SIAM J. SCI. COMPUT.
Vol. 16, No. 1, pp. 40–49, January 1995

WAVEFORM RELAXATION WITH OVERLAPPING SPLITTINGS*

ROLF JELTSCH† AND BERT POHL†

Abstract. In this paper, an extension of the waveform relaxation algorithm for solving large systems of ordinary differential equations is presented. The waveform relaxation algorithm 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 disjointedness, an extension of this algorithm is obtained. This new algorithm, the so-called multisplitting algorithm, is also well suited for parallel computation. As numerical examples demonstrate, this overlapping of the subsystems heavily reduces the computation time.

Key words. waveform relaxation, parallel computing, differential equations, overlapping

AMS subject classifications. 65L05 65Y05

1. Introduction.

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

$$x'(t) = f(t, x(t)), \qquad x(0) = x_0 \tag{1}$$

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 [11]. 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 an m-dimensional nonlinear system of equations at each timepoint before advancing to the next timepoint.

In the beginning of the 1980s a new approach for solving these problems was developed at the Electronic Research Laboratory at Berkeley that circumvents these difficulties. In this approach, called the 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 of using several processors in parallel and the smaller dimension of the subsystems but also of using 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.

Multisplitting methods were first introduced by O'Leary and White in [6] for solving large linear system of equations on a parallel computer. This idea was extended to nonlinear problems by White in [10].

In this paper we adapt the ideas of O'Leary and White and study the use of multisplitting 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

$$x'(t) + Ax(t) = f(t), \qquad x(0) = x_0 \tag{2}$$

*Received by the editors October 9, 1991; accepted for publication (in revised form) October 12, 1993.

†Seminar für Angewandte Mathematik, Eidgenössische Technische Hochschule, CH-8092 Zürich, Switzerland (bpohl@sam.math.ethz.ch, jeltsch@sam.math.ethz.ch).

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 multisplitting algorithm; a first analysis of which is given in the fourth section where we restrict ourselves to finite intervals of integration. The analysis for infinite intervals is presented in the fifth section. In the sixth section we discuss a practical implementation of the multisplitting algorithm. Finally, we present some numerical examples that demonstrate the faster convergence of the multisplitting 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 ith row and jth column of a matrix C by $C(i, j)$. The matrix I_s indicates the $s \times s$ identity matrix.

2. The waveform relaxation algorithm.

For linear initial value problems (2) the waveform relaxation algorithm is based on a splitting of the matrix A into $A = M - N$ which yields

$$x'(t) + Mx(t) = Nx(t) + f(t), \qquad x(0) = x_0.$$

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

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

The starting function $x_0(t)$ is chosen as constant initial values $x_0(t) \equiv x_0$. In the case of block 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 multisplitting 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 $A = M_l - N_l$ for $l = 1, \ldots, L$. Frommer and Mayer demonstrated in [1] that it can have computational advantages to split A more than once and to overlap the subsystems.

We need the following definition.

[1]It has turned out that it is more advantageous to break the domain of integration into time blocks, so-called windows, $[0, T_0]$, $[T_0, T_1]$, $\ldots$, $[T_s, T]$ and to perform the integration only on one window. After convergence is reached on that particular window one proceeds to the next window; see also §4.

DEFINITION 1. *Let* $L \geq 1$ *be a fixed integer that 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) *for* $l = 1, \dots, L$ *is called a* multisplitting *of* A *if*

(i)

$$A = M_l - N_l \quad \textit{for } l = 1, \dots, L. \tag{3}$$

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

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

Using (3) we can rewrite (2) as

$$x'(t) + M_l x(t) = N_l x(t) + f(t), \qquad x(0) = x_0 \quad \text{for } l = 1, \dots, L.$$

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 for $l = 1, \dots, L$,

$$y'_{l,n+1}(t) + M_l y_{l,n+1}(t) = N_l x_n(t) + f(t), \qquad y_{l,n+1}(0) = x_0. \tag{4}$$

For any $l \in \{1, \dots, L\}$ we will refer to (4) as a *subsystem* of (2). After having solved each subsystem we compute a new approximation to the solution of (2) by

$$x_{n+1}(t) = \sum_{l=1}^{L} E_l y_{l,n+1}(t). \tag{5}$$

If we take a closer look at (4) we see that there is no interaction between two different subsystems. Therefore we can solve these subsystems on different processors in parallel.

