ON MULTIGRID FOR OVERLAPPING GRIDS

WILLIAM D. HENSHAW*

Abstract. The solution of elliptic partial differential equations on composite overlapping grids using multigrid is discussed. An approach is described that provides a fast and memory efficient scheme for the solution of boundary value problems in complex geometries. The key aspects of the new scheme are an automatic coarse grid generation algorithm, an adaptive smoothing technique for adjusting residuals on different component grids, and the use of local smoothing near interpolation boundaries. Other important features include optimizations for Cartesian component grids, the use of over-relaxed Red-Black smoothers and the generation of coarse grid operators through Galerkin averaging. Numerical results in two and three dimensions show that very good multigrid convergence rates can be obtained for both Dirichlet and Neumann/mixed boundary conditions. A comparison to Krylov based solvers shows that the multigrid solver can be much faster and require significantly less memory.

Key words. overlapping grids, multigrid, elliptic PDE

AMS subject classifications. 65N20

1. Introduction. The multigrid method is an effective way for efficiently solving a wide class of partial differential equation (PDE) boundary value problems. Composite overlapping grids are an effective approach for building a collection of structured grids on a complicated domain. The combination of multigrid with overlapping grids provides an efficient approach for solving PDE boundary value problems on complicated domains. One of the benefits of using overlapping grids is that efficient geometric multigrid algorithms can be applied. Geometric multigrid relies on the ability to explicitly generate grid coarsenings in contrast to algebraic multigrid which builds coarsenings with no reference to a grid. Although each component grid in an overlapping grid is a logically rectangular grid and can thus be easily coarsened, the collection of coarser grids must be connected with interpolation to achieve good multigrid convergence rates. Due to the difficulty in connecting the coarse level grids through interpolation most if not all other researchers have left the coarse grids uncoupled, applying a zero Dirichlet or Neumann type boundary condition at interpolation points [26, 27, 13, 32, 14]. If the coarse grids are not connected with interpolation, one will in general experience a degradation in the convergence rate since the decoupled coarse grids cannot represent some of the low frequency components of the error. In this paper a new approach is presented for automatically generating the coarse level overlapping grids. By relaxing the requirements for interpolation between component grids, and allowing the grids to grow larger as they are coarsened, very coarse grids can be generated. The algorithm is both fast and robust. This is important for the case of moving grid problems where the coarse grids need to be regenerated each time a component grid moves. The equations on the coarse grids can be automatically generated by averaging the fine grid equations. This so-called Galerkin coarse grid operator [25] has the benefits of making the solver more automatic, is fast to compute, and usually improves the convergence rates.

A crucial aspect of any multigrid solver is the design of a good smoother. An adaptive smoothing algorithm for overlapping grids is presented that significantly

*Centre for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA 94551, henshaw1@llnl.gov. This research was supported under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

improves convergence rates. The adaptive method uses different smoothers on each component grid and adjusts the number of sub-smooths per grid in order to keep the residuals on the different grids approximately the same size. The addition of local smoothing near interpolation boundaries is also shown to be significant.

The algorithm presented in this paper has been implemented in the Ogmg solver, part of the Overture object-oriented framework [5, 4] for solving PDEs on composite grids [1]. The solver has been optimised for some commonly occurring problems such as equations defined with the Laplace operator. Ogmg is particularly efficient when a majority of the grid points belong to Cartesian component grids; this is often the case when grids become sufficiently fine. For predefined equations on Cartesian grids there is no need to store the grid point locations or coefficients of the operators, resulting in significant savings. This manuscript deals only with the case of second-order accurate discretizations; fourth-order accurate discretizations are considered in a forth-coming paper.

The first overlapping grid computations were apparently performed by Starius who solved elliptic and hyperbolic problems [21, 22]. Since then the method has been widely used to solve a wide variety of problems including aerodynamics [23, 6], combustion, blood flow [16], and Hele-Shaw flow [8]. The multigrid method has been coupled to the overlapping grid method in a variety of works. The first consideration of multigrid for overlapping grids seems to be the work of J. Linden who showed results for a model problem [24]. Henshaw extended the overlapping grid generator of B. Kreiss [17] to generate multigrid levels and used these for ocean flow computations [10]. Chesshire and Henshaw extended the CMPGRD overlapping grid generator [7] to generate multigrid levels for general two-dimensional domains. These grids were used to solve elliptic problems in two dimensions for general domains and showed good multigrid convergence rates [12]. Tu and Fuchs [26, 27] use a multigrid method on overlapping grids for applications to internal-combustion. In their approach, however, the coarse grids are not coupled by interpolation. Johnson and Belk [15] and Jespersen, Pulliam and Buning [14] use multigrid to accelerate the convergence of solutions to the Euler and Navier-Stokes equations on overlapping grids. Hinatsu and Ferziger [13] introduce the notion of *incomplete composite multigrid* (ICMG) and *complete composite multigrid* (CCMG). With ICMG the component grids are connected through overlapping grid interpolation only on the finest level. On coarser levels the grids are decoupled. With CCMG overlapping grid interpolation is performed at all levels, as is done with the algorithm presented here. They find that ICMG gives reasonable results although this conclusion was based on solving problems on fairly coarse grids. Perng and Street [19] use ICMG on "block structured" grids. Zang and Street [32] use an ICMG approach to solve a pressure equation for the incompressible Navier-Stokes equations. The multigrid algorithm for overlapping grids also bears similarities to approaches for adaptive mesh refinement grids [2, 18] and domain decomposition methods [20]. For further details on the multigrid approach one may refer to one of the numerous good books on the subject such as Trottenberg et.al. [25], Briggs et.al. [3], Wesseling [28] and Hackbusch [9].

Here is an outline of the paper. Section (2) establishes notation and gives a brief description of the overlapping grid approach. This is followed by a description of how a PDE boundary value problem can be discretized to second-order accuracy. Section (4) discusses the ways in which the smoothing, restriction and prolongation operators are altered for use with overlapping grids. In section (5) a technique is described

[1] The **Overture** software is available from `http://www.llnl.gov/casc/Overture`

for the generation of coarse multigrid levels which permits the generation of much coarser grids compared to previous approaches. The generation of the Galerkin coarse grid operator is described in section (5). Numerical results, presented in section (6), demonstrate that excellent multigrid convergence rates can be obtained on overlapping grids. In many cases the convergence rates for overlapping grids are almost as good as those for a single rectangular grid. A comparison is also made to some popular Krylov-based solvers.

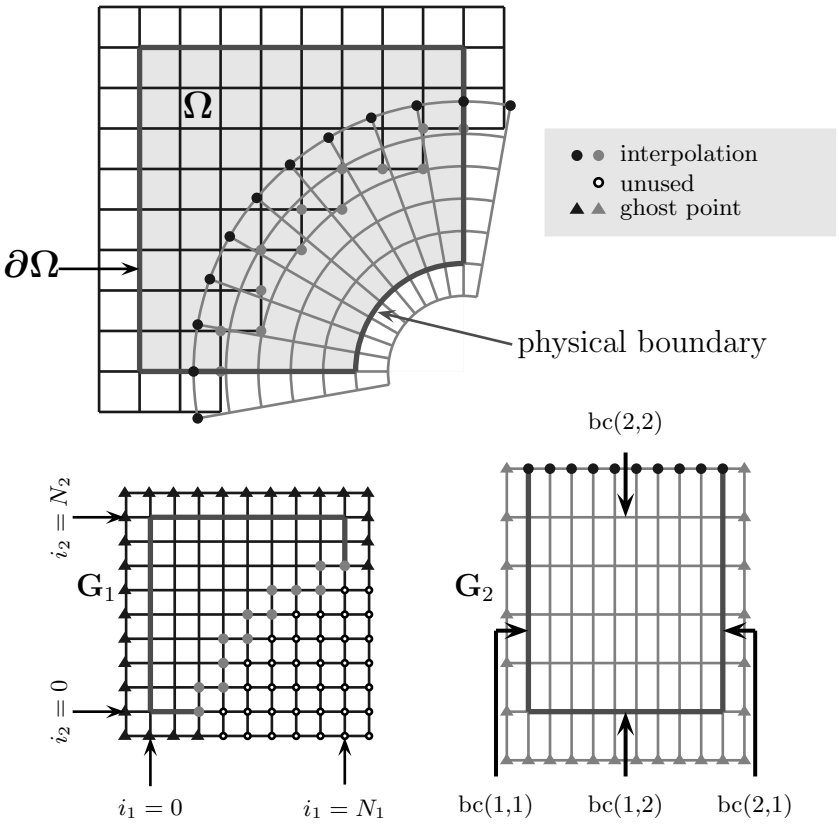


FIG. 2.1. *An overlapping grid consisting of two structured curvilinear component grids. Each component grid is represented by a mapping from the unit square to physical space. Each grid point is classified as either a discretization point, interpolation point or unused point. Ghost points are used to apply boundary conditions.*

2. Composite overlapping grids. This section introduces some of the basic features of a composite overlapping grid, as illustrated in figure (2.1). An *overlapping grid* $\mathcal{G}$ in d space dimensions consists of a set of *component grids*, G_g,

$$\mathcal{G} = \{G_g\}, \qquad g = 1, 2, \ldots, n_g$$

A component grid is a logically-rectangular structured grid. The component grid is defined by a mapping from the unit-square or unit-cube to physical space

$$\mathbf{x} = \mathbf{C}_g(\mathbf{r}), \qquad \mathbf{r} \in [0,1]^d, \quad \mathbf{x} \in \mathbb{R}^d$$

