SIAM J. SCI. COMPUT.
Vol. 16, No. 5, pp. 1137–1158, September 1995

008

OPTIMAL CONVOLUTION SOR ACCELERATION OF WAVEFORM RELAXATION WITH APPLICATION TO PARALLEL SIMULATION OF SEMICONDUCTOR DEVICES*

MARK W. REICHELT†, JACOB K. WHITE‡, AND JONATHAN ALLEN‡

Abstract. In this paper we describe a novel generalized successive overrelaxation (SOR) algorithm for accelerating the convergence of the dynamic iteration method known as waveform relaxation. A new waveform convolution SOR algorithm is presented, along with a theorem for determining the optimal convolution SOR parameter. Both analytic and experimental results are given to demonstrate that the convergence of the waveform convolution SOR algorithm is substantially faster than that of the more obvious ordinary waveform SOR algorithm. Finally, to demonstrate the general applicability of this new method, it is used to solve the differential-algebraic system generated by spatial discretization of the time-dependent semiconductor device equations. Results from experiments on serial and parallel machines are presented to indicate a dramatic speedup over a more conventional method such as pointwise GMRES.

Key words. initial value problems, iterative methods, waveform relaxation, successive overrelaxation, dynamic iteration

AMS subject classifications. 65L05, 65F10

1. Introduction.

To achieve highest performance on a parallel computer, a numerical method must avoid frequent parallel synchronization [1]. The waveform relaxation (WR) approach to solving time-dependent initial-value problems is just such a method, as the iterates are vector waveforms over an interval, rather than vectors at single timepoints [10], [28], [7]. Like any relaxation scheme, efficiency depends on rapid convergence, and there have been several investigations into how to accelerate waveform relaxation [10], [23], including using multigrid [8], [26] and conjugate direction techniques [9].

In this paper, we investigate using successive overrelaxation (SOR) to accelerate WR convergence. In particular, we show that the pessimistic results about waveform SOR (WSOR) derived in [10] can be substantially improved by replacing multiplication with a fixed SOR parameter by convolution with an SOR kernel. A formula for the optimal SOR kernel is derived and is used to demonstrate the effectiveness of the convolution SOR (CSOR) by applying the approach to a model parabolic problem. Then, the general applicability of CSOR is demonstrated by using the method to solve the nonlinear time-dependent drift-diffusion equations associated with modeling semiconductor devices. Finally, results from experiments on serial and parallel machines are presented. These results indicate that on serial machines CSOR is competitive with conventional methods, such as pointwise GMRES, but that on parallel machines CSOR can have a dramatic advantage.

We begin in §2 by reviewing WSOR, and in §3 we relate the algorithm to pointwise SOR, to demonstrate the difficulty in accelerating WR with a fixed SOR parameter. In §4, we motivate the use of CSOR by examining the WSOR operator using simple Fourier analysis techniques. It is shown that on the infinite interval, the iteration operators associated with fixed parameter WSOR magnify errors in certain frequencies, even when the SOR parameter is in the range [0, 2]. The Fourier analysis in §4 easily leads to the correct conclusion that WR convergence is better accelerated using a frequency-dependent SOR parameter. However, the analysis is not

*Received by the editors July 12, 1993; accepted for publication (in revised form) July 7, 1994. This work was supported by a grant from IBM, the Defense Advanced Research Projects Agency contracts N00014-91-J-1698 and N00014-87-K-825, and National Science Foundation grant MIP-9117724.

†The MathWorks, Inc., 24 Prime Park Way, Natick, Massachusetts 01760 (mwr@mathworks.com).

‡Research Laboratory of Electronics, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139 (white@rle-vlsi.mit.edu, jallen@rle-vlsi.mit.edu).

directly applicable to the case of solving time-discretized initial-value problems. Instead, in §5, we use the connection between the z-transform and the spectrum of semi-infinite Toeplitz matrices to derive and to prove the optimality of a convolution SOR sequence. In §6, we briefly describe some aspects of implementing the CSOR algorithm, including the natural relaxation-Newton extension for solving nonlinear problems. We apply the method to device simulation on both serial and parallel machines in §7, and give conclusions and acknowledgments in §8.

2. Waveform SOR. In this section, we consider applying waveform relaxation methods to the model linear initial-value problem

$$\left(\tfrac{d}{dt} + \boldsymbol{A}\right)\boldsymbol{x}(t) = \boldsymbol{b}(t) \quad \text{with} \quad \boldsymbol{x}(0) = \boldsymbol{x}_0, \tag{1}$$

where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\boldsymbol{b}(t) \in \mathbb{R}^n$ is given for all $t \in [0, T]$, and $\boldsymbol{x}(t) \in \mathbb{R}^n$ is to be computed.

Consider the standard relaxation splitting $\boldsymbol{A} = \boldsymbol{D} - \boldsymbol{L} - \boldsymbol{U}$, where $\boldsymbol{D}$, $\boldsymbol{L}$, and $\boldsymbol{U}$ are the diagonal, strictly lower triangular, and strictly upper triangular pieces of $\boldsymbol{A}$ [27], [29]. Subtracting successive waveform relaxation iterations, the waveform Gauss–Jacobi (WGJ) and waveform Gauss–Seidel (WGS) iteration equations, respectively, may be written as

$$\left(\tfrac{d}{dt} + \boldsymbol{D}\right)\Delta\boldsymbol{x}^{k+1}(t) = (\boldsymbol{L} + \boldsymbol{U})\,\Delta\boldsymbol{x}^{k}(t), \tag{2}$$

$$\left(\tfrac{d}{dt} + \boldsymbol{D} - \boldsymbol{L}\right)\Delta\boldsymbol{x}^{k+1}(t) = \boldsymbol{U}\,\Delta\boldsymbol{x}^{k}(t), \tag{3}$$

where $\Delta\boldsymbol{x}^{k+1}(t) = \boldsymbol{x}^{k+1}(t) - \boldsymbol{x}^{k}(t)$.

The WSOR method for acceleration of WGS is a simple extension of algebraic SOR. To derive the WSOR iteration equation, compute a waveform $\hat{x}_i^{k+1}(t)$ on $t \in [0, T]$, solving an initial value problem as in WGS:

$$\begin{aligned}\left(\frac{d}{dt} + a_{ii}\right)\hat{x}_i^{k+1}(t) &= b_i(t) - \sum_{j=1}^{i-1} a_{ij}x_j^{k+1}(t) - \sum_{j=i+1}^{n} a_{ij}x_j^{k}(t),\\ \hat{x}_i^{k+1}(0) &= x_{0_i},\end{aligned} \tag{4}$$

and then update $x_i^k(t)$ in the iteration direction by multiplication with an overrelaxation parameter ω,

$$x_i^{k+1}(t) = x_i^k(t) + \omega \cdot \left[\hat{x}_i^{k+1}(t) - x_i^k(t)\right]. \tag{5}$$

Combining equations (4) and (5) yields

$$\begin{aligned}&\left(\frac{d}{dt} + a_{ii}\right)x_i^{k+1}(t)\\ &= (1-\omega)\left[\left(\frac{d}{dt} + a_{ii}\right)x_i^{k}(t)\right] + \omega\left[b_i(t) - \sum_{j=1}^{i-1} a_{ij}x_j^{k+1}(t) - \sum_{j=i+1}^{n} a_{ij}x_j^{k}(t)\right],\end{aligned}$$

which, after subtracting successive waveform relaxation iterations, leads to

$$\left(\tfrac{d}{dt} + \boldsymbol{D} - \omega\boldsymbol{L}\right)\Delta\boldsymbol{x}^{k+1}(t) = \left[(1-\omega)\left(\tfrac{d}{dt} + \boldsymbol{D}\right) + \omega\boldsymbol{U}\right]\Delta\boldsymbol{x}^{k}(t), \tag{6}$$

where $\Delta\boldsymbol{x}^{k+1}(t) = \boldsymbol{x}^{k+1}(t) - \boldsymbol{x}^{k}(t)$.

Note that deleting the $\frac{d}{dt}$ terms in equations (2), (3), and (6) results in precisely the standard algebraic relaxation and SOR iteration equations. Also note that WSOR as defined by (6) is *not* the same as the dynamic SOR iteration considered in [10] because of the derivative term on the right-hand side.

In this paper, we will consider the effect of replacing the overrelaxation multiplication in (5) by an *overrelaxation convolution* with a causal time-dependent SOR kernel $\omega(t)$. In the continuous time case, the overrelaxation equation becomes

$$x_i^{k+1}(t) = x_i^k(t) + \int_0^t \omega(\tau) \cdot \left[\hat{x}_i^{k+1}(t-\tau) - x_i^k(t-\tau)\right] d\tau, \tag{7}$$

and the iteration equation for the resulting method is

$$\begin{aligned} &\left(\frac{d}{dt} + \boldsymbol{D}\right) \Delta \boldsymbol{x}^{k+1}(t) - \boldsymbol{L} \int_0^t \omega(\tau)\, \Delta \boldsymbol{x}^{k+1}(t-\tau)\, d\tau \\ &= \left(\frac{d}{dt} + \boldsymbol{D}\right) \Delta \boldsymbol{x}^{k}(t) + \left[\boldsymbol{U} - \left(\frac{d}{dt} + \boldsymbol{D}\right)\right] \int_0^t \omega(\tau)\, \Delta \boldsymbol{x}^{k}(t-\tau)\, d\tau. \end{aligned} \tag{8}$$

3. Relation to pointwise SOR. Discretizing (1) in time using a multistep integration method [4] yields

$$\begin{aligned} &(\boldsymbol{I} + h\beta_0 \boldsymbol{A})\, \boldsymbol{x}[m] \\ &= h\beta_0 \boldsymbol{b}[m] + \sum_{j=1}^{s} \left\{h\beta_j\, (\boldsymbol{b}[m-j] - \boldsymbol{A}\boldsymbol{x}[m-j]) - \alpha_j \boldsymbol{x}[m-j]\right\}, \end{aligned} \tag{9}$$

where α_j and β_j are the coefficients of the multistep method, and $\boldsymbol{x}[m]$ denotes the discrete approximation to $\boldsymbol{x}(t)$ at $t = mh$. We now compare the convergence rate of the WSOR method to the convergence rate of pointwise SOR, in which algebraic SOR is used to solve the matrix problem at each timepoint.

