

PARALLEL PERFORMANCE OF SOME MULTIGRID SOLVERS FOR THREE-DIMENSIONAL PARABOLIC EQUATIONS*

MICHAEL HOLST[†] AND FAISAL SAIED[†]

Abstract. We consider the solution of parabolic PDEs in three spatial dimensions by multigrid methods on parallel architectures. The objective is to develop high performance multigrid solvers for large scale time-dependent problems. We begin with temporal semi-discretizations with several explicit and implicit schemes, followed by a uniform spatial discretization in a 3D rectangular region. A nearly optimal time-step is then determined empirically for each of the resulting methods when applied to a given test problem, and the sequential performance of each method is then compared. We examine the effect of the step size on the convergence rates of multigrid, and present results of experiments on two parallel architectures: a four processor Cray Y-MP and a fourteen processor Alliant FX/2800. Our results demonstrate that multigrid can achieve very high parallel efficiency on coarse-grained shared memory machines, and that explicit methods are not likely to be competitive with multigrid based implicit methods on such machines.

Key Words. multigrid methods, partial differential equations, time-dependent, three-dimensional, parabolic equations, parallel algorithms, Cray Y-MP, Alliant FX/2800, shared memory architectures

1. Introduction. In this paper, we consider the solution of three-dimensional parabolic partial differential equations. The objective is to develop high performance solvers using multigrid techniques for large scale time-dependent problems of this type. The target machines are shared memory multiprocessors, including specifically the Alliant FX/2800 and the Cray Y-MP. We begin the paper with a discussion of explicit and implicit temporal semi-discretization schemes, and examine the elliptic problems that must be solved at each time step in the implicit schemes. Our approach to solving these shifted 3D elliptic problems is to use the multigrid method. The use of implicit time stepping schemes coupled with multigrid for solution of the resulting elliptic problems has been examined by Hackbusch, Brandt et al. in [8, 17] for 2D problems. We examine the behavior of the multigrid method for implicit time discretizations of 3D problems, and focus on the efficient parallel solution of the shifted elliptic partial differential equations arising at each time step. Issues of step size selection, multigrid initialization at each time step, and multigrid convergence rates for the shifted problems naturally arise, and we attempt to answer some of these questions.

The remainder of this paper is structured as follows. In § 2, we introduce the class of problems we wish to solve. In § 3, we discuss time discretization techniques, and in particular we outline the approach taken in our implementations. The elliptic problems arising in implicit time discretizations are considered in § 4, and we discuss the effect of the shift introduced into the operator as a result of the time step. In § 5 we review multigrid methods, and discuss the current state of research on multigrid for 3D parabolic problems. Two test problems are presented in § 6: the heat equation in three spatial dimensions, and a variable coefficient problem. In § 7, we present some experiments investigating the performance of explicit and implicit methods for the test problems on a sequential computer, and examine in more detail the effect of the shifted operator on the performance of multigrid. We present performance results for our implementations on some advanced computer architectures in § 8, namely the Alliant FX/2800 and the Cray Y-MP. Finally, we summarize our results in § 9.

2. Parabolic Problems in Three Space Dimensions. We seek the solution $u = u(x, y, z, t)$ of a parabolic partial differential equation of the form

$$(1) \quad \begin{aligned} u_t &= Lu + f \quad \text{in } \Omega \subset \mathbb{R}^3, \\ u(x, y, z, t) &= g(x, y, z, t) \quad \text{on } \partial\Omega, \\ u(x, y, z, 0) &= h(x, y, z) \quad \text{at time } t = 0, \end{aligned}$$

* THIS WORK WAS SUPPORTED IN PART BY NATIONAL SCIENCE FOUNDATION GRANT NO. DMS 89-11410 AND BY RESEARCH BOARD OF THE UNIVERSITY OF ILLINOIS GRANT NO. RES BRD IC SAIED F.

[†] NUMERICAL COMPUTING GROUP, DEPARTMENT OF COMPUTER SCIENCE, UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN, 1304 W. SPRINGFIELD AVENUE, URBANA, IL 61801.

where Ω is some cuboidal region in \mathbb{R}^3 . We consider operators of the form $L = \nabla \cdot (a\nabla) + \vec{b} \cdot \nabla + c$, and in particular the case when the tensor a is diagonal:

$$(2) \quad Lu = \frac{\partial}{\partial x} \left(a_{11} \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial y} \left(a_{22} \frac{\partial u}{\partial y} \right) + \frac{\partial}{\partial z} \left(a_{33} \frac{\partial u}{\partial z} \right) + b_1 \frac{\partial u}{\partial x} + b_2 \frac{\partial u}{\partial y} + b_3 \frac{\partial u}{\partial z} + cu,$$

where a_{ii}, b_i , and c are functions of x, y, z , and possibly t .

Problems of this structure arise in numerous applications involving diffusive or thermodynamic processes. Examples include groundwater and petroleum reservoir simulation, quantum mechanics, ocean acoustics, and neutron transport.

3. Time Discretizations of Parabolic Problems. The semi-discretization of the time variable in the parabolic pde $u_t = Lu + f$ can be accomplished in several ways, involving the use of explicit or implicit techniques [29, 35]. Explicit methods, in which the solution at the new time step is formed by a combination of previous time step solutions, are simple to code and understand, and inexpensive computationally to apply at a single time step. Examples include Forward Euler, Leapfrog, DuFort-Frankel, and Runge Kutta methods. For example, we can represent the semi-discretization using Forward Euler as:

$$(3) \quad u^{n+1} = u^n + \Delta t (L^n u^n + f^n),$$

where Δt is the time step from time t_n to time t_{n+1} , u^n is the solution, f^n is the forcing function, and L^n is the elliptic operator at time t_n . The function u^{n+1} is then the new approximate solution at time t_{n+1} , with a discretization error of $O(\Delta t)$ for this method. Of course with explicit methods there will be stability limit on the time step size.

To avoid the stability considerations implicit methods can be used, and the step size may be chosen purely on the basis of accuracy. They are more costly to apply, requiring the solution of linear algebraic equations at each time step (or nonlinear algebraic equations, if the elliptic operator is nonlinear). Possible choices include the Backward Euler (4) and Crank-Nicolson (5) methods:

$$(4) \quad \left(L^{n+1} - \frac{1}{\Delta t} \right) u^{n+1} = - \left(\frac{1}{\Delta t} \right) u^n - f^{n+1},$$

$$(5) \quad \left(L^{n+1} - \frac{2}{\Delta t} \right) u^{n+1} = - \left(L^n + \frac{2}{\Delta t} \right) u^n - (f^n + f^{n+1}).$$

Note that by introducing the variable V^n , a recursion can be used in the Crank-Nicolson method to avoid the application of the operator L on the right hand side:

$$(6) \quad \left(L^{n+1} - \frac{2}{\Delta t} \right) u^{n+1} = V^n,$$

where V^n is defined by:

$$V^0 = - \left(L^0 + \frac{2}{\Delta t} \right) u^0 - (f^0 + f^1),$$

$$V^n = -V^{n-1} - \left(\frac{4}{\Delta t} \right) u^n - (f^{n+1} + f^n).$$

This technique can result in an appreciable saving in computations (a matrix-vector product with the discrete analogue of the operator L of equation (2)) when compared to the original method (5). This approach is similar in spirit to one used in [12] for saving matrix-vector products in the ADI method.

4. Elliptic Problems Arising in Implicit Schemes. In each of the implicit methods discussed in the previous section, an elliptic partial differential equation of the following form must be solved at each time step:

$$(7) \quad \left(L - \frac{\alpha}{\Delta t}\right) u = f.$$

For example, with Crank-Nicolson $\alpha = 2$, while $\alpha = 1$ for Backward Euler. Our approach to solving this shifted 3D elliptic problem is to use the multigrid method, as was done for implicit time discretizations of parabolic problems in two space dimensions in [8, 17].

Another approach appears in [28, 32, 33, 34], where 2D parabolic problems are first discretized in space, followed by time discretization to solve the resulting system of ordinary differential equations. In particular, several explicit and implicit time discretization methods are compared in [34], involving the use of various linear solvers (including multigrid) for the linear algebraic equations arising at each time step of the implicit methods.

Our purpose is to examine and compare several semi-discretizations in time of 3D parabolic problems, study the behavior of the multigrid method for different time discretizations, and focus on the efficient parallel solution of the shifted elliptic partial differential equations arising at each time step.