Here $\mathbf{x} = (x_1, x_2, x_3) = (x, y, z)$ and $\mathbf{r} = (r_1, r_2, r_3) = (r, s, t)$ for $d = 3$. Variables defined on a component grid are stored in rectangular arrays. The grid point vertices are

$$\mathbf{x}^g_{\mathbf{i}} \ , \quad \mathbf{i} = (i_1, i_2, i_3) \ , \quad i_\alpha = 0, 1, 2, \ldots, N^g_\alpha$$

where N^g_α is the number of grid points in direction α. The Jacobian derivatives, $\partial x_m / \partial r_n$, are computed directly from the mapping, $\mathbf{C}_g(\mathbf{r})$, and are used when forming discrete approximations. The component grids are usually created with one or more lines of ghost points as shown in figure (2.1). Ghost points are useful for applying boundary conditions. Each face of each grid is classified as a physical, periodic or interpolation boundary. Each point on an overlapping grid is classified as one of discretization, interpolation or unused. Interpolation points may extend out to the ghost points on interpolation boundaries. A list is kept of all the interpolation points, the donor grid from which they interpolate and the location of the interpolation point in the unit square coordinates of the donor grid. In particular, if grid g has $n^g_{\mathcal{I}}$ interpolation points then for each $n = 1, 2, \ldots, n^g_{\mathcal{I}}$, let

$$\begin{aligned}
\mathbf{i} &= \mathrm{ip}^g_n && \text{(interpolation point } n \text{ on grid } g) \\
d &= \mathrm{dg}^g_n && \text{(donor grid for interpolation)} \\
w &= \mathrm{iw}^g_n && \text{(width of the interpolation formula)} \\
\mathbf{r} &= \mathrm{dc}^g_n && \text{(donor grid location, } \mathbf{r} = \mathbf{C}_d^{-1}(\mathbf{x}^g_{\mathbf{i}})) \\
\mathbf{j} &= \mathrm{ds}^g_n && \text{(lower left corner of the donor grid stencil)}
\end{aligned}$$

denote the *interpolation data* associated with the interpolation point. The interpolation formula in two-dimensions is given by standard Lagrange interpolation,

$$U^g_{\mathbf{i}} = \sum_{m_1=0}^{w-1} \sum_{m_2=0}^{w-1} \beta_{\mathbf{m}} U^d_{\mathbf{j}+\mathbf{m}} \ , \quad \beta_{\mathbf{m}} = \mathcal{L}^w_{m_1}(\tilde{r}_1)\, \mathcal{L}^w_{m_2}(\tilde{r}_2) \ , \quad \tilde{r}_\alpha \ = r_\alpha \Delta r_\alpha - j_\alpha \ . \tag{2.1}$$

Here $\mathbf{m} = (m_1, m_2)$, $\Delta r_\alpha = 1/N^g_\alpha$, and the Lagrange polynomials $\mathcal{L}^w_\mu$ are defined in the usual way as $\mathcal{L}^w_\mu(r) = \prod_{j=0, j\neq\mu}^{w-1}(r - j) / \prod_{j=0, j\neq\mu}^{w-1}(\mu - j)$.

3. Discretization on overlapping grids. Consider an elliptic boundary value problem in $d = 2, 3$ space dimensions,

$$\begin{aligned}
Lu &= f && \mathbf{x} \in \Omega \\
Bu &= g && \mathbf{x} \in \partial\Omega
\end{aligned}$$

where L is an elliptic operator, and B the boundary operator. For the purposes of this manuscript, L is chosen to be a second-order, linear, variable-coefficient operator and B is chosen to define a Dirichlet, Neumann or mixed boundary condition. In two space dimensions these take the form

$$Lu := \tilde{c}_{11} u_{xx} + \tilde{c}_{12} u_{xy} + \tilde{c}_{22} u_{yy} + \tilde{c}_1 u_x + \tilde{c}_2 u_y + \tilde{c}_0 u = f \tag{3.1}$$

$$Bu := \tilde{\alpha}_1 \partial_n u + \tilde{\alpha}_0 u = g \tag{3.2}$$

Here $\partial_n u = \mathbf{n} \cdot \mathrm{grad}\, u$ is the normal derivative of u, with $\mathbf{n}$ the unit outward normal to the boundary $\partial\Omega$. The extension to three-dimensions is straight forward.

There are many ways that a discrete approximation to these equations can be defined for which the multigrid algorithm could be applied. For this paper a straightforward discretization based on the well known *mapping method* will be used. Consider a problem on a two-dimensional overlapping grid, $\mathcal{G}$. For each component grid G_g the equations (3.1-3.2) are transformed to the unit square coordinates $\mathbf{r} = (r_1, r_2) = (r, s)$,

$$Lu := c_{11}u_{rr} + c_{12}u_{rs} + c_{22}u_{ss} + c_1u_r + c_2u_s + c_0u = f \ , \tag{3.3}$$
$$Bu = \alpha_1\partial_r u + \alpha_2\partial_s u + \alpha_0 u \ , \tag{3.4}$$

where, for example,

$$c_{11} = \tilde{c}_{11}\ r_x^2 + \tilde{c}_{22}\ r_y^2 \ , \qquad c_1 = \tilde{c}_1\ r_x + \tilde{c}_2\ r_y + \tilde{c}_{11}\ r_{xx} + \tilde{c}_{22}\ r_{yy} \ .$$

The inverse Jacobian derivatives $\partial \mathbf{r}_m/\partial \mathbf{x}_n$ are determined from the mapping $\mathbf{x} = \mathbf{C}_g(\mathbf{r})$. Let $U^g_{\mathbf{i}}$ denote the numerical approximation to the solution on grid G_g, $U^g_{\mathbf{i}} \approx u(\mathbf{x}^g_{\mathbf{i}})$. The equations are discretized to second-order accuracy using standard centred approximations such as

$$\partial_r \approx D_{2r} \equiv D_{0r} \ , \qquad \partial_s \approx D_{2s} \equiv D_{0s} \ ,$$
$$\partial_r^2 \approx D_{2rr} \equiv D_{+r}D_{-r} \ , \qquad \partial_s^2 \approx D_{2ss} \equiv D_{+s}D_{-s} \ , \qquad \partial_r\partial_s \approx D_{2rs} \equiv D_{2r}D_{2s} \ ,$$

where D_{+r}, D_{-r}, and D_{0r} are the forward, backward and central divided difference operators. For example $D_{+r}U_{\mathbf{i}} = (U_{i_1+1,i_2} - U_{\mathbf{i}})/\Delta r_1$, with $\Delta r_1 = 1/N^g_1$. The second-order discrete approximation to (3.3-3.4) is given by

$$L_hU := c_{11}D_{2rr}U_{\mathbf{i}} + c_{12}D_{2rs}U_{\mathbf{i}} + c_{22}D_{2ss}U_{\mathbf{i}} + c_1D_{2r}U_{\mathbf{i}} + c_2D_{2s}U_{\mathbf{i}} + c_0U_{\mathbf{i}} = f_{\mathbf{i}} \tag{3.5}$$
$$B_hU := \alpha_1D_{2r}U_{\mathbf{i}} + \alpha_2D_{2s}U_{\mathbf{i}} + \alpha_0U_{\mathbf{i}} = g_{\mathbf{i}}$$

Consider a problem with a boundary condition at $r_1 = 0$, $i_1 = 0$. On a boundary with a Dirichlet condition, the discrete equations will be of the form

$$\begin{aligned} L_hU_{\mathbf{i}} &= f_{\mathbf{i}} \qquad i_1 = 1, 2, 3, \ldots \\ U_{\mathbf{i}} &= g_{\mathbf{i}} \qquad i_1 = 0 \end{aligned}$$

On a boundary with a Neumann or mixed condition the interior PDE is applied on the boundary and one ghost line is introduced,

$$\begin{aligned} L_hU_{\mathbf{i}} &= f_{\mathbf{i}} \qquad i_1 = 0, 1, 2, 3, \ldots \\ \alpha_1D_{2r}U_{\mathbf{i}} + \alpha_2D_{2s}U_{\mathbf{i}} + \alpha_0U_{\mathbf{i}} &= g_{\mathbf{i}} \qquad i_1 = 0 \end{aligned} \tag{3.6}$$

The Neumann boundary condition is thought to define the value of $U_{\mathbf{i}}$ at $i_1 = -1$.

To give a concrete example, consider the discretization of a one-dimensional Poisson equation,

$$\begin{aligned} u_{xx} &= f \qquad x \in (0, 1) \\ u_x(0) = g_0, \quad u(1) &= g_1 \end{aligned}$$

on the overlapping grid shown in figure (3.1).

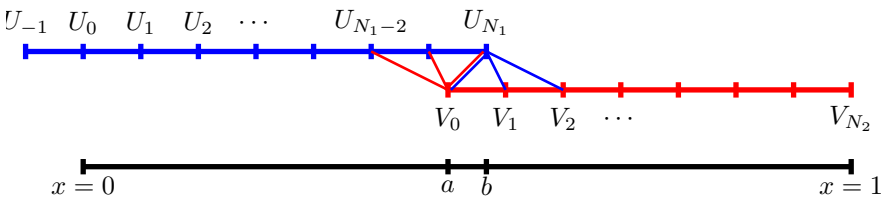


FIG. 3.1. *Overlapping grid in one dimension. The values at the interpolation points, U_{N_1} and V_0 are obtained by three-point interpolation.*

A second-order discretization to this problem is

$$\begin{aligned}
(U_1 - U_{-1})/(2h_1) &= g_0 && \text{(Neumann BC)} \\
(U_{i-1} - 2U_i + U_{i+1})/h_1^2 &= f(x_i^1) && i = 0, 1, 2, \ldots, N_1 - 1 \\
U_{N_1} - \big(\alpha_0 V_0 + \alpha_1 V_1 + \alpha_2 V_2\big) &= 0 && \text{(interpolation)} \\
V_0 - \big(\beta_0 U_{N_1-2} + \beta_1 U_{N_1-1} + \beta_2 U_{N_1}\big) &= 0 && \text{(interpolation)} \\
(V_{i-1} - 2V_i + V_{i+1})/h_2^2 &= f(x_i^2) && i = 1, 2, 3, \ldots, N_2 - 1 \\
V_{N_2} &= g_1 && \text{(Dirichlet BC)}
\end{aligned} \tag{3.7}$$

Here $U_i \approx u(x_i^1)$, $x_i^1 = ih_1$, $h_1 = b/N_1$, denotes the solution on the sub-interval $[0, b]$ and $V_i \approx u(x_i^2)$, $x_i^2 = a + ih_2$, $h_2 = (1-a)/N_2$, denotes the solution on the sub-interval $[a, 1]$. The interpolation weights α_m and β_n are chosen appropriately, see equation (2.1). The value at the ghost point, U_{-1}, is usually considered to be defined by the Neumann boundary condition, specifically $U_{-1} = U_{+1} - 2h_1\ g_0$. Further discussion of the use of ghost points can be found, for example, in Trottenberg et.al.[25].