The pointwise SOR iteration equations are derived by applying the relaxation splitting $\boldsymbol{A} = \boldsymbol{D} - \boldsymbol{L} - \boldsymbol{U}$ to equation (9) and taking the difference between the $(k+1)$st and kth iterations. More precisely, the pointwise SOR iteration equation applied to solve (9) for $\Delta \boldsymbol{x}^{k+1}[m] = \boldsymbol{x}^{k+1}[m] - \boldsymbol{x}^k[m]$ is

$$\begin{aligned} &[(\boldsymbol{I} + h\beta_0 \boldsymbol{D}) - \omega h\beta_0 \boldsymbol{L}]\, \Delta \boldsymbol{x}^{k+1}[m] \\ &= [(1-\omega)\, (\boldsymbol{I} + h\beta_0 \boldsymbol{D}) + \omega h\beta_0 \boldsymbol{U}]\, \Delta \boldsymbol{x}^k[m], \end{aligned} \tag{10}$$

where ω is the SOR parameter. It follows that the spectral radius of the iteration matrix generated by pointwise SOR at the mth timestep is

$$\rho\left([(\boldsymbol{I} + h\beta_0 \boldsymbol{D}) - \omega h\beta_0 \boldsymbol{L}]^{-1}[(1-\omega)\, (\boldsymbol{I} + h\beta_0 \boldsymbol{D}) + \omega h\beta_0 \boldsymbol{U}]\right). \tag{11}$$

If WSOR is used to solve the model problem (1), and a multistep method is used to solve iteration equation (6), then $\Delta \boldsymbol{x}^{k+1}[m]$ satisfies

$$\begin{aligned} &\sum_{j=0}^{s} \alpha_j \left[\Delta \boldsymbol{x}^{k+1}[m-j] - (1-\omega)\, \Delta \boldsymbol{x}^k[m-j]\right] \\ &= h \sum_{j=0}^{s} \beta_j \Big\{ -(\boldsymbol{D} - \omega \boldsymbol{L})\, \Delta \boldsymbol{x}^{k+1}[m-j] + [(1-\omega)\, \boldsymbol{D} + \omega \boldsymbol{U}]\, \Delta \boldsymbol{x}^k[m-j]\Big\}. \end{aligned} \tag{12}$$

This can be rewritten as the discrete-time analogue of (6):

$$\begin{aligned} &\sum_{j=0}^{s} [(\alpha_j \boldsymbol{I} + h\beta_j \boldsymbol{D}) - \omega h\beta_j \boldsymbol{L}] \Delta \boldsymbol{x}^{k+1}[m-j] \\ &= \sum_{j=0}^{s} [(1-\omega)\, (\alpha_j \boldsymbol{I} + h\beta_j \boldsymbol{D}) + \omega h\beta_j \boldsymbol{U}] \Delta \boldsymbol{x}^k[m-j]. \end{aligned} \tag{13}$$

As the similarities of equations (10) and (13) suggest, if the time interval is finite, i.e., the number of timesteps is some finite L, then for a given timestep h and a given SOR parameter ω, the time-discretized WSOR method has the same asymptotic convergence rate as the pointwise SOR method.

THEOREM 3.1. *On a finite simulation interval, the iterations defined by* (10) *and* (13) *have the same asymptotic convergence rate.*

Proof. Let $\boldsymbol{y}^k$ denote the large vector consisting of the concatenation of vectors $\Delta\boldsymbol{x}^k[m]$ at all L discrete timepoints; i.e., $\boldsymbol{y}^k = \left[\Delta\boldsymbol{x}^k[1]^T, \ldots, \Delta\boldsymbol{x}^k[L]^T\right]^T$. Collecting the equations (13) generated at each timepoint into one large matrix equation in terms of vectors $\boldsymbol{y}^{k+1}$ and $\boldsymbol{y}^k$ yields $\boldsymbol{M}\Delta\boldsymbol{y}^{k+1} = \boldsymbol{N}\Delta\boldsymbol{y}^k$, where $\boldsymbol{M}, \boldsymbol{N} \in \mathbb{R}^{Ln\times Ln}$ are block lower triangular banded matrices, with blocks of size $n \times n$, and with block bandwidth s. It is then easily seen that $\boldsymbol{M}^{-1}\boldsymbol{N}$ is block lower triangular, and therefore has eigenvalues given by the eigenvalues of its diagonal blocks, which are

$$[(\boldsymbol{I} + h\beta_0\boldsymbol{D}) - \omega h\beta_0\boldsymbol{L}]^{-1}[(1-\omega)(\boldsymbol{I} + h\beta_0\boldsymbol{D}) + \omega h\beta_0\boldsymbol{U}]. \tag{14}$$

Therefore, $\rho(\boldsymbol{M}^{-1}\boldsymbol{N})$ is given by (11), implying that the iterations defined by (10) and (13) have identical asymptotic convergence rates. □

Theorem 3.1 suggests that parameter ω for WSOR should be chosen to be precisely equal to the optimum parameter for the pointwise SOR method. However, this does not necessarily lead to fastest convergence, as the following example illustrates.

Example 3.1. Let $t \in [0, 2048]$, $\boldsymbol{x}(0) = 0$, and let matrix $\boldsymbol{A} \in \mathbb{R}^{32\times 32}$ and time-dependent input vector $\boldsymbol{b}(t) \in \mathbb{R}^{32}$ of the model problem (1) be given by

$$\boldsymbol{A} = \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix} \quad \text{and}$$

$$\boldsymbol{b}(t) = \begin{bmatrix} b_1(t) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \text{where} \quad b_1(t) = \begin{cases} 1 - \cos\left(\dfrac{2\pi t}{256}\right) & \text{if } t \le 256, \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

Consider the four problems generated by discretizing in time with the first-order backward difference formula, using 64, 128, 256, and 512 uniform timesteps of size $h = 32$, 16, 8, and 4, respectively.

Since the tridiagonal matrix $\boldsymbol{A}$ is symmetric and is consistently ordered [29], [27], the matrix $(\boldsymbol{I} + h\beta_0\boldsymbol{A})$ of the pointwise time-discretized model problem (9) is also consistently ordered and the optimum pointwise SOR parameter ω_{opt} is given by

$$\omega_{\text{opt}} = \frac{2}{1 + \sqrt{1 - \mu_1^2}} \tag{16}$$

where $\mu_1 = \rho(\boldsymbol{H}_{GJ})$ is the spectral radius of the pointwise Gauss–Jacobi iteration matrix $\boldsymbol{H}_{GJ} = (\boldsymbol{I} + h\beta_0\boldsymbol{D})^{-1}(h\beta_0\boldsymbol{L} + h\beta_0\boldsymbol{U})$. For the four problems with 64, 128, 256, and 512 timesteps, the optimum pointwise parameters ω_{opt} are approximately 1.669, 1.586, 1.482, and 1.364, respectively.

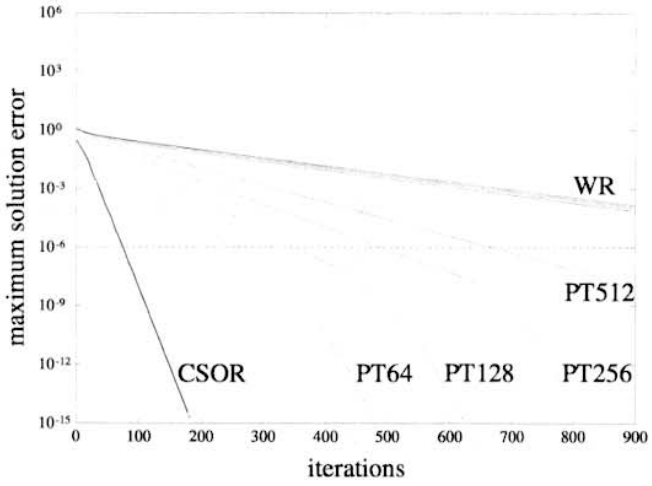

FIG. 1. *Convergence of WSOR using the pointwise optimal parameter (PT) compared with WR and CSOR, with* 64, 128, 256, *and* 512 *timesteps. Note that the four CSOR experiments have the same convergence rate.*

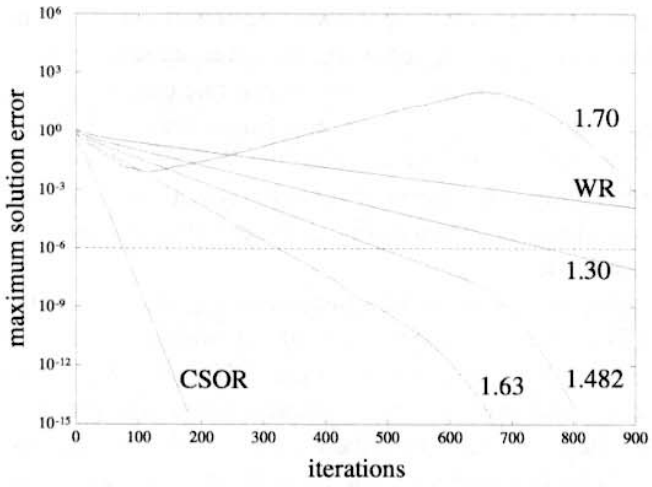

FIG. 2. *Effect on convergence of the* 256*-timestep WSOR example of varying the SOR parameter from the pointwise optimum* $\omega_{\text{opt}} = 1.482$.

Curves PT64, PT128, PT256, and PT512 of Fig. 1 show convergence versus iteration of the WSOR method for the four problems with their optimum pointwise SOR parameters ω_{opt}. Note that as the total number of timesteps is increased, the initial convergence rate slows, approaching a limiting value of the convergence rate of the unaccelerated Gauss–Seidel WR algorithm (shown as WR in Fig. 1). In each case, the convergence rate of WSOR eventually approaches the expected asymptotic value of $\omega_{\text{opt}} - 1$. Note that if the iteration is terminated when the error is less than a reasonable tolerance, for example 10^{-6}, the asymptotic convergence rate is *never* reached. For comparison, Fig. 1 also shows four convergence plots of the new CSOR method to be introduced in the following sections. The four CSOR experiments have the same convergence rate, independent of the number of timesteps, so the four separate plots appear as one line in Fig. 1.

