Multigrid Methods for Computational Acoustics on Vector and Parallel Computers

Faisal Saied
Michael J. Holst

Department of Computer Science
University of Illinois at Urbana-Champaign
1304 W Springfield Ave
Urbana, IL 61801

Abstract

We consider the parabolic approximation to the three–dimensional Helmholtz equation for the acoustic pressure. The parabolic equation is semi–discretized in the range variable using an implicit scheme (e.g., Crank–Nicolson). This leads to a complex elliptic partial differential equation that must be solved at each range step. We use a multigrid method to solve this partial differential equation, which gives rise to matrices that are complex, symmetric (but non-Hermitean). In this sense, multigrid is an alternative to other approaches such as Krylov subspace methods and ADI.

We present results for the Alliant FX2800 (a shared memory machine based on i860 processors), and the Cray Y-MP. Our results demonstrate that multigrid maps effectively to these supercomputer architectures, and that high performance can be achieved through the parallelizing compilers with relatively little user intervention.

1 Introduction

In this paper, we discuss how multigrid methods can be applied to problems arising in Computational Acoustics, and how these methods can be implemented efficiently on high performance computers.

The problem we consider is the three-dimensional parabolic wave equation in underwater acoustics. This partial differential equation arises when the parabolic approximation is applied to the Helmholtz equation for the acoustic pressure:

$$\nabla^2 p + k^2 p = 0$$

In cyclinderical polar coordinates, the 3D parabolic equation has the form:

$$\frac{\partial u}{\partial r} = \frac{i}{2k_0}\frac{\partial^2 u}{\partial z^2} + \frac{i}{2k_0 r^2}\frac{\partial^2 u}{\partial \theta^2} + \frac{ik_0}{2}(n^2-1)u = i\mathcal{L}u. \tag{1}$$

Here k_0 is a reference wave number and $n = n(r, \theta, z)$ is the index of refraction.

With appropriate initial and boundary conditions, this equation can be solved numerically in a range stepping fashion. This equation is the 3D version of the narrow angle equation of Tappert [9]. Techniques similar to the ones reported in this paper can be applied to the Lee-Saad-Schultz model [6].

In the remainder of this paper, we will outline the discretization methods used, in particular for the range variable, and describe the elliptic problem that needs to be solved at each range step. We then use multigrid to solve this problem. A brief overview of the multigrid philosophy is included. Finally, performance results are presented for two parallel computers: a 14 processor Alliant FX-2800 and an 8 processor Cray Y-MP. These results indicate that multigrid can be parallelized effectively on these architectures, almost automatically.

Multigrid methods are discussed in [1], and parallel implementations in [2]. Multigrid methods for vector and parallel computers in the context of underwater acoustics have been discussed in [4], [3]. Parallel multigrid methods for parabolic equations of diffusion type in three spatial dimensions were implemented in [5]. The opportunities for exploiting parallel computers for computational acoustics are outlines in [7], [8].

2 Implicit Finite Difference Methods for the Parabolic Equation

We consider an implicit finite difference discretization of the 3D parabolic wave equation,

$$u_r = i\mathcal{L}u$$

Our discussion centers on the Crank-Nicolson discretization, but the results are applicable to other implicit range stepping methods, such as Backward Euler. Crank-Nicolson for the 3D parabolic equation (1) can be written as

$$\left(I - i\frac{\Delta r}{2}\mathcal{L}\right) u^{n+1} = \left(I + i\frac{\Delta r}{2}\mathcal{L}\right) u^{n} \tag{2}$$

The most significant part of the work at each range step is the solution of an elliptic partial differential equation of the form

$$\mathcal{L}u + i\frac{2}{\Delta r}u = f. \tag{3}$$

We will focus on the efficient solution of this problem. From the form of (3) we can see that if we discretize the depth and azimuthal variables using the standard centered 5-point stencil, the resulting matrix is complex and symmetric (but not Hermitean!). In particular, if A_0 is a real symmetric matrix, then

$$A = A_0 + i\alpha I$$

is a complex symmetric matrix[1].