4. The multigrid algorithm for overlapping grids. This section discusses the multigrid algorithm as applied to overlapping grids. The equations defining a discretization of an elliptic boundary value problem on an overlapping grid, $\mathcal{G}_h$, discussed in the previous section, can be written in the form

$$\begin{aligned}
L_h u_h &= f_h && \mathbf{x_i} \in \Omega_h && \text{(interior equations)} \\
B_h u_h &= g_h && \mathbf{x_i} \in \Gamma_h && \text{(boundary equations)} \\
\mathcal{I}_h u_h &= 0 && \mathbf{x_i} \in \Gamma_h^I && \text{(interpolation equations from (2.1))}
\end{aligned}$$

These *fine grid equations* are written as a single matrix equation,

$$A_h u_h = b_h\ . \tag{4.1}$$

Assume that is possible to define a coarse grid, $\mathcal{G}_H$ defined on a discrete domain Ω_H, with $H = 2h$. The *coarse grid equations*,

$$\begin{aligned}
L_H u_H &= f_H && \mathbf{x_i} \in \Omega_H \\
B_H u_H &= g_H && \mathbf{x_i} \in \Gamma_H \\
\mathcal{I}_H u_H &= 0 && \mathbf{x_i} \in \Gamma_H^I
\end{aligned}$$

are written as

$$A_H u_H = b_H\ .$$

The equations appearing in the matrix A_h or A_H are not scaled by any factors of the grid spacing but are left in the form (2.1,3.5,3.6). The scaling of the equations could be important when residuals are averaged to a coarse grid. However, in practice the boundary conditions and interpolation conditions are treated as constraints so that residuals in these equations will generally be zero. In addition, the residuals from the interior equations are not averaged with the residuals from the boundary conditions or interpolation equations. Therefore the relative scaling between interior, boundary and interpolation equations, as they appear in the matrix, is not critical.

The fundamental structure of the multigrid algorithm for overlapping grids remains the same as for a single grid. Introduce the following operators

$\mathbf{S}_h$: the composite smoothing operator, an iteration that is effective at reducing the high frequency components of the error to the fine grid equations (4.1).

$\mathbf{I}_h^H$: restriction operator, the operator that transfers a grid function from the fine grid to the coarse grid.

$\mathbf{I}_H^h$: prolongation operator, the operator that transfers a grid function from the coarse grid to the fine grid.

Although the operators $\mathbf{I}_h^H$ and $\mathbf{I}_H^h$ represent a form of *interpolation*, in this manuscript the term interpolation will always refer to the updating of the overlapping grid interpolation points (2.1), unless explicitly stated otherwise. The standard defect correction multigrid procedure is given as algorithm 1.

ALGORITHM 1 (The defect correction multigrid algorithm).

procedure multigrid(u_h, f_h)
{
  while *not converged* **do**
    $v_h \leftarrow \mathbf{S}_h^{\nu_1} v_h$ *(smooth ν_1 times)*
    $f_H \leftarrow \mathbf{I}_h^H (b_h - A_h v_h)$ *(transfer defect to coarser grid)*
    $A_H v_H \approx f_H$ *("solve" the defect equation)*
    $v_h \leftarrow v_h + \mathbf{I}_H^h v_H$ *(add correction)*
    $v_h \leftarrow \mathbf{S}_h^{\nu_2} v_h$ *(smooth ν_2 times)*
  end while
}

The coarse grid equations can be approximately solved in a recursive manner by using an even coarser grid. On the very coarsest grid the equations are solved with a sparse matrix solver using either an iterative or direct method. Algorithm 1 is only appropriate for linear problems. For nonlinear problems the above algorithm could be used, for example, to solve the linearized problems resulting from a Newton iteration. Alternatively the full approximation scheme, FAS, could be used. With the exception of the use of the Galerkin coarse grid operator, the extension of the present scheme to nonlinear problems should be straight-forward.

An overlapping grid for some "shapes" and the corresponding multigrid levels are shown in figure (5.3). The component grids on the finest level are not arranged in any particular hierarchy although each grid is given a *priority*. The priority is used when the overlapping grid is first generated; higher priority grids will tend to retain grid points in regions of overlap with lower priority grids. Background Cartesian grids are usually given a low priority, for example. Each component grid is coarsened by a factor of two in each coordinate direction to form the grid on the next coarser level. Each grid belongs to one and only one level and there is a unique coarse grid for each component grid. The coarse grids are generated so that the restriction and prolongation operators can be efficiently implemented. For example, a standard full weighting restriction operator can be applied at all coarse grid points where the

interior equation is applied. More generally one could allow for coarsening factors other than two. In particular, allowing a coarsening factor of one could be useful for component grids with relatively few grid points as this would permit a greater number of multigrid levels to built.

In the following sections a description will be given of the operators $\mathbf{S}$, $\mathbf{I}_h^H$, $\mathbf{I}_H^h$ and the coarse grid equations A_H, as they are defined for overlapping grids.

4.1. Composite smoothing operator. Application of the operator $\mathbf{S}_h$ represents a *composite-smooth* where each component grid in turn is smoothed. The composite smoother should be designed so that after smoothing the resulting defect, $d_h = b_h - A_h v_h$, is smooth on the entire overlapping grid. Since not all grids will be smoothed to the same degree, each component grid is permitted to have a variable number of sub-smooth steps. Let ν_g denote the number of sub-smooths for component grid G_g. The pseudo-code given in algorithm 2 outlines the composite-smooth.

ALGORITHM 2 (The composite smoothing algorithm).

```
procedure compositeSmooth(u)
{
    determineSubSmooths(ν_g)          (given by algorithm 3)
    for  g = 1,...,n_g do             (loop over component grids)
        if g > 1   interpolate(u_g)   (interpolate grid g, equation (2.1))
        for m = 1,...,ν_g             (multiple sub-smooths)
            u_g ← smooth(u_g)         (component grid smoother)
            apply boundary conditions to u_g
        end for
    end for
    interpolate(u)                    (update interp. points, equation (2.1))
    interpolationBoundarySmoother(u)  (extra smoothing near interp. points)
}
```

It is found that the best results are usually obtained when the very latest values for interpolation points are used when smoothing a grid. For Neumann or mixed boundary conditions it is important to apply the boundary conditions after each sub-smooth so that the grid function maintains the proper symmetries at the boundary. This will be discussed more in section (4.8).

4.2. Variable sub-smooths. A variety of component grid smoothers have been implemented in the Ogmg multigrid solver including Jacobi, Gauss-Seidel, Red-Black, and line smoothers. In addition, a variety of Krylov based algorithms can be used for smoothers. Over-relaxed Red-Black smoothers as suggested by Yavneh [31] have been found to be very effective for Cartesian grids and curvilinear grids with mild stretching. The type of smoother and the number of sub-smooths may vary from component grid to component grid. The number of sub-smooths, ν_g, on each component grid G_g is chosen to keep the discrete L_2-norms of the residuals on the different grids roughly the same size. The discrete L_2-norm is defined by $\|u\|_h^2 = N^{-1} \sum_{\mathbf{i}} |u_{\mathbf{i}}|^2$ where the sum is taken over all valid points on the grid and N is the number of terms in the sum.

Before each composite smooth, ν_g may be increased or decreased by one depending on the relative size of the residual on grid G_g compared to that of a reference grid $G_{g_{\rm ref}}$. The number of sub-smooths on the reference grid is fixed (usually to be one). Although the reference grid might be chosen as the one with minimum residual; a better selection seems to be the component grid with the largest number of grid points. This latter choice avoids the unwanted situation whereby a grid with a small number of points and with a small residual forces a large number of sub-smooths

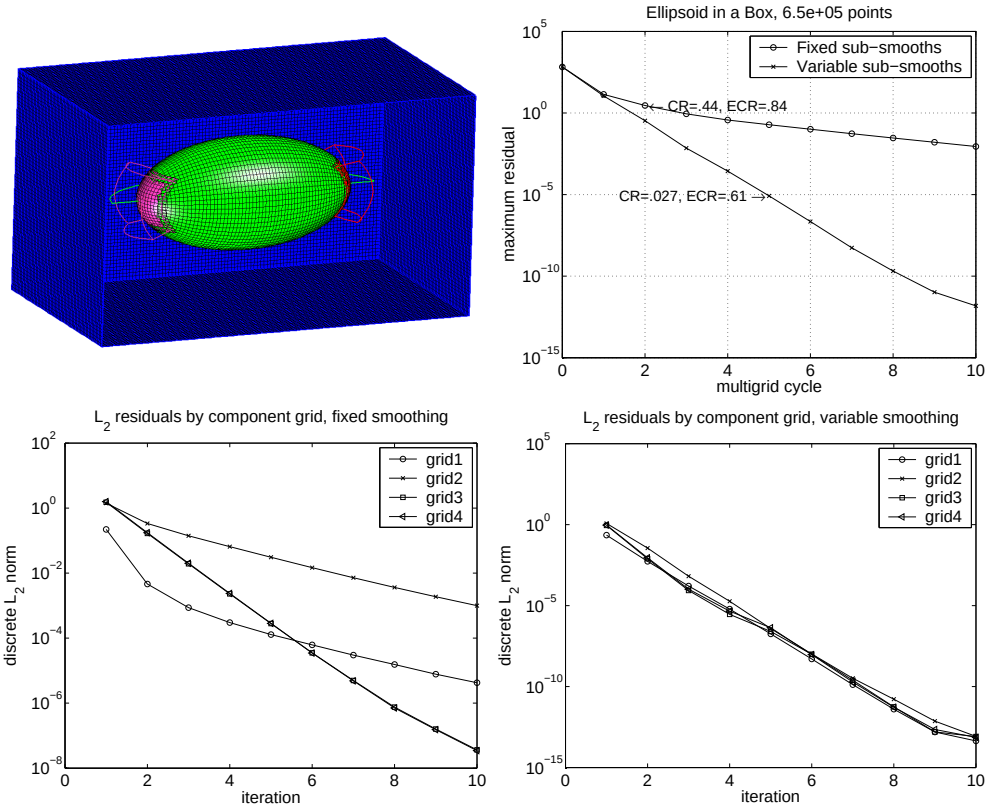


FIG. 4.1. *The convergence rate is improved when a variable number of sub-smooths is used on each component grid. The number of sub-smooths is chosen to keep the discrete L_2-norm of the component grid residuals nearly the same. Results are shown for an ellipsoid in a box with a V[2,1] cycle.*

to be taken on a grid with a large number of points. The procedure for choosing ν_g is given by algorithm 3. The number of sub-smooths, ν_g, is increased by one if the residual on the grid is a factor $(\sigma_+)^{1/\nu_g}$ larger than the residual on the reference grid. The number of sub-smooths is decreased by one if the residual ratio is less than $(\sigma_-)^{1/\nu_g}$. The number of sub-smooths is usually restricted to be greater than zero and less than some predefined maximum value. Comparing the residual ratio to $(\sigma_+)^{1/\nu_g}$ instead of σ_+ allows larger values of ν_g to change more easily. Through numerical experimentation the values of $(\sigma_-, \sigma_+) = (\frac{1}{2}, 2)$ seem to give reasonable results. For efficiency, the discrete L_2-norms of the defects, d_g, are computed approximately using a sub-set of the total number of points.