To illustrate the effect of choosing a different SOR parameter ω, Fig. 2 shows the convergence versus iteration of the 256-timestep example for WSOR with values of the SOR parameter ω not equal to the pointwise optimum $\omega_{\text{opt}} = 1.482$. When $\omega = 1.30 < \omega_{\text{opt}}$,

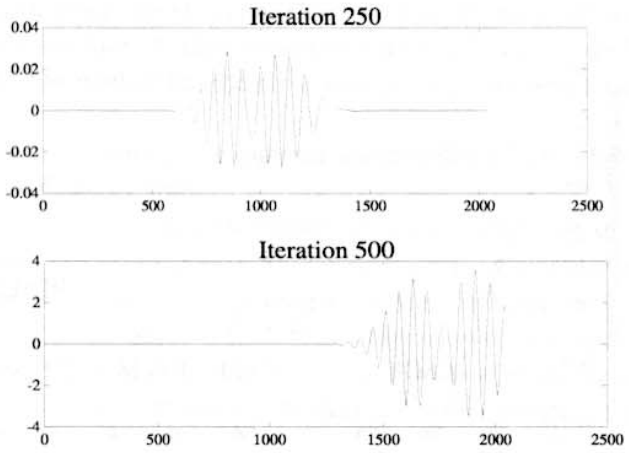

FIG. 3. *Delta waveform* $\Delta x_{16}^{k+1}(t) = x_{16}^{k+1}(t) - x_{16}^{k}(t)$ *versus time after iterations* 250 *and* 500, *for the* 256-*timestep WSOR method using* $\omega = 1.70$, *showing the growth and translation of an oscillating region. Note that the vertical scales on the axes differ by two orders of magnitude.*

the convergence curve lies between the pointwise optimum curve and the WR convergence curve; i.e., both initial and asymptotic convergence rates are slower. By increasing the SOR parameter to $\omega = 1.63 > \omega_{\text{opt}}$, the initial convergence rate can be made faster at the expense of slowing down the asymptotic convergence rate. But as the $\omega = 1.70$ curve shows, once the SOR parameter is increased beyond some point, the WSOR method may appear to diverge before eventually converging. The intermediate solutions produced by the $\omega = 1.70$ example contain spurious oscillations, as shown in Fig. 3. Note both the growth and translation of the oscillating region with iteration.

In general, the optimum pointwise SOR parameter ω_{opt} does not dramatically improve the convergence rate of WSOR because the matrix $\boldsymbol{M}^{-1}\boldsymbol{N}$ which describes WSOR convergence is far from normal. Note that this is the case even if $\boldsymbol{D}^{-1}(\boldsymbol{L} + \boldsymbol{U})$ is normal. This suggests that although the spectral radius of the iteration matrix determines the *asymptotic* convergence rate of WSOR, it does not determine the effective observable convergence rate. The effective convergence rate could be characterized, for example, by computing the pseudoeigenvalues of the WSOR iteration matrix [25]. In the following section, we take an alternate approach, based on results in [13] and [14].

4. Frequency-dependent SOR. The seemingly spontaneous appearance of spurious oscillations in the fixed-parameter WSOR iterates, as shown in Fig. 3, is most easily explained by examining the behavior of iterates on the semi-infinite interval, $t \in [0, \infty)$, which contain primarily a single Fourier component. Specifically, suppose that

$$\Delta \boldsymbol{x}^k(t) = \Delta \boldsymbol{x}^k(i\Omega)e^{-i\Omega t} + B^k(t)$$

where $\lim_{t\to\infty} B^k(t) = 0$. Then, if the eigenvalues of $\boldsymbol{D}$ lie in the open left half plane,

$$\Delta \boldsymbol{x}^{k+1}(i\Omega) = \boldsymbol{H}(i\Omega)\ \Delta \boldsymbol{x}^k(i\Omega) + B^{k+1}(t)$$

where $\lim_{t\to\infty} B^{k+1}(t) = 0$. For the WGJ, WGS, and WSOR methods, $\boldsymbol{H}(i\Omega)$ is given by

$$\boldsymbol{H}_{\text{GJ}}(i\Omega) = \boldsymbol{D}_{\Omega}^{-1}(\boldsymbol{L} + \boldsymbol{U}), \tag{17}$$

$$\boldsymbol{H}_{\text{GS}}(i\Omega) = (\boldsymbol{D}_{\Omega} - \boldsymbol{L})^{-1}\boldsymbol{U}, \tag{18}$$

$$\boldsymbol{H}_{\text{SOR}}(i\Omega) = (\boldsymbol{D}_{\Omega} - \omega\boldsymbol{L})^{-1}[(1-\omega)\boldsymbol{D}_{\Omega} + \omega\boldsymbol{U}], \tag{19}$$

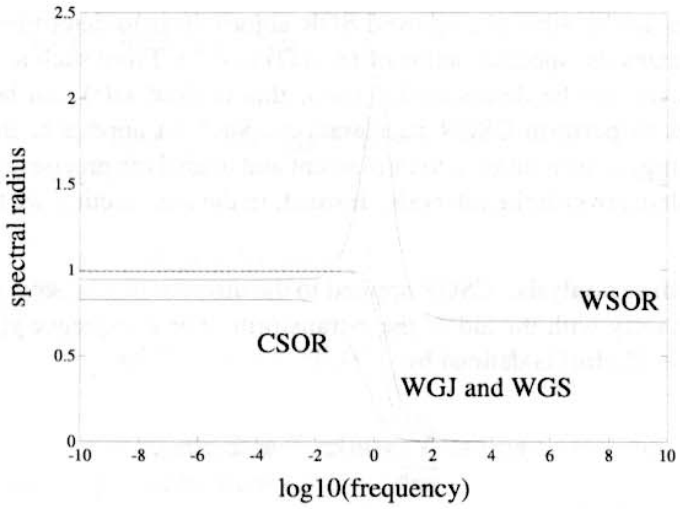

FIG. 4. *The spectral radii as functions of frequency Ω of the WGJ, WGS (dashed) and WSOR iteration matrices for the 32 × 32 version of the continuous-time problem of Example 3.1, with $\omega = 1.70$ for WSOR. For comparison, the plot also shows the spectral radius versus frequency for the CSOR method using the optimal CSOR sequence.*

respectively, where $\boldsymbol{D}_\Omega = i\Omega\boldsymbol{I} + \boldsymbol{D}$. An interpretation of equations (17)–(19) is that when examining convergence on the semi-infinite interval, and ignoring behavior which decays to zero for large t, the spectral radius $\rho(\boldsymbol{H}(i\Omega))$ gives the asymptotic convergence rate for errors in Fourier component Ω.

Figure 4 is a plot of the spectral radii of $\boldsymbol{H}_{\mathrm{GJ}}(i\Omega)$, $\boldsymbol{H}_{\mathrm{GS}}(i\Omega)$, and $\boldsymbol{H}_{\mathrm{SOR}}(i\Omega)$ for the 32 × 32 continuous-time problem given in Example 3.1, using $\omega = 1.70$ for $\boldsymbol{H}_{\mathrm{SOR}}(i\Omega)$. For comparison, the plot also shows the spectral radius versus frequency for the CSOR method using the optimal CSOR sequence, as will be described in the next section. For the 32 × 32 problem, the WGJ and WGS spectral radii are only slightly less than unity for low frequencies, and high frequency components of the error are damped much more quickly than low frequency components. However, the spectral radius $\rho(\boldsymbol{H}_{\mathrm{SOR}}(i\Omega))$ is greater than unity over a range of frequencies, and therefore the WSOR iteration magnifies errors in this higher frequency range. This aspect of WR convergence is also predicted by the more formal treatment in [10], and explains the appearance and growth, but not the time-translation, of the spurious oscillations seen in Figs. 2 and 3.

As is clear from the above, the growth in errors at higher frequencies is caused by selecting a fixed SOR parameter based on insuring rapid decay of low frequency errors. Then, at the higher frequencies, where unaccelerated relaxation is already reducing errors rapidly, this SOR parameter is too large. Therefore, it is clear that this growth in high frequency errors can be eliminated without slowing low frequency convergence by simply allowing the SOR parameter to vary with frequency. That is, if $\boldsymbol{x}^k(t) = \boldsymbol{x}^k(i\Omega)e^{-i\Omega t}$, then equation (5) can be generalized to

$$x_i^{k+1}(i\Omega) = x_i^k(i\Omega) + \omega(i\Omega)\cdot\left[\hat{x}_i^{k+1}(i\Omega) - x_i^k(i\Omega)\right], \tag{20}$$

where $\omega(i\Omega)$ is the frequency-dependent SOR parameter. The resulting frequency-dependent SOR operator is given by

$$\boldsymbol{H}_C(i\Omega) = [\boldsymbol{D}_\Omega - \omega(i\Omega)\,\boldsymbol{L}]^{-1}[(1-\omega(i\Omega))\boldsymbol{D}_\Omega + \omega(i\Omega)\,\boldsymbol{U}], \tag{21}$$

where $\boldsymbol{D}_\Omega = i\Omega\boldsymbol{I} + \boldsymbol{D}$, as before.

One approach to deriving an improved SOR algorithm is to determine, for each Ω, the $\omega(i\Omega)$ that minimizes the spectral radius of $\boldsymbol{H}_C(i\Omega)$ in (21). From such a frequency-domain description, an $\omega(t)$ can be determined. Then, this derived $\omega(t)$ can be convolved with iterate differences to perform CSOR acceleration. Such an approach, though simple and intuitively appealing, is difficult both to implement and to analyze precisely when considering initial-value problems over finite intervals. Instead, in the next section we take a more direct approach.

5. Discrete-time analysis. CSOR applied to the discrete-time system (9) can be derived and analyzed precisely with the aid of the z-transform. For a sequence $\boldsymbol{y}[m]$, the unilateral z-transform $\boldsymbol{y}(z) = \mathcal{Z}\,\boldsymbol{y}[m]$ is defined by

$$\boldsymbol{y}(z) \equiv \sum_{m=0}^{\infty} \boldsymbol{y}[m]\, z^{-m} \equiv \mathcal{Z}\,\boldsymbol{y}[m], \tag{22}$$

where $z \in \mathbb{C}$ [15]. Using the transform representation, the WGJ, WGS, and ordinary WSOR methods applied to the discrete-time problem (9) are of the form $\Delta\boldsymbol{x}^{k+1}(z) = \boldsymbol{H}(z)\ \Delta\boldsymbol{x}^{k}(z)$, where

$$\boldsymbol{H}_{\mathrm{GJ}}(z) = \boldsymbol{D}_z^{-1}(\boldsymbol{L} + \boldsymbol{U}), \tag{23}$$

$$\boldsymbol{H}_{\mathrm{GS}}(z) = (\boldsymbol{D}_z - \boldsymbol{L})^{-1}\boldsymbol{U}, \tag{24}$$

$$\boldsymbol{H}_{\mathrm{SOR}}(z) = (\boldsymbol{D}_z - \omega\boldsymbol{L})^{-1}[(1-\omega)\boldsymbol{D}_z + \omega\boldsymbol{U}]. \tag{25}$$

For a general multistep method with uniform timestep h, the diagonal matrix $\boldsymbol{D}_z$ is

$$\boldsymbol{D}_z = \frac{\sum_{j=0}^{s} \alpha_j z^{-j}}{h \sum_{j=0}^{s} \beta_j z^{-j}}\, \boldsymbol{I} + \boldsymbol{D}. \tag{26}$$

Like equations (17)–(19), these operators correspond to standard relaxation and SOR matrices with $\boldsymbol{D}$ replaced by the diagonal matrix $\boldsymbol{D}_z$. Note that $\boldsymbol{H}_{\mathrm{SOR}}(z)$ in (25) can be obtained by applying the z-transform to (13).