These matrices arise in a number of applications. A number of linear algebra techniques can be adapted to take advantage of this structure, such as preconditioned Krylov subspace methods, or Alternating Direction Implicit (ADI) methods.

We propose solving the elliptic problems that arise at each range step by multigrid methods, which we will review below.

[1]A somewhat more general form is $A = A_0 + iD$, where D is a diagonal matrix.

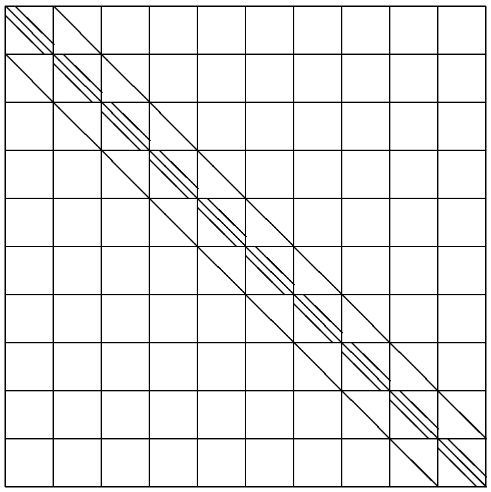

Figure 1: Non-zero structure of the matrix

We include a comment about data structures that turned out to be important from the point of view of performance. With the 5-point stencil and the "natural ordering" of the unknowns, it is well known that the discrete operator has the non-zero structure depicted in Figure 1. If the non-zero entries of the matrix are stored by diagonals, vectorizing compilers can generate extremely efficient code for operations like a matrix vector product, which are used in in multigrid methods (in the residual computation and the weighted Jacobi smoother, discussed in the next section).

3 Multigrid methods

Multigrid methods for partial differential equations use multiple grids for resolving various features of the solution on the appropriate spatial scales. They derive their efficiency by not attempting to resolve coarse scale features on the finest grid.

The basic idea of multigrid is depicted in Figure 2, for the two-grid version.

Starting with and initial guess, u_h^{old}, on the finest grid, we apply ν_1 iterations of a smoothing method, such as Jacobi or Gauss-Seidel and form the residual r_h of the resulting grid vector $\overline{u}_h$. This is "restricted" down to the coarse grid, where it is used as the right hand side (r_{2h}) of the coarse grid correction equation, $L_{2h}c = r_{2h}$. The solution to this problem (c_h) is interpolated back to the fine grid where it is added to the current approximation. Finally an additional ν_2 sweeps of the smoother are applied to the corrected approximation, to obtain u_h^{new}.

The bulk of the work is in the smooting iterations and the residual computation, followed by the grid transfer operations (restriction and interpolation).

The smoother we have used is weighted (or underrelaxed) Jacobi, which, for $Ax = b$ and $A = D - L - U$, is defined by

$$x^{(n+1)} = [(1-\omega)I + \omega D^{-1}(L+U)]x^{(n)} + \omega D^{-1}f$$

The grid transfers were Full Weighting for the restriction and cubic interpolation for the coarse to

$$\begin{tikzcd}
u_h^{old} \arrow[r, "R_h^{\nu_1}"] & \overline{u}_h \arrow[r] & r_h = f_h - L_h\overline{u}_h \arrow[d, "I_h^{2h}"'] & & c_h \arrow[r] & \overline{u}_h + c_h \arrow[r, "R_h^{\nu_2}"] & u_h^{new} \\
& & r_{2h} \arrow[r] & \overset{\text{Solve}}{L_{2h}c = r_{2h}} \arrow[r] & c_{2h} \arrow[u, "I_{2h}^h"'] & &
\end{tikzcd}$$