We conclude this section by describing the *multisplitting algorithm* in an algorithmic way and presenting an example of a multisplitting based on an (overlapping) decomposition of the set of components.

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

We will now discuss an example of the multisplitting algorithm, which is based on an (overlapping) block decomposition of the matrix A. Let us denote by $S = \{1, \dots, m\}$ the *set of the components*. First we choose L subsets $S_1, S_2, \dots, S_L$ 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

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

and we have equality if $S_1, \ldots, S_L$ are disjoint. If there exists at least one pair of indices $i \neq j$ with $i, j \in \{1, \ldots, L\}$ so that $S_i \cap S_j \neq \emptyset$, then we will call a multisplitting an *overlapping multisplitting*; otherwise, it will be named a *disjoint multisplitting*.

Here we will only 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. By *overlap k* we mean that

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

Next we define the elements of the E_l matrices by setting

$$E_l(i, i) = 0 \quad \text{for } i \notin S_l.$$

By computing values for components that will be thrown away afterward, not only computing time is wasted but also more memory is used. Therefore we will compute a component i of subsystem l only if $i \in S_l$. Using the subsets $S_1, S_2, \ldots, S_L$ we can define L projection matrices $P_1, P_2, \ldots, P_L$ in the following way:

$$P_l \text{ are diagonal matrices with } P_l(i, i) = 1 \Leftrightarrow i \in S_l.$$

By using these projection matrices we project problem (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. We observe that in a disjoint multisplitting the matrices P_l and E_l coincide.

LEMMA 3.1. *The following relation holds:* $P_l E_l = E_l P_l = E_l$ *for* $l = 1, \ldots, L$.

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

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

where Lemma 3.1 is used. This means that in the lth subsystem we only have to compute $P_l y_{l,n+1}$. In order to have only the components i with $i \in S_l$ as unknowns in the lth subsystem *in a practical implementation* of the algorithm, not the matrix A but the projected matrix $P_l A$ is split.

As a final remark we observe that the case $L = 1$ yields exactly the waveform relaxation algorithm as presented in §2. Moreover the block Jacobi iteration can be expressed as a disjoint multisplitting with $L = r$ and $M_l = P_l A P_l$.

4. The analysis of the multisplitting algorithm. In [5] Nevanlinna analyzed the waveform relaxation algorithm. In this section we will adapt the means presented in that paper to analyze the multisplitting algorithm. Since the proofs of this section can be obtained in an analogous way or can be found in [3], we will omit them here. In this section we will restrict ourselves to finite time intervals and discuss infinite intervals in the next section.

First we introduce the following notation:

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

$k_l(t)$ is the kernel of the linear integral operator K_l acting on elements of the space $X = \mathcal{L}^p([0,T],\mathbb{R}^m)$, $1 \le p \le \infty$ defined by

$$K_l u(t) = \int_0^t k_l(t-s)u(s)\,ds \qquad \text{for } l = 1,\dots,L.$$

If we denote

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

we can write the solution $y_l(t)$ of a subsystem (4) in the $(n+1)$st sweep using the "variation of constants formula" as

$$y_{l,n+1}(t) = K_l x_n(t) + \varphi_l(t).$$

Having computed the solution of each subsystem we have to weight the solutions by the E_l matrices and to sum the weighted solutions over all subsystems to get a new approximation to the solution of (2). Therefore the following notation will be used frequently

$$k(t) = \sum_{l=1}^{L} E_l k_l(t),$$

$$\mathcal{K}u(t) = \sum_{l=1}^{L} E_l K_l u(t),$$

$$\varphi(t) = \sum_{l=1}^{L} E_l \varphi_l(t).$$

Using this notation the next iteration can be written as

$$x_{n+1}(t) = \mathcal{K}x_n(t) + \varphi(t). \tag{6}$$

LEMMA 4.1. *The following statements are equivalent*:
1. $x(t)$ *is a solution of* (2);
2. $x(t)$ *is a solution of* $x(t) = K_l x(t) + \varphi_l(t), x(0) = x_0, l = 1,\dots,L$;
3. $x(t)$ *is the solution of the fixpoint equation* $x(t) = \mathcal{K}x(t) + \varphi(t)$.

