10
12/5/91 JSD

ornl

ORNL/TM-11963

OAK RIDGE
NATIONAL
LABORATORY

MARTIN MARIETTA

Parallelizing Across Time When Solving Time-Dependent Partial Differential Equations

Patrick H. Worley

MANAGED BY
MARTIN MARIETTA ENERGY SYSTEMS, INC.
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from the Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831; prices available from (615) 576-8401, FTS 626-8401.

Available to the public from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Rd., Springfield, VA 22161.

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ORNL/TM--11963

DE92 003733

Engineering Physics and Mathematics Division

Mathematical Sciences Section

PARALLELIZING ACROSS TIME WHEN SOLVING TIME-DEPENDENT PARTIAL DIFFERENTIAL EQUATIONS

Patrick H. Worley

Oak Ridge National Laboratory
Mathematical Sciences Section
P. O. Box 2008, Bldg. 6012
Oak Ridge, TN 37831-6367

worley@msr.epm.ornl.gov

Date Published: September 1991

Research was supported by the Applied Mathematical Sciences Research Program of the Office of Energy Research, U.S. Department of Energy.

Prepared by the
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831
Managed By
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
U.S. DEPARTMENT OF ENERGY
under Contract No. DE-AC05-84OR21400

MASTER

Contents

1 Introduction 1
2 Parabolic PDEs 2
  2.1 Waveform relaxation 2
  2.2 Waveform relaxation multigrid 3
  2.3 Waveform relaxation multigrid cyclic reduction 4
  2.4 Numerical Results 5
  2.5 Discussion 6
  2.6 Generalizations 8
3 Hyperbolic PDEs 9
4 Conclusions 10
5 Acknowledgements 10
6 References 10

PARALLELIZING ACROSS TIME WHEN SOLVING TIME-DEPENDENT PARTIAL DIFFERENTIAL EQUATIONS

Patrick H. Worley

Abstract

The standard numerical algorithms for solving time-dependent partial differential equations (PDEs) are inherently sequential in the time direction. This paper describes algorithms for the time-accurate solution of certain classes of linear hyperbolic and parabolic PDEs that can be parallelized in both time and space and have serial complexities that are proportional to the serial complexities of the best known algorithms. The algorithms for parabolic PDEs are variants of the waveform relaxation multigrid method (WFMG) of Lubich and Ostermann where the scalar ordinary differential equations (ODEs) that make up the kernel of WFMG are solved using a cyclic reduction type algorithm. The algorithms for hyperbolic PDEs use the cyclic reduction algorithm to solve ODEs along characteristics.

1. Introduction

For many numerical problems in scientific computation, the execution time grows without bound as a function of the problem size, independent of the number of processors and of the algorithm used [40], [43]. In particular, for most linear partial differential equations (PDEs) arising in mathematical physics, the parallel complexity grows as $\log N$, where N is a particular measure of the problem size. The proof is based on deriving upper and lower bounds on the execution time of *optimal parallel algorithms* for multiprocessors with an *unlimited number of processors* and *no interprocessor communication costs*. (Lower bounds for the case when communication costs are not zero can also be calculated [40], [41], [43].) Due to the assumption on the number of processors, these optimal parallel algorithms can have very large serial complexities, and the tightness of the bounds on the parallel execution time for practical algorithms is not established by this analysis.

An analysis of standard numerical algorithms for linear PDEs indicates that growth in the parallel execution time for these algorithms has an important effect when using scaled speed-up models to evaluate multiprocessor performance [42]. In this analysis, there is a strong dichotomy in the nature of the growth in the parallel execution time between algorithms for the solution of time-dependent and time-independent PDEs, *a dichotomy that is not present in the algorithm-independent analysis*. For example, when approximating elliptic PDEs using finite difference or finite element discretizations, the serial complexity is at least $\Theta(N_s)$, where N_s is the size of the underlying grid and $\Theta(x)$ denotes a positive quantity whose leading order term is proportional to x [16, p. 31]. This linear serial complexity can often be achieved using a full multigrid V-cycle algorithm, weighted Jacobi or multi-color Gauss-Seidel relaxation, and local restriction/prolongation operators, which has a parallel complexity of $\Theta(\log^2 N_s)$ on a multiprocessor with $\Theta(N_s)$ processors [2], [3], [5], [11]. So, for these problems, there exists an algorithm whose serial complexity is proportional to that of the best serial algorithm and whose parallel complexity is a polylog function of the serial complexity.

Timestepping methods are commonly used to calculate the time-accurate solution of time-dependent PDEs. For a time-accurate solution, the solution is required at a sequence of times $\{t_i \mid i = 0, \ldots, N_t\}$, where $t_{i-1} < t_i$ and $t_i - t_{i-1}$ is small enough to allow accurate interpolation of the solution at all times in between. Timestepping algorithms calculate the approximate solution for each time level in sequence, calculating the solution at time t_i from the approximate solution at times $t_j, j < i$. Standard timestepping algorithms based on finite difference or finite element discretizations of hyperbolic and parabolic PDEs have serial complexities that are linear in the number of space-time locations where the solution function is approximated. Thus, the serial complexity is $\Theta(N_s \cdot N_t)$, where N_s is the size of the underlying grid at a fixed time and N_t is the number of time levels. The calculation of each time level is usually

easily parallelized, but the time direction in a timestepping algorithm is inherently sequential. Thus, the parallel complexity is always at least $\Theta(N_t)$, i.e. not a polylog function of the serial complexity. (Variants of the standard timestepping algorithms have been proposed that begin the calculation for later time levels before the current time level is finished [12], [30], [39], but these algorithms do not alter the sequential nature of the time direction.) This paper addresses the question of whether good serial algorithms for time-dependent PDEs are intrinsically less parallel than good serial algorithms for time-independent PDEs. We pose the question in the following form:

For a given time-dependent PDE, is there a numerical algorithm for the time-accurate approximation of the solution with the properties:

(1) *Let $C_s(N)$ be the serial complexity of the algorithm for a problem of size N, and let $C_{s,\ \mathrm{opt}}(N)$ be the serial complexity of the best known serial algorithm for this problem. Then $C_s(N) = \Theta(C_{s,\ \mathrm{opt}}(N))$.*

(2) *The algorithm can be parallelized in the time direction as well as the spatial directions, so that the achievable parallel complexity given an unlimited number of processors, $C_p(N)$, satisfies $C_p(N) = \Theta(\log^\gamma C_s(N))$ for some finite constant γ.*

In this paper,[1] we describe a class of algorithms that have properties (1) and (2) for a large class of linear parabolic PDEs. Not only does this class of algorithms answer the above theoretical question, it may also have practical applications on massively parallel multiprocessors. We also briefly describe a different class of algorithms with properties (1) and (2) for a particular class of linear hyperbolic PDEs.