Figure 2: The Two-grid version of Multigrid

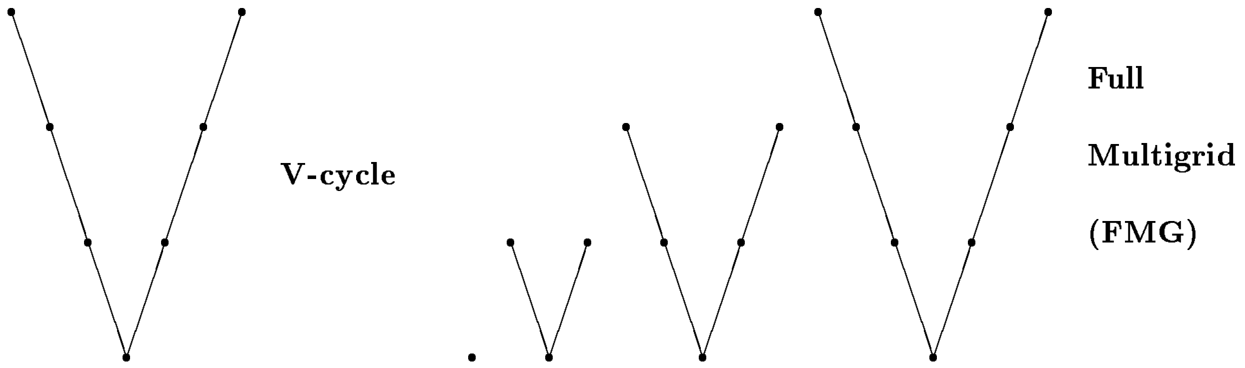

Figure 3: The V-cycle and Full Multigrid (the finest grid is at the top).

fine transfers. The coarse grid solves were done using banded Gaussian elimination.

In practice, the two-grid algorithm is applied recursively. The most common approach is the V-cycle, where an initial guess must be supplied on the finest grid. The Full Multigrid (FMG) method goes one step further and starting from the coarsest grid, "bootstraps" itself up to the finest grid, before doing the V-cycle. In this sense, FMG generates its own (usually very good) initial guess on the finest grid.

4 Test Problem

The elliptic problem to be solved at each range step is

$$\mathcal{L}u - i\frac{2}{\Delta r}u = f,$$

where

$$\mathcal{L}u \equiv \frac{1}{2k_0}\frac{\partial^2 u}{\partial z^2} + \frac{1}{2k_0 r^2}\frac{\partial^2 u}{\partial \theta^2} + \frac{k_0}{2}(n^2 - 1)u.$$

This can be viewed as solving a parabolic equation of the form

$$u_t = i(\alpha u_{xx} + \frac{\beta}{t^2} u_{yy} + \gamma u)$$

where

$$\gamma = \gamma(x, y, t) = \frac{1}{2}(n^2 - 1) - i\frac{2}{\Delta r}.$$

The coefficient γ depends on the sound speed profile and on the range step. (α and β are constants.) We used a synthetheic index of refraction. The performance data presented below is for a single range step.

5 Performance Results for the Alliant FX-2800

The Alliant FX-2800 is a shared memory parallel computer, with Intel's i860 chips on each CPU. We used the 14 processor machine at the Center for Supercomputing Research and Development at the University of Illinois at Urbana-Champaign.

Before we discuss the performance results for the Alliant, we will review some terminology that is common in this context. A Megaflop represents a computational rate of one million floating point operations per second (a Gigaflop is a thousand Megaflops). If a program requires times T_1 on one processor, and T_P on P processors, then (parallel) speedup is defined to be

$$\text{Speedup} = \frac{T_1}{T_P}.$$

This is not the most stringent measure of speedup, but it is the one we have employed.

There is an effective parallelizing compiler available on the Alliant FX-2800. Our experience on this machine was that the parallelizing software did a fairly good job, but that it was worthwhile to insert compiler directives judiciously, to enhance the parallel performance. The FORTRAN compiler generated code that executed relatively slowly on a single processor (6 Megaflops, compared to the $40 - 50$ that the i860 chip is capable of.

In the tests reported here, we were able to use only 12 of the 14 processors. In Figure 4, the execution time is plotted as a function of the number of processors and in Figure 5, the speed-up values corresponding to the data in Figure 4 is plotted as a function of the number of processors. Whereas the speedup for 4 processors is very close to 4, for 12 processors it is about 7. Part of the reason may lie in the bus speed, and the fact that with multigrid, where $O(N)$ operations are performed on $O(N)$ data, the caching algorithm is less likely to do a good job, than, say, for dense Gaussian elimination, where $O(N^3)$ operations are performed on $O(N^2)$ data.

