
Numerical Algorithms **29:** 1–19, 2002.

High accuracy solution of three-dimensional biharmonic equations

Irfan Altas [a], Jocelyne Erhel [b] and Murli M. Gupta [c]

[a] *School of Information Studies, Charles Sturt University, Wagga Wagga, NSW 2678, Australia*
E-mail: ialtas@csu.edu.au
[b] *IRISA/INRIA, Campus de Beaulieu, 35042 Rennes cedex, France*
E-mail: Jocelyne.Erhel@irisa.fr
[c] *Department of Mathematics, The George Washington University, Washington, DC 20052, USA*
E-mail: mmg@gwu.edu

Received 9 November 2000; revised 9 March 2001

In this paper, we consider several finite-difference approximations for the three-dimensional biharmonic equation. A symbolic algebra package is utilized to derive a family of finite-difference approximations for the biharmonic equation on a 27 point compact stencil. The unknown solution and its first derivatives are carried as unknowns at selected grid points. This formulation allows us to incorporate the Dirichlet boundary conditions automatically and there is no need to define special formulas near the boundaries, as is the case with the standard discretizations of biharmonic equations. We exhibit the standard second-order, finite-difference approximation that requires 25 grid points. We also exhibit two compact formulations of the 3D biharmonic equations; these compact formulas are defined on a 27 point cubic grid. The fourth-order approximations are used to solve a set of test problems and produce high accuracy numerical solutions. The system of linear equations is solved using a variety of iterative methods. We employ multigrid and preconditioned Krylov iterative methods to solve the system of equations. Test results from two test problems are reported. In these experiments, the multigrid method gives excellent results. The multigrid preconditioning also gives good results using Krylov methods.

Keywords: multigrid method, three-dimensional biharmonic equation, high accuracy, compact approximations, finite differences

AMS subject classification: 65N06, 65N55

1. Introduction

Consider the three-dimensional biharmonic equation:

$$\Delta^2 u(x, y, z) = f(x, y, z), \tag{1.1}$$

or

$$\frac{\partial^4 u}{\partial x^4} + \frac{\partial^4 u}{\partial y^4} + \frac{\partial^4 u}{\partial z^4} + 2\frac{\partial^4 u}{\partial x^2 \partial y^2} + 2\frac{\partial^4 u}{\partial x^2 \partial z^2} + 2\frac{\partial^4 u}{\partial y^2 \partial z^2} = f(x, y, z),$$

with $(x, y, z) \in \Omega$, and Dirichlet boundary conditions of the first kind

$$u = g_1(x, y, z), \quad \frac{\partial u}{\partial n} = g_2(x, y, z), \qquad (x, y, z) \in \partial\Omega. \tag{1.2}$$

Here Ω is a closed convex domain in three dimensions and $\partial\Omega$ is its boundary.

Equation (1.1) is the three-dimensional version of the 2D biharmonic equation

$$\frac{\partial^4 u}{\partial x^4} + \frac{\partial^4 u}{\partial y^4} + 2\frac{\partial^4 u}{\partial x^2 \partial y^2} = f(x, y), \quad (x, y) \in \Omega, \tag{1.3}$$

that has been considered extensively in the literature. Altas et al. [1] presented a family of compact finite-difference approximations for the 2D biharmonic equation; these compact approximations were obtained through the use of a symbolic algebra package *Mathematica*.

In [2], Altas and Gupta presented another method, also derived through *Mathematica*, to solve the second biharmonic boundary value problem where values of u and its second normal derivative $\partial^2 u/\partial n^2$ are prescribed on $\partial\Omega$.

Various approaches for the numerical solution of the 2D biharmonic equation (1.3) have been considered in the literature. A popular technique is to split it into two coupled Poisson equations for u and v:

$$\begin{cases} \dfrac{\partial^2 u}{\partial x^2} + \dfrac{\partial^2 u}{\partial y^2} = v, \\ \dfrac{\partial^2 v}{\partial x^2} + \dfrac{\partial^2 v}{\partial y^2} = f, \end{cases} \tag{1.4}$$

each of which may be discretized using the standard approximations and solved using any of the Poisson solvers. Difficulty with this approach is that the boundary conditions for the new variable v are undefined and need to be approximated from the discrete form of equation (1.4). The coupled equation approach has been used by many authors for many years (see [1,12,14] and other references contained therein for detailed background), and there have also been efforts to introduce multigrid techniques with the coupled equation approach. However, these computations are dependent on accurate evaluation of the missing boundary values for v, and the computational procedures are often unsatisfactory. Mixed approximations have also been considered, see, for example, [11], and multigrid preconditioners have been designed for this approach [20].

Not many authors have tried to solve the three-dimensional biharmonic equation. The reason, of necessity, is that three-dimensional problems require large computing power and place huge amounts of memory requirements on the computational systems. Such computing power has only recently begun to become available for academic research.

A conventional approach for solving the three-dimensional biharmonic equations is to discretize equation (1.1) on a uniform grid using a 25 point approximation with truncation error of order h^2. Such an approximation of 3D biharmonic equation is not generally available in the literature and is exhibited below. Using a 25-point finite-difference "star", this approximation was derived by Ribeiro Dos Santos [18] in 1967 who also proposed certain modifications required near the boundaries.

Consider a three-dimensional uniform grid centered at the point (x_i, y_j, z_k); values of unknown solution u and other functions at the point (x_i, y_j, z_k) are written as $u_{i,j,k}$. The conventional 25-point finite-difference approximation of equation (1.1) at the point (x_i, y_j, z_k) may be written as

$$\begin{aligned} &42u_{i,j,k} - 12(u_{i+1,j,k} + u_{i,j+1,k} + u_{i,j,k+1} + u_{i-1,j,k} + u_{i,j-1,k} + u_{i,j,k-1}) \\ &\quad + u_{i+2,j,k} + u_{i,j+2,k} + u_{i,j,k+2} + u_{i-2,j,k} + u_{i,j-2,k} + u_{i,j,k-2} \\ &\quad + 2(u_{i+1,j,k+1} + u_{i,j+1,k+1} + u_{i-1,j,k+1} + u_{i,j-1,k+1} + u_{i+1,j,k-1} + u_{i,j+1,k-1} \\ &\quad + u_{i-1,j,k-1} + u_{i,j-1,k-1} + u_{i+1,j+1,k} + u_{i-1,j+1,k} + u_{i-1,j-1,k} + u_{i+1,j-1,k}) \\ &\quad = h^4 f_{i,j,k}. \end{aligned} \tag{1.5}$$

This approximation has a truncation error of order h^2 and connects the values of $u_{i,j,k}$ in terms of 24 neighboring values of u in a $5\times5\times5$ "star" grid; the two-dimensional biharmonic equation is similarly approximated by the 13-point formula in a 5×5 grid. We note that the value of $u_{i,j,k}$ is connected to grid points two grids away in each direction from the point (x_i, y_j, z_k) and the above difference approximation needs to be modified at grid points near the boundaries. In case of 2D biharmonic equations, many such modifications are discussed in [15]. However, there are serious computational difficulties with solution of the linear systems obtained through the 13-point discretization of the 2D biharmonic equations [8,15]; these problems are worse in 3D. Approximations using compact cells avoid these difficulties.