2. Parabolic PDEs

2.1. Waveform relaxation

Waveform relaxation is a technique for solving systems of ordinary differential equations of initial-value type [26], [29]. It is based on applying standard point and block iterative methods for the solution of linear systems [37] to the solution of a system of ODEs. For example, let A be an $n \times n$ matrix, and consider the problem $dU/dt + AU = F$, where U and F are vector functions of time. Then the kth step of a Jacobi-based iterative method for the solution of this system is

$$\frac{d}{dt}U^{(k)} + DU^{(k)} = F + (D - A)U^{(k-1)}, \tag{1}$$

[1] An earlier version of this paper appears in the Proceedings of the Fifth SIAM Conference on Parallel Processing for Scientific Computing.

where D is the diagonal of A. Thus, each step of the method involves the solution of n independent scalar ODEs. The decoupling of the system allows different discretizations and timesteps to be used for each of the ODEs, which can lead to significant savings for some applications.

To solve parabolic PDEs, the spatial derivatives are discretized to generate a semi-discrete problem and the resulting system of ODEs is solved using waveform relaxation. Miekkala and Nevanlinna have analyzed the convergence of waveform relaxation for linear operators [28]. They showed that, for linear PDEs of the form $u_t + Lu = f$ where L is an elliptic operator and for standard spatial discretizations, the convergence rates for Jacobi and Gauss-Seidel iterations for the semi-discrete problem are similar to those for the analogous linear system.

2.2. Waveform relaxation multigrid

Let L_h represent the discrete operator generated by discretizing an elliptic operator L. The correspondence between the convergence rate of waveform relaxation applied to $dU/dt + L_h U = F$ and the convergence rate of the analogous matrix iterative method applied to $L_h U = F$ has two immediate implications. First, the convergence rate is too slow for waveform relaxation to be competitive with standard timestepping algorithms. Second, multigrid techniques may be effective at accelerating convergence of the iteration.

Multigrid acceleration has been analyzed by Lubich and Ostermann [27]. Among their results, they showed that full multigrid performance can be achieved for the semi-discrete problem if L_h is symmetric positive definite, and either L_h has constant diagonal entries, in which case weighted Jacobi relaxation is used, or L_h has the form

$$\begin{pmatrix} D_1 & B \\ A & D_2 \end{pmatrix}, \tag{2}$$

where D_1 and D_2 are diagonal, in which case Gauss-Seidel relaxation is used. Note that this latter matrix structure commonly occurs when using a red-black ordering with standard finite difference stencils. They also show that full multigrid performance can be achieved for the fully discrete problem (i.e. linear serial complexity) if, in addition to the above conditions, all ODEs are discretized by the same method, the time direction is not coarsened, and the ODE solver is an A-stable linear multistep or Runge-Kutta method. Full multigrid performance also holds for $A(\alpha)$-stable linear multistep or Runge-Kutta methods for suitably large α. Note that all of these conditions are sufficient, but not necessary.

This *waveform relaxation multigrid* algorithm has been tested extensively [32], [33], [34], [35], [36], and has been shown to work well for a variety of parabolic problems, both linear and nonlinear, on both serial and parallel computers.

2.3. Waveform relaxation multigrid cyclic reduction

The waveform relaxation multigrid algorithm is normally implemented in a fashion that is still intrinsically sequential in the time direction. But computation in the time direction only involves solving linear scalar ODEs. If the ODEs are solved using a linear multistep method with a statically determined timestep, then each ODE solution corresponds to the solution of a banded lower triangular matrix equation, or, equivalently, a linear recurrence. Parallelizing linear recurrence equations has been studied extensively [9], [14], [15], [21], [22], [23], [24], [31]. In particular, if a cyclic reduction approach is used to parallelize the linear recurrence, then parallelism is introduced without increasing the order of the serial complexity. For example, if a two-level scheme, like backward Euler or Crank-Nicolson, is used to discretize the scalar ODE, then a lower bidiagonal matrix equation must be solved. Cyclic reduction combines even numbered equations with odd numbered equations to generate a new bidiagonal matrix equation of half the size. If the original matrix has the form

$$\begin{pmatrix} a_{11} & & & \\ a_{21} & a_{22} & & \\ & a_{32} & a_{33} & \\ & & a_{43} & a_{44} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{pmatrix}, \tag{3}$$

then one step of the cyclic reduction algorithms generates a new matrix equation of the form

$$\begin{pmatrix} a_{22} & \\ -\frac{a_{43}}{a_{33}} a_{32} & a_{44} \end{pmatrix} \cdot \begin{pmatrix} x_2 \\ x_4 \end{pmatrix} = \begin{pmatrix} f_2 - \frac{a_{21}}{a_{11}} f_1 \\ f_4 - \frac{a_{43}}{a_{33}} f_3 \end{pmatrix}. \tag{4}$$

The solution vector of the smaller system is identical to the even-numbered elements of the solution vector of the original matrix equation. This process is repeated until only two equations are left, at which time the two-by-two linear system is solved. The solution values for the small system are then used to calculate the unresolved values in the next larger linear system. For example, in (3), $x_3 = (f_3 - a_{32}x_2)/a_{33}$, which can be calculated immediately since x_2 was determined when solving the smaller system. By repeating this process, the solution to the original matrix equation is calculated. Note that each step of the reduction stage is perfectly parallel, in the sense that combining each pair of equations is independent. Similarly, injecting solution values into the next larger system and solving for the unresolved variables can be done independently for each unresolved variable. Thus, cyclic reduction allows us to parallelize the time direction.