In Figure 6, we give the Megaflop rates for the multigrid solver overall, and for a few major components (smoothing, residual computation, restriction and interpolation). We see that the FX-2800 goes from 6 Megaflops on 1 processor to about 45 Megaflops on 12 processors.

6 Performance Results for the Cray Y-MP/8

The Cray Y-MP/8 is a vector supercomputer with 8 processors. Thus in addition to the very high vector performance of each processor, there is the possibility of multiplying performance by using

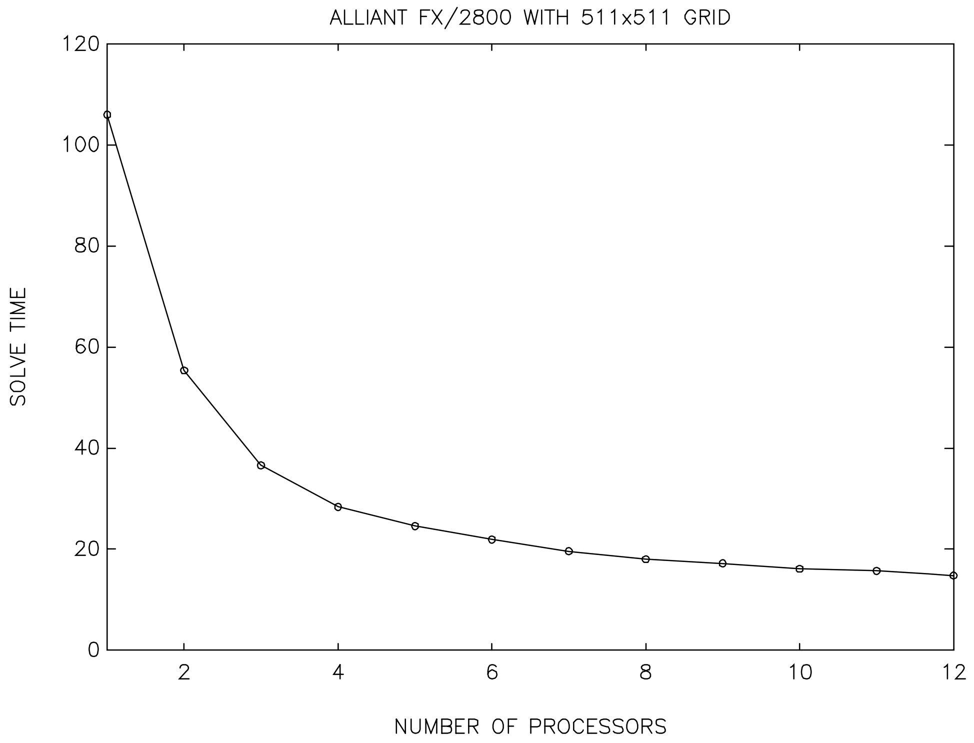

Figure 4: Execution times in seconds for one multigrid solve on a 511×511 grid vs. number of processors.