Using (6) inductively yields:

$$x_n(t) = \mathcal{K}^n x_0(t) + \sum_{i=0}^{n-1} \mathcal{K}^i \varphi(t). \tag{7}$$

To investigate convergence we observe that $\mathcal{K}^n$ can be written as an n-fold convolution:

$$\mathcal{K}^n u(t) = \int_0^t k^{n*}(t-s)u(s)ds,$$

where

$$k^{(i+1)*} = k * k^{i*}, \qquad k^{1*} = k.$$

Let us introduce the following norm:

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

where $|\cdot|$ denotes an L_p-norm. By $\|\cdot\|$ we also denote the matrix norm induced by the vector norm $|\cdot|$.

Since T is finite there exist constants C_l such that

$$\|k_l\|_T \le C_l \quad \text{for } l = 1, \dots, L. \tag{8}$$

Since $|\cdot|$ is an L_p-norm and E_l is a diagonal matrix with entries in the interval $[0, 1]$, we have $\|E_l\|_T \le 1$ for $l = 1, \dots, L$. One easily obtains the estimate

$$|k(t)| \le \|k\|_T \le C := \sum_{l=1}^{L} C_l.$$

In exactly the same way as it was done in [5] for the waveform relaxation (see also [3], [7]) one finds the estimate

$$|k^{n*}(t)| \le C\frac{(Ct)^{n-1}}{(n-1)!}$$

and finally

$$\|\mathcal{K}^n\|_T \le \frac{(CT)^n}{n!}. \tag{9}$$

Note that in [3] general norms have been assumed. In that case one has to replace C by CE. We return to the fixpoint equation and we will show that it has a unique solution.

Since $\|\mathcal{K}\|_T \le CT$ we can find an interval $[0, T_0]$ with $T_0 < \frac{1}{C}$. Therefore the operator $\mathcal{K}$ is a contraction operator on that particular interval.

Using Stirling's formula we get immediately for $\rho(\mathcal{K})$

$$\rho(\mathcal{K}) = \lim_{n\to\infty} \|\mathcal{K}^n\|_T^{\frac{1}{n}} \le \lim_{n\to\infty} \left(\frac{(CT)^n}{n!}\right)^{\frac{1}{n}} = 0. \tag{10}$$

It follows that the spectral radius $\rho(\mathcal{K}) = 0$ on any finite interval $[0, T]$. If the product CT is large, however, a large n is necessary before the term on the right-hand side of (9) starts to decrease. The convergence behaviour for these cases is better described by the analysis for infinite domains of integration, which is discussed in the next section.

The 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 $[T_{i-1}, T_i]$.

A first form of the error of the nth approximation is given by

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

Using this result we can prove that the error of the nth approximation is bounded by

$$\|x - x_n\|_T \le \frac{(CT)^n}{n!}\left(\exp(CT)\|\varphi\|_T + \|x_0\|_T\right).$$

We summarize the results of this section in the following theorem.

THEOREM 4.2. *The multisplitting algorithm converges on any finite interval to the solution of* (2) *without restriction on A or the splitting.*

5. The stiff case. When one treats stiff ordinary differential equations, Lipschitz constants are usually large; even so, solutions are smooth. Hence the "classical" estimates from §4 would force us to use small windows. To treat the stiff situation adequately we should show convergence independently of the size of Lipschitz constants, or what is equivalent to this for arbitrary window size. Hence we shall give bounds and show convergence independent of the interval. This is reminiscent of the classical linear stability of schemes for stiff ordinary differential equations where one fixes the step size, but lets the number of steps go to infinity; i.e., one wants stability when integrating on an infinite interval. As with that classical theory one has to assume some dissipativeness of the differential equation. Here we request that A is an $\mathcal{M}$-matrix.

Let $X = \mathcal{L}^p(\mathbb{R}_+, \mathbb{R}^m)$ with $1 \leq p \leq \infty$. We want to show that in X,

$$\rho(\mathcal{K}) < 1.$$

Similar to how it was done in [4], one can prove

$$\rho(\mathcal{K}) = \sup_{\Re(z)\geq 0} \rho(\mathrm{K}(z)), \tag{11}$$

where K denotes the Laplace transform of the kernel $k(t)$.

Our goal is to show the following inequality:

$$\sup_{\Re(z)\geq 0} \rho\left(\sum_{l=1}^{L} E_l(zI + M_l)^{-1}N_l\right) < 1. \tag{12}$$

In the previous section we could achieve convergence without any restrictions on the matrix A or on the multisplitting. This is not possible here. We have to restrict ourselves to certain matrices and multisplittings.