If a k-step linear multistep algorithm is used to solve the ODE, then the banded lower triangular matrix defining the linear recurrence has a bandwidth of k. The cyclic reduction

algorithm again halves the number of equations at each step, but now k consecutive equations are needed to transform the dependencies in a given equation from the previous $k-1$ values in the current matrix equation to the $k-1$ previous values in the new smaller system. As before, this process is continued until only k equations are left. If $k > 2$, then solving the $k \times k$ system and injecting the solution back into the next larger system does not decouple the calculation of the unresolved variables. Instead, it produces a new banded lower triangular matrix equation to solve, one whose bandwidth is $\lceil k/2 \rceil$. Repeatedly applying the cyclic reduction algorithm continues to halve the bandwidth until all of the unresolved variables are calculated, at which time they can be injected into the next larger system to reduce its bandwidth.

The cyclic reduction algorithm is more expensive than the standard serial algorithm, but the complexity is still $\Theta(N_t)$ for each ODE solution (if k is independent of N_t). For example, for a two-level scheme, the serial complexity of the standard algorithm is $3N_t$, while for the cyclic reduction algorithm it is $5N_t$ or $7N_t$, depending on whether certain values are precomputed. For a three-level scheme, the complexity of the standard algorithm is $5N_t$, compared to $11N_t$ or $21N_t$ for the cyclic reduction algorithm. As long as the complexity of the ODE solver is $\Theta(N_t)$, the waveform relaxation multigrid algorithm remains an $\Theta(N_s \cdot N_t)$ complexity algorithm.

The parallel complexity of the cyclic reduction algorithm is a function of the number of time levels used in the discretization of the ODE. For a k-level scheme, it is $\Theta(\log^\gamma N_t)$, where $\gamma = \lceil k/2 \rceil$. Thus, including the parallel cyclic reduction algorithm in a parallel waveform relaxation multigrid algorithm based on weighted Jacobi or red-black Gauss-Seidel iteration results in a total parallel complexity of the form $\Theta(\log^2 N_s \cdot \log^\gamma N_t)$, which is worse than for elliptic problems, but is still polylog.

2.4. Numerical Results

Parallel implementations of multigrid and cyclic reduction have been discussed elsewhere. See [11], [19], and [38] for pointers to the literature. In this section, we verify the predicted linear serial complexity of the waveform relaxation multigrid algorithms for a specific example problem.

We solved the heat equation $u_t + \nabla u = f$ on the unit square $[0,1] \times [0,1]$ in the time interval $[0,1]$ using Dirichlet boundary conditions. Standard centered differencing was used to discretize the spatial derivatives. Crank-Nicolson, a two-level scheme, and the 2nd order backward difference formula (BDF), a three-level scheme, were used to discretize the time derivative. The same timestep and spacestep were used in the discretization, $\sqrt{N_s} = N_t$, and a sequence of problem sizes was examined.

Three algorithms were tried: waveform relaxation mutigrid with the cyclic reduction ODE solver (WFMGCR), waveform relaxation mutigrid with the standard ODE solver (WFMG), and

a timestepping algorithm that uses multigrid at each timestep. The convergence of the multigrid algorithm was essentially identical for all three algorithms, and four full V-cycle multigrid cycles, with 1 relaxation sweep before and after each coarse grid correction, was sufficient to identify convergence (small residual and little change between successive iterates) for all problem sizes and forcing functions tried. The approximate solutions were also essentially identical, indicating that the cyclic reduction algorithm is no less stable (for these problems) than the standard ODE solver. The numbers of floating point operations (flops) required to solve the problems are displayed in Figures 1 and 2. The data indicate linear growth in complexity for all three methods, with WFMGCR being somewhat more expensive than WFMG because of its more expensive ODE solver.

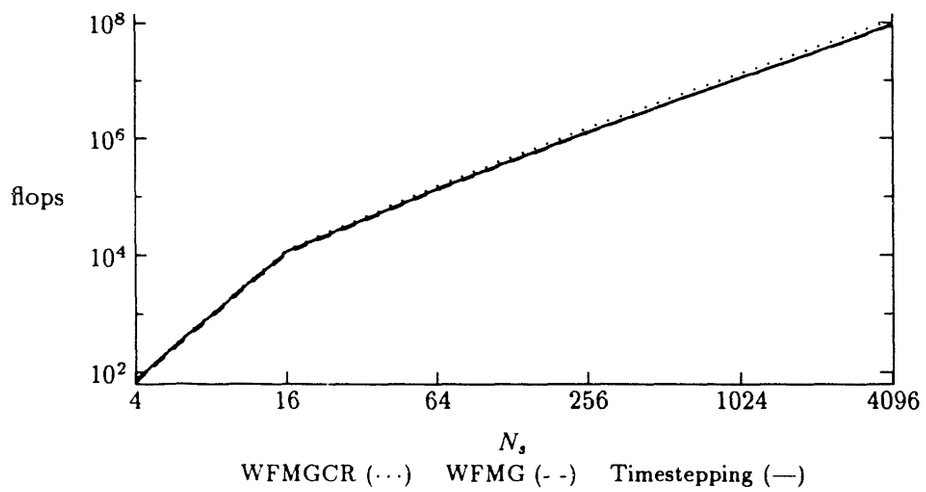

Figure 1: Serial complexity for Crank-Nicolson time discretization.

2.5. Discussion

Using cyclic reduction with the waveform relaxation multigrid algorithm has the desired properties (linear serial complexity and polylog parallel complexity) for all problems for which (a) the waveform relaxation multigrid algorithm has a linear serial complexity when using a relaxation technique that can be efficiently parallelized, like Jacobi or multi-color Gauss-Seidel, and (b) cyclic reduction is a stable algorithm for solving the linear recurrences arising from discretizing the scalar ODEs. Both theory and empirical evidence indicate that (a) is true for a large class of parabolic problems, both linear and nonlinear. While some work on the stability of parallelizing recurrence equations has been done [21], [31] and the numerical examples described here give no indication of stability problems, more work must be done to establish the