ALGORITHM 3 (Determine the number of sub-smooths per component grid).

procedure **determineSubSmooths**(ν_g)

{

$\nu_{\max} \leftarrow$ *maximum number of sub-iterations allowed (e.g. 10).*

$d_g \leftarrow \|A_g u_g - b_g\|_h \ , \quad g = 1, \ldots, n_g$ *(discrete L_2-norm of the residual on G_g)*

g_{ref} *: $G_{g_{\text{ref}}}$ is the component grid with the most grid points.*

$d_{\text{ref}} \leftarrow d_{g_{\text{ref}}}$ *(residual on the reference grid)*

for $g = 1, \ldots, n_g$ **do**

if $d_g/d_{\text{ref}} < (\sigma_-)^{1/\nu_g}$

$\nu_g \leftarrow \max(1, \nu_g - 1)$ *(decrease sub-smooths)*

```
        else if  d_g/d_ref > (σ_+)^{1/ν_g}
            ν_g ← min(ν_max, ν_g + 1)        (increase sub-smooths)
        end if
    end for
}
```

Figure (4.1) presents some convergence results for solving a three-dimensional Poisson equation on a domain exterior to an ellipsoid and interior to a box. Convergence rates are shown with and without the variable smoothing algorithm. A comparison of the residuals on the different component grids shows that when the number of sub-smooths is fixed the residual becomes much larger on one grid than the others (the residual is largest on the large curvilinear grid that covers most of the ellipsoid), see figure (4.1). However, when the number of sub-smooths is allowed to vary, the residuals on the different component grids are nearly the same size. The convergence rate, CR, and effective convergence rate, ECR, as defined in section (6), are both significantly better with a variable number of sub-smooths. With fixed sub-smooths, a maximum residual of 6.2e-10 was achieved using 40 cycles and 26.8 CPU seconds. With variable sub-smooths, a maximum residual of 2.1e-10 was achieved using 8 cycles and 9.0 CPU seconds; a speedup of almost a factor of 3.

4.3. Interpolation boundary smoothing (IBS). During the multigrid iteration the defect can sometimes become large in a narrow region next to the interpolation points. This can cause a degradation of the convergence rate. The source of the problem is illustrated in figure (4.2) which shows three sub-steps in the composite smoothing algorithm applied to a one-dimensional overlapping grid function. As shown in the figure, the solution may lose smoothness when the interpolation points are updated. When there is a small overlap, say some fraction of a mesh width, the loss of smoothness is usually minor. However, when the overlap is larger, which may occur when the grid spacings on adjacent grids are not commensurate, a large defect can form near the interpolation points and the convergence rate can slow. To

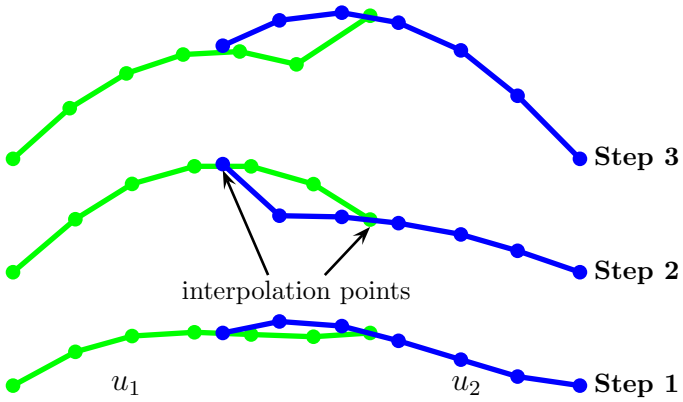


FIG. 4.2. *The solution at different steps in the composite smooth. Step 1: initial guess at the solution. Step 2: u_1 is smoothed and the left endpoint of u_2 is interpolated. Step 3: u_2 is smoothed and the right endpoint of u_1 is interpolated. The solution can be non-smooth near the interpolation points, especially when the overlap is large.*

remedy this problem, an additional interpolation-boundary-smoothing (IBS) step is applied as the last stage of the composite-smooth. The IBS procedure is outlined in algorithm 4. Extra smoothing is performed at discretization points that lie near inter-

polation points. A Gauss-Seidel smoother with $\omega = 1$ is usually used. For efficiency, the list of the interpolation neighbours that require smoothing is pre-computed.

ALGORITHM 4 (Smoothing the solution near interpolation boundaries).

procedure **IBS: interpolationBoundarySmoother***(u)*
{
  for $\mu = 1, \ldots, n_I$ **do** *(global smoothing iterations)*
    for $g = 1, \ldots, n_g$ **do** *(n_g component grids)*
      for $i = 1, \ldots, m_I$ **do** *(smoothing points on grid G_g)*
        for *each discretization point* **i** *on grid g that lies within w_I grid points of an interpolation point of grid g.* **do**
          $u_{\mathbf{i}} \leftarrow$ **Smooth***($u_{\mathbf{i}}$)*
        end for
      end for
    end for
    interpolate*(**u**)* *(update all interpolation points)*
  end for
}

Algorithm 4 contains three parameters, the number of global smoothing iterations, n_I, the number of local smoothing iterations, m_I and the number of layers of grid points, near interpolation points, to be smoothed, w_I. Typical choices for the parameters are $n_I = 2$, $m_I = 2$, and $w_I = 4$. These values would mean that for each of $n_I = 2$ global iterations, $w_I = 4$ layers of discretization points near interpolation points are smoothed, with $m_I = 2$ sub-iterations of Gauss-Seidel. Figure (4.3) shows an example where smoothing near the interpolation boundaries improves the convergence rate. With no IBS smoothing, a maximum residual of $2.1e$-9 was reached using 9 cycles and 14.9 CPU seconds. With IBS smoothing, a maximum residual of $2.1e$-9 was achieved in 7 cycles and 12.7 CPU seconds.

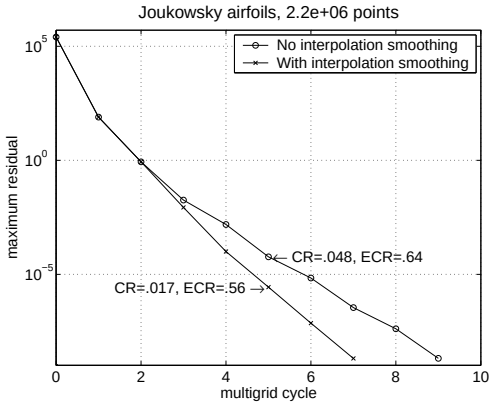


FIG. 4.3. *A comparison showing the benefit of using interpolation boundary smoothing (IBS) for two Joukowsky airfoils in a channel, W[2,1].*

4.4. Over-relaxed Red-Black Smoothers. The Red-Black Gauss-Seidel smoother, RB-GS, is an excellent smoother for Poisson's equation on Cartesian grids. The local smoothing factor on a Cartesian grid with equal grid spacing in each direction is $\frac{1}{4}$ in two-dimensions and $\frac{4}{9}$ in three-dimensions. The smoothing factor, defined for example in [25], measures the reduction in the rough modes of the error per smooth. The smoothing rate of Red-Black Gauss-Seidel smoothers with a relaxation parameter, $\omega-$RB-GS, can be improved through the appropriate choice of ω, as described in

Yavneh [31]. For example, the smoothing factor in three-dimensions improves from $\mu \approx .444$ for $\omega = 1$ to $\mu \approx .23$ for $\omega = 1.13$. In two-dimensions the smoothing factor improves from $\mu = .25$ for $\omega = 1$ to a value of $\mu \approx .16$ for $\omega = 1.05$. Following the suggestion in[31], the value for ω is allowed to vary on curvilinear grids, based on the relative local sizes of the coefficients in the operator. The resulting smoother works well on curvilinear grids provided the aspect ratios of the cells do not become too large. Over-relaxation can also be used for zebra-line smoothers in three-dimensions.

4.5. Fine to coarse grid restriction operator. The restriction operator $\mathbf{I}_h^H$ is used to transfer the defect on the fine grid $\mathcal{G}_h$ to the coarse grid $\mathcal{G}_H$. For example, a commonly used restriction is the **full-weighting** operator which in two-dimensions is

$$\begin{aligned} d^H_{\mathbf{i}} &= \frac{1}{4} d^h_{2\mathbf{i}} + \frac{1}{8}(d^h_{2i_1+1,2i_2} + d^h_{2i_1-1,2i_2} + d^h_{2i_1,2i_2+1} + d^h_{2i_1,2i_2-1}) \\ &\quad + \frac{1}{16}(d^h_{2i_1+1,2i_2+1} + d^h_{2i_1-1,2i_2+1} + d^h_{2i_1+1,2i_2-1} + d^h_{2i_1-1,2i_2-1}) \end{aligned}$$

(For ease of notation, in this section the fine and coarse grid defects will be denoted by d^h and d^H instead of d_h and d_H.) Some care is required when transferring the defect near physical boundaries and interpolation boundaries. The basic philosophy for averaging the defect from a fine grid to a coarse grid is to only average defects that arise from equations of the same type; see, for example, the discussion in [25](section 5.6.1). Thus defects from the interior equations should not be averaged with defects from the equations that define the boundary conditions or defects from the interpolation equations. In general the defects for the boundary conditions should be transfered separately, using a lower dimensional restriction operator that only operates on the defects in the boundary conditions. However, this step is not needed if the defects in the boundary condition equations are zero, as is the case here.