4.1. Shifted Elliptic Operators. It is noted in [8] that when a semi-discretization in time is performed in the case of the 2D heat equation, the resulting shifted elliptic operators at each time step are Helmholtz operators with shift $\frac{\alpha}{\Delta t}$, and the behavior of classical relaxation methods, as well as that of the multigrid method, is analyzed in [30] for model problems of this type.

Here, we examine the effect of the time shift on the convergence behavior of multigrid for 3D elliptic operators, including Helmholtz-like operators, as well as more general variable coefficient problems.

The effect of the shift is a better conditioned problem, which will have a beneficial effect on most solution techniques. In (§ 7), we present numerical results demonstrating how multigrid methods benefit from the shift.

4.2. Spatial Discretization of the Elliptic Problem. We put down a uniform 3D mesh of width h on Ω , and discretize the shifted elliptic operator using centered finite difference approximations (the 7-point stencil). Given that the elliptic operator is the general second order linear operator in equation (2), and that we order the discrete unknowns using the natural ordering, a block tri-diagonal linear system $L_h u_h = f_h$ results. We denote the block tri-diagonal system corresponding to a mesh-width of $2h$ as $L_{2h} u_{2h} = f_{2h}$, and so on.

Since cross derivatives are not present in (2), the matrix L_h has the non-zero structure of the discrete 3D Laplacian (Figure 1), where the nonzeros of the matrix lie on seven diagonals. We use a data structure in our implementations that reflects the diagonal form of the discrete operator, and as a result matrix-vector operations are extremely efficient on vector computers such as the Cray Y-MP (e.g. [25]). For example, this storage format can achieve a factor of five speedup in the time taken to compute a matrix-vector product when compared to the equivalent operation implemented with more general sparse matrix storage formats, even when the gather-scatter hardware is used [27].

5. Multigrid Methods for 3D Elliptic Problems. Multigrid methods are highly efficient techniques for solving certain types of partial differential equations. The study of these methods constitute a very active research area in numerical analysis and scientific computing, and this section represents a survey of some of the basic methods, mentioning special considerations for 3D problems. More detailed discussions can be found in any of [5, 6, 18, 30].

5.1. The Multigrid Idea. The basic idea behind multigrid methods for partial differential equations is the use of multiple grids to resolve different features of the solution on the appropriate scales. In particular, multigrid methods avoid the inefficiency of dealing with “coarse scale” phenomenon (low frequency errors) on the finest grid.

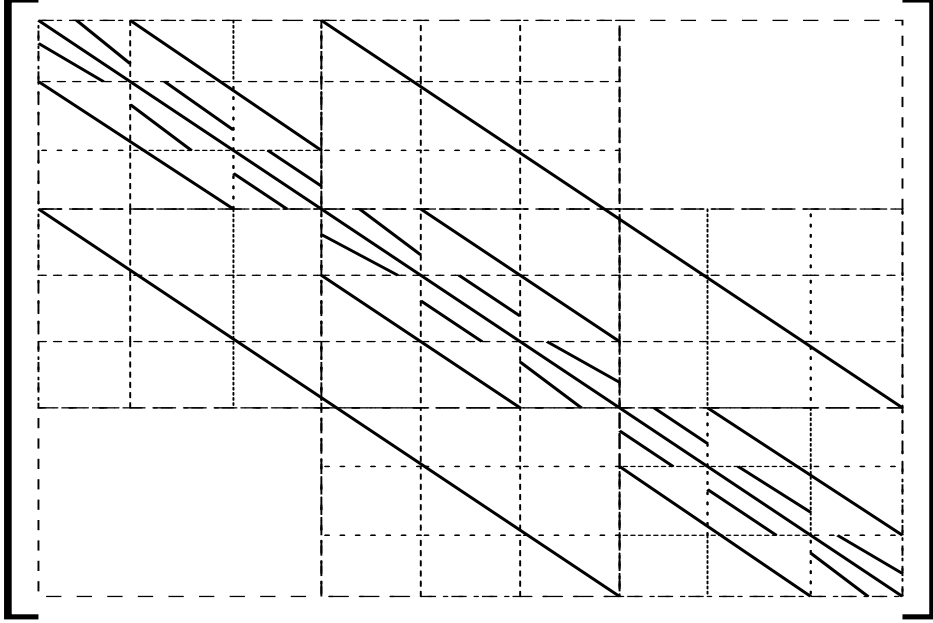


FIG. 1. Block tri-diagonal form of the discretized 3D elliptic operator for a $3 \times 3 \times 3$ mesh.

The Two-Grid Algorithm. If uniform three dimensional meshes of widths h and $2h$ are used to discretize the elliptic partial differential equation discussed in § 4, we will then have two sets of equations representing discrete approximations to the original PDE; namely $L_h u_h = f_h$, and $L_{2h} u_{2h} = f_{2h}$. A two-grid *Correction Storage (CS)* multigrid scheme [5, 30] using both the fine grid and coarse grid equations to solve the underlying PDE can be represented graphically as in Figure 2 (cf. [30]).

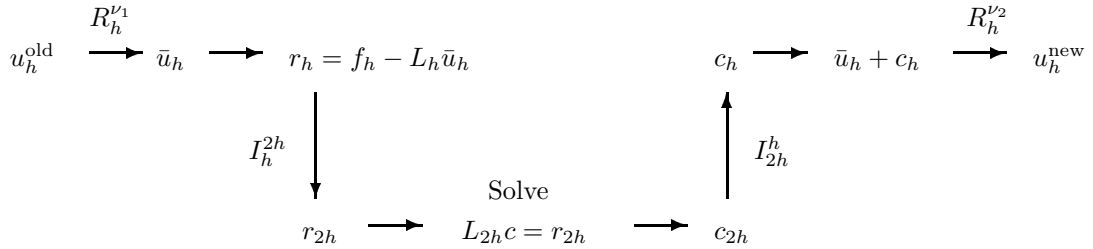


FIG. 2. The two grid Correction Storage (CS) scheme.

The method begins with an initial approximation u_h^{old} on the finest grid. A *relaxation* (or *smoothing*) method R_h is then employed for ν_1 iterations, resulting in an approximation \bar{u}_h with smooth error and residual. Next, a residual computation is performed, and this residual r_h is *restricted* to the coarse grid via the restriction operator I_h^{2h} , resulting in a residual on the coarse grid, r_{2h} .

The coarse grid problem is solved in some way (perhaps directly, or by more relaxations), and the correction c_{2h} obtained from the residual equation is then *interpolated* back to the fine grid via the operator I_{2h}^h , resulting in the correction c_h . The correction is added to the approximate solution, and ν_2 smoothings are done to reduce any high frequency error that has been introduced by the interpolation operator, resulting in the final approximation u_h^{new} from the multigrid cycle.

Note that linearity was used to motivate the CS scheme correction equation: $Le = L(u^* - u) = Lu^* - Lu = f - Lu = r$. While there is no direct analogue in the case of a nonlinear operator \mathcal{L} , a modified correction equation can be used, and the resulting algorithm will require a nonlinear relaxation operator \mathcal{R}_h , in addition to an analogous coarse-grid correction step. (If the nonlinearities

occur in the discrete equations in particularly simple forms, which is the case in many applications such as computational fluid dynamics, then nonlinear analogues of the classical relaxation procedures such as Jacobi iteration are easily defined [7].)

A description of the two-grid *Full Approximation Storage (FAS)* scheme employing the more general correction scheme can be found in [5, 18]. In this scheme, a combination of the fine grid residual r_h and the fine grid solution \bar{u}_h are transferred to the coarse grid to become the right-hand side of the coarse grid correction equation:

$$\bar{f}_{2h} = \mathcal{L}_{2h}(\bar{I}_h^{2h}\bar{u}_h) + I_h^{2h}(f_h - \mathcal{L}_h(\bar{u}_h)).$$

The correction u_{2h} is then returned to the fine grid as:

$$c_h = \bar{I}_{2h}^h(u_{2h} - I_h^{2h}\bar{u}_h).$$

(Note that $\bar{I}_{h_1}^{h_2}$ and $I_{h_1}^{h_2}$ may be defined independently.) While the FAS scheme is more expensive than the CS scheme because of the more complicated formula for \bar{f}_{2h} , it has several advantages: it can be used for both linear and nonlinear problems, whereas the CS scheme is restricted to the linear case; truncation error estimates are easily obtained, enabling adaptive implementations, including adaptive time step-size control for the parabolic problems we are considering in this paper.

Smoothers. The smoothing operator R_h can be chosen to be one of the classical methods such as Jacobi, Red/Black Gauss-Seidel (point, line, and plane versions), or weighted Jacobi. Of particular interest for 3D problems are the alternating plane relaxations, which become increasingly important for problems with anisotropic or discontinuous coefficients [9, 31]. Other methods that have been proposed include incomplete *LU* factorizations, as well as the conjugate gradient method [1]. The 2D multigrid method itself has been proposed as a smoother for 3D multigrid [9, 31].