Certain second- and fourth-order finite-difference approximations for the biharmonic equation (1.1) on a 9-point compact cell have been known for some time [1,16, 21]. The compact approach involves discretizing the biharmonic equation using not just the grid values of the unknown solution u but also the values of the gradients u_x, u_y and u_z at selected grid points. Introducing u_x, u_y and u_z as unknowns may look as if the computational cost of our method would increase fourfold when compared with a method with only u as unknown. However, as demonstrated for the 2D problems in [1], the compact fourth-order difference scheme produced an accuracy with a 16×16 grid that was comparable to the accuracy of a conventional second-order method using a 256×256 grid. As described in detail in [1], our approach is advantageous because

(i) the given Dirichlet boundary conditions are exactly satisfied and no approximations need to be carried out at the boundaries, in contrary to the splitting method (1.4);

(ii) the proposed finite-difference approximations are derived on a compact cell and no modifications are needed at grid points near the boundaries; this eliminates the problems of the conventional approach described above;

(iii) the values of gradients u_x, u_y and u_z are already available at all grid points and need not be approximated from the computed values of the solution u;

(iv) the proposed methodology represents a streamfunction–velocity formulation of fluid flow problems (such as Stokes' flow problems represented by the biharmonic equation, and the Navier–Stokes equations), and produces physically meaningful solutions with an efficient formulation and computational procedure.

In summary, we claim that the finite-difference approximation using a compact cell has several advantages. We extend the 2D *Mathematica* code presented in [1] to three dimensions and obtain a family of finite-difference approximations on the 27-point compact cubic cell. In section 2, we present finite-difference approximations with truncation errors of order h^2 and h^4. In section 3, we briefly introduce multigrid, Krylov and classical iterative schemes that are employed to solve the system of equations arising from the fourth-order discretisation of the biharmonic equation (1.1). Results of our computations show that the proposed finite-difference approximations yield highly accurate solutions that exhibit a fourth-order convergence. A brief comparison of convergence behaviour of Krylov and multigrid methods for two test problems is introduced in section 4. Conclusions are presented in section 5.

2. Compact finite-difference approximations

Using our *Mathematica* code, we can obtain a variety of finite-difference approximations by choosing various combinations of the grid values of u, u_x, u_y and u_z to be used in the derivations. For example, an efficient finite-difference approximation of order h^2 may be obtained by choosing the values of u at 18 neighboring points of (x_i, y_j, z_k) and the values of u_x, u_y and u_z at two neighboring points in the respective directions. With these choices, the *Mathematica* code produces the following finite-difference approximation:

$$\begin{aligned}
&48u_{i,j,k} - 10(u_{i+1,j,k} + u_{i,j+1,k} + u_{i,j,k+1} + u_{i-1,j,k} + u_{i,j-1,k} + u_{i,j,k-1}) \\
&\quad + (u_{i+1,j,k+1} + u_{i,j+1,k+1} + u_{i-1,j,k+1} + u_{i,j-1,k+1} + u_{i+1,j,k-1} + u_{i,j+1,k-1} \\
&\quad + u_{i-1,j,k-1} + u_{i,j-1,k-1} + u_{i+1,j+1,k} + u_{i-1,j+1,k} + u_{i-1,j-1,k} + u_{i+1,j-1,k}) \\
&\quad + 3h(u_{x_{i+1,j,k}} - u_{x_{i-1,j,k}} + u_{y_{i,j+1,k}} - u_{y_{i,j-1,k}} + u_{z_{i,j,k+1}} - u_{z_{i,j,k-1}}) \\
&\qquad = \frac{h^4}{2} f_{i,j,k}. \qquad (2.1)
\end{aligned}$$

We observe that this approximation utilizes the values of u in the compact cubic cell and does not require the values of u two grid points away from the central point (x_i, y_j, z_k). Corresponding finite-difference approximations for the gradients u_x, u_y and u_z at (x_i, y_j, z_k) may also be obtained from the *Mathematica* code:

$$hu_{x_{i,j,k}} = \frac{3}{4}(u_{i+1,j,k} - u_{i-1,j,k}) - \frac{h}{4}(u_{x_{i+1,j,k}} + u_{x_{i-1,j,k}}), \tag{2.2}$$

$$hu_{y_{i,j,k}} = \frac{3}{4}(u_{i,j+1,k} - u_{i,j-1,k}) - \frac{h}{4}(u_{y_{i,j+1,k}} + u_{y_{i,j-1,k}}), \tag{2.3}$$

$$hu_{z_{i,j,k}} = \frac{3}{4}(u_{i,j,k+1} - u_{i,j,k-1}) - \frac{h}{4}(u_{z_{i,j,k+1}} + u_{z_{i,j,k-1}}). \tag{2.4}$$

A fourth-order compact finite-difference approximation for the three-dimensional biharmonic equation (1.1) is given below:

$$\begin{aligned}
&u_{i,j,k} - \frac{19}{104}(u_{i+1,j,k} + u_{i,j+1,k} + u_{i,j,k+1} + u_{i-1,j,k} + u_{i,j-1,k} + u_{i,j,k-1}) \\
&\quad + \frac{1}{208}(u_{i+1,j,k+1} + u_{i,j+1,k+1} + u_{i-1,j,k+1} + u_{i,j-1,k+1} \\
&\qquad + u_{i+1,j,k-1} + u_{i,j+1,k-1} + u_{i-1,j,k-1} + u_{i,j-1,k-1} \\
&\qquad + u_{i+1,j+1,k} + u_{i-1,j+1,k} + u_{i-1,j-1,k} + u_{i+1,j-1,k} \\
&\qquad + u_{i+1,j+1,k+1} + u_{i-1,j+1,k+1} + u_{i-1,j-1,k+1} + u_{i+1,j-1,k+1} \\
&\qquad + u_{i+1,j+1,k-1} + u_{i-1,j+1,k-1} + u_{i-1,j-1,k-1} + u_{i+1,j-1,k-1}) \\
&\quad + \frac{13}{208}h(u_{x_{i+1,j,k}} - u_{x_{i-1,j,k}} + u_{y_{i,j+1,k}} - u_{y_{i,j-1,k}} + u_{z_{i,j,k+1}} - u_{z_{i,j,k-1}}) \\
&\quad + \frac{1}{416}h(u_{x_{i+1,j,k+1}} - u_{x_{i-1,j,k+1}} + u_{x_{i+1,j,k-1}} - u_{x_{i-1,j,k-1}} \\
&\qquad + u_{x_{i+1,j+1,k}} - u_{x_{i-1,j+1,k}} + u_{x_{i+1,j-1,k}} - u_{x_{i-1,j-1,k}} \\
&\qquad + u_{y_{i,j+1,k+1}} - u_{y_{i,j-1,k+1}} + u_{y_{i,j+1,k-1}} - u_{y_{i,j-1,k-1}} \\
&\qquad + u_{y_{i+1,j+1,k}} + u_{y_{i-1,j+1,k}} - u_{y_{i-1,j-1,k}} - u_{y_{i+1,j-1,k}} \\
&\qquad + u_{z_{i+1,j,k+1}} + u_{z_{i,j+1,k+1}} + u_{z_{i,j-1,k+1}} + u_{z_{i-1,j,k+1}} \\
&\qquad - u_{z_{i+1,j,k-1}} - u_{z_{i,j+1,k-1}} - u_{z_{i,j-1,k-1}} - u_{z_{i-1,j,k-1}}) \\
&\quad = \frac{h^4}{1248}\big[9f_{i,j,k} + (f_{i+1,j,k} + f_{i,j+1,k} + f_{i,j,k+1} \\
&\qquad\qquad + f_{i-1,j,k} + f_{i,j-1,k} + f_{i,j,k-1})\big].
\end{aligned} \tag{2.5}$$

This finite-difference approximation discretizes equation (1.1) at a grid point (x_i, y_j, z_k) using the 26 neighboring values of u; it also uses 12 values of u_x, 12 values of u_y and 12 values of u_z at appropriate neighboring points. Truncation error of this approximation is of order h^4.