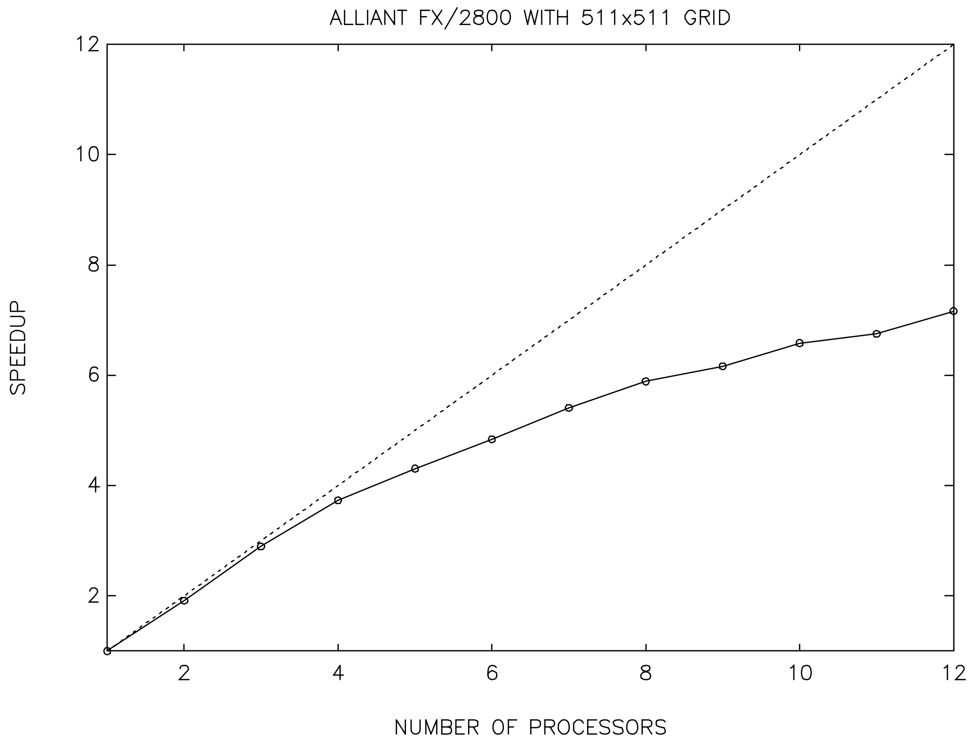

Figure 5: Speedup vs. number of processors, based on the times from Figure 4.

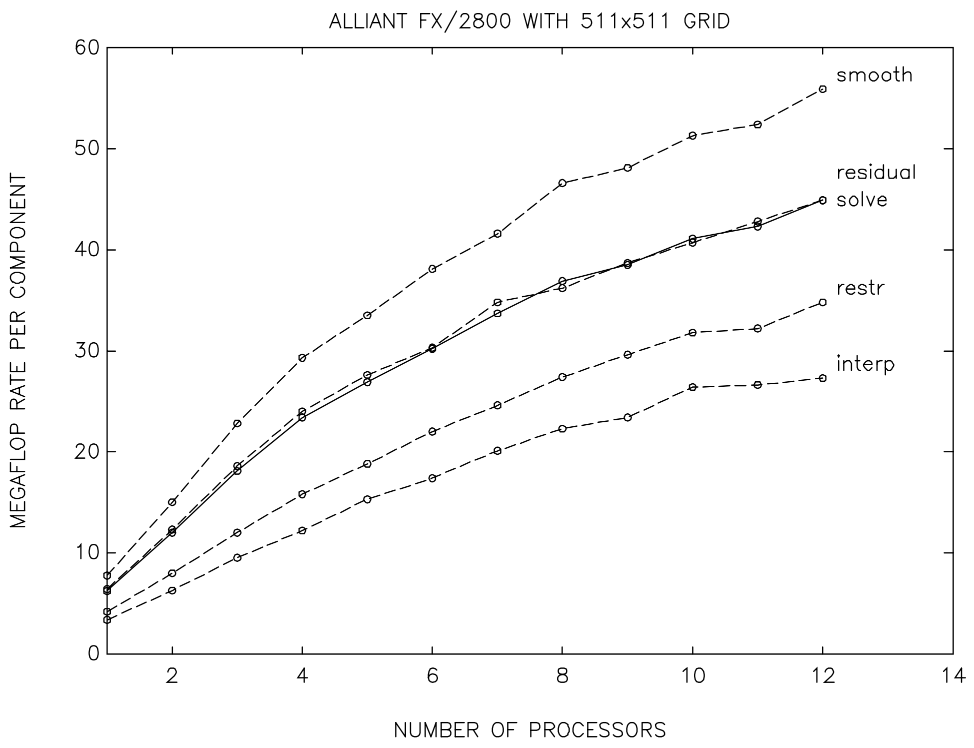

Figure 6: Megaflop rate vs. number of processors based on the times from Figure 3. Separate curves are included for the major components of multigrid.