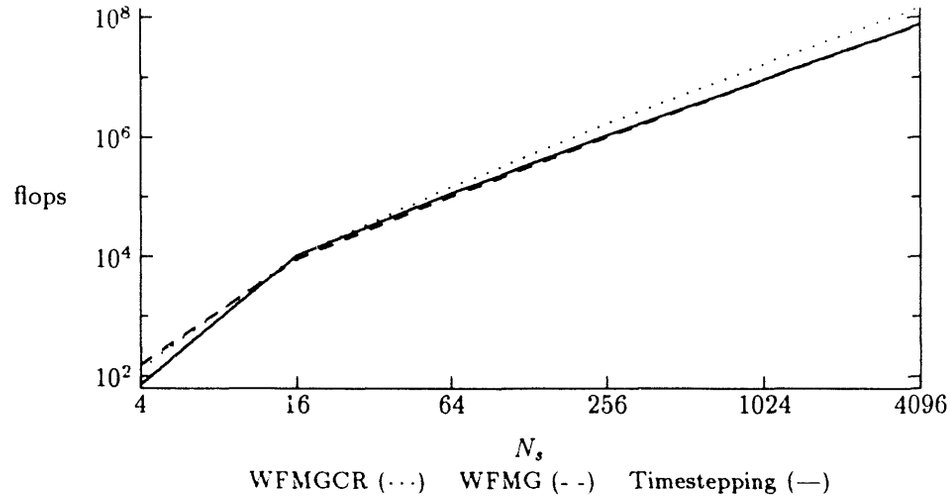

Figure 2: Serial complexity for second order BDF time discretization.

stability of this algorithm, especially when the bandwidth of the recurrence (k) is large. This is especially true given the stability problems of the straightforward implementation of cyclic reduction for elliptic problems [8].

When properties (a) and (b) hold, we have shown the algorithm to have an achievable parallel complexity of $\Theta(\log^2 N_s \cdot \log^\gamma N_t)$, where $\gamma = \lceil k/2 \rceil$. This is higher than that for elliptic problems, and the question arises as to whether it can be decreased. There is some hope since it is normally not necessary to solve the "subproblems" exactly in a multigrid solver. The simplest approach is to use Jacobi iteration to approximately solve the banded triangular systems, instead of using cyclic reduction. This is essentially the method of Hackbusch [13], which has been shown to lead to an efficient parallel algorithm for many applications, if some care is taken in choosing the discretization [1], [6], [7], [17], [18]. But the following argument shows that the total number of Jacobi iterations used to approximate the solution of the triangular systems over the course of the algorithm must increase at least linearly as a function of N_t.

Consider solving for the solution at a given location in the space-time grid. As the error due to the discretization decreases, and N_t increases, data throughout an increasing portion of the domain of dependence of the solution operator for this location must be sampled to accurately approximate the solution. Since this domain is independent of N_t, and since some fixed fraction α of the N_t grid points used to approximate the scalar ODE falls in this domain, at least αN_t Jacobi iterations are required for the data at these grid points to be used in approximating the solution at the given location. Thus, asymptotically, the parallel complexity will have a term that grows like N_t. The same argument applies to other point iterative methods that

might be used to solve the triangular systems in parallel. Note that this is not necessarily a condemnation of Hackbusch's method since this growth as a function of N_t can be very small, and may not show up for realistic sized problems.

The obvious approach to avoiding the problem indicated above is to coarsen the grid in time (as well as space) during the multigrid process. In this way, data from more distant grid points can propagate through the coarse grids. Unfortunately, the theory of Lubich and Ostermann does not hold in this situation, and experiments indicate that naive implementations of this approach do not work. Local mode (Fourier) analysis for the example problem and discretizations described in §2.4 confirm the experimental evidence, that coarsening in time does not work, but more sophisticated discretizations, possibly differing between grids, may still allow the time direction to be treated in an analogous way to the spatial directions [4]. In summary, whether or not the parallel complexity can be further reduced, without a corresponding increase in the serial complexity, is not yet known.

2.6. Generalizations

The *waveform relaxation multigrid cyclic reduction* algorithm described above was motivated by the theoretical question introduced in §1. The following generalizations are motivated by practical issues.

Fine grain parallel algorithms By imitating Hockney's PARACR algorithm [15], we can lower the parallel complexity of the parallel cyclic reduction algorithm to $\Theta(\log N_t)$, independent of the length of the recurrence, without increasing the number of processors needed. The trick is to modify all equations at each reduction step. Thus, after $\log_2 N_t$ steps, there are N_t/k independent $k \times k$ systems whose solutions solve the original problem. While the resulting algorithm has a serial complexity of $\Theta(N_t \log N_t)$, this is unimportant on a multiprocessor with $\Theta(N_s \cdot N_t)$ processors.

Coarse grain parallel algorithms - I By imitating the blocked cyclic reduction algorithm discussed in [19] and [20], the communication cost can be reduced to a manageable size for distributed memory multiprocessors. For example, if P_t processors are assigned to the solution of an ODE, the blocked algorithm generates a $P_t \times P_t$ linear system whose solution introduces P_t-way parallelism into the rest of the calculation.

Coarse grain parallel algorithms - II Since each relaxation in the multigrid algorithm involves the solution of many ODEs, much of the analysis used in determining how best to parallelize ADI algorithms for elliptic problems applies immediately [19], [20]. In particular, this analysis addresses the issue of whether to move data for a single ODE to a single processor

and use a fast serial algorithm or to use a cyclic reduction algorithm to parallelize the ODE solution, attempting to overlap communication with computation since parts of many ODE calculations may be assigned to the same processor.

Coarse grain parallel algorithms - III Selectively exploiting parallelism in time can alleviate the inefficiency of solving on coarse grids in the multigrid algorithm. For example, the cyclic reduction algorithm might be used only on coarse grids, when processors have been idled by the coarsening.

3. Hyperbolic PDEs

While waveform relaxation can be used to solve hyperbolic PDEs, multigrid does not accelerate the convergence, and the serial complexity of the resulting algorithm is not $\Theta(N_s \cdot N_t)$. But the same approach to parallelizing in time can be applied to any algorithm whose computational kernel is solving a linear scalar ODE. In this section we briefly describe such an algorithm for constant coefficient hyperbolic PDEs that can be written in the form

$$U_t + \sum_{i=1}^{d} A_i \cdot U_{x_i} = F, \tag{5}$$

where the $n \times n$ matrices $\{A_i\}$ can be simultaneously diagonalized. Here, the problem is defined in d space dimensions and F is a function of both $\bar{x}$ and t, where $\bar{x} = (x_1, \ldots, x_d)$.