Compatible approximations for u_x, u_y and u_z at (x_i, y_j, z_k) are given, respectively, by

$$
\begin{aligned}
hu_{x_{i,j,k}} &= \frac{5}{12}(u_{i+1,j,k} - u_{i-1,j,k}) \\
&\quad + \frac{11}{120}(u_{i+1,j,k+1} - u_{i-1,j,k+1} + u_{i+1,j,k-1} - u_{i-1,j,k-1} \\
&\qquad + u_{i+1,j+1,k} - u_{i-1,j+1,k} - u_{i-1,j-1,k} + u_{i+1,j-1,k}) \\
&\quad + \frac{1}{120}(-u_{i+1,j+1,k+1} + u_{i-1,j+1,k+1} + u_{i-1,j-1,k+1} - u_{i+1,j-1,k+1} \\
&\qquad - u_{i+1,j+1,k-1} + u_{i-1,j+1,k-1} + u_{i-1,j-1,k-1} - u_{i+1,j-1,k-1}) \\
&\quad - \frac{3}{20}h(u_{x_{i+1,j,k}} + u_{x_{i-1,j,k}}) \\
&\quad - \frac{1}{40}h(u_{x_{i+1,j,k+1}} + u_{x_{i-1,j,k+1}} + u_{x_{i+1,j,k-1}} + u_{x_{i-1,j,k-1}} \\
&\qquad + u_{x_{i+1,j+1,k}} + u_{x_{i-1,j+1,k}} + u_{x_{i-1,j-1,k}} + u_{x_{i+1,j-1,k}} \\
&\qquad + u_{y_{i+1,j+1,k}} - u_{y_{i-1,j+1,k}} + u_{y_{i-1,j-1,k}} - u_{y_{i+1,j-1,k}} \\
&\qquad + u_{z_{i+1,j,k+1}} - u_{z_{i-1,j,k+1}}) - u_{z_{i+1,j,k-1}} + u_{z_{i-1,j,k-1}}) \\
&\quad + \frac{h^4}{240}(f_{i+1,j,k} - f_{i-1,j,k}),
\end{aligned} \tag{2.6}
$$

$$
\begin{aligned}
hu_{y_{i,j,k}} &= \frac{5}{12}(u_{i,j+1,k} - u_{i,j-1,k}) \\
&\quad + \frac{11}{120}(u_{i,j+1,k+1} - u_{i,j-1,k+1} + u_{i,j+1,k-1} - u_{i,j-1,k-1} \\
&\qquad + u_{i+1,j+1,k} + u_{i-1,j+1,k} - u_{i-1,j-1,k} - u_{i+1,j-1,k}) \\
&\quad + \frac{1}{120}(-u_{i+1,j+1,k+1} - u_{i-1,j+1,k+1} + u_{i-1,j-1,k+1} + u_{i+1,j-1,k+1} \\
&\qquad - u_{i+1,j+1,k-1} - u_{i-1,j+1,k-1} + u_{i-1,j-1,k-1} + u_{i+1,j-1,k-1}) \\
&\quad - \frac{3}{20}h(u_{y_{i,j+1,k}} + u_{y_{i,j-1,k}}) \\
&\quad - \frac{1}{40}h(u_{x_{i+1,j+1,k}} - u_{x_{i-1,j+1,k}} + u_{x_{i-1,j-1,k}} - u_{x_{i+1,j-1,k}} \\
&\qquad + u_{y_{i,j+1,k+1}} + u_{y_{i,j-1,k+1}} + u_{y_{i,j+1,k-1}} + u_{y_{i,j-1,k-1}} \\
&\qquad + u_{y_{i+1,j+1,k}} + u_{y_{i-1,j+1,k}} + u_{y_{i-1,j-1,k}} + u_{y_{i+1,j-1,k}} \\
&\qquad + u_{z_{i,j+1,k+1}} - u_{z_{i,j-1,k+1}} + u_{z_{i,j-1,k-1}} - u_{z_{i,j+1,k-1}}) \\
&\quad + \frac{h^4}{240}(f_{i,j+1,k} - f_{i,j-1,k}),
\end{aligned} \tag{2.7}
$$

$$
\begin{aligned}
hu_{z_{i,j,k}} &= \frac{5}{12}(u_{i,j,k+1} - u_{i,j,k-1}) \\
&\quad + \frac{11}{120}(u_{i+1,j,k+1} + u_{i,j+1,k+1} + u_{i-1,j,k+1} + u_{i,j-1,k+1} \\
&\qquad - u_{i+1,j,k-1} - u_{i,j+1,k-1} - u_{i-1,j,k-1} - u_{i,j-1,k-1})
\end{aligned}
$$

$$
\begin{aligned}
&+\frac{1}{120}(-u_{i+1,j+1,k+1}-u_{i-1,j+1,k+1}-u_{i-1,j-1,k+1}-u_{i+1,j-1,k+1}\\
&\qquad+u_{i+1,j+1,k-1}+u_{i-1,j+1,k-1}+u_{i-1,j-1,k-1}+u_{i+1,j-1,k-1})\\
&-\frac{3}{20}h(u_{z_{i,j,k+1}}+u_{z_{i,j,k-1}})\\
&-\frac{1}{40}h(u_{x_{i+1,j,k+1}}-u_{x_{i-1,j,k+1}}-u_{x_{i+1,j,k-1}}+u_{x_{i-1,j,k-1}}\\
&\qquad+u_{y_{i,j+1,k+1}}-u_{y_{i,j-1,k+1}}-u_{y_{i,j+1,k-1}}+u_{y_{i,j-1,k-1}}\\
&\qquad+u_{z_{i+1,j,k+1}}+u_{z_{i,j+1,k+1}}+u_{z_{i,j-1,k+1}}+u_{z_{i-1,j,k+1}}\\
&\qquad+u_{z_{i+1,j,k-1}}+u_{z_{i,j+1,k-1}}+u_{z_{i,j-1,k-1}}+u_{z_{i-1,j,k-1}})\\
&+\frac{h^4}{240}(f_{i,j,k+1}-f_{i,j,k-1}). \qquad (2.8)
\end{aligned}
$$

We discuss the solution of linear systems associated with the above finite-difference approximations in the next section.

3. Solution of linear systems

By writing equations (2.5)–(2.8) at every interior grid points one obtains a system of linear algebraic equations for equation (1.1).

Direct solution of these linear systems is impractical because of the huge size of the coefficient matrix and enormous storage requirements even for moderate values of grid size h.

On the other hand, the condition number of the coefficient matrix increases rapidly with the grid size h and one must be very cautious when attempting to solve such linear systems using iterative linear solvers (see [2,13,15] for two-dimensional problems).

The performance of iterative solvers is sensitive to the number of equations to be solved, the type of boundary conditions applied, the condition number and other factors. Classical iterative solvers such as Gauss–Seidel and SOR are attractive for their low storage requirements as long as convergence is guaranteed. Employing these classical methods to solve the linear systems arising from the fourth-order discretization suffers from extremely slow convergence. For example, the Gauss–Seidel method requires over 13,000 and 200,000 iterations to solve the linear system arising from the discretization of test problem 1 (see section 4) on $16 \times 16 \times 16$ and $32 \times 32 \times 32$ grids, respectively. The required error tolerance for the maximum absolute iteration error is 10^{-10}. This example clearly demonstrates that one needs some other approaches that may produce faster convergence rates. In this paper, our aim is to solve the linear systems arising from (2.5)–(2.8) by employing various iterative solvers and to compare results. One of the approaches to accomplish faster convergence rates is the implementation of a multigrid algorithm discussed in section 3.1. Another approach is to use preconditioned Krylov methods, as discussed in section 3.2.

3.1. Multigrid

The rate of convergence of basic iterative methods can be improved by employing multigrid methods. Since the pioneering work of Brandt [3] in the early 1970s, multigrid methods have been widely applied to the numerical solution of differential equations. A good introductory text on multigrid is the book by Briggs [5], more advanced treatment is given by Brandt in [4]. We present a brief description of how multigrid works.