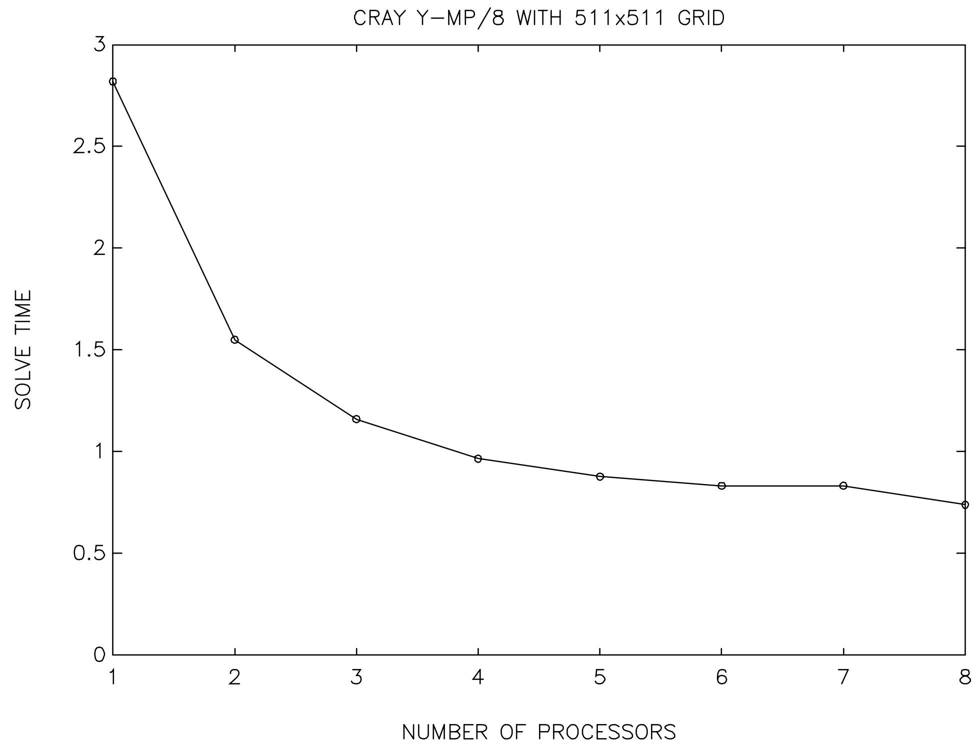

Figure 7: Execution times in seconds for one multigrid solve on a 511×511 grid vs. number of processors. These times were obtained in non-dedicated mode.

multiple processors. This parallelism can be achieved through software supplied by Cray Research. We have used the microtasking mechanism, which parallelizes at the do-loop level. As with the Alliant, it was necessary to add compiler directives to certain key loops to enhance performance. Nevertheless, it is fair to say that the modifications needed to the code were minor.

In Figure 7, the execution time is plotted as a function of the number of processors. In Figure 8, the speed-up values corresponding to the data in Figure 7 is plotted as a function of the number of processors. These runs were done in non-dedicated mode, which means that our program did not necessarily have the stated number of processors for any given portion of its execution. For these runs, the speedup for a small number of processors is reasonable (close to 3 on 4 processors), but is only 3.8 on 8 processors.

In Figure 9, we give the Megaflop rates for the multigrid solver overall, and for a few major components (smoothing, residual computation, restriction and interpolation), This figure shows that the performance on a single processor is very high, and that with 1, 2, 4 and 8 processors, the Megaflops rate is close to 230, 430, 650 and 900, respectively. Individual components of multigrid are running at over 1 Gigaflop, and we feel that in dedicated mode, the overall performance could be in this range.