Let T be the matrix that diagonalizes $\{A_i \mid i = 1, \ldots, d\}$, $TA_iT^{-1} = \Lambda_i$. Define $V = TU$ and $G = TF$. Then (5) can be written as

$$V_t + \sum_{i=1}^{d} \Lambda_i \cdot V_{x_i} = G. \tag{6}$$

Equation (6) is actually n independent scalar PDEs of the form

$$v_t + \sum_{i=1}^{d} \lambda_i v_{x_i} = g, \tag{7}$$

each of which can be solved be solved by integrating the ODE

$$\frac{d}{dt} v(\lambda_1\xi_1 + t, \ldots, \lambda_d\xi_d + t, t) = g(\lambda_1\xi_1 + t, \ldots, \lambda_d\xi_d + t, t) \tag{8}$$

along the characteristic defined by the set of equations $\{\xi_i = x_i - \lambda_i t \mid i = 1, \ldots, d\}$ for each point $(\xi_1, \ldots, \xi_d)$ in the problem domain [10], [25]. For a numerical algorithm, we would specify a grid in the space-time domain and only track characteristics that exit the space-time domain

at a grid point. To recover the desired variables requires interpolating from characteristics back to the desired space-time grid, and calculating $T^{-1}V$ at each grid location. Since this overhead is a linear function of the number of grid points, the serial complexity of the resulting algorithm is still linear. Both the interpolation and the inversion are "local" processes, and all ODE calculations are independent. Therefore, the parallel complexity of the overall algorithm is $\Theta(\log^{\gamma} N_t)$ when using cyclic reduction, where γ is again determined by the number of levels in the discretization of the ODE.

Note that the form of (5) is very general. For example, the wave equation $u_{tt} - u_{xx} = f$ can be rewritten as

$$\begin{pmatrix} v_1 \\ v_2 \end{pmatrix}_t + \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}_x = \begin{pmatrix} f \\ 0 \end{pmatrix}, \tag{9}$$

where $v_1 = u_t$ and $v_2 = u_x$. After applying the above algorithm to solve for v_1 and v_2, u can be recovered by solving the ODE $u_t = v_1$ for each spatial grid point, using the parallel cyclic reduction algorithm as before. This extra step alters neither the order of the serial complexity nor the order of the parallel complexity.

4. Conclusions

The algorithms described in this paper establish that major classes of linear time-dependent PDEs can be solved in polylog parallel time without giving up linear serial complexity. Beyond the theoretical question, WFMGCR has promise as a practical parallel algorithm, as indicated in §2.6, since WFMG is a competitive serial algorithm for many applications. Additionally, WFMGCR can be used for nonlinear problems since many multigrid solvers automatically linearize the ODEs.

5. Acknowledgements

We thank Stefan Vandewalle for describing how to apply WFMGCR to nonlinear problems, and for the clear explanation of waveform relaxation found in his papers.

6. References

[1] P. Bastain, J. Burmeister, and G. Horton, *Implementation of a parallel multigrid method for parabolic partial differential equations*, in Parallel Algorithms for PDEs, Proceedings of the Sixth GAMM-Seminar, W. Hackbusch, ed., Vieweg-Verlag, Weisbaden, 1990.

[2] A. BRANDT, *Multigrid solvers on parallel computers*, in Elliptic Problem Solvers, M. H. Schultz, ed., Academic Press, New York, 1981, pp. 39–84.

[3] ——, *Guide to multigrid development*, in Multigrid Methods, W. Hackbusch and U. Trottenberg, eds., Springer-Verlag, Berlin, 1982, pp. 220–312.

[4] ——. Personal communication, 1991.

[5] W. L. BRIGGS, *A Multigrid Tutorial*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1987.

[6] J. BURMEISTER, *Paralleles lösen diskreter parabolischer probleme mit mehrgittertechniken*, diplom thesis, University of Kiel, 1985.

[7] J. BURMEISTER AND G. HORTON, *Time-parallel multigrid solution of the Navier-Stokes equations*, in Proceedings of the Third European Multigrid Conference, W. Hackbusch and U. Trottenberg, eds., Birkhäuser Verlag, Basel, 1991.

[8] B. L. BUZBEE, G. H. GOLUB, AND C. W. NIELSON, *On direct methods for solving poisson's equations*, SIAM J. Numer. Anal., 7 (1970), pp. 627–656.

[9] S. C. CHEN AND D. J. KUCK, *Time and parallel processor bounds for linear recurrence systems*. IEEE Trans. Comput., C-24 (1975), pp. 701–717.

[10] R. COURANT AND D. HILBERT, *Methods of Mathematical Physics*, vol. II, Interscience Publishers, New York, 1962.

[11] N. H. DECKER, *Note on the parallel efficiency of the Frederickson-McBryan multigrid algorithm*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 208–220.

[12] H. EISSFELLER AND S. M. MÜLLER, *The triangle method for saving startup time in parallel computers*, in The Fifth Distributed Memory Computing Conference, D. W. Walker and Q. F. Stout, eds., IEEE Computer Society Press, Los Alamitos, CA, 1990, pp. 568–572.

[13] W. HACKBUSCH, *Parabolic multi-grid methods*, in Computing Methods in Applied Sciences and Engineering, R. Glowinski and J.-L. Lions, eds., North-Holland, Amsterdam, 1984, pp. 189–197.

[14] D. HELLER, *On the efficient computation of recurrence relations*, report, Institute for Computer Applications in Science and Engineering, NASA Langley Res. Center, Hampton, VA, June 1974.

[15] R. W. HOCKNEY AND C. R. JESSHOPE, *Parallel Computers: Architecture, Programming, and Algorithms*, Adam Hilger Ltd., Bristol, UK, 1981.

[16] E. HOROWITZ AND S. SAHNI, *Fundamentals of Computer Algorithms*, Computer Science Press, Rockville, Maryland, 1978.

[17] G. HORTON, *Time-parallel multigrid solution of the Navier-Stokes equations*, in Proceedings of the Conference on Applications of Supercomputers in Engineering, C. Brebbia, ed., Computational Mechanics Publ., 1991.

[18] G. HORTON AND R. KNIRSCH, *Time-parallel multigrid in extrapolation method for time-dependent partial differential equations*, Tech. Report Report 1/91, IMMD 3, Universität Erlangen-Nürnberg, D-8520 Erlangen, Germany, 1991.

[19] S. L. JOHNSSON, *Solving tridiagonal systems on ensemble architectures*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 354–392.