To derive the iteration equation for discrete-time CSOR, let overrelaxation equation (5) be replaced by a convolution sum with sequence $\omega[m]$,

$$x_i^{k+1}[m] = x_i^k[m] + \sum_{\ell=0}^{m} \omega[\ell] \cdot \left(\hat{x}_i^{k+1}[m-\ell] - x_i^k[m-\ell]\right). \tag{27}$$

The z-transform of this convolution equation is

$$x_i^{k+1}(z) = x_i^k(z) + \omega(z) \cdot \left[\hat{x}_i^{k+1}(z) - x_i^k(z)\right], \tag{28}$$

where $\omega(z)$ is the z-transform of the sequence $\omega[m]$. The z-transform of the resulting discrete-time CSOR operator is given by

$$\boldsymbol{H}_C(z) = [\boldsymbol{D}_z - \omega(z)\,\boldsymbol{L}]^{-1}[(1-\omega(z))\boldsymbol{D}_z + \omega(z)\,\boldsymbol{U}], \tag{29}$$

with $\omega(z) = \mathcal{Z}\,\omega[m]$ and $\boldsymbol{D}_z$ given by (26) above.

The CSOR operator on sequences is derived by combining the convolution equation (27) with the discrete-time problem (9) and subtracting successive iterations. The resulting CSOR iteration equation for a general multistep method is given by

$$
\begin{aligned}
&\sum_{j=0}^{s} \left(\alpha_j \boldsymbol{I} + h\beta_j \boldsymbol{D}\right) \Delta \boldsymbol{x}^{k+1}[m-j] \\
&\quad - \sum_{j=0}^{s} h\beta_j \boldsymbol{L} \sum_{\ell=0}^{m-j} \omega[\ell]\, \Delta \boldsymbol{x}^{k+1}[m-j-\ell] \\
&= \sum_{j=0}^{s} \left(\alpha_j \boldsymbol{I} + h\beta_j \boldsymbol{D}\right) \Delta \boldsymbol{x}^{k}[m-j] \\
&\quad + \sum_{j=0}^{s} [h\beta_j \boldsymbol{U} - (\alpha_j \boldsymbol{I} + h\beta_j \boldsymbol{D})] \sum_{\ell=0}^{m-j} \omega[\ell]\, \Delta \boldsymbol{x}^{k}[m-j-\ell].
\end{aligned} \tag{30}
$$

Though lengthy, this is precisely the same as the WSOR iteration equation (13), with overrelaxation multiplication replaced by a convolution sum.

Let $\mathcal{K}$ denote the CSOR operator mapping from sequence $\Delta \boldsymbol{x}^k[m]$ to $\Delta \boldsymbol{x}^{k+1}[m]$, defined by

$$
\Delta \boldsymbol{x}^{k+1}[m] = \mathcal{K} \Delta \boldsymbol{x}^k[m] \equiv \sum_{\ell=0}^{m} \boldsymbol{h}_C[\ell]\, \Delta \boldsymbol{x}^k[m-\ell], \tag{31}
$$

where $\boldsymbol{h}_C[m] \in \mathbb{R}^{n \times n}$ is derived from equation (30). Concatenating the vectors $\Delta \boldsymbol{x}[m]$ at all timepoints, the operator $\mathcal{K}$ may be written in block matrix form as

$$
\begin{bmatrix} \boldsymbol{M}_1 & & & \\ \boldsymbol{M}_2 & \boldsymbol{M}_1 & & \\ \boldsymbol{M}_3 & \boldsymbol{M}_2 & \boldsymbol{M}_1 & \\ \vdots & \ddots & \ddots & \ddots \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{x}^{k+1}[1] \\ \Delta \boldsymbol{x}^{k+1}[2] \\ \Delta \boldsymbol{x}^{k+1}[3] \\ \vdots \end{bmatrix} = \begin{bmatrix} \boldsymbol{N}_1 & & & \\ \boldsymbol{N}_2 & \boldsymbol{N}_1 & & \\ \boldsymbol{N}_3 & \boldsymbol{N}_2 & \boldsymbol{N}_1 & \\ \vdots & \ddots & \ddots & \ddots \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{x}^{k}[1] \\ \Delta \boldsymbol{x}^{k}[2] \\ \Delta \boldsymbol{x}^{k}[3] \\ \vdots \end{bmatrix}. \tag{32}
$$

From this, it is easily seen that the CSOR operator $\mathcal{K}$ is a block Toeplitz operator [17], corresponding to the semi-infinite, block lower triangular matrix $\boldsymbol{M}^{-1}\boldsymbol{N}$ specified by (30). This leads to the following lemma.

LEMMA 5.1. *On the infinite interval, the spectral radius of the operator $\mathcal{K}$ defined in* (31) *is determined by the spectral radius of the matrix $\boldsymbol{H}_C(z)$ given by* (29),

$$
\rho(\mathcal{K}) = \max_{|z| \geq 1} \rho(\boldsymbol{H}_C(z)). \tag{33}
$$

Proof. Because operator $\mathcal{K}$ is a block lower triangular semi-infinite Toeplitz operator, the spectrum of $\mathcal{K}$ equals the spectrum of the matrix-valued symbol $f(z)$ of the Toeplitz operator, where z ranges over the exterior of the unit disk [11]. The symbol of the Toeplitz operator $\mathcal{K}$ is precisely the matrix $\boldsymbol{H}_C(z)$. □

Thus, minimizing the spectral radius of the CSOR operator on sequences is the same as minimizing the maximum over $|z| \geq 1$ of the spectral radius of the matrix $\boldsymbol{H}_C(z)$ given by (29). This leads to the following theorem, the main result of this paper.

THEOREM 5.2. *Consider the time-discretized model problem* (9), *where $\boldsymbol{A}$ is consistently ordered. If, at a particular $z \in \mathbb{C}$, the spectrum $\mu(z)$ of $\boldsymbol{H}_{GJ}(z)$ lies on a line segment $[-\mu_1(z), \mu_1(z)]$, with $\mu_1(z) \in \mathbb{C}$ and $|\mu_1(z)| < 1$, then the spectral radius of $\boldsymbol{H}_C(z)$ is minimized by the unique optimum $\omega_{\text{opt}}(z) \in \mathbb{C}$ given by*

$$
\omega_{\text{opt}}(z) = \frac{2}{1 + \sqrt{1 - \mu_1(z)^2}} \tag{34}
$$

where $\sqrt{\cdot}$ denotes the root with the positive real part. Furthermore, the sequence $\omega_{\text{opt}}[m] = \mathcal{Z}^{-1}\omega_{\text{opt}}(z)$ is optimal in the sense that it minimizes the spectral radius of the operator $\mathcal{K}$.

Proof. For brevity, the argument (z) will be omitted in the following; i.e., ω will denote $\omega(z)$ and $\boldsymbol{H}_C$ will denote the CSOR operator (evaluated at z) computed using CSOR parameter $\omega(z)$.

Let $\mu_i = r_i\mu_1$, where $r_i \in [-1, 1]$, denote each eigenvalue of $\boldsymbol{H}_{\text{GJ}}$ given by (23). Classical SOR theory [29], [27] guarantees that for each $\mu_i = r_i\mu_1$, there is an eigenvalue λ_i of $\boldsymbol{H}_C$ which satisfies

$$\lambda_i - \omega r_i \mu_1 \sqrt{\lambda_i} + (\omega - 1) = 0, \tag{35}$$

and therefore, from the quadratic formula,

$$\sqrt{\lambda_i} = \frac{r_i\mu_1\omega}{2} + \sqrt{\left(\frac{r_i\mu_1\omega}{2}\right)^2 - \omega + 1}. \tag{36}$$

Let ω be the conjectured optimal ω_{opt}. Combining equation (34) with (36) yields

$$\sqrt{|\lambda_i|} = \left|\tfrac{1}{2}\mu_1\omega_{\text{opt}}\left[r_i + \sqrt{r_i^2 - 1}\right]\right| = \left|\tfrac{1}{2}\mu_1\omega_{\text{opt}}\right|, \tag{37}$$

where the last equality follows from the fact that $\left|r_i + \sqrt{r_i^2 - 1}\right| = 1$ for $r_i \in [-1, 1]$. And as (37) holds for all i, with parameter $\omega = \omega_{\text{opt}}$,

$$\rho(\boldsymbol{H}_C) = |\lambda_i| = \left|\left(\tfrac{1}{2}\mu_1\omega_{\text{opt}}\right)^2\right| = \left|\omega_{\text{opt}} - 1\right|. \tag{38}$$

Equation (38) implies that $\rho(\boldsymbol{H}_C)$ cannot be decreased by picking a value of ω such that $|\omega - 1| > \left|\omega_{\text{opt}} - 1\right|$. This follows from the fact that

$$\rho(\boldsymbol{H}_C) \geq |\omega - 1| \tag{39}$$

for any ω [29], [27].

To show that $\rho(\boldsymbol{H}_C)$ also cannot be decreased by choosing a value of ω such that $|\omega - 1| < \left|\omega_{\text{opt}} - 1\right|$, consider the eigenvalue λ_j corresponding to μ_1:

$$\sqrt{\lambda_j} = f_+(\omega) = \frac{\mu_1\omega}{2} + \sqrt{\frac{\mu_1^2\omega^2}{4} - \omega + 1} \tag{40}$$

and note that $f_+ : \mathbb{C} \to \mathbb{C}$, given by equation (40), is a single-valued, continuous function that is analytic except at

$$\omega_1, \omega_2 = \frac{2}{1 \pm \sqrt{1 - \mu_1^2}}. \tag{41}$$

Since $|\mu_1| < 1$, points ω_1 and ω_2 lie in the interior and exterior, respectively, of the circle $|\omega - 1| = 1$ in the complex ω-plane. Note that ω_1 equals the conjectured ω_{opt} from equation (34).