The results presented in this paper are based on the weighted Jacobi method, which can be written for a linear system $Ax = b$ as:

$$x^{(n+1)} = x^{(n)} + \omega D^{-1}(b - Ax^{(n)}),$$

where D represents the diagonal of the matrix A .

Grid Transfer Operators. The grid transfer operators we have used are the natural 3D extensions of some of the standard 2D operators, as described in [24]. In particular, we use a 3D full weighting restriction operator for fine to coarse grid transfers, which involves twenty seven fine grid points to form a single coarse grid point, and a tri-linear interpolation operator for coarse to fine grid transfers.

Solution of the Coarse Grid Problem. Our choice of the coarse grid solution technique was banded Gaussian elimination [10]. For reasons discussed in [22], we typically use enough grid levels so that the percentage of time spent in coarse grid solves amounts to roughly five percent of the total solution time or less. This implies a small coarse grid problem, and a banded direct solve is quite fast for these problem sizes. Although sparse Gaussian elimination is asymptotically cheaper, banded Gaussian elimination can solve the small coarse grid problems in comparable time (if not faster, particularly on vector processors).

5.2. The V-cycle and Full Multigrid. Multigrid is a recursive application of the two-grid algorithms discussed above. This idea can be applied successively, until the cost of solving the coarse grid problem is negligible. If the algorithm begins with the fine grid, cycles down to the coarse grid, and then returns to the fine grid, it is called a *V-cycle (MG)* [5, 18]. An alternative procedure begins with the coarse grid, interpolates the solution (not a correction) to a finer grid, performs a V-cycle from that grid level, interpolates the solution to still a finer grid, and repeats the process, until a V-cycle is performed on the finest grid. This is called *full multigrid (FMG)* or *nested iteration* [5, 18]. Both algorithms are depicted in Figure 3. The dashed lines in the FMG picture represent coarse-to-fine interpolations in which a fine grid is visited for the first time. Ideally, an accurate interpolation scheme (e.g., cubic) should be used in these steps, but in the results presented in this paper, we have used tri-linear interpolation.

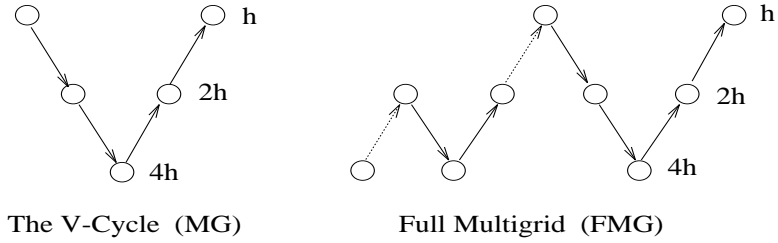


FIG. 3. Two multigrid algorithms.

The advantage of FMG over MG is that it provides a more accurate initial guess for the V-Cycle than an arbitrary or zero initial guess, at a small additional cost. (Discretization error accuracy can usually be reached in one FMG iteration, whereas several MG iterations may be required to reach the same accuracy.) However, time-dependent problems naturally provide a third option: we can exploit information available from previous time steps (the simplest approach is to use the solution at time t_n as the initial guess for multigrid at time t_{n+1}). We discuss this further in § 7.

For an $n \times n \times n$ grid, the cost of dense, banded, and sparse Gaussian elimination is $O(n^9)$, $O(n^7)$, and $O(n^6)$, respectively. In contrast, one iteration of multigrid costs $O(n^3)$ operations, and the number of iterations of multigrid required to solve the problem to discretization error accuracy is typically very small. For certain classes of problems, it can be shown rigorously that the number of iterations required is independent of n , which implies that multigrid methods are of optimal order for these problems.

In three dimensions, the additional memory required by multigrid is not excessive. If M_h is the memory required to describe the problem on the finest grid, then the total memory M needed by multigrid is approximately:

$$M = M_h + M_{2h} + M_{4h} + \dots = M_h \left(1 + \frac{1}{8} + \frac{1}{16} + \dots\right) \leq \frac{8}{7} M_h.$$

Many convergence proofs can be found in the literature [1, 2, 4, 16] and are based on an analysis of the iteration matrix of the multigrid method, which for the two-grid CS scheme is given by:

$$G_h^{2h} = S_h^{\nu_2} (I_h - I_{2h}^h L_{2h}^{-1} I_h^{2h} L_h) S_h^{\nu_1},$$

where S_h represents the iteration matrix for the relaxation process R_h .

6. Test Problems. Our test problems are defined on the unit cube and are of the form given in equations (1) and (2). We will present results for the following two test problems.

Problem 1. For this problem, L is chosen to be the Laplacian, i.e., in (2) we set

$$a_{11} = a_{22} = a_{33} = 1, \quad b_1 = b_2 = c = 0.$$

The boundary and the initial conditions are chosen to agree with a test solution which we have chosen, namely:

$$u(x, y, z, t) = e^{-t} \sin(\pi x) \sin(\pi y) \sin(\pi z).$$

Problem 2. This problem is a generalization to three dimensions of a 2D test problem that appeared in [34]. In particular, L is a time-dependent, variable coefficient, non-self-adjoint, elliptic operator. The coefficients in (2) are defined as follows:

$$a_{11} = \frac{t}{6(x+1)^2}, \quad a_{22} = \frac{t}{6(y+1)^2}, \quad a_{33} = \frac{t}{6(z+1)^2},$$

$$b_1 = \frac{-t}{6(x+1)^3}, \quad b_2 = \frac{-t}{6(y+1)^3}, \quad b_3 = \frac{-t}{6(z+1)^3}, \quad c = 0.$$

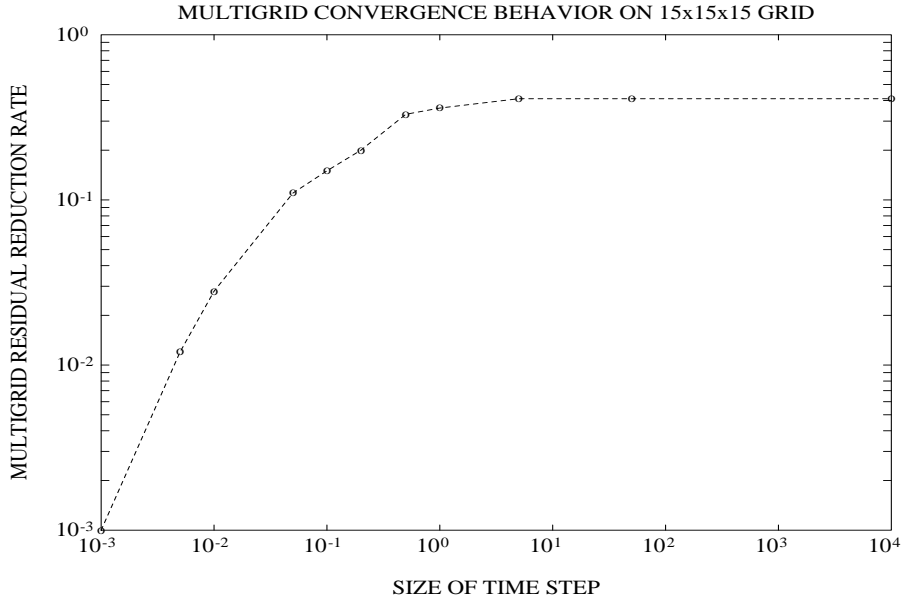


FIG. 4. Effect of varying the Crank-Nicolson time shift in Problem 2 on a $15 \times 15 \times 15$ mesh.

The boundary and initial conditions are chosen to agree with the test solution which we choose as

$$u(x, y, z, t) = e^{-t^2} \sin((x+1)^2 + (y+1)^2 + (z+1)^2).$$

The results which we present in § 7 are for Problem 2, while the results presented in § 8 are for Problem 1.

7. Sequential Behavior and Performance. In this section, we first consider the issue of step size selection for both the explicit and implicit time stepping schemes. Next, we mention several possible techniques for starting the multigrid iteration at a particular time step of an implicit scheme, and explain the approach we took. We then examine the sequential behavior and performance of the three time stepping methods we implemented, when each method uses nearly optimal step sizes. Finally, we make some comments about the effect of the time step on multigrid convergence rates.