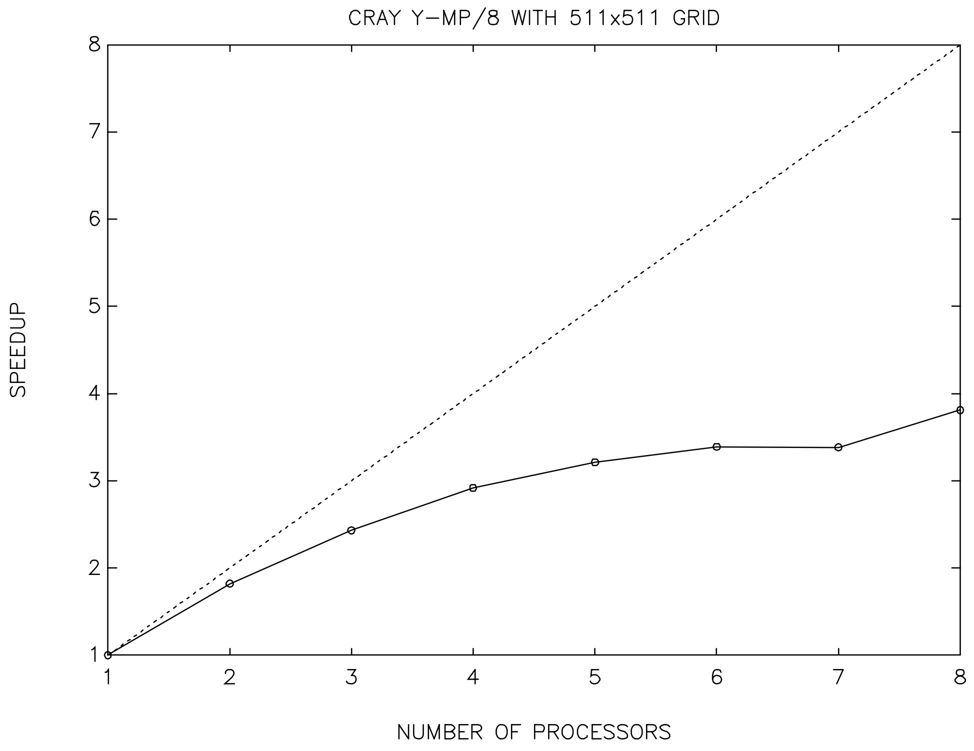

Figure 8: Speedup vs. number of processors, based on the times from Figure 7.

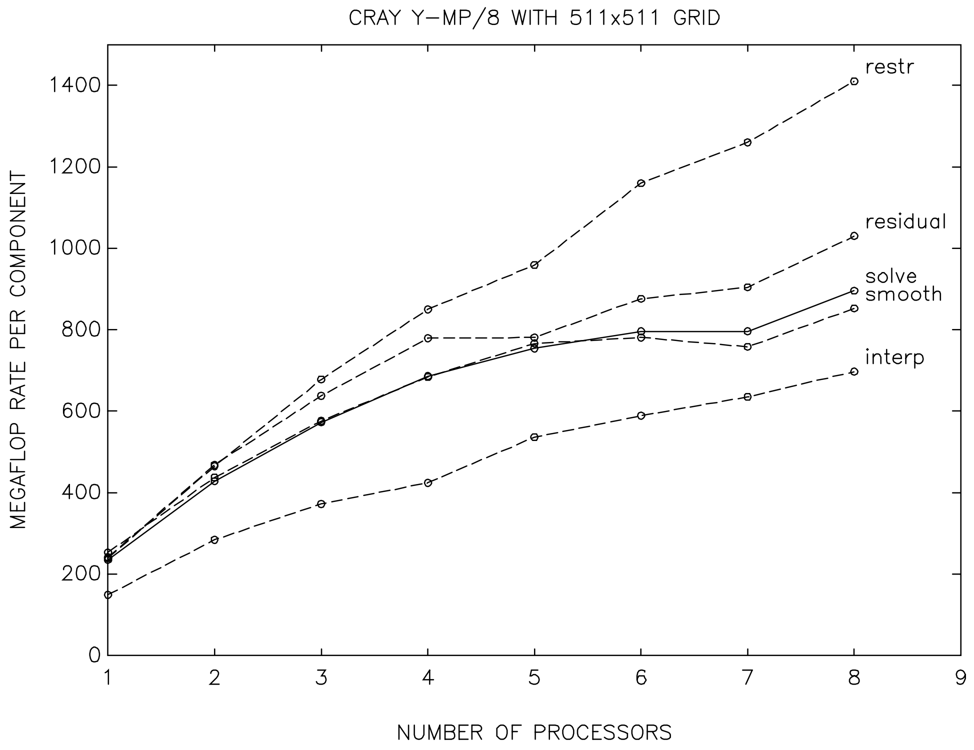

Figure 9: Megaflop rate vs. number of processors based on the times from Figure 6. Separate curves are included for the major components of multigrid.

