapping components. If we increase the overlap however the number of necessary iterations suddenly increases. We can explain this by considering a component i for which either $i-1$ or $i+1$ is not an element of the same subset. Lets suppose that $i \in S_l, i+1 \notin S_l, i+1 \in S_{l+1}$. We have two approximate solutions of component i, one solution computed from subsystem l and one solution computed from subsystem $l+1$. Since the phenomenon only occurs if we use a large number of overlapping components we can assume that $i-2$, $i-1$, i, $i+1$ and $i+2$ are elements of S_{l+1}. Therefore we expect that the computation of component i from the subsystem $l+1$ is more accurate than the other computation. But with the above mentioned weighting scheme we use the same weight for both solutions and we introduce an error that causes an increased number of iteration steps required.

Instead of the above choice of E_l we now give a different setting of the weight matrices wherefore we need the following definition:

Definition 6.1:

We call a component i of S_l a **border component** if $(i-1) \notin S_l$ or $(i+1) \notin S_l$. The lower border component of S_l is denoted by l_{min} and the upper border component is denoted by l_{max}. For the first and last subsystem we define $1_{min} = 1$ and $L_{max} = m$. The **distance $d_l(i)$ from the border** of component i is defined as $d_l(i) = l_{max} - i$.

Suppose we use overlap k. We propose the following choice of weights for the overlapping components $i = l_{max} - k + 1, \ldots, l_{max}$

$$E_l(i,i) = \frac{d_l(i) + 1}{\text{overlap} + 1} \quad \text{for } l = 1, \ldots L-1.$$

By this we define the last k diagonal elements of $E_1, E_2, \ldots, E_{L-1}$. Since the E_l matrices satisfy the consistency condition the first k diagonal elements of $E_2, E_3, \ldots, E_L$

are also uniquely defined. Using this weight scheme we get for the matrices E_1 and E_2 of Example 5.9 with overlap $k = 2$:

$$E_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{2}{3} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{3} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad E_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & \frac{2}{3} & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

We will vary the number of splittings L and the number of overlapping components. The quantity that is given in the tables is the necessary number of iterations that (6.1) is satisfied.

Table 1: **number of subsystems** $L = 2$

overlap	0	1	2	3	4	5	6	7	8	9
iterations	8	7	6	5	5	4	4	4	4	4
time[%]	100	90	79	68	70	58	59	61	63	64

Table 2: **number of subsystems** $L = 3$

overlap	0	1	2	3	4	5	6	7	8	9
iterations	8	7	6	5	5	4	4	4	4	4
time[%]	100	93	82	71	73	60	61	62	64	66

Table 3: **number of subsystems** $L = 6$

overlap	0	1	2	3	4	5	6	7	8	9
iterations	8	7	6	5	5	4	4	4	4	4
time[%]	100	93	82	71	73	60	61	62	64	66

References:

[FrMa90] A. Frommer, G. Mayer, *"Theoretische und praktische Ergebnisse zu Multisplitting-Verfahren auf Parallelrechnern"*, *ZAMM Vol.70 (1990) No.6, T 600 - T 602*

[Neva89] O. Nevanlinna, *"Remarks on Picard-Lindelöf Iterations"*, *Part I, Bit 29 (1989), pp. 328-346, Part II, Bit 29 (1989), pp. 535-562,*

[OLWh85] D.P. O'Leary, R.E. White, *"Multi-Splittings of Matrices and Parallel Solution of Linear Systems"*, *SIAM J.Alg. Disc. Meth. Vol.6, No.4, (1986), pp. 630-640*

[Whit86] R.E. White, *"Parallel Algorithms for nonlinear Problems"*, *SIAM J. Alg. Disc. Meth. Vol.7, No.1, (1986), pp. 137-149*

[WhSa87] J. White, A.L. Sangiovanni-Vincentelli, *"Relaxation Techniques for the Simulation on VLSI Circuits"*, *Kluwer Academic Publishers, Boston, (1987)*