We call a matrix $A \in \mathbb{R}^{m\times m}$ an *$\mathcal{M}$-matrix* [8] if $A(i,j) \leq 0$, for $i \neq j$, $1 \leq i, j \leq m$, and if the inverse A^{-1} exists with $A^{-1} \geq 0$. A splitting (M, N) of $A \in \mathbb{R}^{m\times m}$ is called *$\mathcal{M}$-splitting* if M is an $\mathcal{M}$-matrix and $N \geq 0$. Throughout this paper, we will denote by $A \geq 0$ that each entry of A is nonnegative. Let $|A|$ denote the matrix, where each element is the absolute value of the corresponding element of A. The *comparison matrix* $\langle A\rangle$ is defined by taking the negative absolute value of all off-diagonal elements and the absolute value of all diagonal elements.

We first present three lemmas, which will be used in the proof of the main theorem of this section.

LEMMA 5.1. *Let $A \in \mathbb{C}^{m\times m}$, then $\rho(A) \leq \rho(|A|)$.*

LEMMA 5.2. *Let $A \in \mathbb{R}^{m\times m}$ be a nonnegative matrix and $u \in \mathbb{R}^m$ be a positive vector. If $Au < u$, then $\rho(A) < 1$.*

LEMMA 5.3. *Let $z \in \mathbb{C}$ with $\Re(z) \geq 0$ and $A \in \mathbb{R}^{m\times m}$ be an $\mathcal{M}$-matrix. Then*

$$|(zI + A)^{-1}| \leq \langle zI + A\rangle^{-1}. \tag{13}$$

Lemma 5.2 is a well-known result that can be found in [2]. The proofs of Lemmas 5.1 and 5.3 for real matrices are given also in [2] and for complex matrices in [7]. Now we can prove the following theorem.

THEOREM 5.4. *Let $A \in \mathbb{R}^{m\times m}$ be an $\mathcal{M}$-matrix, and for each $l = 1, \ldots, L$ let (M_l, N_l) be an $\mathcal{M}$-splitting of A. Furthermore let $z \in \mathbb{C}$ with $\Re(z) \geq 0$. Then*

$$\rho\left(\sum_{l=1}^{L} E_l(zI + M_l)^{-1}N_l\right) < 1. \tag{14}$$

Proof. Using Lemma 5.1 it suffices to show that

$$\rho\left(\left|\sum_{l=1}^{L} E_l(zI + M_l)^{-1}N_l\right|\right) < 1.$$

Obviously $|\sum_{l=1}^{L} E_l(zI + M_l)^{-1}N_l|$ is a nonnegative, real matrix. With Theorem 2.8 from [9] we obtain that

$$\rho\left(\left|\sum_{l=1}^{L} E_l(zI + M_l)^{-1}N_l\right|\right) \le \rho\left(\sum_{l=1}^{L} |E_l|\,|(zI + M_l)^{-1}|\,|N_l|\right)$$

holds. Using that E_l and N_l are nonnegative, it suffices to show that

$$\rho\left(\sum_{l=1}^{L} E_l|(zI + M_l)^{-1}|\,N_l\right) < 1.$$

With Lemma 5.3 this reduces to show that

$$\rho\left(\sum_{l=1}^{L} E_l\langle zI + M_l\rangle^{-1}N_l\right) < 1.$$

For simplicity we will denote the above matrix by J.

It is easy to see that J is nonnegative and we obtain

$$N_l = |N_l| = \langle M_l\rangle - \langle A\rangle = \langle zI + M_l\rangle - \langle zI + A\rangle.$$

We get

$$\begin{aligned} J &= \sum_{l=1}^{L} E_l\langle zI + M_l\rangle^{-1}N_l \\ &= I - \sum_{l=1}^{L} E_l\langle zI + M_l\rangle^{-1}\langle zI + A\rangle. \end{aligned}$$

We choose a positive vector $u \in \mathbb{R}^m$ as

$$u = \langle zI + A\rangle^{-1}\mathbf{1}^T \quad \text{with } \mathbf{1} = (\underbrace{1, \dots, 1}_{m\text{times}}).$$

Then $Ju < u$ and we can use Lemma 5.2 to complete the proof. □

6. Numerical results. The problem we will look at is the semidiscretized one-dimensional heat equation

$$u_t = -u_{xx}, \quad u(0, x) = 0, \quad t \in [0, 1], \quad x \in [0, 1]$$

with boundary conditions $u(t, 0) = u(t, 1) = 1$. The dimension of the problem is $m = 400$, we use only one window, and as a stopping criterion we require that all components i of the solution satisfy

$$\sum_t |x_{n+1,i}(t) - x_{n,i}(t)| \le 10^{-2}, \tag{15}$$

where t is a timepoint. The subsystems were solved with the implicit Euler method and a constant stepsize $h = \frac{1}{20}$ is used.