While iterative processes are sometimes slow to solve differential equations, they tend to make good *smoothers*. That is, analyzing Fourier components of the *error*, an iterative solver will typically sharply reduce the oscillatory components, while leaving the smooth components virtually unchanged. These smooth components can be solved for on a coarser grid by computing the residual of the equation, *restricting* it to the coarse grid, and solving. This is more efficient, both due to the smaller number of coarse grid points and to the fact that smooth fine grid components become oscillatory on the coarse grid ("smoothness" being measured in gridpoints per wavelength), thus, are efficiently solved by the iterative method. Components that are still slow to converge on the coarse grid are transferred to a yet coarser grid, and so on, until a grid is reached where all components can be efficiently resolved. The error components solved for on the coarse grid are added to the fine grid solution, using interpolation to determine the correction values at fine grid points.

A multigrid *cycle* starts with a number (ν) of relaxations of the iterative scheme, transfers the (now smoothed) error to a coarser grid where a number (γ) of multigrid cycles are performed before the solution is interpolated back to the fine grid, and some (μ) more relaxations performed. Setting $\gamma = 1$ results in what is called a "V cycle", while $\gamma = 2$ gives a "W cycle".

A good initial guess for the multigrid cycle may be obtained cheaply by solving a coarsened version of the problem and interpolating it to a finer grid. The FMG (Full Multigrid) algorithm uses this idea recursively, starting at a relatively coarse grid and going to progressively finer grids. This minimizes the work done on fine grids – starting out with the interpolated coarse grid solution.

3.2. Krylov methods

Krylov methods are now commonly used to get fast convergence rates. Recent surveys are given, for example, in books from Saad [19], Meurant [17] and Bruaset [6].

Here the linear algebraic system to solve is nonsymmetric with no storage of the matrix required. In order to minimize storage, we have chosen to implement BiCGSTAB [22] and QMR [10] which rely on short recurrences so that storage requirement is well controlled. Thanks to the compact grid, it is easy to implement both matrix–vector products Ax and $A^{\mathrm{T}}x$ without storing the matrix A (matrix-free version). Hence, the Transpose-Free QMR [9] algorithm is not required here. Both methods rely on the nonsymmetric Lanczos process.

Usually, it is necessary to use preconditioned versions of Krylov methods. Classical preconditioners are SSOR, where the preconditioning matrix comes from the SSOR

decomposition or m-step SSOR where the SSOR iteration is applied m times. Here, preconditioners such as Incomplete Cholesky or Approximate Inverse are not considered, because the matrices are not stored. One of our objectives is to compare multigrid by itself with multigrid used as a preconditioner of Krylov methods. Therefore, multigrid preconditioning is implemented with a fixed number of cycles.

4. Numerical experiments

4.1. Test problem 1

We consider the biharmonic boundary value problem (1) in a unit cube with the exact solution

$$u(x, y, z) = (1 - \cos 2\pi x)(1 - \cos 2\pi y)(1 - \cos 2\pi z). \tag{4.1}$$

The forcing term $f(x, y, z)$ and boundary data g_1, g_2 are obtained from u.

4.2. Test problem 2

We consider the biharmonic boundary value problem (1) in a unit cube with the exact solution given by

$$u(x, y, z) = \left(x^2 - x\right)\left(y^2 - y\right)\left(z^2 - z\right)\mathrm{e}^{q[(x-0.5)^2+(y-0.5)^2+(z-p)^2]}. \tag{4.2}$$

The forcing term $f(x, y, z)$ and boundary data g_1, g_2 are obtained from u.

The exact solution of this test problem is strongly peaked for large values of the parameter q. The second parameter p moves the peak along the z-direction. The computational results are presented here for $p = 0.2$ and $q = 10$.

4.3. Computed solution error of the method

First we measure the solution error in order to check the fourth-order convergence behaviour of our method. Figures 1 and 2 show respectively the exact solution u and the computed solution u_h for problem 1 (on a grid of size 32^3). Figure 3 shows the computed solution u_h for problem 2 (on a grid of size 64^3).

Since the exact solution is known, it is easy to compute the discretisation errors (i.e., the solution errors) for u and its derivatives u_x, u_y and u_z. Figures 4 and 5 show the Euclidean norm of the discretisation errors with respect to the grid size for problems 1 and 2, respectively. It is clear that the errors decay with order h^4 as the mesh size h is reduced. Also, as expected, problem 2 is more difficult to solve and the error for u can be approximated as $0.4 \times 10^4 \times h^4$, whereas it can be approximated as $1 \times h^4$ in problem 1.

4.4. Multigrid results

In this work we have implemented both V and W cycle versions of multigrid. We observe that W cycle produces better convergence rates than V cycle implementations.

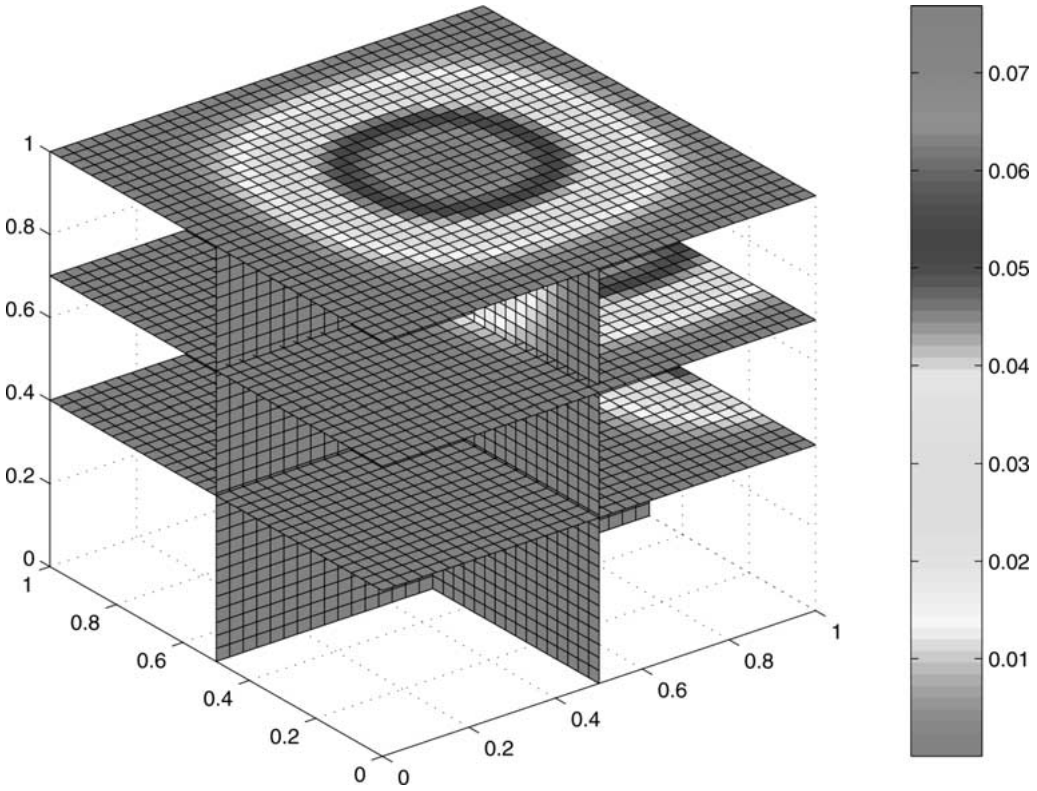

Figure 1. Exact solution for problem 1.

As a smoothing scheme we have tried different iteration schemes such as Gauss–Seidel, SOR, and zebra SOR. For our test problems SOR iteration scheme works reasonably well with the relaxation parameters 1.25, 1, 1 and 1 for equations (2.5), (2.6), (2.7) and (2.8), respectively. We pick $\nu = 4$ and $\mu = 4$. The multigrid algorithm for our test problems is sensitive to the residual transfer. We employ an order 0 residual method. That is, the residuals are directly calculated from the corresponding difference schemes and injected to a lower level with the following ratios. Three quarter of the residuals of equation (2.5) are transferred to the coarser grid. Only one quarter of the residuals from the derivative equations (2.6)–(2.8) are transferred to the coarser grid. As the interpolation procedure we employ the standard approach. That is, we directly transfer values to finer levels at the points common to both levels, and the rest of the values are transferred as the average of either two or four closest grid points. We also implemented full multigrid with the above parameters. However, this implementation produces similar results to the one without full multigrid.