The restriction procedure for overlapping grids is given in algorithm 5. The first step in the algorithm is to assign values at the interpolation points of the fine grid defect, d^h. The reason for this step is as follows. A valid defect for the interior PDE can be computed at all discretization points but not at interpolation points. In some cases a discretization point on the coarse grid will, using the full-weighting operator, require a defect to be defined at an interpolation point. For example, in the one-dimensional Poisson problem, equation (3.7), a defect in the interior equations may be needed at the interpolation points $x^1_{N_1}$ and x^2_0. Rather than define a special restriction operator for some points, values are assigned to the interpolation points of d^h, These values should approximate the defect in the interior equations. It has been found that interpolating the defect from neighbouring grids gives, in general, better results than using a one-sided approximation.

The next step in algorithm 5 is to loop over the grids and assign values of the defect on the ghost points of boundaries where a Neumann (or mixed) boundary condition is assigned. The value of d^h on the ghost point is set equal to the value of d^h on the first point inside, for example $d^h_{-1,i_2} = d^h_{1,i_2}$ for a boundary where $i_1 = 0$. When the full weighting operator is subsequently applied to a point on the boundary, the result will be equivalent to using a modified full-weighting operator as discussed, for example, in [25] (section 5.6.2). After updating the values on periodic boundaries of d_h, the coarse grid defect d_H is assigned using the full-weighting operator. This operator is applied at all points where the interior equation is applied; this includes boundary points on Neumann boundaries. The value of d^H is zero at interpolation points, boundary points of Dirichlet boundaries and ghost points of Neumann boundaries.

ALGORITHM 5 (Restrict the defect to the coarse grid).
procedure $\mathbf{I}_h^H$*:* **restriction** *(*d^h*,*d^H*)*
{
  interpolate*(* d^h *)* *(assign defect interp. points, equation (2.1))*
  for $g = 1, \ldots, n_g$ **do**
    for *each Neumann/mixed boundary*
      $d^{h,g}$*(ghost-line)* $\leftarrow$ $d^{h,g}$*(first line in)* *(symmetry condition)*
    end for
    Update periodic boundaries of $d^{h,g}$*.*
    foreach *point of* $\mathcal{G}_H$ *where the interior equation is applied*
      $d^{H,g} \leftarrow$ **fullWeighting***(*$d^{h,g}$*)* *(interior and Neumann bndry pts)*
    else
      $d^{H,g} \leftarrow 0$ *(interpolation, Dirichlet boundary and Neumann ghost pts)*
    end foreach
  end for
}

4.6. The coarse to fine prolongation operator. The prolongation operator $\mathbf{I}_H^h$ adds the coarse grid correction, u_H, to the current fine grid solution, u_h. It is straight-forward to define the composite prolongation operator for overlapping grids. The standard prolongation operator (such as linear interpolation) is applied to the solution on each component grid. The boundary conditions are applied and the overlapping grid interpolation points are updated to be consistent with the corrected solution.

4.7. Coarse grid operators. The operator on the coarse grid can be constructed using the same discrete approximation as on the fine grid, such as equation (3.5). Another way is to use the *Galerkin coarse grid operator* defined as

$$A_H := \mathbf{I}_h^H A_h \mathbf{I}_H^h \tag{4.2}$$

where $\mathbf{I}_h^H$ and $\mathbf{I}_H^h$ are the restriction and prolongation operators. There are a number of advantages to using equation (4.2):

1. The coarse grid operator can be computed with little knowledge of the original PDE allowing the multigrid solver to act in the manner of a black box solver.
2. Use of the Galerkin operator can lead to improved convergence rates as shown later in section (6). Although the Galerkin operator may be somewhat more costly to evaluate (a 5 point stencil for the Laplace operator in 2D becomes a 9 point operator) this extra cost appears to be generally insignificant on modern cache-based machines.
3. The Galerkin coarse grid operator is useful for problems with discontinuous coefficients.
4. Even for problems with smooth coefficients, but where the geometry contains fine scale features, the Galerkin operator will effectively smooth out the geometry on coarser grids. To see this, note that the coefficients of the elliptic operator depend on the geometry through the Jacobian derivatives of the mapping. These coefficients are averaged when the coarse grid operator is formed.

Formally the Galerkin coarse grid operator (4.2) is constructed by multiplying together the matrices defined by the restriction operator, the fine grid operator and the prolongation operator. In practice the coefficients of the Galerkin coarse grid operator can be efficiently computed by taking the appropriate weighted averages of the

coefficients of the fine grid operator. Note, however, that equations corresponding to boundary conditions should not be averaged with equations coming from the PDE. Similarly, on an overlapping grid, the interpolation equations should not be averaged with the neighbouring equations for the PDE. In practice, the interpolation equations are not directly added to the discrete operator; a discrete approximation to the PDE is inserted in their place to facilitate the averaging process. The interpolation equations are not averaged in any way on the coarse grid operator.

4.8. Neumann and mixed boundary conditions. It is well known that care is required in the treatment of Neumann or mixed boundary conditions to avoid a degradation in the convergence rate. As indicated earlier, on a boundary with a Neumann or mixed boundary condition, both the interior PDE and the boundary condition are applied discretely on the boundary and one ghost line is introduced, see equation (3.6) or equation (3.7).

It is common practice to eliminate the values at the ghost points by combining the interior equation and the boundary condition, see for example the discussion in Wesseling [28]. Even if the ghost points are not eliminated, as is the case here, the same effect can be achieved by treating the boundary condition as a constraint that should always be satisfied. For example, in the one-dimensional problem, equation (3.7), U_{-1} should be set equal $U_1 - 2h_1 g_0$ before it is used in a smoothing step. Convergence results can sometimes be quite sensitive to the enforcement of the Neumann boundary condition. Figure (4.4) compares the convergence rates when the Neumann boundary conditions are applied after both Red and Black sub-steps of a RB-GS smoother (CR=.011), to only applying the boundary conditions once at the end of each Red-Black smooth (CR=.038).

Some insight into this phenomena can be gained by considering the multigrid algorithm applied to the model problem of Poisson's equation on the unit square; only a brief discussion will be given here. Standard Fourier analysis for this model problem generally relies on properties of the discrete eigenfunctions [9, 28, 25]. When using an $\omega-$Jacobi smoother and Dirichlet boundary conditions, the eigenfunctions of both the discrete operator and the smoother are sines while for Neumann boundary conditions they are cosines. The multigrid convergence rates for the Dirichlet and Neumann problems are essentially the same. However, if the Neumann boundary condition is not treated as a constraint, the discrete eigenfunctions of the smoother may become a linear combination of both sines and cosines. The Neumann problem no-longer behaves in the same way as the Dirichlet case. The eigenfunctions will no longer have an even symmetry property at the boundary and high-frequency eigenfunctions may be significantly changed. Whether the convergence rate is affected depends, of course, on the detailed interactions between the smoother, prolongation, restriction and coarse grid operators.

As a general principle, it seems wise to treat Neumann and mixed boundary conditions as constraints. Furthermore, use of the symmetry properties of the eigenfunctions, leads one to choose appropriate restriction operators at Neumann boundaries (as described in section 4.5) and to choose appropriate conditions for corner ghost points as described in the next section.

4.8.1. Boundary conditions for corner and edge ghost points. On a two dimensional grid it is necessary to specify boundary conditions to determine values of the solution at the ghost points in the four corners, $u_{-1,-1}$, $u_{N_1+1,-1}$, u_{-1,N_2+1} and u_{N_1+1,N_2+1}. On a three dimensional grid values are required for the ghost points along the twelve edges (such as $u_{i_1,-1,-1}$, u_{-1,i_2,N_2+1}, u_{N_1+1,N_2+1,i_3} etc.) and the eight

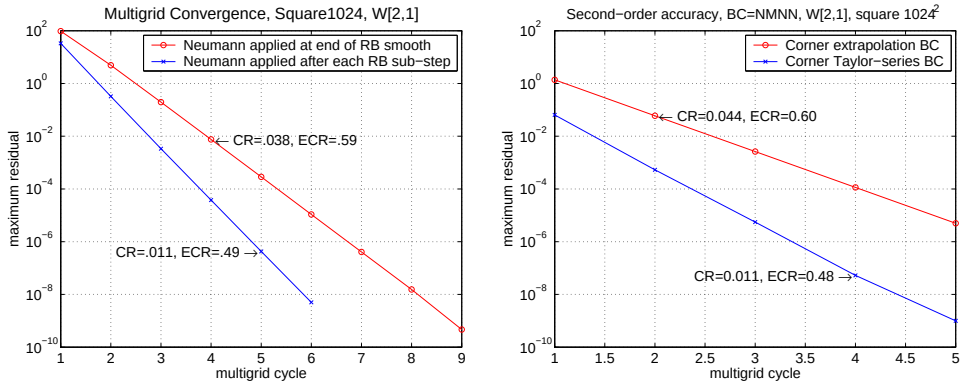


FIG. 4.4. *The proper assignment of ghost points is important for Neumann and mixed boundary conditions. Left: the convergence rate degrades if the ghost points are only updated at the end of the Red-Black smooth as compared to updating after the both the Red and Black sub-steps. Right: A boundary condition for the corner ghost points that preserves even symmetry performs better than extrapolation. The corner ghost points appear in the Galerkin coarse grid approximation for the Laplace operator. Results are shown for a W[2,1] cycle for a* 1024^2 *square with three Neumann and one mixed boundary condition.*

corners (such as $u_{-1,-1,-1}$, $u_{-1,-1,N_2+1}$, $u_{N_1+1,N_2+1,-1}$ etc.) For Neumann and mixed boundary conditions the value of the solution at the corner ghost point are chosen with an approximation that will be both accurate and preserve an even symmetry condition if the solution has this symmetry. A straightforward use of extrapolation, for example, can lead to a significant reduction in the convergence rate as shown in figure (4.4).