7.1. Step Size Selection. In our experiments, we first select the spatial meshwidth Δx , and then determine a time step Δt that is as large as possible while maintaining a degree of accuracy at the final time point commensurate with the spatial discretization error.

An important issue regarding the time step selection arises when using the multigrid method for the implicit equations. It was noted in § 4.1 that the size of the time step chosen effects the conditioning of the elliptic equation (7). As Δt approaches zero, the operator becomes more well-conditioned, and this is reflected in a smaller convergence factor for multigrid; this is shown in Figure 4. Naturally, we are interested in taking large time steps and it is clear that the multigrid convergence rates for the shifted elliptic operator in equation (7) is never larger than for the unshifted operator L^1 .

When solving an application problem rather than a test problem, the luxury of determining the optimal step size, either analytically or empirically, does not exist, and therefore either a fixed time step must be chosen a priori, or some type of adaptive time stepping strategy must be employed. If one has a spatial error estimate, then one could attempt to maintain maximal accuracy while taking the fewest possible time steps by exploiting the spatial error information. The structure of multigrid

¹ It is interesting to note that the ill-posedness of time-reversed parabolic equations is reflected in poor convergence behavior of multigrid when Δt is chosen to be negative. Standard multigrid procedures fail to converge if the coarse grid problem is too small [14, 23].

methods makes it easy to estimate the accuracy of the computed solution. We outline two simple possibilities.

In the CS multigrid scheme, the solution on the $2h$ grid, when transferred to the finest h grid, can be viewed as an approximation to the error on the finest grid. In the notation of Figure 2, $\|c_h\|/\|\bar{u}_h\|$ could be used as an error estimator in the spirit of iterative improvement (e.g., [15]).

Alternatively, if the FMG and/or FAS multigrid approach is used, we have approximate solutions on each grid. The solutions on the h and $2h$ grids can be combined to produce an error estimate, using either Richardson extrapolation or τ extrapolation [7].

Ideally, the spatial error estimate thus obtained would be combined with an error estimate in time, to implement adaptive time stepping. Eventually, it would be desirable to implement methods which are adaptive in both space and time for 3D parabolic problems. This question is addressed for nonlinear parabolic problems in one space dimension in [26].

7.2. Initial Guess for Multigrid. The Multigrid V-cycle solution of the discrete equations at time t_{n+1} can in principle be initialized in one of several ways, including:

- the solution u^n from the previous time step,
- an initial approximation from FMG,
- use of a predictor.

Since explicit methods used as predictors typically violate their stability limits, they may excite high frequency modes in the predicted value, and therefore may create more work to be done on the fine grid of the next time step [8]; for this reason, the predictor approach may not be worthwhile.

Brandt et al. [8] argue that while the previous time step or some extrapolated approximation other than a predictor can be used, the FMG approach coupled with a variation of the V-Cycle, the *F-Cycle*, is optimal. For the particular problems we have looked at, and for our implementation, our experiments seem to indicate that the previous time step is a marginally better initialization to the V-cycle than is the FMG approach. Whether this would remain true for other problems or multigrid method formulations (e.g., if cubic interpolation is used in FMG; cf., Figure 3) remains open.

7.3. Comparisons of Explicit and Implicit Methods. We integrated the test problem, using our empirically determined nearly optimal time step, with each of the three methods to time $t = 1$, noting the cost of each method on a sequential computer. Note that the implicit equations were solved with the multigrid method at each time step.

Table 1 presents some results, in which we give the number of time steps required to reach a given tolerance, the cost of each step, and the total cost of all time steps, for the solution of a variable coefficient problem (Problem 2) on a $15 \times 15 \times 15$ mesh. Table 2 presents the results of the same experiments for a $31 \times 31 \times 31$ mesh.

The number of steps required by each of the methods to reach the final time point is dictated by the stability limit in the Forward Euler method, while accuracy considerations constrain the Backward Euler method to taking more time steps than Crank-Nicolson. However, the cost for each time step is more expensive for the implicit methods. The question then arises as to whether the explicit method is so inexpensive to apply that it can overcome the penalty imposed by the stability limit.

The results presented in Tables 1 and 2 indicate that this is not the case on sequential computers, at least when an efficient solver (multigrid in this case) is used for the implicit equations. On parallel computers this situation could change, if the explicit method ran at a much higher parallel efficiency than the implicit method. However, noting that the ratio of the total sequential times given in Table 1 for Forward Euler and Crank-Nicolson is roughly five, this would imply that the explicit method would have to achieve a parallel efficiency five times greater than the implicit method to be competitive. In this scenario, if the implicit method achieves over twenty percent parallel efficiency, then the explicit method is not likely to be competitive. As we will see in § 8.1 and § 8.2, we are able to achieve substantially more than twenty percent parallel efficiency, from FORTRAN, for our implementation of the Crank-Nicolson method with multigrid.

From Table 2 we see that the advantage of the implicit method increases as we go to larger problems. For the types of problems we are considering, we do not expect explicit methods to be

TABLE 1
Solution of Problem 2 on a $15 \times 15 \times 15$ mesh. CPU seconds on a Convex C240 to reach $t_{final} = 1$.

Method	Number of Steps N	Time for 1 step	Time for N steps	Accuracy
Forward Euler	1010	$1.87E - 02$	18.9	$1.6E - 03$
Backward Euler	105	$1.97E - 01$	20.7	$2.0E - 03$
Crank-Nicolson	20	$2.01E - 01$	4.0	$2.6E - 03$

TABLE 2
Solution of Problem 2 on a $31 \times 31 \times 31$ mesh. CPU seconds on a Convex C240 to reach $t_{final} = 1$.

Method	Number of Steps N	Time for 1 step	Time for N steps	Accuracy
Forward Euler	4040	$1.51E - 01$	610.0	$3.9E - 04$
Backward Euler	350	$1.51E + 00$	528.5	$5.0E - 04$
Crank-Nicolson	30	$1.55E + 00$	46.5	$6.2E - 04$

competitive with implicit methods, for sequential or coarse-grained parallel computers.

8. Performance on Advanced Computer Architectures. We have implemented a 3D multigrid solver for a class of problems that includes the elliptic problems arising at each time step of an implicit method. This solver has been interfaced with a parabolic equation integrator which implements both Backward Euler and Crank-Nicolson time stepping. The multigrid solver can use either the V-Cycle algorithm (with either zero or the previous time level solution as the initial guess) or the FMG algorithm, with weighted Jacobi smoothing, full-weighting restrictions, tri-linear interpolation, and banded Gaussian elimination for the coarse grid solves.

The FORTRAN code executes with high vector and parallel efficiency on machines such as the Alliant FX/2800, the Cray Y-MP, and also executes efficiently on the IBM RS6000, the Convex C240, and the Sun SPARC-1. The multigrid code consists of four central routines: the smoothing routine, the residual routine, and the restriction and interpolation routines. The restriction and interpolation routines consist of triply nested loops which scan through the grid functions by plane, line, and point; appropriate compiler directives must be placed in the code in order for parallelization across the planes to take place. The smoothing and residual routines, on the other hand, consist of long single loops as a result of the diagonal matrix storage scheme. These loops may need to be restructured in order for parallelization to be enabled on different architectures.

The performance figures presented below, as well as the summary in Table 3 refer to identical code, other than the insertion of necessary compiler directives mentioned above to force parallelization across planes in the restriction and interpolation routines (this was automated for a number of compilers), and hand restructuring of long single loops in the smoothing and residual routines to enable parallelization in the case of the Cray Y-MP.

In what follows, we examine the parallel and vector performance on the Cray and the Alliant in detail. Table 3 summarizes these results along with the overall performance of the code on several other machines. For the results that follow in § 8.1 and § 8.2, we use the notions of parallel speedup S_P , parallel efficiency E_P , and Megaflops as performance indicators. The standard definitions for these measures are (for a program running on P processors of a parallel computer):

$$S_P \equiv \frac{\text{Execution time on 1 processor}}{\text{Execution time on } P \text{ processors}}, \quad E_P \equiv S_P \times \frac{1}{P} \times 100\%.$$

8.1. Parallel Performance on the Alliant FX/2800. The Alliant FX/2800 used for our experiments is located at the Center for Supercomputing Research and Development at the University of Illinois, Urbana-Champaign. The architecture is shared memory multiprocessor, with fourteen Intel i860 processors. There is compiler support for parallel programming in FORTRAN. In particular, the sequential code was run on this machine with no alterations, other than the insertion of a few compiler directives.

TABLE 3
Performance summary of the 3D parabolic multigrid solver on several architectures.