We performed the following calculations on Sun Ultra Enterprise 450 with 3 UltraSPARC-II 248 MHz CPU and 1 Gbyte memory. The stopping criterion here is the absolute maximum value of the differences of values of $u(x, y, z)$ from two consecutive multigrid cycles. We stop the multigrid algorithm when the criterion is less

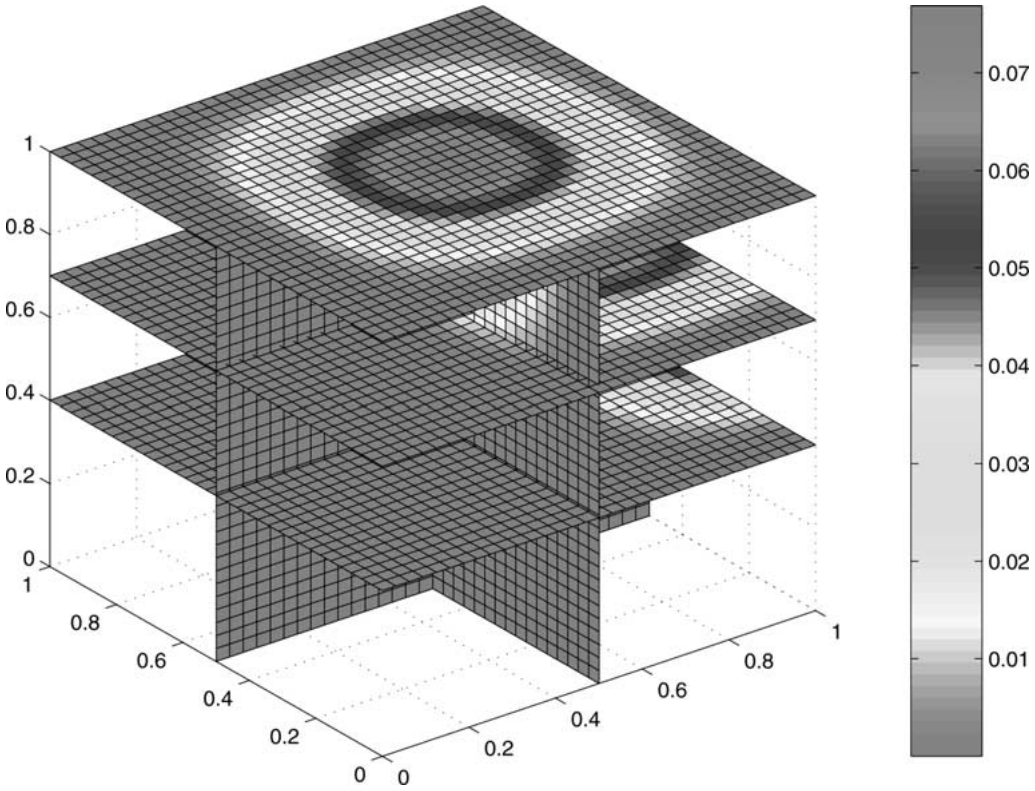

Figure 2. Computed solution for problem 1.

than 10^{-10}. Results for problems 1 and 2 are reported in tables 1 and 2, respectively. As expected, the number of cycles is almost independent of the mesh size, thus indicating true multigrid performance. On the other hand, the CPU time is roughly proportional to $1/h^3$.

Since the exact solution is known, we have also computed the error during the computational cycles. Figures 6 and 7 plot the criterion and the solution errors for mesh sizes 16, 32, and 64. Here the criterion is the Euclidean norm of the difference between two consecutive cycles normalized by the Euclidean norm of the right-hand side. The solution error is the Euclidean norm of the difference between the exact solution and the computed solution, normalized by the Euclidean norm of the exact solution. We observe a linear convergence of the criterion and the solution error; the solution error decreases down to a plateau which corresponds to the discretization error. Further cycles have no effect. Therefore, an efficient criterion needs to be able to estimate this error and stop early enough.

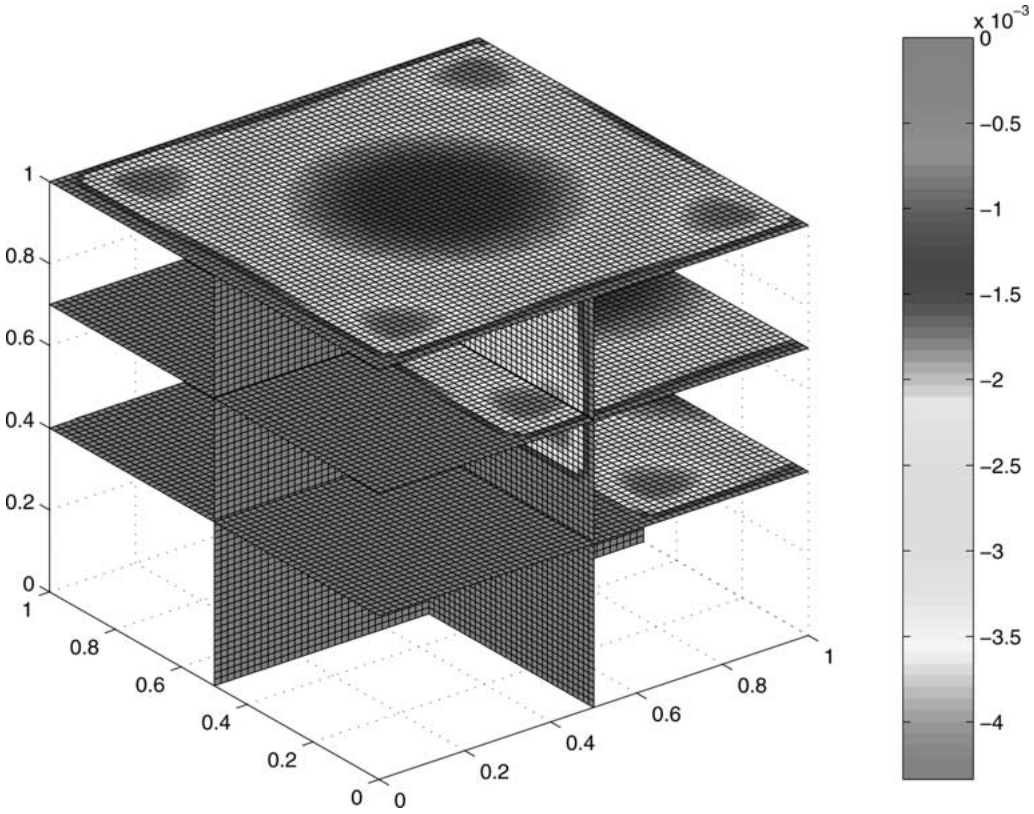

Figure 3. Computed solution for problem 2.

Table 1
Multigrid results for problem 1.

Mesh size	Discretisation error	Time (s)	W cycles
$16 \times 16 \times 16$	3.9×10^{-5}	64.7	20
$32 \times 32 \times 32$	2.3×10^{-6}	679.5	22
$64 \times 64 \times 64$	3.0×10^{-7}	5747.9	22

Table 2
Multigrid results for problem 2.