[20] S. L. JOHNSSON, Y. SAAD, AND M. H. SCHULTZ, *Alternating direction methods on multiprocessors*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 686–700.

[21] P. M. KOGGE, *The numerical stability of parallel algorithms for solving recurrence problems*, Technical Report No. 44, Digital Systems Laboratory, Stanford Electronics Laboratories, Stanford University, Stanford, CA, September 1972.

[22] ——, *Parallel solution of recurrence problems*, IBM Journal of Research and Development, 18 (1974), pp. 138–148.

[23] P. M. KOGGE AND H. S. STONE, *A parallel algorithm for the efficient solution of a general class of recurrence equations*, IEEE Trans. Comput., C-22 (1973), pp. 786–793.

[24] D. J. KUCK, *The Structure of Computers and Computations*, vol. 1, John Wiley, New York, 1978.

[25] P. D. LAX, *Hyperbolic Systems of Conservation Laws and the Mathematical Theory of Shock Waves*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1973.

[26] 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. Computer-Aided Design, 1 (1982), pp. 131–145.

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

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

[29] A. R. NEWTON AND A. L. SANGIOVANNI-VINCENTELLI, *Relaxation-based electrical simulation*, IEEE Trans. Computer-Aided Design, 3 (1984), pp. 308–331.

[30] J. H. SALTZ, V. K. NAIK, AND D. M. NICOL, *Reduction of the effects of the communication delays in scientific algorithms on message passing mimd architectures*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. s118–s138.

[31] A. H. SAMEH AND R. BRENT, *Solving triangular systems on a parallel computer*, SIAM J. Numer. Anal., 14 (1977), pp. 1101–1113.

[32] S. VANDEWALLE, *Waveform relaxation methods for solving parabolic partial differential equations*, in The Proceedings of the Fifth Distributed Memory Computing Conference, D. W. Walker and Q. F. Stout, eds., IEEE Computer Society Press, Los Alamitos, CA, 1990, pp. 575–584.

[33] S. VANDEWALLE AND R. PIESSENS, *Efficient parallel algorithms for solving initial-boundary value and time-periodic parabolic partial differential equations*, Report TW 139, Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A B-3030 Leuven, Belgium, November 1990.

[34] ——, *Numerical experiments with multigrid waveform relaxation on a parallel processor*, Report TW 142, Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A B-3030 Leuven, Belgium, November 1990.

[35] S. VANDEWALLE AND D. ROOSE, *The parallel waveform relaxation multigrid method*, in The Proceedings of the Third SIAM Conference on Parallel Processing for Scientific Computing, G. Rodrigue, ed., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1989, pp. 152–156.

[36] S. VANDEWALLE, D. ROOSE, AND R. PIESSENS, *A comparison of two parallel multigrid methods for the numerical solution of parabolic partial differential equations*, in The Proceedings of the Fourth Conference on Hypercubes, Concurrent Computers, and Applications, J. L. Gustafson, ed., Golden Gate Enterprises, P.O. Box 428, Los Altos, CA, 1990, pp. 1287–1290.

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

[38] D. E. WOMBLE, *Multigrid on massively parallel computers*, in The Proceedings of the Fifth Distributed Memory Computing Conference, D. W. Walker and Q. F. Stout, eds., IEEE Computer Society Press, Los Alamitos, CA, 1990, pp. 559–563.

[39] ——, *A time-stepping algorithm for parallel computers*, SIAM J. Sci. Statist. Comput., 11 (1991), pp. 824–837.

[40] P. H. WORLEY, *Information Requirements and the Implications for Parallel Computation*, Ph.D. thesis, Stanford University, Stanford, CA, June 1988.

[41] ——, *The effect of multiprocessor radius on scaling*, Tech. Report ORNL/TM-11579, Oak Ridge National Laboratory, Oak Ridge, TN, June 1990.

[42] ——, *The effect of time constraints on scaled speedup*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 838–858.

[43] ——, *Limits on parallelism in the numerical solution of linear PDEs*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 1–35.

INTERNAL DISTRIBUTION

1. V. Alexiades
2. B. R. Appleton
3. E. F. D'Azevedo
4–5. T. S. Darland
6. J. M. Donato
7. J. J. Dongarra
8. J. B. Drake
9. G. A. Geist
10. L. Gray
11. M. R. Leuze
12. E. G. Ng
13. C. E. Oliver
14. B. W. Peyton
15–19. S. A. Raby
20. C. H. Romine
21. T. H. Rowan
22–26. R. F. Sincovec
27. D. W. Walker
28–32. R. C. Ward
33–37. P. H. Worley
38. Central Research Library
39. ORNL Patent Office
40. K-25 Applied Technology Library
41. Y-12 Technical Library / Document Reference Station
42. Laboratory Records - RC
43–44. Laboratory Records Department

EXTERNAL DISTRIBUTION

45. Loyce M. Adams, Applied Mathematics, FS-20, University of Washington, Seattle, WA 98195

46. Christopher R. Anderson, Department of Mathematics, University of California, Los Angeles, CA 90024

47. Donald M. Austin, 6196 EECS Bldg, University of Minnesota, 200 Union St., S.E., Minneapolis, MN 55455

48. Robert G. Babb, Oregon Graduate Center, CSE Department, 19600 N.W. Walker Road, Beaverton, OR 97006

49. David H. Bailey, NASA Ames, Mail Stop 258-5, NASA Ames Research Center, Moffet Field, CA 94035

50. Edward H. Barsis, Computer Science and Mathematics, P. O. Box 5800, Sandia National Laboratory, Albuquerque, NM 87185

51. Robert E. Benner, Parallel Processing Division 1413, Sandia National Laboratories, P. O. Box 5800, Albuquerque, NM 87185

52. Marsha J. Berger, Courant Institute of Mathematical Sciences, 251 Mercer Street, New York, NY 10012

53. Ake Bjorck, Department of Mathematics, Linkoping University, S-581 83 Linkoping, Sweden

54. John H. Bolstad, L-16, Lawrence Livermore National Laboratory, P. O. Box 808, Livermore, CA 94550

55. Achi Brandt, Weizmann Institute of Science, Department of Applied Mathematics, Rehovot, Israel

56. Bill L. Buzbee, Scientific Computing Division, National Center for Atmospheric Research, P. O. Box 3000, Boulder, CO 80307