Machine	Megaflops
Cray Y-MP (4 processors)	641.0
Cray Y-MP (1 processor)	224.0
Alliant FX/2800 (14 processors)	51.1
Alliant FX/2800 (1 processor)	6.4
Convex C240	16.1
IBM RS6000	12.7
Sun SPARC 1	0.6

TABLE 4
Percentage of solution time spent in main multigrid components when solving Problem 2 on a $31 \times 31 \times 31$ mesh.

Multigrid Component	Alliant FX/2800		Cray Y-MP	
	P=1	P=14	P=1	P=4
Smoothing	76%	75%	75%	79%
Residual	14%	14%	12%	10%
Interpolation	4%	5%	8%	6%
Restriction	2%	3%	3%	3%
Coarse Grid Solve	< 1%	< 1%	< 1%	< 1%

Figures 5 through 8 display the execution times, speedup, parallel efficiency for the overall solution process, and the megaflop rate breakdown for each major component in the multigrid method at one particular time step.

Figure 5 shows the reduction in execution time as more processors are used. Figure 6 shows a speedup of nearly eight on fourteen processors, which when translated to parallel efficiency in Figure 7 shows nearly sixty percent efficiency. Figure 8 shows an overall performance of over fifty megaflops on fourteen processors, and also reveals a routine that lags in performance: the interpolation routine. (Note that interpolation is only about three to eight percent of the overall solution time, and therefore does not constitute a substantial bottleneck; see Table 4.)

From these figures we can make a few observations. First, the code runs at about six megaflops on a single processor. This is somewhat disappointing, given the high peak performance of the i860 chips (thirty to forty megaflops for SAXPYs [11, 21]). Secondly, the leveling off in the parallel efficiency curve is at first glance somewhat unexpected. However, for a method like multigrid where $O(N)$ operations are performed on data of size N , it is harder for the cache management algorithm to do a good job, than, say, for Gaussian elimination, where $O(N^3)$ operations are performed on $O(N^2)$ data. This in turn leads to greater contention for the bus, as the number of processors is increased.

8.2. Parallel and Vector Performance on the Cray Y-MP. Our implementation was also run on the Cray Y-MP, located at the National Center for Supercomputer Applications at the University of Illinois, Urbana-Champaign. The architecture is also a shared memory multiprocessor, with four Cray vector processors. We used the micro-tasking facility (do-loop level) rather than macro-tasking (subroutine level) to parallelize our code. There is compiler support for some parallel programming, which is used through a compiler front-end which inserts micro-tasking (“autotasking”) compiler directives. However, we were forced to hand nest the long single loops of the smoothing and residual routines in order for the autotasking directives to be correctly inserted by the compiler front-end; this was not necessary for the Alliant compiler.

Shown in Figures 9 through 12 are the Cray performance results for the same problem we considered in the previous section: execution times, speedup, parallel efficiency, and megaflop rate breakdown. Figure 9 shows the usual execution time reduction as more processors are used. Figure 10 shows a speedup of nearly three on four processors, which translates to a parallel efficiency of over seventy percent, shown in Figure 11. Figure 12 shows the overall performance of over two hundred

and twenty megaflops on one processor, and over six hundred and forty megaflops on four processors. (Again, the lagging interpolation routine is revealed.)

From these figures we can see that the choice of diagonal matrix data structures paid a large dividend in performance: the smoothing and residual routines are the fastest components of the multigrid method on the Cray. Secondly, on one processor of the Cray Y-MP, we achieve over two hundred and twenty megaflops, which is nearly the peak performance of the older Cray X-MP. Finally, if an eight processor version of the Cray Y-MP had been available, a performance of over a gigaflop seems plausible, given the slope and curvature of the speedup plot on four processors in Figure 10.

9. Conclusions.

We conclude by making a few observations. First, it is clear that if the implicit equations are solved using the multigrid method, not only is the resulting method sequentially very efficient, but it can also achieve high vector and parallel efficiencies. This is of course not new [3, 13], but what can be said is that this performance is achievable from FORTRAN with very little special coding considerations.

Second, the shifted elliptic operators arising at each time step of an implicit time discretization results in rapid multigrid convergence. Furthermore, multigrid can exploit information available from previous time steps by using the previous solution as an initial guess for the multigrid V-Cycle.

Third, explicit methods may achieve somewhat higher computation rates, but not enough to overcome their inherent disadvantages, at least on coarse grained parallel architectures such as the ones considered in this paper. In particular, for the methods we implemented, the implicit methods were superior even for small problems, and as the problem size grows, the superiority of the implicit methods will only increase.

Finally, although we did not implement this idea, we feel that adaptive step-size selection will be straightforward to implement when using multigrid for the implicit equations, and lead to even greater efficiency.

Acknowledgments. Dr. Ahmed Sameh, at the Center for Supercomputing Research and Development at the University of Illinois at Urbana-Champaign, was kind enough to allow us to use the Alliant FX/2800 for some of our numerical experiments. Also, Stan Kerr at the University of Illinois' Computing Services Organization arranged for us to have access to the Cray Y-MP at the National Center for Supercomputer Applications, also located at the University of Illinois.

REFERENCES

- [1] R. E. BANK AND C. C. DOUGLAS, *Sharp estimates for multigrid rates of convergence with general smoothing and acceleration*, SIAM J. Numer. Anal., 22 (1985), pp. 617–633.
- [2] R. E. BANK AND T. F. DUPONT, *An optimal order process for solving finite element equations*, Math. Comp., 36 (1981), pp. 35–51.
- [3] E. BARSZCZ, T. F. CHAN, D. C. JESPERSON, AND R. S. TUMINARO, *Performance of a parallel code for the Euler equations on hypercube computers*, Tech. Report NASA 102260, NASA, 1990.
- [4] D. BRAESS AND W. HACKBUSCH, *A new convergence proof for the multigrid method including the V-cycle*, SIAM J. Numer. Anal., 20 (1983), pp. 967–975.
- [5] A. BRANDT, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31 (1977), pp. 333–390.
- [6] ———, *Guide to multigrid development*, in Multigrid Methods: Proceedings of Köln-Porz Conference on Multigrid Methods, Lecture notes in Mathematics 960, W. Hackbusch and U. Trottenberg, eds., Berlin, Germany, 1982, Springer-Verlag.
- [7] ———, *Multigrid techniques: 1984 guide with applications to fluid dynamics*, Tech. Report GMD-Studien Nr. 85, GMD, Bonn, 1984.
- [8] A. BRANDT AND J. GREENWALD, *Parabolic multigrid revisited*, in Third European Conference on Multigrid Methods, Bonn, Germany, 1990, Springer-Verlag.
- [9] J. E. DENDY, JR., *Two multigrid methods for three-dimensional problems with discontinuous and anisotropic coefficients*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 673–685.
- [10] J. J. DONGARRA, J. R. BUNCH, C. B. MOLER, AND G. W. STEWART, *LINPACK Users' Guide*, SIAM, Philadelphia, PA, 1979.
- [11] T. H. DUNIGAN, *Performance of the Intel iPSC/860 hypercube*, Tech. Report ORNL/TM-11491, Oak Ridge National Laboratory, 1990.
- [12] D. J. EVANS, *Fast ADI methods for the solution of linear parabolic partial differential equations involving two space dimensions*, BIT, 17 (1977), pp. 486–491.
- [13] P. O. FREDERICKSON AND O. A. MCBRYAN, *Parallel superconvergent multigrid*, in Multigrid Methods: Theory, Applications, and Supercomputing, S. McCormick, ed., New York, NY, 1988, Marcel Dekker, Inc., pp. 195–210.