7 Summary and Conclusions

We have presented results for the parallel performance of multigrid when it is used to solve the complex symmetric problems that arise at each range step, when the Crank-Nicolson scheme is applied to the 3D parabolic equation of underwater acoustics.

By choosing vectorizable and parallelizable components of multigrid, and by choosing an efficient data structure for the discretized operator, we were able to achieve high efficiencies. In particular, we achieved speedups of over 7 on 12 processors of the FX-2800 and 3.8 on 8 processors of a Cray Y-MP (in non-dedicated mode). The Cray performance was very close to 900 Megaflops.

Our results indicate that multigrid has excellent potential as a high performance solver for the parabolic wave equation of underwater acoustics.

Acknowledgements We would like to thank Prof. Ahmed Sameh for providing us access to the 14 processor Alliant FX–2800 at the Center for Supercomputing Research and Development, at the University of Illinois at Urbana-Champaign, and to Cray Research Inc. for providing us with access to an eight processor Cray Y-MP.

The work of the second author was supported in part by National Science Foundation Grant No. DMS 89–11410 and by the Research Board of the University of Illinois, Grant No. RES BRD IC SAIED F.

References

[1] A. Brandt. Adaptive multilevel solution to boundary value problems. *Math. Comp.*, Vol. 31, pp. 333-391, 1977.

[2] A. Brandt. Multigrid Solvers on Parallel Computers. in *Elliptic Problem Solvers*, M. H. Schultz, Ed., pp. 39–84, Academic Press, New York, 1981.

[3] C. C. Douglas, S. C. Ma, and W. L. Miranker. Generating Parallel Algorithms through Multigrid and Aggregation/Disaggregation Techniques. in *Computational Acoustics: Algorithms and Applications*, D. Lee, R. L. Sternberg, and M. H. Schultz, Eds., pp. 133–148, North Holland, New York, 1988.

[4] C. I. Goldstein. Multigrid Preconditioners applied to three dimensional parabolic equation type models. in *Computational Acoustics: Wave Propagation*, D. Lee, R. L. Sternberg, and M. H. Schultz, Eds., pp. 57–74, North Holland, New York, 1988

[5] M. J. Holst and F. Saied. Parallel Performance of some Multigrid Solvers for Three-Dimensional Parabolic Equations. Dept. of Computer Science, University of Illinois at Urbana-Champaign, Report No. UIUC Report No. UIUCDCS-R-91-1697, 1991.

[6] D. Lee, Y. Saad, and M. H. Schultz. An efficient method for solving the three-dimensional wide angle wave equation. in *Computational Acoustics: Wave Propagation*, D. Lee, R. L. Sternberg, and M. H. Schultz, Eds., pp. 75–90, North Holland, New York, 1988

[7] M. H. Schultz. Multiple Array Processors for Ocean Acoustic Problems. in *Computational Ocean Acoustics*, M. H. Schultz and D. Lee, Eds., pp. 777–786, Pergamon Press, New York, 1985.

[8] M. H. Schultz. Harnessing Supercomputers for Computational Underwater Acoustics. in *Computational Acoustics, Vol. 1*, D. Lee, A. Cakmak, and R. Vichnevetsky, Eds., pp. 239–242, Elsevier Science Publishers, Amsterdam, 1990.

[9] F. D. Tappert. The parabolic approximation method. In *Wave Propagation and Underwater Acoustics*, J. B. Keller and J. S. Papadakis (Eds.), Lecture Notes in Physics, Vol. 70, Springer-Verlag, New York, 1977.