Let D denote the interior of the curve given by the perimeter of the circle $|\omega - 1| = 1$ with a cut along the line defined by the circle's center and ω_1. The cut follows the line from the perimeter down to ω_1, and then back up the other side to the perimeter, as shown in Fig. 5. The function f_+ is nonzero everywhere within D, since equation (40) implies that a zero can

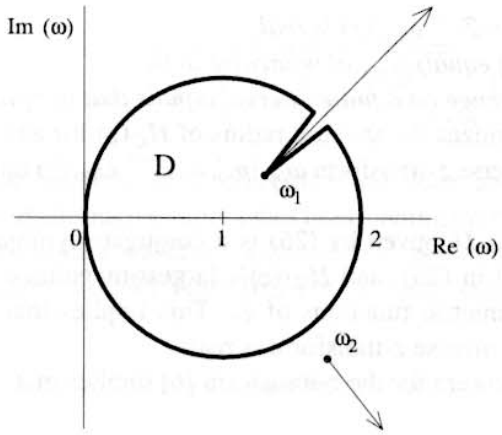

FIG. 5. *The region D and branch cuts in the complex ω-plane.*

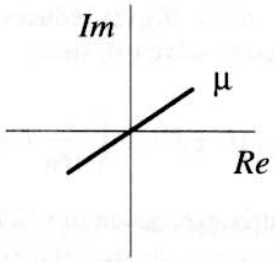

FIG. 6. *The optimal CSOR parameter theorem requires the spectrum $\mu(z)$ to lie on a line segment $[-\mu_1(z), \mu_1(z)]$, with $\mu_1(z) \in \mathbb{C}$ and $|\mu_1(z)| < 1$.*

occur only at $\omega = 1$, and $f_+(1) = \mu_1$. Therefore, the minimum modulus theorem [20] implies that $|f_+(\omega)|$ attains its minimum value somewhere on the boundary of D. Finally, the lower bound in (39) implies that $\omega_1 = \omega_{\text{opt}}$ in (34) is the only point on D which can achieve as low a $\rho(\boldsymbol{H}_C)$ as given in (38), completing the proof. □

Note that when the eigenvalues μ lie on a real line segment, this theorem contains the classic SOR theorem [29], [27] as a special case, and extends the theorem to a limited class of complex matrices [3], [29]. Whereas the classic SOR theorem requires the spectrum μ of the Gauss–Jacobi matrix $\boldsymbol{H}_{\text{GJ}}$ to lie on the real line segment $[-\mu_1, \mu_1]$ with $|\mu_1| < 1$, the CSOR theorem requires the spectrum $\mu(z)$ of $\boldsymbol{H}_{\text{GJ}}(z)$ to lie on a line segment $[-\mu_1(z), \mu_1(z)]$, with $\mu_1(z) \in \mathbb{C}$ and $|\mu_1(z)| < 1$, as shown in Fig. 6. For the discrete-time problem (9), this requirement is satisfied, for example, by the class of diagonally dominant symmetric matrices $\boldsymbol{A}$ with constant diagonal $\boldsymbol{D} = d\boldsymbol{I}$, since the spectrum $\mu(z)$ of the $\boldsymbol{H}_{\text{GJ}}(z)$ for such matrices is

$$\mu(z) = \frac{\mu_0 dh \sum_{j=0}^{s} \beta_j z^{-j}}{\sum_{j=0}^{s} \left[\alpha_j z^{-j} + dh\beta_j z^{-j}\right]} \tag{42}$$

where μ_0 denotes the eigenvalues of $\boldsymbol{D}^{-1}(\boldsymbol{L} + \boldsymbol{U})$. For any particular z, the real line segment μ_0 is mapped to a rotated line segment.

Theorem 5.2 leads immediately to the following corollary.

COROLLARY 5.3. *If $\omega_{\text{opt}}(z)$ given by* (34) *is analytic, then the corresponding sequence $\omega_{\text{opt}}[m] = \mathcal{Z}^{-1}\,\omega_{\text{opt}}(z)$ is optimal for discrete-time convolution SOR; i.e., it minimizes $\rho(\mathcal{K})$, the spectral radius of the operator on sequences. In addition, the following is true.*

1. *Sequence* $\omega_{\text{opt}}[m] = \mathcal{Z}^{-1}\,\omega_{\text{opt}}(z)$ *is real.*
2. *Initial value* $\omega_{\text{opt}}[0]$ *equals* ω_{opt} *of pointwise SOR.*
3. *Asymptotic convergence on a finite interval equals that of optimal pointwise SOR.*

Proof. Since $\omega(z)$ minimizes the spectral radius of $\boldsymbol{H}_C(z)$ for any z, Lemma 5.1 proves that convolution with the inverse z-transform $\omega_{\text{opt}}[m] = \mathcal{Z}^{-1}\,\omega_{\text{opt}}(z)$ optimally minimizes the spectral radius of operator $\mathcal{K}$.

1. The diagonal matrix $\boldsymbol{D}_z$ given by (26) is a conjugate-symmetric function of z, so that $\boldsymbol{H}_{\text{GJ}}(z)$ defined in (23), and $\boldsymbol{H}_{\text{GJ}}(z)$'s largest-magnitude eigenvalue $\mu_1(z)$ are also conjugate-symmetric functions of z. This implies that $\omega_{\text{opt}}(z)$ is conjugate-symmetric, and the inverse z-transform is real.
2. The initial value theorem for the z-transform [6] implies that

$$\omega_{\text{opt}}[0] = \lim_{z\to\infty} \omega_{\text{opt}}(z) = \frac{2}{1+\sqrt{1-\left(\lim\limits_{z\to\infty}\mu_1(z)\right)^2}}.$$

   In the limit $z \to \infty$, the matrix $\boldsymbol{H}_{\text{GJ}}(z)$ reduces precisely to the pointwise Gauss–Jacobi iteration matrix used to solve (9), since

$$\lim_{z\to\infty} \boldsymbol{D}_z^{-1}(\boldsymbol{L}+\boldsymbol{U}) = \left(\frac{1}{h\beta_0}\,\boldsymbol{I}+\boldsymbol{D}\right)^{-1}(\boldsymbol{L}+\boldsymbol{U}).$$

3. Referring to the block matrix expression of CSOR operator $\mathcal{K}$ (32), it is easily seen that on a finite simulation interval, the spectrum of the block lower triangular operator $\mathcal{K}$ is equal to the spectrum of its block diagonal $\boldsymbol{M}_1^{-1}\boldsymbol{N}_1$ where

$$\begin{aligned}\boldsymbol{M}_1 &= [(\boldsymbol{I}+h\boldsymbol{D}) - \omega_{\text{opt}}[0]h\boldsymbol{L}],\\ \boldsymbol{N}_1 &= [(1-\omega_{\text{opt}}[0])(\boldsymbol{I}+h\boldsymbol{D}) + \omega_{\text{opt}}[0]h\boldsymbol{U}].\end{aligned}$$

   Since $\omega[0]$ equals the optimal ω_{opt} of pointwise SOR, the matrix $\boldsymbol{M}_1^{-1}\boldsymbol{N}_1$ is precisely equal to the iteration matrix of optimal pointwise SOR. □

Note that the result that the asymptotic convergence of optimal CSOR on a finite interval equals that of optimal pointwise SOR is nearly irrelevant, since the operator $\mathcal{K}$ is so far from normal. As Example 3.1 and Fig. 1 showed for WSOR, the asymptotic convergence rate given by the spectral radius of $\boldsymbol{M}^{-1}\boldsymbol{N}$ may have little to do with the convergence rate observed in practice.

6. Implementation of convolution SOR. In practice, standard SOR algorithms use the results of several Gauss–Jacobi iterations to estimate the optimal SOR parameter [5], but it is not yet clear how to apply this approach to CSOR. One might first try to estimate the largest-magnitude eigenvalue $\mu_1(z)$ by computing the transforms of successive WR iterates $\Delta\boldsymbol{x}^{k+1}[m]$. Then an adaptive scheme might be used to approximate $\omega_{\text{opt}}[m]$. Of course, one difficulty is that it is the complex value of $\mu_1(z)$ that is required, not the magnitude $|\mu_1(z)|$. In addition, since the iterates are not known for m greater than some finite L, it is not possible, in general, to compute the transform of the iterates. A more successful approach has been to consider the spectrum of the SOR operator and use a power method to estimate $\mu_1(z)$ and $\omega_{\text{opt}}[m]$.

For the Poisson problem in Example 3.1 and Figs. 1 and 2, the optimal CSOR sequence $\omega_{\text{opt}}[m]$ can be computed analytically. The largest-magnitude eigenvalue $\mu_1 = \rho(\boldsymbol{D}^{-1}(\boldsymbol{L}+\boldsymbol{U}))$ for the Poisson matrix is easily calculated, and then Theorem 5.2 implies that the z-transform of the optimal CSOR sequence is given by

$$\omega_{\text{opt}}(z) = \frac{2}{1+\sqrt{1-\left(\dfrac{\mu_1 dh}{1-z^{-1}+dh}\right)^2}}. \tag{43}$$

Finally, series expansion can be used to invert the z-transform and obtain $\omega_{\text{opt}}[m]$.

There are a variety of alternative approaches to extending the CSOR algorithm to problems with nonlinearities. We used a waveform extension of relaxation-Newton (WRN) methods for solving nonlinear algebraic problems [16]. For example, consider the nonlinear problem of the form

$$\tfrac{d}{dt}\boldsymbol{x}(t) + \boldsymbol{F}(\boldsymbol{x}(t), t) = 0, \tag{44}$$

where $\boldsymbol{F} : \mathbb{R}^n \to \mathbb{R}^n$. The iteration update equation for the ith component of $\boldsymbol{x}$ in a CSOR-Newton algorithm is then given by

$$\frac{d}{dt}\hat{x}_i^{k+1}(t) + \frac{\partial F_i(\boldsymbol{x}^k(t))}{\partial x_i}\left(\hat{x}_i^{k+1}(t) - x_i^k(t)\right) + \boldsymbol{F}_i(\boldsymbol{x}^k(t), t) = 0, \tag{45}$$

followed by

$$x_i^{k+1}(t) = x_i^k(t) + \int_0^t \omega(\tau)\cdot\left[\hat{x}_i^{k+1}(t-\tau) - x_i^k(t-\tau)\right]\, d\tau. \tag{46}$$

7. Application to semiconductor device simulation. In this section, the robustness of the CSOR method is demonstrated by using it to solve a practical problem for which the theory does not precisely apply, in particular, the transient simulation of semiconductor devices [2], [22], [18]. We begin in §7.1 with a brief review of the problem of semiconductor device simulation. Then in §7.2, experimental results are presented, comparing the performance on a serial machine of the WRN method with and without CSOR acceleration with that of standard pointwise solution methods. Finally, simulation results on a parallel machine are presented in §7.3.

7.1. Background for semiconductor device simulation. A key component of modern VLSI circuits is a semiconductor device known as a MOSFET (metal-oxide semiconductor field effect transistor). Although a MOSFET is a three-dimensional structure consisting of several different regions of silicon, oxide, and metal, a MOSFET may be modeled by a two-dimensional slice of the device, as shown in Fig. 7. Here, thick lines represent metal contacts to the drain, source, substrate, and gate oxide regions, to which external voltage boundary conditions are applied.