- [14] C. I. GOLDSTEIN, *Multigrid preconditioners applied to the iterative solution of singularly perturbed elliptic boundary value problems and scattering problems*, Tech. Report BNL-37225, Brookhaven National Laboratory, 1990.
- [15] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, MD, second ed., 1989.
- [16] W. HACKBUSCH, *On the convergence of multi-grid iterations*, Beiträge Numer. Math., 9 (1981), pp. 213–239.
- [17] ———, *Parabolic multi-grid methods*, in Computing Methods in Applied Sciences and Engineering, VI, New York, NY, 1984, North Holland.
- [18] ———, *Multi-grid Methods and Applications*, Springer-Verlag, Berlin, Germany, 1985.
- [19] W. HACKBUSCH AND U. TROTTENBERG, eds., Berlin, Germany, 1982, Springer-Verlag.
- [20] ———, eds., *Multigrid Methods II: Proceedings of the Second European Conference on Multigrid Methods held at Cologne*, Berlin, Germany, 1986, Springer-Verlag.
- [21] M. T. HEATH, G. A. GEIST, AND J. B. DRAKE, *Early experience with the Intel iPSC/860 at Oak Ridge National Laboratory*, Tech. Report ORNL/TM-11655, Oak Ridge National Laboratory, 1990.
- [22] M. HOLST AND F. SAIED, *Vector multigrid: An accuracy and performance study*, Tech. Report UIUCDCS-R-90-1636, Numerical Computing Group, Department of Computer Science, University of Illinois at Urbana-Champaign, 1990.
- [23] ———, *Multigrid methods for computational ocean acoustics on vector and parallel computers*, in Proceedings of the Third IMACS Symposium on Computational Acoustics, New York, NY, North Holland, 1991.
- [24] W. H. HOLTER, *A vectorized multigrid solver for the three-dimensional Poisson equation*, Appl. Math. Comp., 19 (1986), pp. 127–144.
- [25] T. L. JORDAN, *A guide to parallel computation and some Cray-1 experiences*, in Parallel Computations, G. Rodrigue, ed., Academic Press, New York, NY, 1982, pp. 1–50.
- [26] J. LAWSON, M. BERZINS, AND P. M. DEW, *Balancing space and time errors in the method of lines for parabolic problems*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 573–594.
- [27] J. G. LEWIS AND H. D. SIMON, *The impact of hardware gather/scatter on sparse gaussian elimination*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 304–311.
- [28] C. LUBICH AND A. OSTERMANN, *Multi-grid dynamic iteration for parabolic equations*, BIT, 27 (1987), pp. 217–234.
- [29] R. D. RICHTMYER AND K. W. MORTON, *Difference Methods for Initial-Value Problems*, Interscience Publishers, New York, NY, second ed., 1957.
- [30] K. STÜBEN AND U. TROTTENBERG, *Multigrid methods: Fundamental algorithms, model problem analysis and applications*, in Multigrid Methods: Proceedings of Köln-Porz Conference on Multigrid Methods, Lecture notes in Mathematics 960, W. Hackbusch and U. Trottenberg, eds., Berlin, Germany, 1982, Springer-Verlag.
- [31] C.-A. THOLE AND U. TROTTENBERG, *Basic smoothing procedures for the multigrid treatment of elliptic 3d-operators*, Tech. Report GMD-Studien Nr. 141, GMD, Bonn, 1985.
- [32] S. VANDEWALLE AND R. PIESENS, *Efficient parallel algorithms for solving initial-boundary value and time-periodic parabolic partial differential equations*, Tech. Report TW 139, Department of Computer Science, K.U.Leuven, Heverlee, Belgium, 1990.
- [33] ———, *Numerical experiments with multigrid waveform relaxation on a parallel processor*, Tech. Report TW 142, Department of Computer Science, K.U.Leuven, Heverlee, Belgium, 1990.
- [34] S. VANDEWALLE, R. VAN DRIESSCHE, AND R. PIESENS, *The parallel implementation of standard parabolic marching schemes*, Tech. Report TW 125, Department of Computer Science, K.U.Leuven, Heverlee, Belgium, 1989.
- [35] R. S. VARGA, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1962.

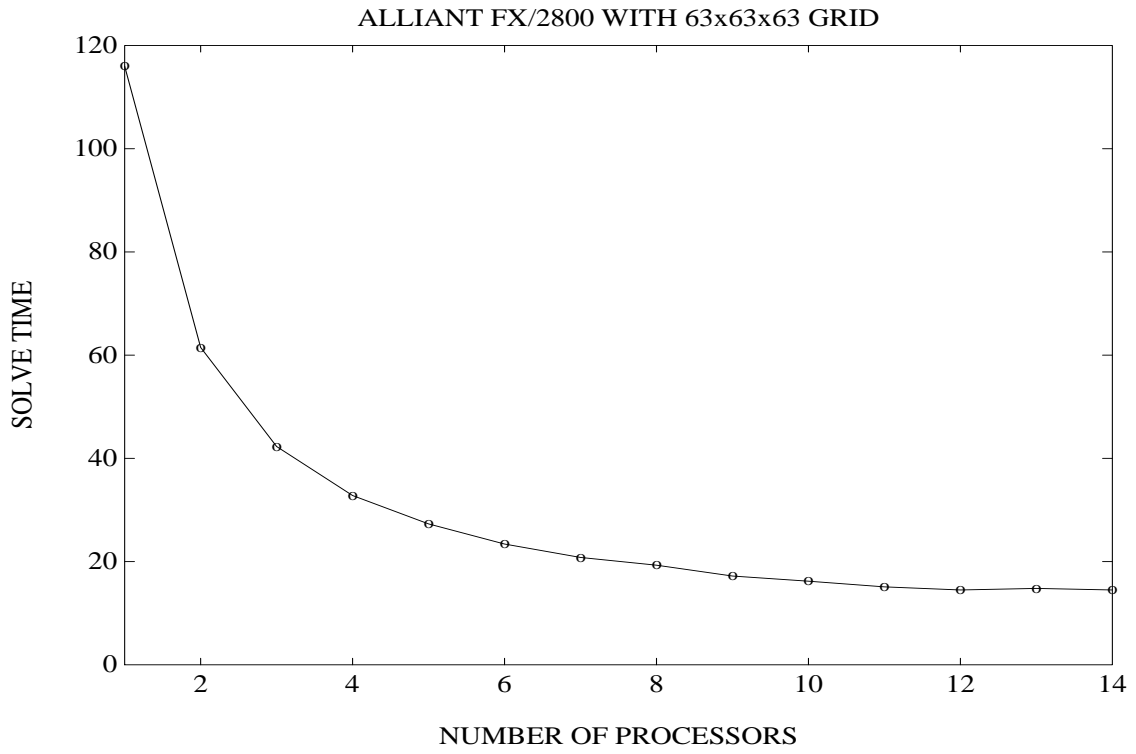


FIG. 5. Execution times over 14 processors on the Alliant FX/2800.

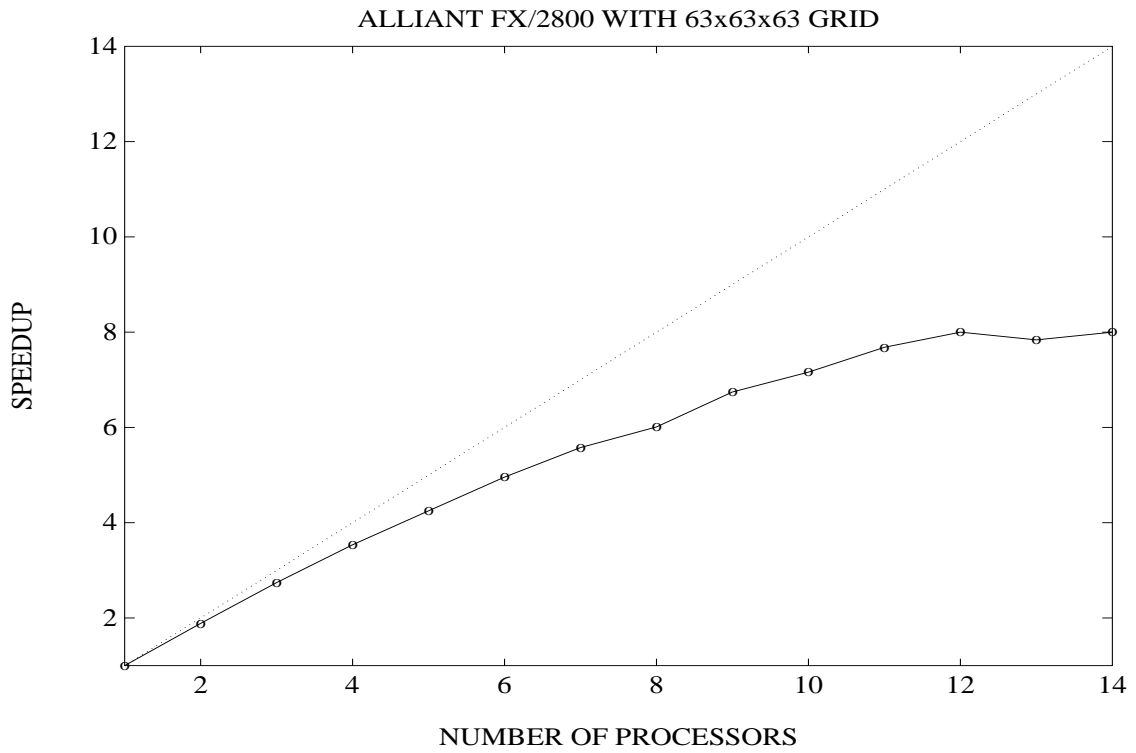


FIG. 6. Speedup over 14 processors on the Alliant FX/2800.