Mesh size	Discretisation error	Time (s)	W cycles
$16 \times 16 \times 16$	8.9×10^{-2}	60.9	19
$32 \times 32 \times 32$	7.2×10^{-3}	594.9	20
$64 \times 64 \times 64$	4.8×10^{-4}	5388.0	21

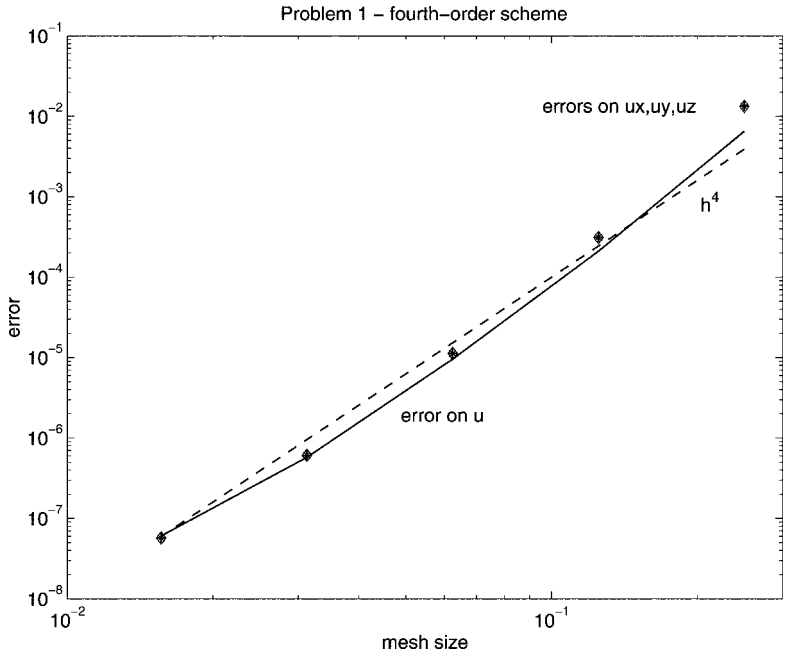

Figure 4. Discretisation errors for problem 1.

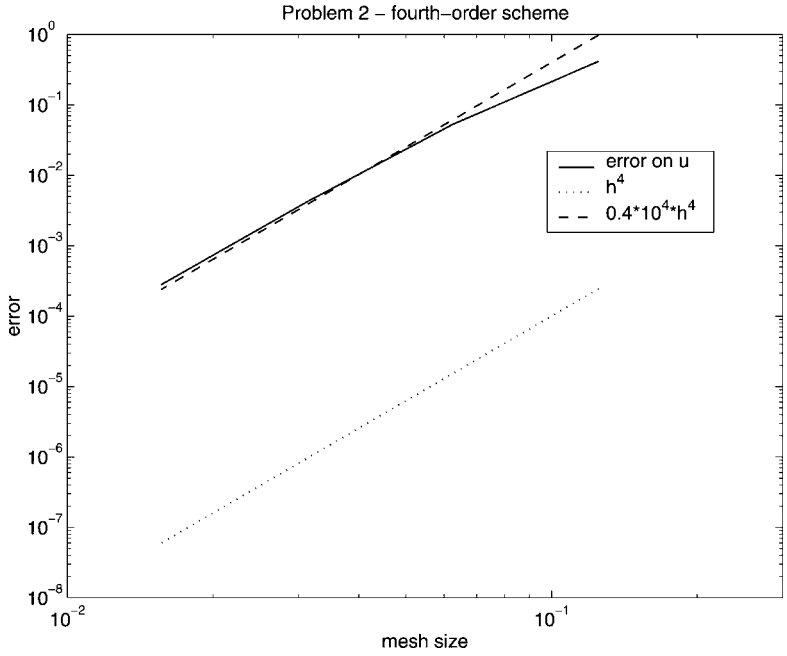

Figure 5. Discretisation errors for problem 2.

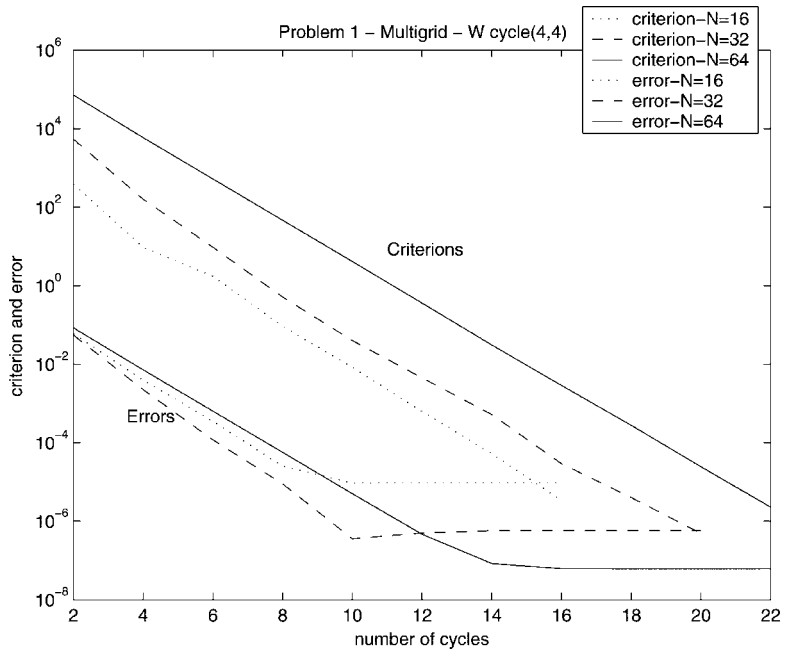

Figure 6. Criterion and error histories for problem 1.

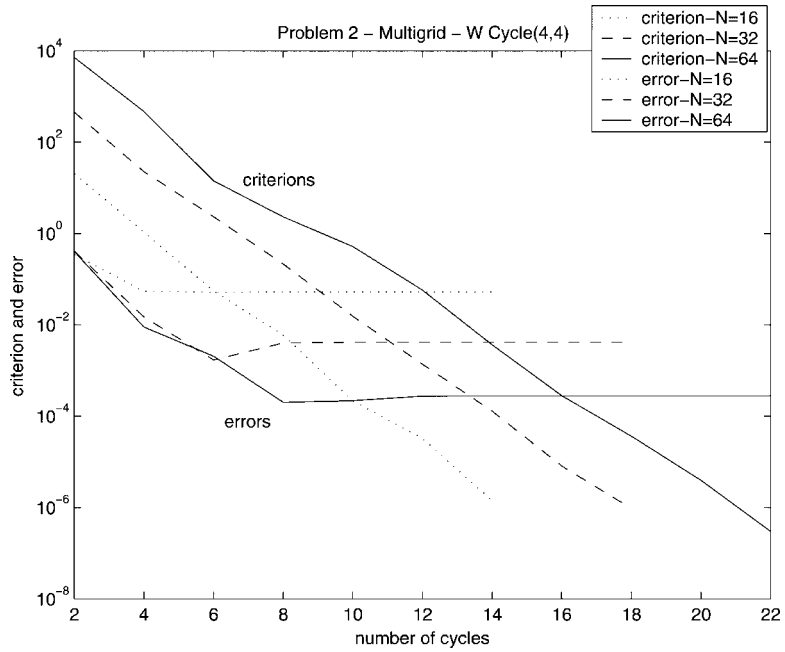

Figure 7. Criterion and error histories for problem 2.