The following approximation derived from Taylor series seems to work well and leads to convergence rates for Neumann and mixed boundary conditions that are almost the same as the Dirichlet case. The equations will be derived for the corner point $u(-\Delta r, -\Delta s)$, expressions for the other corners follow easily. Adding the Taylor series approximation for $u(\Delta r, \Delta s)$ to that for $u(-\Delta r, -\Delta s)$ gives

$$u(-\Delta r, -\Delta s) = u(\Delta r, \Delta s) - 2\Delta r\ u_r - 2\Delta s\ u_s + O(\max(\Delta r, \Delta s)^3)\ .$$

Using the approximations $u_r \approx D_{0r}U_{0,0}$ and $u_s \approx D_{0s}U_{0,0}$ results in the following (third order accurate) expression to be used at corners,

$$U_{-1,-1} = U_{1,1} - (U_{1,0} - U_{-1,0}) - (U_{0,1} - U_{0,-1}) \tag{4.3}$$

This *even-symmetry Taylor boundary condition* (4.3) will reduce to an even symmetry boundary condition if the neighbouring points also satisfy the symmetry condition. Note that an odd-symmetry boundary condition can also be derived by subtracting the Taylor series $u(\Delta r, \Delta s)$ and $u(-\Delta r, -\Delta s)$. Equation (4.3) can immediately be extended for use at edges in three dimensions. For a corner in three dimensions one can follow the above argument to arrive at the even-symmetry Taylor-series boundary condition

$$U_{-1,-1,-1} = U_{1,1,1} - (U_{1,0,0} - U_{-1,0,0}) - (U_{0,1,0} - U_{0,-1,0}) - (U_{0,0,1} - U_{0,0,-1}) \tag{4.4}$$

5. Coarse grid generation. This section describes a new algorithm for generating the multigrid levels for an overlapping grid. The overlapping grid for the finest

level, $\mathcal{G}_h$, can be constructed with the Ogen grid generator[11]. In previous work [12], the overlapping grid generator was also used to build the coarser levels. Usually, however, only a few multigrid levels could be computed before there was insufficient overlap to permit the creation of a valid grid. The new algorithm does not use the general overlapping grid generation algorithm to build the coarse levels. Instead, by using the information contained in the existing valid overlapping grid, $\mathcal{G}_h$, the coarse level overlapping grids and interpolation points can be generated in a faster, simpler and more robust manner. Furthermore, since only approximate solutions of the coarse grid equations are needed, the accuracy requirements on the interpolation can be relaxed as the grids are coarsened; this will allow an overlapping grid to be constructed which would not be considered valid by the general algorithm.

The problem of coarsening an overlapping grid can be characterized as follows. Given an overlapping grid $\mathcal{G}_h$ with corresponding component grids $\{G_{h,g}\}_1^{n_g}$, classification mask $\text{mask}_{\mathbf{i}}^{h,g}$ (that classifies each grid point as one of discretization, interpolation or unused) and interpolation information (see section (2)) the aim is to determine a valid coarse overlapping grid $\mathcal{G}_H$ with component grids $\{G_{H,g}\}_1^{n_g}$ containing half as many points in each direction, a classification mask $\text{mask}_{\mathbf{i}}^{H,g}$ and interpolation information. To be valid the coarse grid should be classified so that each discretization point is surrounded by either discretization points or interpolation points. An example of the mask array for fine and coarse grids is shown in figures 5.1 and 5.2. Of course it is always possible to start from a valid coarse grid and then refine it to get as many multigrid levels as desired. Unfortunately this easier approach is, in general, neither convenient nor practical.

When an overlapping grid is generated, interpolation points are determined so that the overlap distance greater than but nearly equal to a constant times the local mesh spacing, $\alpha h(\mathbf{x})$. Here $h(\mathbf{x})$ is an approximate spacing between grid points. The factor α is called the *effective overlap*. When the discrete stencil has a width of 3 points in each direction the minimum overlap is usually chosen so that $\alpha \geq \frac{1}{2}$ (the equations become singular when $\alpha \to 0$). For *explicit interpolation* all points in the interpolation stencil are discretization points and the effective overlap will be $\alpha \geq \frac{3}{2}$. For *implicit interpolation* some points in the interpolation stencil may be themselves interpolation points and then $\alpha \geq \frac{1}{2}$. The key ingredients to the coarsening algorithm are as follows:

1. Use the overlapping information and interpolation information contained in the fine grid to help determine the properties of the coarse grid. Thus the topology of the coarse region, such as location of holes and boundaries, is given by the fine grid and does not need to be computed.
2. Relax the accuracy and explicitness of interpolation on coarse grids. As the grids are coarsened,
   (a) allow explicit interpolation to become implicit since implicit interpolation requires less overlap.
   (b) allow the width of the interpolation stencil to decrease. Thus each interpolation point may have a possibly different interpolation width.
   (c) allow a coarse grid interpolation point that has extended outside the domain to be given a reasonable value based on nearby values on the boundary. Figure (5.3), for example, shows points that have extended outside the computational domain.
3. interpolate ghost points on interpolation boundaries. In previous work the boundary points were interpolated which meant that the effective overlap

would tend to decrease as the grids were coarsened. However, when ghost points are interpolated, the ghost points on the coarsened grid will extend further into the neighbouring grids and thus the effective overlap will tend to increase.

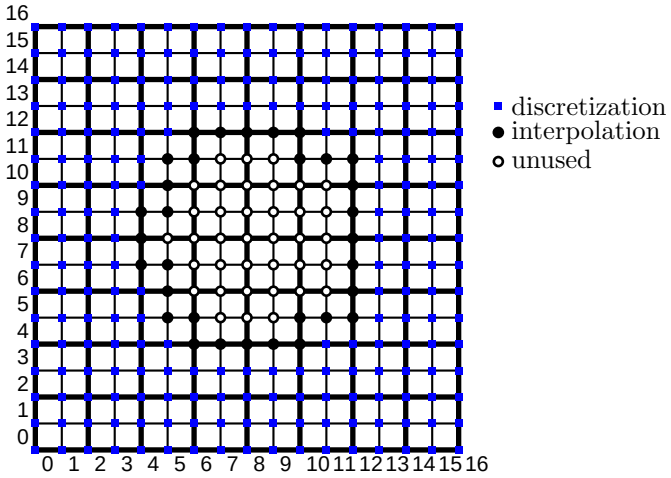


FIG. 5.1. *Mask array for the fine grid.*

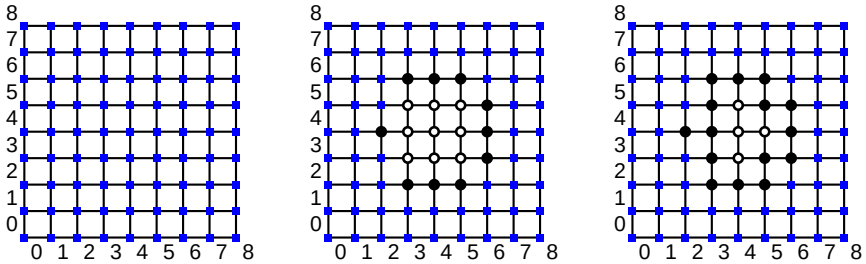


FIG. 5.2. *Mask array for the coarse grid. Left: initial state. Middle: after assigning from fine grid mask. Right: after filling in extra interpolation points.*

ALGORITHM 6 (Generate coarse level overlapping grids).

procedure **buildCoarseMultigridLevel**$(\mathcal{G}_h, \mathcal{G}_H)$

{

for $g = 1, \ldots, n_g$ **do**

$\text{mask}_{\mathbf{i}}^{H,g} \leftarrow \text{mask}_{2\mathbf{i}}^{h,g}$ *for* $\mathbf{i}$ *in the coarse grid* $G_{H,g}$

where *unused point* $\mathbf{i}$ *is next to a discretization point*

$\text{mask}_{\mathbf{i}}^{H,g} \leftarrow$ *interpolation*

end where

end for

for $g = 1, \ldots, n_g$ **do**

where $\text{mask}_{\mathbf{i}}^{H,g}$ *and* $\text{mask}_{2\mathbf{i}}^{h,g}$ *are both interpolation points*

$\text{InterpInfo}_{\mathbf{i}}^{H,g} \leftarrow \text{InterpInfo}_{2\mathbf{i}}^{h,g}$ *(get interpolation data from the fine grid)*

end where

where $\text{mask}_{\mathbf{i}}^{H,g}$ =*interpolation and* $\text{mask}_{2\mathbf{i}}^{h,g} \neq$*interpolation*

Determine the interpolation information for point $\mathbf{i}$ *on* $G_{H,g}$*:*

foreach $\text{mask}_{2\mathbf{i}\pm 1}^{h,g}$ =*interpolation (nearby fine grid interp pts)*

$g_d \leftarrow$ *donor grid from point* $2\mathbf{i} \pm 1$ *(potential donor grid)*

$\mathbf{r} \leftarrow \mathbf{C}_{g_d}^{-1}(\mathbf{x}_{\mathbf{i}}^{H,g})$ *(unit square coords in donor grid)*

if $\mathbf{r}$ *is a good quality location to interpolate from*

```
                    InterpInfo_i^{H,g} ← interpolation information
                    break (this point has been found)
                else
                    save this info but keep looking for a better quality donor
                end if
            end foreach
            if unresolved interpolation points remain
                search other possible grids and choose the best quality donor.
            end if
        end where
    end for
}
```

The coarsening algorithm consists of two main stages. In the first stage, the classification mask on the coarse grid is defined and the location of all interpolation points is determined. In the second stage the interpolation data is evaluated (such as the donor grid and the interpolation coordinates in the donor grid). For efficiency, as much information as possible is derived from the valid fine grid. At the end of the first stage (consisting of the first "for" loop in algorithm 6) the interpolation points in the coarse grid mask will partition the grid points into those that are **discretization** separated from those that are **unused**. Each discretization point will be surrounded by discretization or interpolation points. For coarse grid interpolation points that coincide with fine grid interpolation points the interpolation data can be immediately computed. For a coarse grid point that does not coincide with a fine grid interpolation point a search is made to find nearby interpolation points. In some cases there may be more than one potential donor grid and a decision is made based on the quality of the interpolation. The quality of the interpolation is ranked from most desirable to least desirable:

Quality 1: valid interpolation with the same interpolation width as the fine grid.
Quality 2: valid interpolation with reduced interpolation width.
Quality 3: extrapolation outside a physical boundary.
Quality 4: extrapolation outside an interpolation boundary.

When interpolation data has been found that is quality 1 or 2, the data is considered acceptable and no more searching is performed. When the quality is 3 or 4, the search continues for a better quality donor grid until all possibilities are exhausted. Figure (5.3) shows four multigrid levels for an overlapping grid for some "shapes". Notice how the curvilinear grids grow in size as they are coarsened.