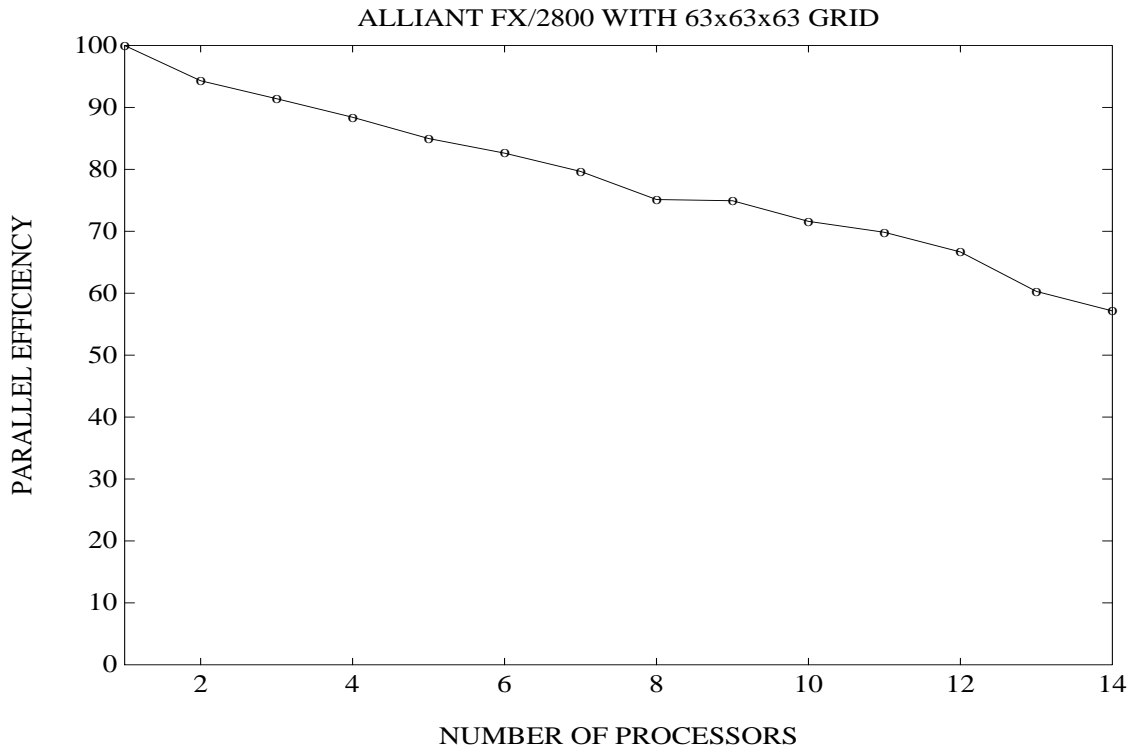


FIG. 7. Parallel efficiency over 14 processors on the Alliant FX/2800.

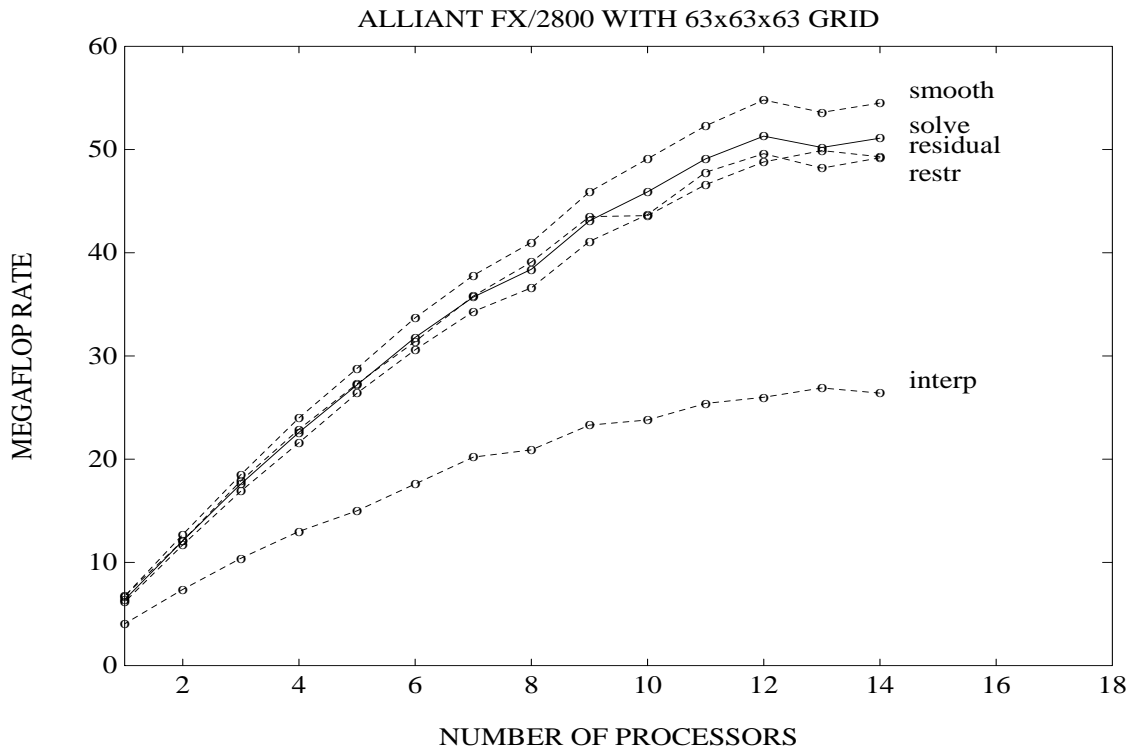


FIG. 8. Megaflop rates for each multigrid component on the Alliant FX/2800.

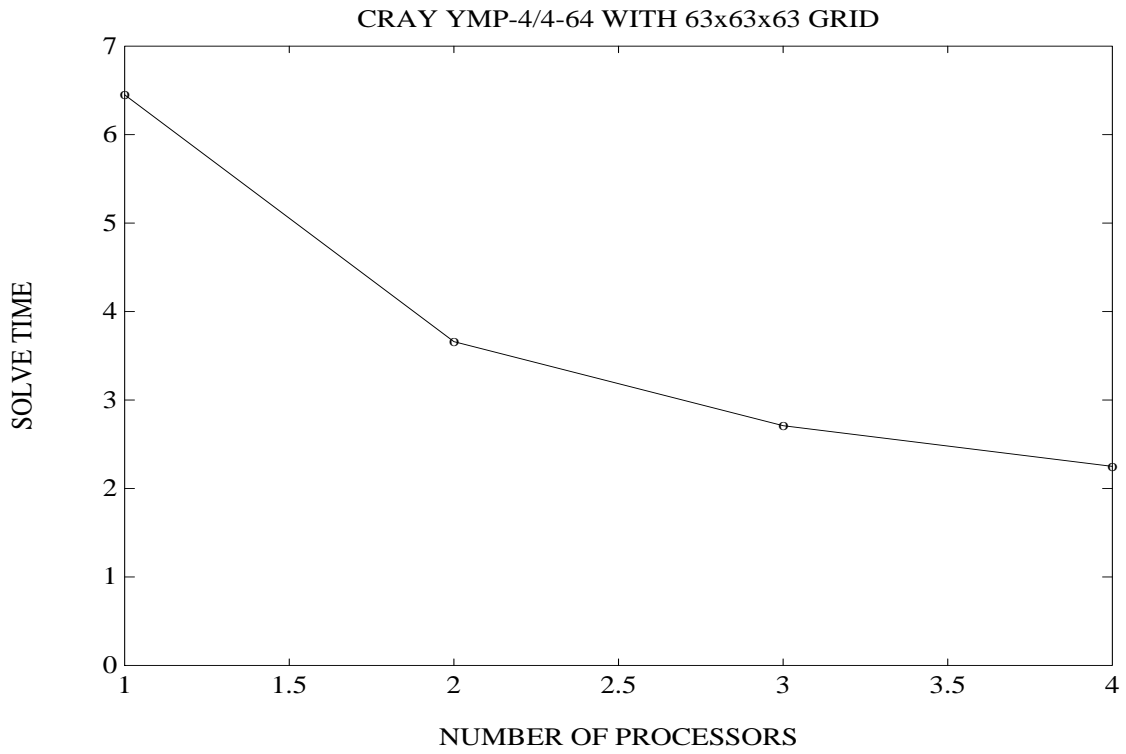


FIG. 9. Execution times over 4 processors on the Cray Y-MP.