How do we split? Since the structure of the matrix A is simple, we have chosen the subsets $S_1, S_2, \dots, S_L$ so 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 $S_{L-\beta+1}, \dots, S_L$ 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.

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 lth subsystem is in two subsystems.

The number of iterations necessary to satisfy a given stopping criterion is expected to be 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. Let us 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, which is better than the above first approach but not optimal. We call a component i of S_l a *border component* if $(i-1) \notin S_l$ or $(i+1) \notin S_l$. The upper border component of S_l is denoted by $l_{\max}$. 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 $i = l_{\max} - k + 1, \dots, l_{\max}$,

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

We will vary the number of splittings L and the number of overlapping components. The quantity that is given in Tables 1–3 is the necessary number of iterations such that (15) is satisfied.

TABLE 1
Number of subsystems $L = 5$.

Overlap	0	2	4	6	8	10	12	14	16	18	20
Iterations	842	395	268	207	170	144	126	113	102	93	86
Time[%]	100	52	36	29	25	22	20	19	18	17	16

TABLE 2
Number of subsystems $L = 10$.

Overlap	0	2	4	6	8	10	12	14	16	18	20
Iterations	824	511	369	293	245	211	187	167	152	140	129
Time[%]	100	78	60	51	46	42	40	38	38	37	37

TABLE 3
Number of subsystems $L = 15$.

Overlap	0	2	4	6	8	10	12	14	16	18	20
Iterations	1514	847	599	470	390	335	295	264	239	219	203
Time[%]	100	72	55	46	42	39	37	35	34	34	33

Numerical examples have also been computed on both shared memory and distributed memory MIMD computers. All show the excellent suitability of the algorithm on parallel machines.

Acknowledgements. The authors would like to thank Christian Lubich for showing us formula (11) and for many helpful discussions.

REFERENCES

[1] A. FROMMER AND G. MAYER, *Theoretische und praktische Ergebnisse zu Multisplitting-Verfahren auf Parallelrechnern*, Z. Angew. Math. Mech., 70 (1990), pp. T600–T602.
[2] A. FROMMER, *Lösung linearer Gleichungssysteme auf Parallelrechnern*, Vieweg-Verlag, Braunschweig, 1990.
[3] R. JELTSCH AND B. POHL, *Waveform relaxation with overlapping splittings*, Research report 91-02, Seminar für Angewandte Mathematik, ETH, Zürich, 1991.
[4] U. MIEKKALA AND O. NEVANLINNA, *Convergence of dynamic iteration methods for initial value problems*, SIAM J. Sci. Stat. Comput., 8 (1987), pp. 459–482.
[5] O. NEVANLINNA, *Remarks on Picard–Lindelöf iteration, Part* I, BIT 29 (1989), pp. 328–346; Part II, BIT 29 (1989), pp. 535–562.
[6] D. O'LEARY AND R. WHITE, *Multi-splittings of matrices and parallel solution of linear systems*, SIAM J. Alg. Disc. Meth., 6 (1985), pp. 630–640.
[7] B. POHL, *Ein Algorithmus zur Lösung von Anfangswertproblemen auf Parallelrechnern*, Informatik-Diss. ETH Zürich, 37, 1992.
[8] H. SCHNEIDER, *Theorems on M-splittings of a singular M-matrix which depend on graph structure*, Linear Algebra Appl., 58 (1984), pp. 407–424.
[9] R. S. VARGA, *Matrix Iterative Analysis*, Prentice Hall, Englewood Cliffs, NJ, 1962.
[10] R. E. WHITE, *Parallel algorithms for nonlinear problems*, SIAM J. Alg. Disc. Meth., 7 (1986), pp. 137–149.
[11] J. K. WHITE AND A. E. SANGIOVANNI-VINCENTELLI, *Relaxation Techniques for the Simulation of* VLSI *Circuits*, Kluwer Academic Publishers, Boston, 1987.