Charge transport within a semiconductor device can be described by a two-carrier model of electrons and holes and is governed by the electrostatic potential u normalized in thermal volts and the electron and hole concentrations n and p. These three unknowns are computed at each point in a device by solving the Poisson equation, and the electron and hole continuity equations:

$$\epsilon V_t \nabla^2 u + q\,(p - n + N) = 0, \tag{47}$$

$$D_n \nabla^2 n - \mu_n V_t\left(\nabla n \nabla u + n\,\nabla^2 u\right) = \frac{\partial n}{\partial t}, \tag{48}$$

$$D_p \nabla^2 p + \mu_p V_t\left(\nabla p \nabla u + p\,\nabla^2 u\right) = \frac{\partial p}{\partial t}, \tag{49}$$

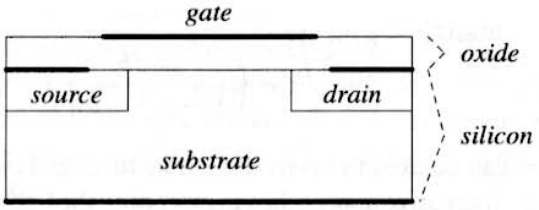

FIG. 7. *A two-dimensional slice of a MOSFET device.*

where N is a known background concentration typically ranging from $-10^{16}\ cm^{-3}$ to $10^{20}\ cm^{-3}$, $V_t = 0.026\ V$ is the thermal voltage constant, $q = 1.60\times 10^{-19}\ C$ is the magnitude of electronic charge, D_n and D_p are the electron and hole diffusion constants, and μ_n and μ_p are the electron and hole mobilities. The diffusion constants D_n and D_p are related to the mobilities by the Einstein relations $D_n = V_t\mu_n$ and $D_p = V_t\mu_p$. The mobilities μ_n and μ_p are used to model many physical mechanisms. One standard approach is to model the mobilities as nonlinear functions of the electric field E; i.e.,

$$\mu_n = \mu_{n_0}\left[1 + \left(\frac{\mu_{n_0}E}{v_{sat}}\right)^{\beta}\right]^{-(1/\beta)}$$

where v_{sat} and β are constants and μ_{n_0} is a doping-dependent mobility [12]. Typical mobility and diffusivity values are $\mu_n = 550$, $\mu_p = 180$, $D_n = 14.3$, and $D_p = 4.68$.

A common approach to spatially discretizing the two-dimensional device equations (47)–(49) is to use a finite-difference formula to discretize the Poisson equation, and an exponentially fit finite-difference formula to discretize the continuity equations [22]. For each node i of an N-node rectangular mesh, the spatial discretization yields

$$V_t \sum_j \left\{\frac{d_{ij}\,\epsilon_{ij}}{L_{ij}}[u_i - u_j]\right\} - qA_i\,(p_i - n_i + N_i) = 0, \tag{50}$$

$$\frac{V_t}{A_i}\sum_j \left\{\frac{d_{ij}\,\mu_{n_{ij}}}{L_{ij}}[n_i B(u_i - u_j) - n_j B(u_j - u_i)]\right\} = \frac{dn_i}{dt}, \tag{51}$$

$$\frac{V_t}{A_i}\sum_j \left\{\frac{d_{ij}\,\mu_{p_{ij}}}{L_{ij}}[p_i B(u_j - u_i) - p_j B(u_i - u_j)]\right\} = \frac{dp_i}{dt}. \tag{52}$$

The summations are taken over the silicon nodes j adjacent to node i. As shown in Fig. 8, for each node j adjacent to node i, L_{ij} is the distance from node i to node j, d_{ij} is the length of the side of the Voronoi box that encloses node i and bisects the edge between nodes i and j, and A_i is the area of the Voronoi box. Similarly, the quantities ϵ_{ij}, $\mu_{n_{ij}}$, and $\mu_{p_{ij}}$ are the dielectric permittivity, electron, and hole mobility, respectively, on the edge between nodes i and j. The Bernoulli function $B(x) = x/(e^x - 1)$ is used to exponentially fit potential variation to electron and hole concentration variations, and effectively upwinds the current equations.

Figure 9 illustrates a typical MOSFET simulation, showing a two-dimensional slice of the silicon device along with the external boundary conditions on the potentials u at the terminals. Here, the potentials at the source and substrate terminals are held at 0, the potential u at the gate terminal is held at 200, and there is a short pulse at the drain terminal. The concentrations n and p at each terminal are held constant at an equilibrium value determined by the background doping concentration at that terminal. For all MOSFET simulations, Dirichlet boundary

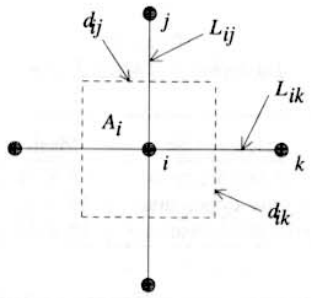

FIG. 8. *Illustration of a mesh node i, the area A_i of its Voronoi box, and the lengths d_{ij} and L_{ij}.*

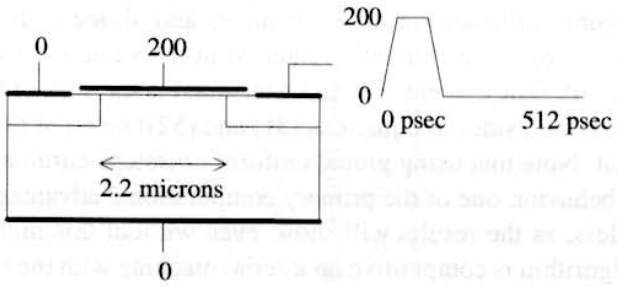

FIG. 9. *Illustration of the drain-driven karD example and the Dirichlet boundary conditions on the terminal potentials u (normalized in thermal volts).*

conditions on the variables u, n, and p are imposed at the source, drain, gate, and substrate terminals, and Neumann reflecting boundary conditions are imposed along the left and right edges of the device.

In the following experiments, the three different MOS devices of Table 1 were used to construct six simulation examples, each device being subjected to either a drain voltage pulse with the gate held high (the D examples), or a gate voltage pulse with the drain held high (the G examples). To observe the effect on WR convergence, two additional gate-pulsed examples were constructed by refining the device meshes of karG and soiG to contain 64 vertical lines.

7.2. Results on a serial machine. The waveform methods used for experiments below are based on red/black block Gauss–Seidel WR, where the blocks correspond to vertical mesh lines. To generate a good initial guess, 16 or 32 iterations of standard WR were performed, and either the WRN method or WRN with CSOR acceleration was used. In each block corresponding to a vertical line, the sequence of implicit nonlinear algebraic systems generated by the backward difference formula were solved with Newton's method, and the linear equation systems in each block generated by Newton's method were solved with sparse Gaussian elimination. For CSOR acceleration, the CSOR sequence $\omega[m]$ was determined by linearizing the device equations (47)–(49) about the initial condition solution, fitting $\omega_{\text{opt}}(z)$ with a rational function, and inverse transforming. To diminish the effect of the nonlinearity, the overrelaxation convolution was applied only to the potential variables u.

For the pointwise methods, the device equation system of the entire mesh was generated successively at each timepoint, with each vertical line block contributing the equations of that mesh line. The nonlinear algebraic system resulting from the backward difference formula was solved at each timepoint with Newton's method, and the linear equation system generated by Newton's method was solved with either sparse Gaussian elimination or the vertical line block preconditioned generalized minimal residual (GMRES) conjugate direction algorithm [19].

TABLE 1
Description of MOS devices.

Device	Description	Mesh	Unknowns
kar	abrupt junction	19 × 31	1379
ldd	lightly-doped drain	15 × 20	656
soi	silicon-on-insulator	18 × 24	856

To simplify comparisons between the different solution methods, the backward Euler method using 256 equal timesteps was used for most experiments, with a simulation interval of 51.2 or 512 picoseconds. Some experiments were performed using the second-order backward-difference method, to show that the selection of integration method has limited impact on the results. The convergence criterion for all experiments and all methods was the requirement that the maximum error over the simulation interval in the value of any terminal current be less than one part in 10^4. The current entering a terminal is the sum of the electron and hole current fluxes (the left-hand sides of equations (51) and (52)) on all of the mesh line segments touching the terminal. Note that using global uniform timesteps eliminates the ability of WR to exploit multirate behavior, one of the primary computational advantages of WR on a serial machine. Nevertheless, as the results will show, even without this multirate advantage, the accelerated WRN algorithm is competitive on a serial machine with the best known pointwise methods.

Figure 10 shows the convergence of the eight examples as a function of iteration for the ordinary WRN method and the same method accelerated with CSOR. Despite the nonlinearity of the semiconductor equations, the CSOR algorithm converged substantially faster than the WRN method, demonstrating the robustness of the approach.

Table 2 shows the CPU times in seconds required for solution of the eight examples, for pointwise methods and waveform methods. The serial experiments were performed on an IBM RS/6000 Model 540 workstation, with 256 MB of memory. For the pointwise methods, the *sparse elim* column shows the result of using direct factorization to solve the matrix problem at each timepoint, compared with the vertical line block preconditioned GMRES algorithm. The waveform method columns show the result of using ordinary Gauss–Seidel WRN, and the same algorithm accelerated with CSOR. The results show that on a serial machine, CSOR is competitive with pointwise Newton–GMRES—in fact, CSOR is slightly faster than the pointwise Newton–GMRES method for five of the eight examples. This is especially encouraging since, as noted above, the fixed timestep simulations do not allow the waveform method to take advantage of multirate behavior.

TABLE 2
Comparison of serial times required for pointwise methods and waveform methods. Serial experiments were conducted on an IBM RS/6000 Model 540 workstation.