6. Numerical results. In this section numerical results are presented for some two- and three-dimensional problems. Poisson's equation is solved with Dirichlet, Neumann ($\alpha_0 = 0$, $\alpha_1 = 1$) or mixed ($\alpha_1 = 1$ and $\alpha_0 = 1$) boundary conditions:

$$\Delta u = f \qquad \mathbf{x} \in \Omega \ , \tag{6.1}$$

$$\alpha_1 \frac{\partial u}{\partial n} + \alpha_0 u = g \qquad \mathbf{x} \in \partial\Omega \ . \tag{6.2}$$

The *method of analytic solutions* is used to choose f and g so that the exact solution to (6.1-6.2) is known. For example, any sufficiently smooth function $w(\mathbf{x})$ will be a solution provided $f = \Delta w$ and $g = \alpha_1 \frac{\partial w}{\partial n} + \alpha_0 w$. With this approach the error in the discrete solution can be easily determined. Two common choices for exact solutions are a low degree polynomial or a trigonometric function such as $w(\mathbf{x}) = \cos(f_x \pi x)\cos(f_y \pi y)\cos(f_z \pi z)$. Convergence rates are generally insensitive to

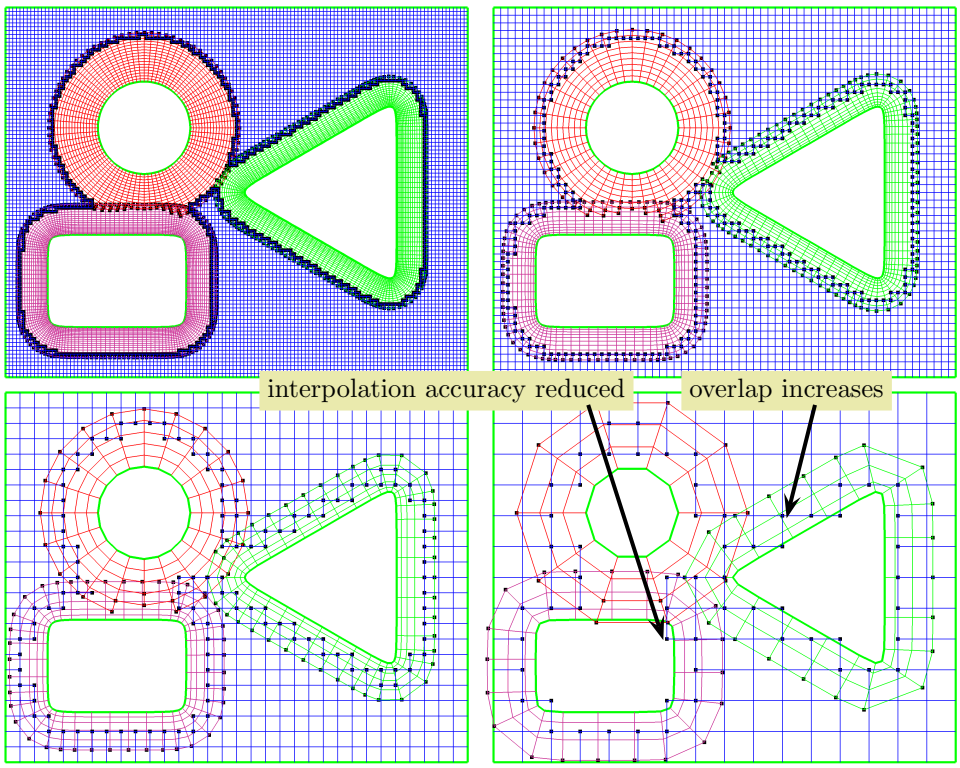


FIG. 5.3. *An overlapping grid for some shapes, 4 multigrid levels.*

the choice of f and g. The trigonometric function will be used in the examples given in this section. Introduce the following notation

$$\begin{aligned}
\text{WU} &= \text{number of work units for a cycle ,} \\
\|\text{res}\|_\infty &= \text{maximum residual ,} \\
\text{CR} &= \text{average convergence rate for a cycle ,} \\
\text{ECR} &= \text{effective convergence rate } = (\|\text{res}_\text{i}\|_\infty / \|\text{res}_{\text{i}-1}\|_\infty)^{1/WU} \ , \\
\text{W}[\nu_1,\nu_2] &= \text{denotes a W cycle with } \nu_1 \text{ pre-smooths and } \nu_2 \text{ post-smooths.}
\end{aligned}$$

A work unit is defined to be the amount of work (number of multiplications) required for a single Jacobi iteration. The work units reported here are only reasonable approximations. The effective convergence rate (ECR) is a normalized convergence rate that takes into account the amount of work required for each multigrid iteration. Roughly speaking, for a given problem, smaller values of the ECR will correspond to smaller computational times.

To allow for a comparison between the different examples, results for the W cycle are given. Although the V cycle is often more efficient (note that a V[1,1] cycle is used for the timing comparison in table 6.3), the W cycle is more robust and gives better results for some of the more difficult cases such as the submarine-in-a-box. As a remark, the F cycle seems to give almost the same results as the W cycle and is slightly more efficient.

6.1. Accuracy. Table 6.1 shows the maximum errors and estimated order of accuracy, error $\propto h^\sigma$, for a square, circle-in-a-channel, box and sphere-in-box. The order of accuracy was estimated by a least squares fit to the log of the errors versus $\log(h)$. Note that as the overlapping grids are refined, the positions of the interpolation points will change since the effective overlap decreases. As a result, the reduction in errors is not always as uniform as those from a single grid. The results demonstrate the second-order accuracy of the discretization.

$\frac{h}{h_0}$	error
1	1.4e−4
2	3.6e−5
4	9.0e−6
8	2.3e−6
16	5.7e−7
32	1.4e−7
σ	2.00
square	
BC=Dirichlet	

$\frac{h}{h_0}$	error
1	8.2e−4
2	2.0e−4
4	5.1e−5
8	1.3e−5
16	3.2e−6
32	8.0e−7
σ	2.00
square	
BC=NMNN	

$\frac{h}{h_0}$	error
1	7.0e−3
2	1.7e−3
4	3.2e−4
8	7.8e−5
16	2.2e−5
32	4.9e−6
σ	2.09
cic	
BC=Dirichlet	

$\frac{h}{h_0}$	error
1	1.9e−4
2	4.8e−5
4	1.2e−5
σ	2.00
box	
BC=NDDDDD	

$\frac{h}{h_0}$	error
1	2.3e−2
2	5.1e−3
4	8.8e−4
6	3.9e−4
σ	2.34
sib	
BC=Dirichlet	

TABLE 6.1

Maximum errors and estimated order of accuracy, error $\propto h^\sigma$*, for Poisson's equation with a trigonometric exact solution. The first column gives the ratio of the grid spacing to that on the coarsest grid. The cic grid is a circle in a channel. The sib grid is a sphere-in-a-box. BC=NMNN means there is a Neumann condition on 3 sides and a mixed condition on the other. Notice that the finest sib grid is only* 1.5 *times as fine as the previous grid.*

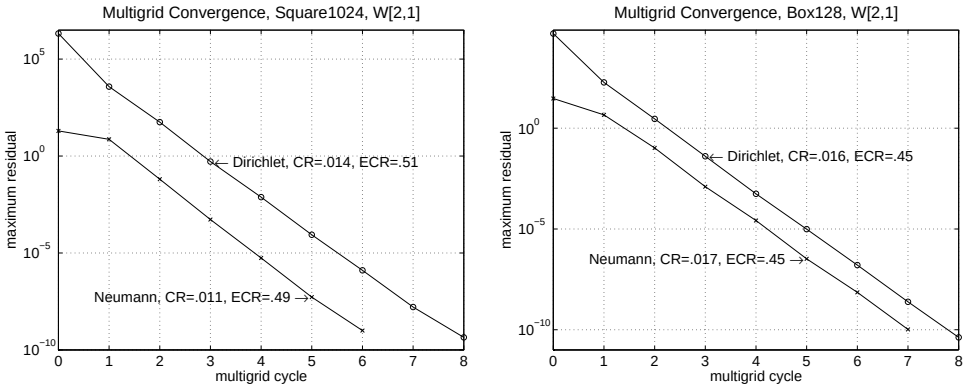
6.2. Rectangular grids. Some convergence results for a 1024^2 square grid and a 128^3 box grid are given in figure (6.1) and serve as a bench mark for comparison with overlapping grids. The results show that when the boundary conditions are handled properly, the convergence rates for Neumann/mixed boundary conditions are as good as the convergence rates for Dirichlet boundary conditions. The ECR's for the three-dimensional box are actually better than those for the square.

ω-RB-GS			LFA (theory)			Computed	
$[\nu_1,\nu_2]$	Galerkin	ω	μ_{loc}	$\rho_{\text{loc}}^{(2G)}$	$\rho_{\text{loc}}^{(3G)}$	CR(W)	CR(V)
$[1,1]$	No	1	.0625	.074	.104	.061	.092
$[1,1]$	Yes	1	.0625	.0625	.066	.059	.058
$[1,1]$	Yes	1.10	.0734	.0388	.052	.034	.037
$[2,1]$	No	1	.0335	.0523	.074	.044	.064
$[2,1]$	Yes	1	.0335	.0284	.040	.026	.034
$[2,1]$	Yes	1.10	.0440	.0157	.024	.014	.015
Two-dimensional, 5 point Laplacian.							

TABLE 6.2

Theoretical smoothing rates and asymptotic convergence factors for various ω-RB-GS cycles compared to the computed convergence rates for a 1024^2 square. ω-RB-GS is the over-relaxed Red-Black Gauss-Seidel smoother with $[\nu_1,\nu_2]$ the number of pre- and post-smooths. Galerkin refers to the use of Galerkin coarse grid operators. CR(V) and CR(W) are the convergence rates for a V and W cycle.

The numerical convergence rates can be validated by comparing them to the results predicted theoretically by local Fourier analysis (LFA). Local Fourier analysis



i	$\|\|\text{res}\|\|_\infty$	CR	WU	ECR
1	$3.8e+03$	0.002	8.1	0.46
2	$5.6e+01$	0.015	6.1	0.50
3	$5.2e-01$	0.009	6.1	0.46
4	$7.6e-03$	0.015	6.1	0.50
5	$8.7e-05$	0.011	6.1	0.48
6	$1.3e-06$	0.015	6.1	0.50
square1024, Dirichlet, W[2,1], 5 levels.				

i	$\|\|\text{res}\|\|_\infty$	CR	WU	ECR
1	$1.9e+02$	0.004	6.9	0.45
2	$2.9e+00$	0.016	4.9	0.43
3	$4.1e-02$	0.014	4.9	0.42
4	$5.5e-04$	0.014	4.9	0.42
5	$9.8e-06$	0.018	4.9	0.44
6	$1.6e-07$	0.016	4.9	0.44
box128, Dirichlet, W[2,1], 5 levels.				