57. John Cavallini, Acting Director, Scientific Computing Staff, Applied Mathematical Sciences, Office of Energy Research, U.S. Department of Energy, Washington, DC 20585

58. Tony Chan, Department of Mathematics, University of California, Los Angeles, 405 Hilgard Avenue, Los Angeles, CA 90024

59. Jagdish Chandra, Army Research Office, P. O. Box 12211, Research Triangle Park, NC 27709

60. Melvyn Ciment, National Science Foundation, 1800 G Street N.W., Washington, DC 20550

61. Tom Coleman, Department of Computer Science, Cornell University, Ithaca, NY 14853

62. Paul Concus, Mathematics and Computing, Lawrence Berkeley Laboratory, Berkeley, CA 94720

63. Jane K. Cullum, IBM T. J. Watson Research Center, P. O. Box 218, Yorktown Heights, NY 10598

64. George Cybenko, Center for Supercomputing Research and Development, University of Illinois, 104 South Wright Street, Urbana, IL 61801-2932

65. John J. Dorning, Department of Nuclear Engineering Physics, Thornton Hall, McCormick Road, University of Virginia, Charlottesville, VA 22901

66. Iain S. Duff, Atlas Centre, Rutherford Appleton Laboratory, Chilton, Oxon OX11 0QX, England

67. Stanley Eisenstat, Department of Computer Science, Yale University, P. O. Box 2158 Yale Station, New Haven, CT 06520

68. Howard C. Elman, Computer Science Department, University of Maryland, College Park, MD 20742

69. Peter G. Eltgroth, L-298, Lawrence Livermore National Laboratory, P. O. Box 808, Livermore, CA 94550

70. Philippe Engrand, Department of Nuclear Engineering, North Carolina State University, Box 7909, Raleigh, NC 27695-7909

71. Geoffrey C. Fox, NPAC, 111 College Place, Syracuse University, Syracuse, NY 13244-4100

72. Chris Fraley, Statistical Sciences, Inc., 1700 Westlake Ave. N, Suite 500, Seattle, WA 98119

73. Paul O. Frederickson, RIACS, MS 230-5, NASA Ames Research Center, Moffet Field, CA 94035

74. Fred N. Fritsch, L-300, Mathematics and Statistics Division, Lawrence Livermore National Laboratory, P. O. Box 808, Livermore, CA 94550

75. Robert E. Funderlic, Department of Computer Science, North Carolina State University, Raleigh, NC 27650

76. Dennis B. Gannon, Computer Science Department, Indiana University, Bloomington, IN 47405

77. C. William Gear, NEC Research Institute, 4 Independence Way, Princeton, NJ 08540

78. W. Morven Gentleman, Division of Electrical Engineering, National Research Council, Building M-50, Room 344, Montreal Road, Ottawa, Ontario, Canada K1A 0R8

79. Alan George, Vice President, Academic and Provost, Needles Hall, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1

80. Gene Golub, Computer Science Department, Stanford University, Stanford, CA 94305

81. Joseph F. Grcar, Division 8331, Sandia National Laboratories, Livermore, CA 94550

82. William D. Gropp, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439

83. Eric Grosse, AT&T Bell Labs 2T-504, Murray Hill, NJ 07974

84. John L. Gustafson, Ames Laboratory, 236 Wilhelm Hall, Iowa State University, Ames, IA 50011-3020

85. Michael T. Heath, National Center for Supercomputing Applications, 4157 Beckman Institute University of Illinois, 405 North Mathews Avenue, Urbana, IL 61801-2300

86. Gerald W. Hedstrom, L-71, Lawrence Livermore National Laboratory, P. O. Box 808, Livermore, CA 94550

87. Don E. Heller, Physics and Computer Science Department, Shell Development Co., P. O. Box 481, Houston, TX 77001

88. N. J. Higham, Department of Mathematics, University of Manchester, Gtr Manchester, M13 9PL, England

89. Charles J. Holland, Air Force Office of Scientific Research, Building 410, Bolling Air Force Base, Washington, DC 20332

90. Kenneth R. Jackson, Computer Science Department, University of Toronto, Toronto, Ontario, M5S 1A4, Canada

91. Lennart Johnsson, Thinking Machines Inc., 245 First Street, Cambridge, MA 02142-1214

92. Harry Jordan, Department of Electrical and Computer Engineering, University of Colorado, Boulder, CO 80309

93. Bo Kagstrom, Institute of Information Processing, University of Umea, 5-901 87 Umea, Sweden

94. Malvyn Kalos, Cornell Theory Center, Engineering and Theory Center Bldg., Cornell University, Ithaca, NY 14853-3901

95. Hans Kaper, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Bldg. 221, Argonne, IL 60439

96. Alan H. Karp, IBM Scientific Center, 1530 Page Mill Road, Palo Alto, CA 94304

97. Linda Kaufman, Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974

98. Joseph B. Keller, Department of Mathematics, Stanford University, Stanford, CA 94305

99. Kenneth Kennedy, Department of Computer Science, Rice University, P.O. Box 1892, Houston, TX 77001

100. A. Q. M. Khaliq, Department of Mathematics, Western Illinois University, Macomb, IL 61455

101. Michael Langston, Department of Computer Science, University of Tennessee, Knoxville, TN 37996-1301

102. Richard Lau, Office of Naval Research, 1030 E. Green Street, Pasadena, CA 91101

103. Robert L. Launer, Army Research Office, P. O. Box 12211, Research Triangle Park, NC 27709

104. Charles Lawson, MS 301-490, Jet Propulsion Laboratory, 4800 Oak Grove Drive, Pasadena, CA 91103

105. Peter D. Lax, Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012

106. James E. Leiss, Rt. 2, Box 142C, Broadway, VA 22815

107. Robert Leland, Sandia National Laboratories, 1424, P. O. Box 5800, Albuquerque, NM 87185-5800

108. Randall J. LeVeque, Applied Mathematics, FS-20, University of Washington, Seattle, WA 98195

109. John G. Lewis, Boeing Computer Services, P. O. Box 24346, M/S 7L-21, Seattle, WA 98124-0346

110. Heather M. Liddell, Center for Parallel Computing, Department of Computer Science and Statistics, Queen Mary College, University of London, Mile End Road, London E1 4NS, England