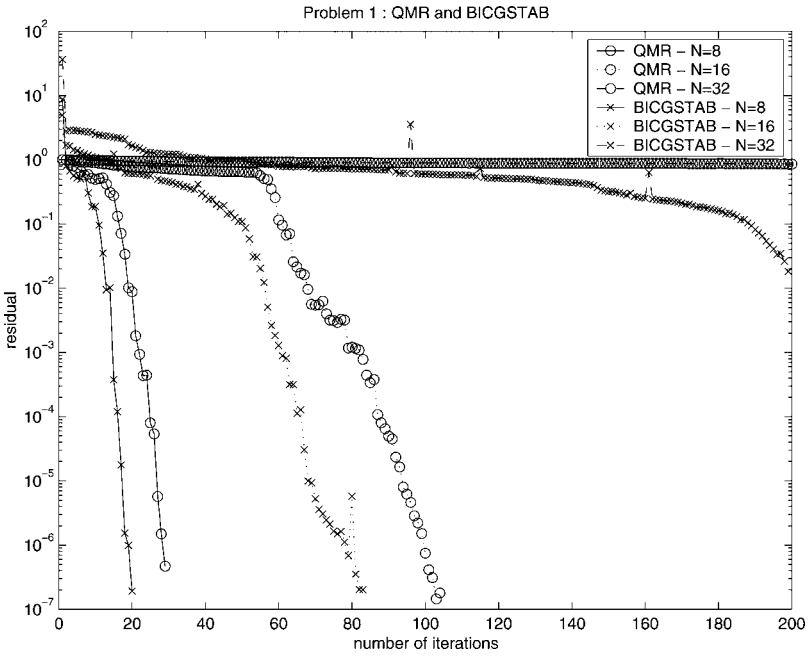

Figure 8. Comparison between BiCGSTAB and QMR for problem 1.

4.5. *BiCGSTAB results*

We need to solve the linear system $Ax = b$ and the residual $r = b - Ax$ is used to control the convergence. In our examples, the residual computed from recurrences in the algorithm and the true residual are equal. The stopping criterion at iteration k is, thus, $\|r_k\|_2/\|b\|_2$.

We first compare BiCGSTAB with QMR for various grid sizes and problem 1. Results are shown on figure 8. For this example, QMR is somewhat less efficient than BiCGSTAB, so we have selected BiCGSTAB for further experiments.

Then we compare several preconditioners and estimate the computational costs for the different preconditioners tested. Since matrix–vector products are the most time-consuming operations, we take this operation as the comparison unit, called U in the following. Hence, one BiCGSTAB iteration requires $2U$ and two preconditioning calls. SSOR preconditioning requires $1U$ and m-step SSOR requires $m \times U$. One V cycle of multigrid requires roughly $(\nu + \mu + 2)U$, where $1U$ is counted for restriction and $1U$ is counted for interpolation. Here smoothing on coarse grids is neglected. One W cycle is about 60% more expensive than one V cycle. In our experiments, we observed that multigrid preconditioning was not too sensitive to the parameters. We have chosen to use V cycles. For the grid size 32, we chose parameters $\nu = 2$, $\mu = 2$ and for the grid size 64, we chose parameters $\nu = 4$, $\mu = 4$.

Results for problem 1 are shown on figures 9 and 10, for grid size 32 and 64. Clearly, SSOR and m-step SSOR are not very efficient especially for the $64 \times 64 \times 64$

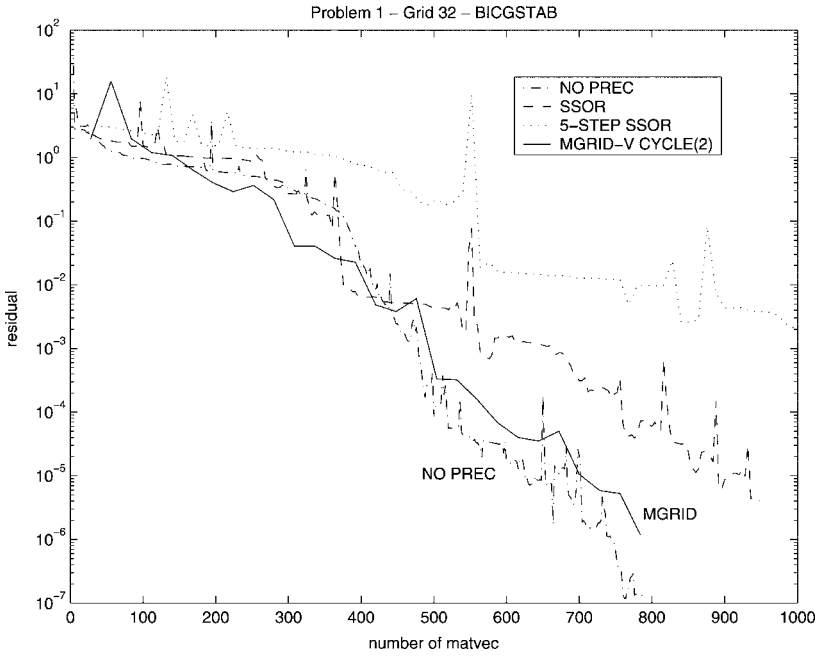

Figure 9. Comparison between several preconditioned BiCGSTAB for problem 1 and for a grid size 32^3.

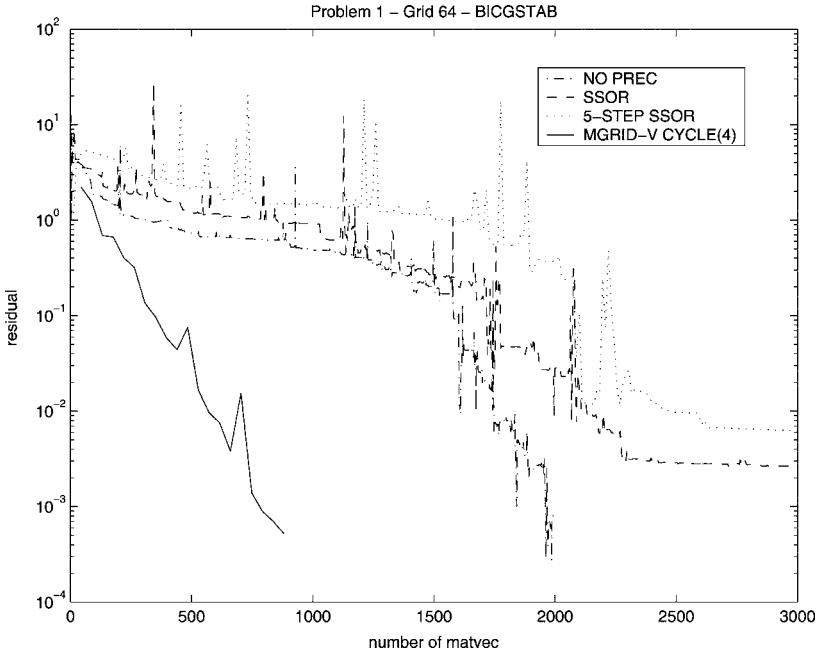

Figure 10. Comparison between several preconditioned BiCGSTAB for problem 1 and for a grid size 64^3.

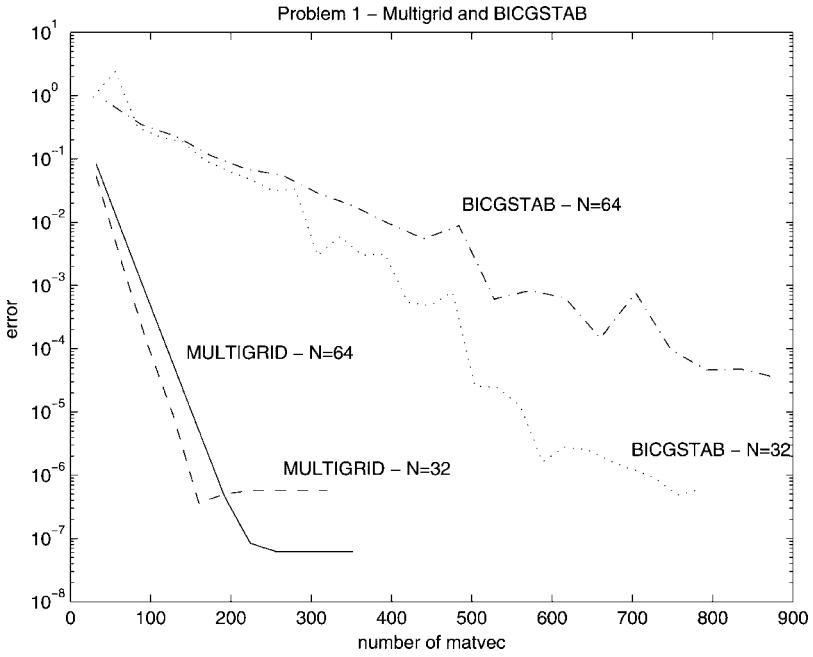

Figure 11. Comparison between BiCGSTAB and MULTIGRID for problem 1.

grid. Indeed, the condition number of the preconditioned matrix depends on h, so that the impact of preconditioning is about the same for all grid sizes. On the other hand, multigrid is a very efficient preconditioner for finer grids and the condition number is almost independent of h.