	Pointwise methods		Waveform methods			
Example	Sparse elim	GMRES	WRN	(Iters)	Conv SOR	(Iters)
lddD	231.65	620.00	7327.24	(1434)	584.81	(100)
lddG	244.46	505.92	3404.19	(657)	360.15	(55)
soiD	401.95	188.53	1832.21	(284)	388.62	(50)
soiG	430.80	205.28	1377.21	(209)	369.16	(45)
karD	1401.60	673.81	5451.02	(515)	1088.41	(92)
karG	1460.38	794.01	4672.49	(440)	712.74	(55)
soiG64	2144.05	2249.01	31399.19	(1885)	2239.39	(119)
karG64	3833.04	5205.89	40246.82	(1885)	3328.85	(141)

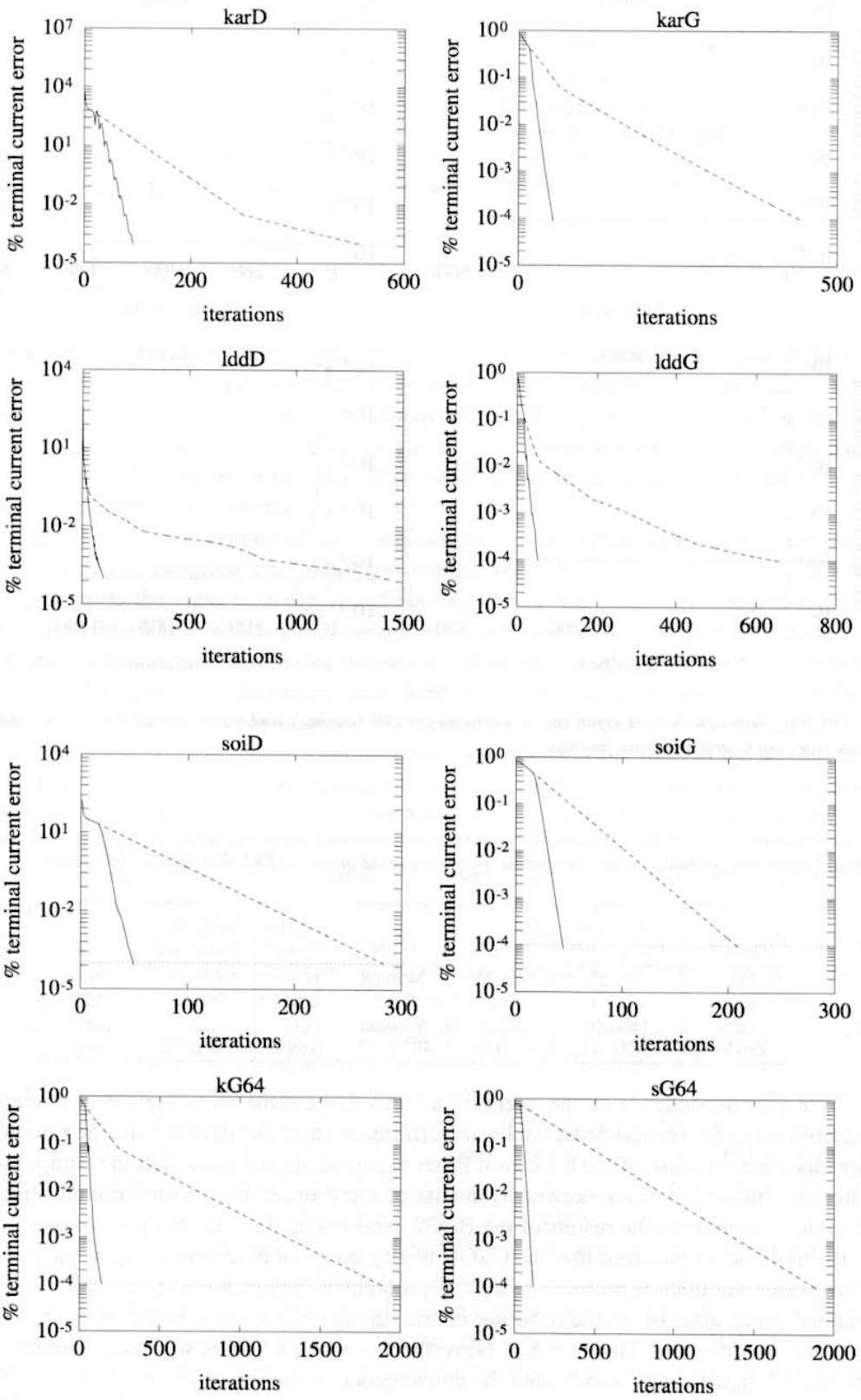

FIG. 10. *Terminal current error versus iteration for WR (dashed) and waveform CSOR (solid). The integration method was backward Euler.*

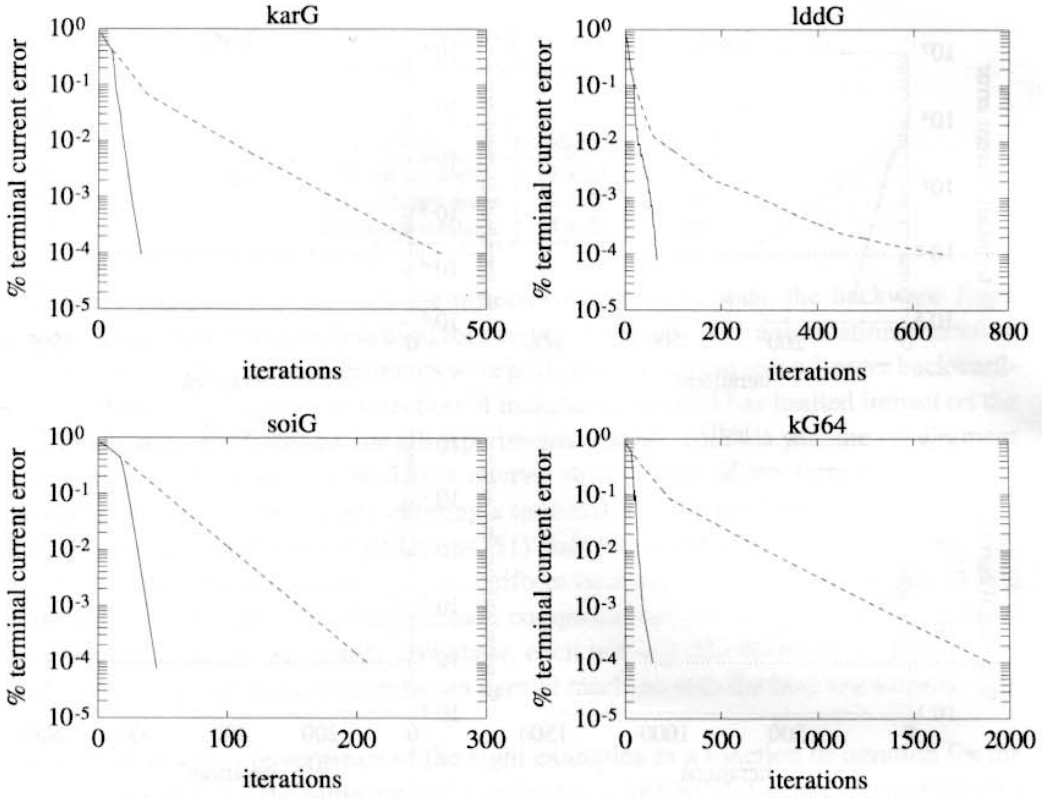

FIG. 11. *Terminal current error versus iteration for WR (dashed) and waveform CSOR (solid), using the second-order backward-difference formula.*

TABLE 3

Comparison of serial times required for pointwise methods and waveform methods, using the second-order backward-difference formula. Serial experiments were conducted on an IBM RS/6000 Model 540 workstation.

	Pointwise methods		Waveform methods			
Example	Sparse elim	GMRES	WRN	(Iters)	Conv SOR	(Iters)
lddG	244.22	505.28	3496.88	(662)	457.58	(66)
soiG	430.94	204.88	1461.23	(208)	399.46	(45)
karG	1460.40	792.63	5086.66	(443)	762.72	(55)
karG64	3834.80	5179.61	40242.77	(1895)	3863.22	(149)

To test for dependence on the integration method, the same set of eight experiments was conducted using the second-order backward-difference formula (BDF2) [4]. The results were almost identical to those of the backward Euler experiments and show that the comparison of the iterative methods was not skewed by the use of a first-order integration formula. Figure 11 and Table 3 summarize the results of the BDF2 experiments for four of the examples.

It should be emphasized that the CSOR theory does not precisely apply to the semiconductor device simulation problem since the problem is highly nonlinear. And even when linearized about some point, the resulting matrix, though block consistently ordered, does not satisfy the conditions of Theorem 5.2. Nevertheless, using a CSOR sequence determined by Theorem 5.2 significantly accelerated the convergence of the waveform method.

7.3. Experimental results on parallel machines. In this section, the implementations of the waveform and pointwise methods for device simulation on the Intel iPSC/860 MIMD parallel machine are outlined and experimental results are presented. As described in §7, the

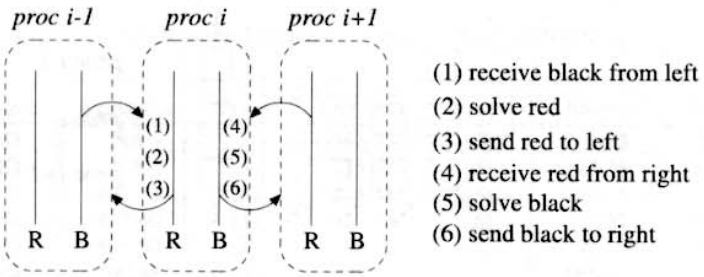

FIG. 12. *Illustration of the communication and computation steps performed by compute node i during one parallel waveform method iteration.*

waveform methods used for device simulation are based on red/black block Gauss–Seidel WR, where the blocks correspond to vertical mesh lines. In the parallel implementation, after the N vertical lines of the device mesh are partitioned into $N/2$ pairs of adjacent red and black lines, each pair is assigned to one of $N/2$ processors or compute nodes. Then in each parallel iteration, the $N/2$ compute nodes first solve the $N/2$ red lines concurrently, and then solve the $N/2$ black lines concurrently.

Figure 12 is an illustration of the communication and computation steps performed by compute node i to complete one parallel waveform iteration. On compute node i, the black line which gives the *right* boundary condition of the red line is already resident within the node, but the black line which gives the *left* boundary condition is on compute node $i-1$. Therefore, the first step in computing iterates for all of the red lines is for each compute node i to receive the black line information sent rightward from compute node $i-1$. Once the left boundary conditions have been received, each compute node i can compute iterates for its red line, and send this red line information leftward. Solving the black lines in the second half of the iteration requires a similar communication pattern. Each compute node i receives the red line waveform solution sent from the right, computes the iterate for its black line, and sends this black line iterate to the right.

Note that each iteration of the parallel waveform algorithm contains only two blocking communications—before computing iterates for a line, each compute node must wait to receive the waveform from a neighboring line. The algorithm therefore requires very little synchronization between the compute nodes, and the communication that is required consists of large packets of information, entire waveforms.

If fewer than $N/2$ compute nodes are available, then each compute node is given multiple pairs of red/black lines that are adjacent in the device mesh. This implies that some of the lines (both red and black) residing on a compute node will depend only on other lines residing in the compute node. In this case, some communication and computation can be overlapped—the red lines that do not depend on other compute node solutions are solved before waiting to receive the black line solutions. Similarly, the black lines that do not depend on other compute node solutions are solved before waiting to receive the red line solutions.