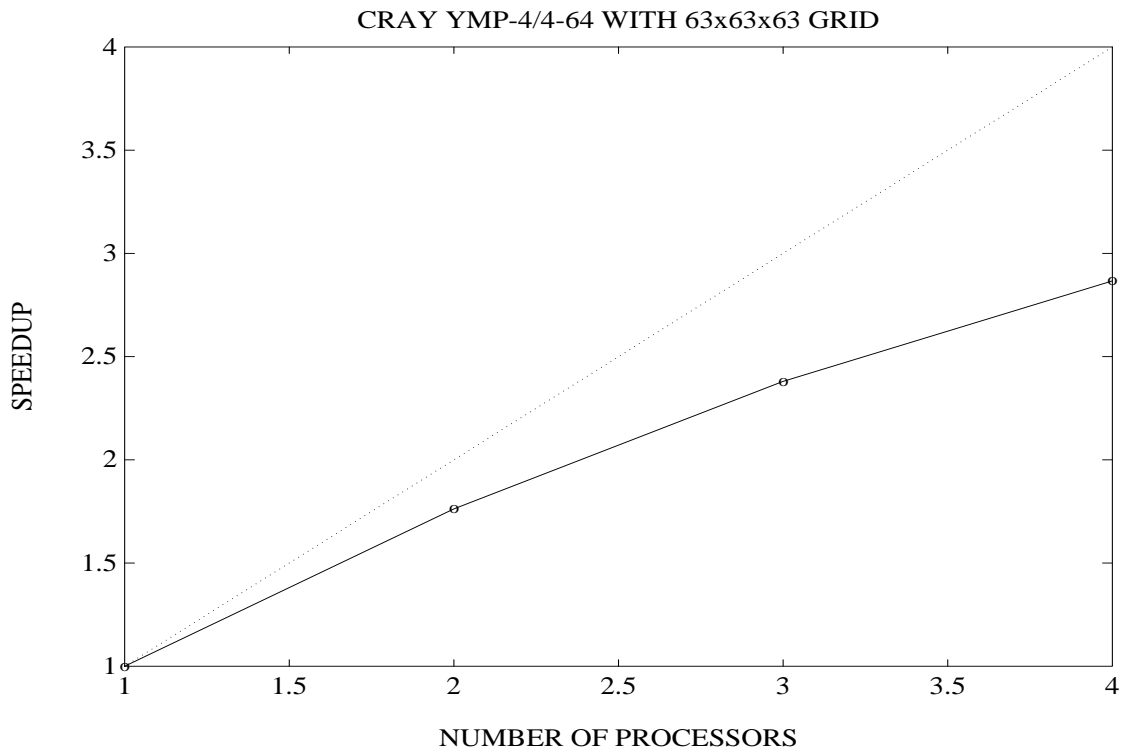


FIG. 10. Speedup over 4 processors on the Cray Y-MP.

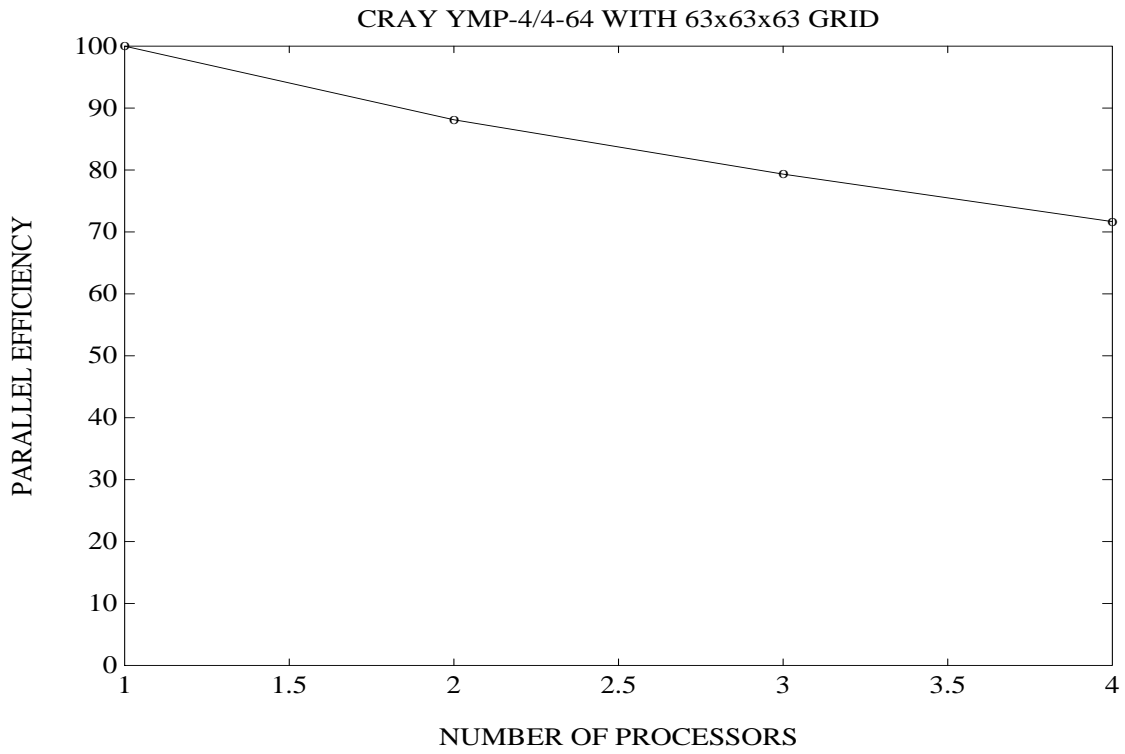


FIG. 11. Parallel efficiency over 4 processors on the Cray Y-MP.

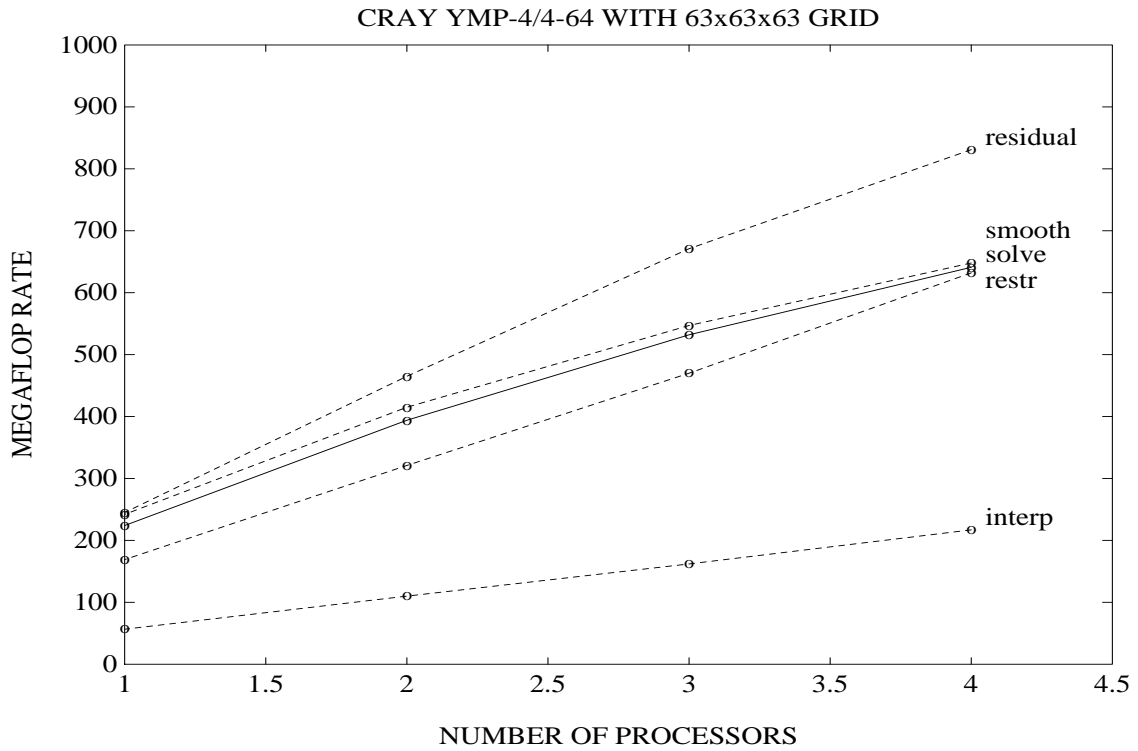


FIG. 12. Megaflop rates for each multigrid component on the Cray Y-MP.