4.6. Comparison between multigrid and BiCGSTAB

In order to compare BiCGSTAB preconditioned by multigrid with multigrid on its own, we plot the computational cost in terms of the number of matvec units versus the solution error. Figures 11 and 12 show the results for problems 1 and 2, respectively, for grid sizes 32 and 64. For these problems, multigrid is clearly very efficient and cannot be further accelerated by a Krylov method. Here the problems are homogeneous and isotropic, the grid is regular, and the grid size is a power of 2. Hence, we can conclude that under these conditions, multigrid is the most efficient algorithm.

5. Conclusions

In this paper we examine a high-accuracy, compact formulation for the three-dimensional biharmonic equation with Dirichlet boundary conditions. The finite-difference approximation is derived on a 27-point compact stencil using the values of the solution and its gradients as the unknowns. The approximations have been derived using a symbolic software package. We solve several test problems using a multigrid method

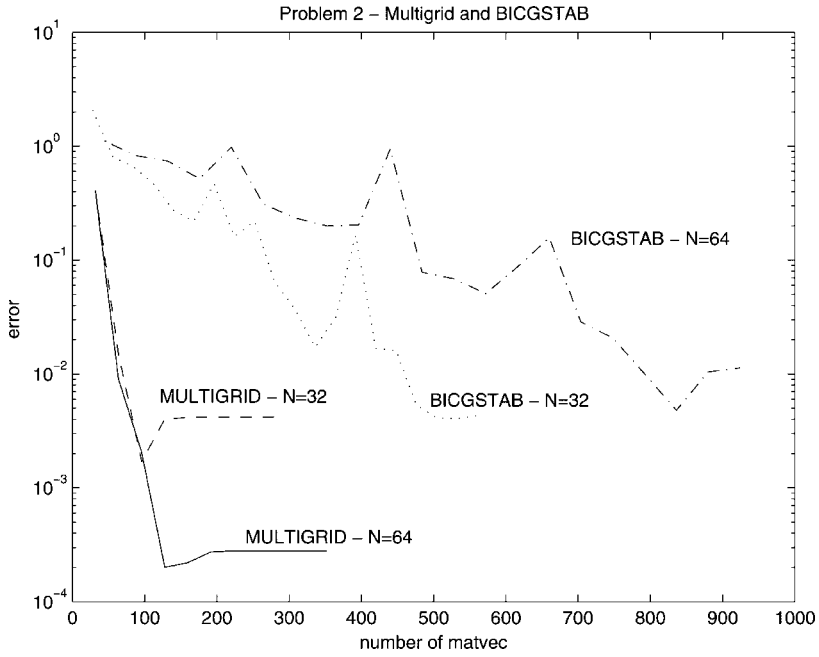

Figure 12. Comparison between BiCGSTAB and MULTIGRID for problem 2.

and a preconditioned BiCGSTAB method. Our results show that multilevel methods are required for large problems in order to get a convergence independent of the mesh size. For our test problems, multigrid by itself is very efficient. For unstructured grids, algebraic multigrid or multilevel preconditioners should be used.

We plan to introduce a stopping criterion based on an error estimation using the results at embedded grid levels. We also plan to implement a parallel version in order to reduce CPU timings.

References

[1] I. Altas, J. Dym, M.M. Gupta and R. Manohar, Multigrid solution of automatically generated high-order discretizations for the biharmonic equation, SIAM J. Sci. Comput. 19 (1998) 1575–1585.

[2] I. Altas and M.M. Gupta, Iterative methods for fixed point problems on high performance computing environment, in: *Proceedings of Complex Systems 1996, Albury, Australia, Complex Systems: From Local Interactions to Global Phenomena*, eds. R. Sticker, H. Jelinek, B. Durnota and T. Bossomaier (IOS Press, Amsterdam, 1996) pp. 121–128, `http://www.csu.edu.au/ci/vol3/altas2/altas2.html`

[3] A. Brandt, Multi-level adaptive technique (MLAT) for fast numerical solution of boundary value problems, in: *Proc. Third Int. Conf. on Numerical Methods in Fluid Mechanics*, Paris, 1972, Lecture Notes in Physics, Vol. 18 (Springer-Verlag, 1973) pp. 82–89.

[4] A. Brandt, Multigrid techniques: 1984 guide with applications to fluid dynamics, in: *GMD-Studien Nr. 85* (Gesselschaft für Mathematik und Datenverarbeitung MBH, St. Augustin, 1984).

[5] W. Briggs, *A Multigrid Tutorial* (SIAM, Philadelphia, 1984).

[6] A.M. Bruaset, *A Survey of Preconditioned Iterative Methods* (Longman Scientific and Technical, 1995).
[7] L. Bauer and E.L. Riess, Block five diagonal matrices and the fast numerical solution of the biharmonic equation, Math. Comp. 26 (1972) 311–326.
[8] S.D. Conte and R.T. Dames, On an alternating direction method for solving the plate problem with mixed boundary conditions, J. Assoc. Comput. Mach. 7 (1960) 264–273.
[9] R. Freund, A transpose free quasi minimal residual algorithm for non Hermitian linear systems, RIACS technical report 91.18, NASA Ames Research Center, Ames, CA (1991).
[10] R. Freund and N. Nachtigal, QMR: a quasi-minimal residual method for non hermitian linear systems, Numer. Math. 60 (1991) 315–339.
[11] R. Glowinski and O. Pironneau, Numerical methods for the first biharmonic equations and for the two-dimensional Stokes problem, SIAM Rev. 21 (1979) 167–212.
[12] M.M. Gupta, Discretization error estimates for certain splitting procedures for solving first biharmonic boundary value problems, SIAM J. Numer. Anal. 12 (1975) 364–377.
[13] M.M. Gupta, Spectrum transformation methods for divergent iterations, NASA technical memorandum 103745, ICOMP-91-02, NASA Lewis Research Center, Cleveland, OH (1991).
[14] M.M. Gupta and L.W. Ehrlich, Some difference schemes for the biharmonic equation, SIAM J. Numer. Anal. 12 (1975) 773–790.
[15] M.M. Gupta and R. Manohar, Direct solution of biharmonic equation using noncoupled approach, J. Comput. Phys. 33 (1979) 236–248.
[16] Y. Kwon, R. Manohar and J.W. Stephenson, Single cell fourth order methods for the biharmonic equation, Congressus Numerantium 34 (1982) 475–482.
[17] G. Meurant, *Computer Solution of Large Linear Systems* (North-Holland, 1999).
[18] J. Ribeiro Dos Santos, Équations aux différences finies pour l'équation biharmonique dans l'espace à trois dimensions, C.R. Acad. Sci. Paris Sér. A–B 264 (1967) A291–A293.
[19] Y. Saad, *Iterative Methods for Sparse Linear Systems* (PWS Publishing Company, 1996).
[20] D.J. Silvester and M.D. Mihajlovic, Efficient preconditioning of the biharmonic equation, Report 362, University of Manchester (2000).
[21] J.W. Stephenson, Single cell discretizations of order two and four for biharmonic problems, J. Comput. Phys. 55 (1984) 65–80.
[22] H. Van der Vorst, BiCGSTAB: a fast and smoothly converging variant of BiCG for the solution of nonsymmetric linear systems, SIAM J. Sci. Comput. 13 (1992) 631–644.