To parallelize the pointwise Newton–GMRES method, the vertical line blocking of the device mesh is used to partition the block tridiagonal matrix of the linearized equation system. First, the host assigns pairs of vertical line blocks to the compute nodes exactly as in the waveform method. Then, given a particular block, the corresponding compute node is responsible for the storage and computation of the corresponding pieces of the matrix and GMRES vectors in that "block row," as shown in Fig. 13.

Unfortunately, partitioning the matrix and the vectors implies that the parallel pointwise Newton–GMRES algorithm requires many communication steps, each consisting of relatively small packets. Before every Newton iteration at every timepoint, a compute node must receive

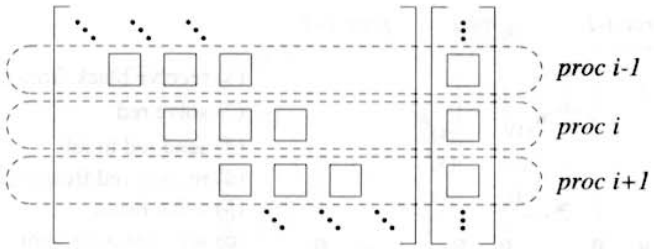

FIG. 13. *To partition the matrix–vector product, each processor is assigned the block rows corresponding to a pair of vertical line blocks.*

TABLE 4
Waveform relaxation Newton timing results on the iPSC/860.

Example	8 blk/proc	(Procs)	4 blk/proc	(Procs)	2 blk/proc	(Procs)
lddD	NA		4721.25	(5)	2473.97	(10)
lddG	NA		2190.19	(5)	1182.51	(10)
soiD	1871.74	(3)	1037.27	(6)	554.67	(12)
soiG	1504.04	(3)	812.72	(6)	429.44	(12)
karD	4378.82	(4)	2256.21	(8)	1224.32	(16)
karG	3745.48	(4)	1939.39	(8)	1041.91	(16)

the two vectors of solutions of the neighboring lines from the left and the right in order to generate the block diagonal and block off-diagonal matrices. To accomplish the matrix–vector product for each GMRES iteration, a compute node must exchange pieces of the multiplicand vector with the neighboring compute nodes. Moreover, each GMRES iteration requires a number of inner product calculations, each of which requires a global sum. Although it may be possible to overlap these communication operations with local computation, a significant amount of interprocessor synchronization is still required. An approach such as that given in [24] might prove to be helpful, however.

Tables 4 and 5 show the CPU times required for solution of the eight examples on the Intel iPSC/860. The times shown are measured elapsed times on the compute nodes. The waveform method displays a remarkable scalability, with a roughly linear speedup up to 32 processors (the soiG64 and karG64 examples). In contrast, the parallel pointwise Newton–GMRES algorithm became significantly slower on more than one processor. This can be attributed to the many small communications and synchronizations required at each time point in the simulation interval. Despite the fairly sophisticated parallel implementation and the dedicated communication hardware in the iPSC/860, the parallel Newton–GMRES method is bounded by the cost of its communications, at every iteration, at every timestep. Table 6 summarizes the best timing results for each method on the iPSC/860.

It is clear that the parallel implementations of both the waveform and the pointwise methods could be improved. The pointwise Newton–GMRES code could be rewritten to avoid some communication with the host, perhaps by choosing one of the i860 compute nodes to act as coordinator. In addition, some of the Newton–GMRES communication could be overlapped with computation. The CSOR code could be improved by using a specialized block tridiagonal matrix solver rather than a general sparse Gaussian elimination package. But none of these improvements will change the inherent difference between the pointwise and the waveform algorithm: the pointwise algorithm requires much more synchronization and performs many more communications.

TABLE 5
Convolution SOR timing results on the iPSC/860.

Example	8 blk/proc	(Procs)	4 blk/proc	(Procs)	2 blk/proc	(Procs)
lddD	NA		382.77	(5)	201.23	(10)
lddG	NA		235.67	(5)	128.59	(10)
soiD	405.93	(3)	225.41	(6)	120.15	(12)
soiG	417.40	(3)	225.74	(6)	117.31	(12)
karD	895.50	(4)	460.27	(8)	248.93	(16)
karG	590.88	(4)	308.29	(8)	165.61	(16)
soiG64	987.17	(8)	507.20	(16)	260.13	(32)
karG64	1386.32	(8)	713.97	(16)	373.53	(32)

TABLE 6
Summary of the best timing results for each method on the iPSC/860.

Example	(Size)	Newton–GMRES	(Procs)	Conv SOR	(Procs)
lddD	(656)	1184.68	(1)	201.23	(10)
lddG	(656)	989.74	(1)	128.59	(10)
soiD	(856)	366.11	(1)	120.15	(12)
soiG	(856)	401.42	(1)	117.31	(12)
karD	(1379)	1262.49	(1)	248.93	(16)
karG	(1379)	1492.17	(1)	165.61	(16)
soiG64	(2292)	4400 est.	(1)	260.13	(32)
karG64	(2854)	9800 est.	(1)	373.53	(32)

8. Conclusion. In this paper, a new waveform convolution SOR algorithm was presented and applied to solving the differential-algebraic system generated by spatial discretization of the time-dependent semiconductor device equations. In the experiments included, CSOR converged robustly, and substantially faster than unaccelerated WR. In addition, on the parallel machines used in these experiments, the CSOR method was substantially more efficient than the pointwise GMRES method.

Example 3.1 and all plots were generated with MATLAB 3.5 running on an IBM RS/6000 Model 540 workstation.

Acknowledgments. The authors would like to thank Andrew Lumsdaine for his collaboration on the parallel WORDS project and L. N. Trefethen, D. Rose, P. Lanzcron, A. Toomre, J. Janssen, S. Vandewalle, and O. Nevanlinna for many valuable suggestions.

REFERENCES

[1] A. AGARWAL, *Limits on interconnection network performance*, IEEE Trans. Parallel Distrib. Systems, 2 (1991), pp. 398–412.

[2] R. E. BANK, W. C. COUGHRAN, JR., W. FICHTNER, E. GROSSE, D. ROSE, and R. SMITH, *Transient simulation of silicon devices and circuits*, IEEE Trans. CAD, 4 (1985), pp. 436–451.

[3] R. C. Y. CHIN AND T. A. MANTEUFFEL, *An analysis of block successive overrelaxation for a class of matrices and complex spectra*, SIAM J. Numer. Anal., 25 (1988), pp. 564–585.

[4] C. W. GEAR, *Numerical Initial Value Problems in Ordinary Differential Equations*, Automatic Computation, Prentice–Hall, Englewood Cliffs, NJ, 1971.

[5] L. A. HAGEMAN AND D. M. YOUNG, *Applied Iterative Methods*, Academic Press, Orlando, FL, 1981.

[6] E. I. JURY, *Theory and Application of the z-Transform Method*, John Wiley & Sons, Inc., New York, 1964.

[7] E. LELARASMEE, A. E. RUEHLI, AND A. L. SANGIOVANNI-VINCENTELLI, *The waveform relaxation method for time domain analysis of large scale integrated circuits*, IEEE Trans. CAD, 1 (1982), pp. 131–145.

[8] C. LUBICH AND A. OSTERMAN, *Multi-grid dynamic iteration for parabolic equations*, BIT, 27 (1987), pp. 216–234.

[9] A. LUMSDAINE, *Theoretical and Practical Aspects of Parallel Numerical Algorithms for Initial Value Problems, with Applications*, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA, 1992.

[10] U. MIEKKALA AND O. NEVANLINNA, *Convergence of dynamic iteration methods for initial value problems*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 459–482.

[11] ———, *Sets of convergence and stability regions*, BIT, 27 (1987), pp. 554–584.

[12] R. S. MULLER AND T. I. KAMINS, *Device Electronics for Integrated Circuits*, John Wiley and Sons, New York, 1986.

[13] O. NEVANLINNA, *Remarks on Picard–Lindelöf iteration, part* I, BIT, 29 (1988), pp. 328–346.

[14] ———, *Remarks on Picard–Lindelöf iteration, part* II, BIT, 29 (1989), pp. 535–562.

[15] A. V. OPPENHEIM AND R. W. SCHAFER, *Discrete-Time Signal Processing*, Prentice–Hall, Englewood Cliffs, New Jersey, 1989.

[16] J. M. ORTEGA AND W. C. RHEINBOLT, *Iterative Solution of Nonlinear Equations in Several Variables*, Computer Science and Applied Mathematics, Academic Press, New York, 1970.

[17] L. REICHEL AND L. N. TREFETHEN, *Eigenvalues and pseudo-eigenvalues of toeplitz matrices*, Linear Algebra Appl., 162–164 (1992), pp. 153–185.

[18] M. REICHELT, J. WHITE, AND J. ALLEN, *Waveform relaxation for transient two-dimensional simulation of MOS devices*, in Proc. International Conference on Computer-Aided Design, Santa Clara, CA, November 1989, pp. 412–415.

[19] Y. SAAD AND M. SCHULTZ, *GMRES: A generalized minimum residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[20] E. B. SAFF AND A. D. SNIDER, *Fundamentals of Complex Analysis for Mathematics, Science, and Engineering*, Prentice–Hall, Inc., Englewood Cliffs, NJ, 1976.

[21] R. SALEH AND J. WHITE, *Accelerating relaxation algorithms for circuit simulation using waveform-newton and step-size refinement*, IEEE Trans. CAD, 9 (1990), pp. 951–958.

[22] S. SELBERHERR, *Analysis and Simulation of Semiconductor Devices*, Springer-Verlag, New York, 1984.

[23] R. D. SKEEL, *Waveform iteration and the shifted Picard splitting*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 756–776.

[24] E. D. STURLER, *A parallel restructured version of GMRES(m)*, in Proceedings of the Copper Mountain Conference on Iterative Methods, Copper Mountin, Colorado, 1992.

[25] L. N. TREFETHEN, *Pseudospectra of matrices*, in Numerical Analysis 1991, D. f. Griffiths and G. A. Watson, eds., Longman Scientific & Technical, Harlow, Essex, England, 1992.

[26] S. VANDEWALLE AND R. PIESSENS, *Efficient parallel algorithms for solving initial-boundary value and time-periodic parabolic partial differential equations*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 1330–1346.

[27] R. S. VARGA, *Matrix Iterative Analysis*, Automatic Computation Series, Prentice–Hall, Englewood Cliffs, NJ, 1962.

[28] J. K. WHITE AND A. SANGIOVANNI-VINCENTELLI, *Relaxation Techniques for the Simulation of VLSI Circuits*, The Kluwer International Series in Engineering and Computer Science, Kluwer Academic Publishers, Boston, 1987.

[29] D. M. YOUNG, *Iterative Solution of Large Linear Systems*, Academic Press, Orlando, FL, 1971.