111. Thomas A. Manteuffel, Department of Mathematics, University of Colorado - Denver, Denver, CO 80202

112. Anita Mayo, IBM T. J. Watson Research Center, P. O. Box 218, Yorktown Heights, NY 10598

113. James McGraw, Lawrence Livermore National Laboratory, L-306, P. O. Box 808, Livermore, CA 94550

114. Paul C. Messina, Mail Code 158-79, California Institute of Technology, 1201 E. California Blvd. Pasadena, CA 91125

115. Willard L. Miranker, IBM T. J. Watson Research Center, P. O. Box 218, Yorktown Heights, NY 10598

116. Neville Moray, Department of Mechanical and Industrial Engineering, University of Illinois, 1206 West Green Street, Urbana, IL 61801

117. William A. Mulder, Koninklijke Shell Exploratie en Produktie Laboratorium, Postbus 60, 2280 AB Rijswijk, THE NETHERLANDS

118. Dianne P. O'Leary, Computer Science Department, University of Maryland, College Park, MD 20742

119. Joseph Oliger, Computer Science Department, Stanford University, Stanford, CA 94305

120. James M. Ortega, Department of Applied Mathematics, Thornton Hall, University of Virginia, Charlottesville, VA 22901

121. Beresford N. Parlett, Department of Mathematics, University of California, Berkeley, CA 94720

122. Merrell Patrick, Department of Computer Science, Duke University, Durham, NC 27706

123. Linda R. Petzold, Computer Science Department, University of Minnesota, 200 Union Street, S.E., Room 4-192, Minneapolis, MN 55455

124. Robert J. Plemmons, Departments of Mathematics and Computer Science, North Carolina State University, Raleigh, NC 27650

125. John K. Reid, Numerical Analysis Group, Central Computing Department, Atlas Centre, Rutherford Appleton Laboratory, Didcot, Oxon OX11 0QX, England

126. Werner C. Rheinboldt, Department of Mathematics and Statistics, University of Pittsburgh, Pittsburgh, PA 15260

127. John R. Rice, Computer Science Department, Purdue University, West Lafayette, IN 47907

128. Garry Rodrigue, Numerical Mathematics Group, Lawrence Livermore National Laboratory, Livermore, CA 94550

129. Donald J. Rose, Department of Computer Science, Duke University, Durham, NC 27706

130. Ahmed H. Sameh, Department of Computer Science, University of Minnesota, 200 Union Street S.E., Minneapolis, MN 55455

131. Jorge Sanz, IBM Almaden Research Center, Department K53/802, 650 Harry Road, San Jose, CA 95120

132. Robert B. Schnabel, Department of Computer Science, University of Colorado at Boulder, ECOT 7-7 Engineering Center, Campus Box 430, Boulder, CO 80309-0430

133. Robert Schreiber, RIACS, MS 230-5, NASA Ames Research Center, Moffet Field, CA 94035

134. Martin H. Schultz, Department of Computer Science, Yale University, P. O. Box 2158 Yale Station, New Haven, CT 06520

135. David S. Scott, Intel Scientific Computers, 15201 N.W. Greenbrier Parkway, Beaverton, OR 97006

136. The Secretary, Department of Computer Science and Statistics, The University of Rhode Island, Kingston, RI 02881

137. Lawrence F. Shampine, Mathematics Department, Southern Methodist University, Dallas, TX 75275

138. Horst D. Simon, Mail Stop 258-5, NASA Ames Research Center, Moffett Field, CA 94035

139. William C. Skamarock, 3973 Escuela Court, Boulder, CO 80301

140. Bob Skeel, Department of Computer Science, University of Illinois at Urbana-Champaign, 1304 West Springfield Avenue, Urbana, IL 61801-2987

141. Tony Skjellum, L-316, Lawrence Livermore National Laboratory, P. O. Box 808, Livermore, CA 94550

142. Burton Smith, Tera Computer Company, 400 North 34th Street, Suite 300, Seattle, WA 98103

143. Danny C. Sorensen, Department of Mathematical Sciences, Rice University, P. O. Box 1892, Houston, TX 77251

144. G. W. Stewart, Computer Science Department, University of Maryland, College Park, MD 20742

145. Steven Suhr, Computer Science Department, Stanford University, Stanford, CA 94305

146. Paul N. Swarztrauber, National Center for Atmospheric Research, P. O. Box 3000, Boulder, CO 80307

147. Wei Pai Tang, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2l 3G1

148. Bernard Tourancheau, LIP, ENS-Lyon, 69364 Lyon cedex 07, France

149. Joseph F. Traub, Department of Computer Science, 450 Computer Science Building, Columbia University, New York, NY 10027

150. Lloyd N. Trefethen, Department of Computer Science, Cornell University, Ithaca, NY 14853

151. Stefan Vandewalle, Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, B-3030 Heverlee, Belgium

152. Charles Van Loan, Department of Computer Science, Cornell University, Ithaca, NY 14853

153. Robert G. Voigt, ICASE, MS 132-C, NASA Langley Research Center, Hampton, VA 23665

154. Bi R. Vona, Center for Numerical Analysis, RLM 13.150, University of Texas at Austin, Austin, TX 78712

155. David Voss, Department of Mathematics, Western Illinois University, Macomb, IL 61455

156. Phuong Vu, Cray Research, Inc., 655F Lone Oak Drive, Eagen, MN 55121

157. A. J. Wathen, School of Mathematics, University Walk, Bristol BSB 1TW, England

158. Mary F. Wheeler, Rice University, Department of Mathematical Sciences, P. O. Box 1892, Houston, TX 77251

159. Andrew B. White, Computing Division, Los Alamos National Laboratory, Los Alamos, NM 87545

160. David E. Womble, Sandia National Laboratories, Division 1422, P. O. Box 5800, Albuquerque, NM 87185

161. David M. Young, Jr., Center for Numerical Analysis, The University of Texas at Austin, RLM 13.150, Austin, TX 78713-8510

162. Office of Assistant Manager for Energy Research and Development, U.S. Department of Energy, Oak Ridge Operations Office, P. O. Box 2001, Oak Ridge, TN 37831-8600

163–172. Office of Scientific & Technical Information, P. O. Box 62, Oak Ridge, TN 37831

DATE
FILMED
01/06/92