FIG. 6.1. *Convergence history for a* 1024^2 *square and* 128^3 *box. A comparison between Dirichlet and Neumann/mixed boundary conditions. In the Neumann case the square has 3 Neumann and 1 mixed boundary condition while the box has 3 Neumann and 3 mixed boundary conditions. The tables show results for Dirichlet boundary conditions.*

can be used to predict the smoothing factor, μ_{loc}, and the asymptotic two-grid, $\rho_{\text{loc}}^{(2G)}$ and asymptotic three-grid convergence factor, $\rho_{\text{loc}}^{(3G)}$, see [25, 30]. The smoothing factor is an indication of the reduction in residual for each smooth, while the asymptotic convergence factors reflect the convergence rate for a cycle. Table 6.2 shows the smoothing factor, two-grid and three-grid asymptotic convergence rates, as computed by local Fourier analysis, for a standard 5-point operator in two-dimensions. These results were computed with the Wienands' excellent LFA software[29]. The last two columns show the numerically computed convergence rates for V and W cycles on a 1024^2 square. In general, one might expect the two-grid convergence factor to reflect the results of a W-cycle while the three-grid convergence factor should be more indicative of a V-cycle. Overall there is reasonable agreement between the theory and computation. For example, The two-grid convergence factor $\rho_{\text{loc}}^{(2G)} = .0157$ for $[\nu_1, \nu_2] = [2, 1]$ using Galerkin coarse grid operators should be compared to the value of about .014 actually obtained on a 1024^2 square. These results illustrate that a significant improvement in convergence rate can be realized by using a Galerkin coarse grid operator and accelerated Red-Black smoothers with $\rho_{\text{loc}}^{(2G)}$ improving from about .044 to approximately .014 for the computed results of a W[2,1] cycle. Although these methods require more work, in practice the additional computational time required is found to be small.

The results in table 6.2 illustrate that even though the smoothing factor is worse for $\omega = 1.1$ compared to $\omega = 1$, the overall convergence rates are better. Local Fourier analysis can be used to determine the value of ω that minimizes $\rho_{\text{loc}}^{(2G)}$ or $\rho_{\text{loc}}^{(3G)}$. These values of ω will, in general, be different from the value that minimizes the smoothing factor, especially as the number of smooths, $\nu_1 + \nu_2$, increases.

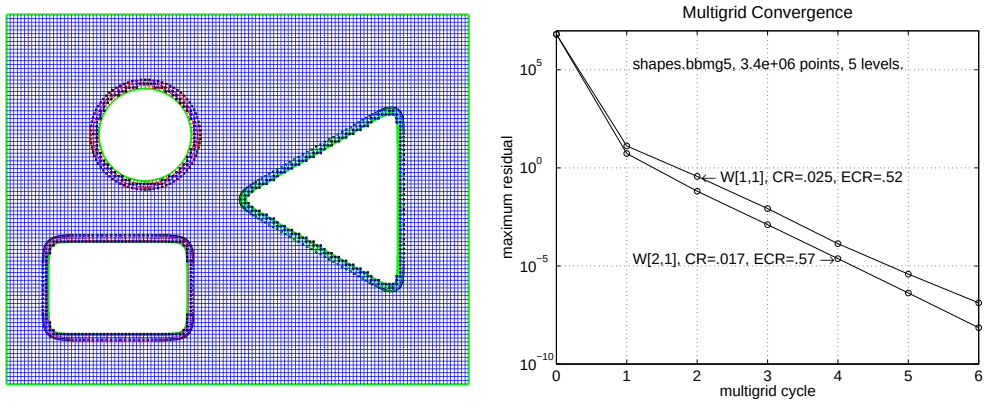


FIG. 6.2. *Left: Multigrid level 5 of the shapes grid. Right: convergence history beginning with a full multigrid cycle. Most of the grid points are on the Cartesian grid. CR and ECR are the average convergence rate and effective convergence rate, not including the first cycle.*

6.3. Overlapping grids. The coarsest level grid and the convergence history or the "shapes" geometry are shown in figure (6.2). The iteration started with a full-multigrid cycle. In full-multigrid, the cycles begin on the coarsest grid and work themselves up to the finest grid. A Red-Black smoother was used on the square, a line zebra smoother was used on the body fitted grids. This choice of smoothers, Red-Black for Cartesian grids and line-zebra for curvilinear grids, usually gives the best results. Results are shown for a W[1,1] and W[2,1] cycle. Although the W[2,1] cycle has a better convergence rate, it is about 30% slower than the W[1,1] cycle; this is reflected in a smaller ECR for the W[1,1] cycle. Since the majority of the grid points on the finest level are on the Cartesian background grid, one might hope that the convergence rates would be close to that of a single Cartesian grid. Indeed the W[2,1] convergence rates of CR=.017 (ECR=.57) are close to that of a square, CR=.014 (ECR=.51). Figure (6.3) shows similar convergence results for a grid about two Joukowsky airfoils.

Figure (6.4) shows the convergence results for a three-dimensional grid for the region exterior to a sphere and inside a box. These results, CR=.010 (ECR=.48) for a W[2,1] cycle, compare favourably to the results for a three-dimensional box, CR=.016 (ECR=.45). In figure (6.5) results are given for a geometry consisting of a collection of spheres in a box. This grid has over 5 million grid points and 15 different component grids. The convergence rates, although not quite as good as for the single sphere are also very good with CR=.028 (ECR=.60) for a W[2,1] cycle. Results for the submarine-in-a-box grid are shown figure (6.6). The overlapping grid consisted of 18 component grids. The convergence rates are not quite as good as for the sphere-in-a box but are still quite acceptable.

6.4. Performance and comparison to other methods. Results from the Ogmg multigrid solver are compared to some good Krylov-based iterative solvers in table 6.3. The first five rows show results for a two-dimensional circle-in-a-channel grid. The last four rows consider a three-dimensional ellipsoid-in-a-box. The stabilized bi-conjugate-gradient (biCG-stab) method with incomplete LU (ILU) preconditioning was found to give the best results amongst the Krylov methods that were tested. The Krylov solvers used here are from the PETSc library [1]. The setup time for Ogmg

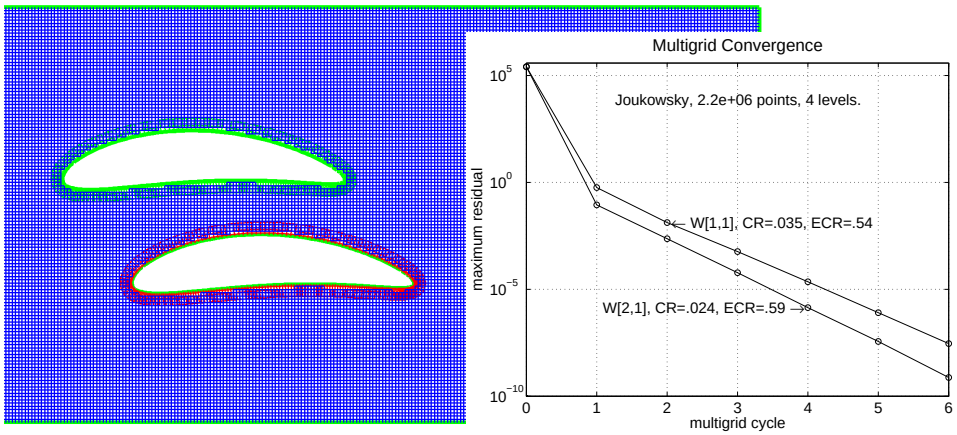
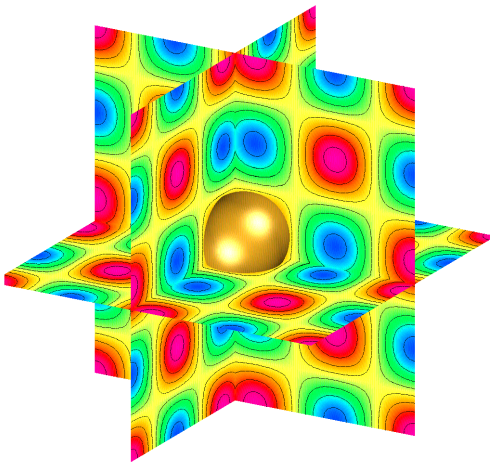


FIG. 6.3. *An overlapping grid for two airfoils in a channel (multigrid level $l = 3$). Convergence history including an initial full multigrid cycle. An alternating zebra smoother is used on the body fitted grids.*



i	$\|\|\text{res}\|\|_\infty$	CR	WU	ECR
1	$1.2e+00$	.0004	8.0	0.38
2	$1.3e-02$	0.010	5.9	0.46
3	$8.6e-05$	0.007	6.4	0.46
4	$8.0e-07$	0.009	6.2	0.47
5	$1.2e-08$	0.015	6.8	0.54
Sphere in a box. Dirichlet BC's. Full Multigrid. W[2,1]: RB $\omega = 1.12$, lz3 $\omega = 1.09$ 2.78e+06 grid-points. 4 levels. Average CR=0.010, ECR=0.48.				

FIG. 6.4. *Left: computed solution for a sphere in a box. Right; convergence history with a full multigrid cycle.*

includes the time needed for generation of the multigrid levels and generation of the coarser grid operators by averaging. The setup time for the Krylov solvers includes the time required to copy the matrix coefficients from Overture to PETSc and the time needed to build the preconditioner. The *reals/pt* column indicates the approximate [2] number of double-precision floating point numbers that are required per grid point.

For the two-dimensional grid circle-in-a-channel, which has about 1.1 million grid points, the computing time for the multigrid solver is about 45 times faster and uses about 11 times less storage than biCG-stab, ILU(5). Compared to biCG-stab ILU(0), Ogmg is about 100 times faster and uses about 7 times less memory. For the three-dimensional case, Ogmg is about 10 times faster than biCG-stab, ILU(2), and

[2]The memory for the PETSc solvers was taken as the `Maximum memory used` obtained with the `-trmalloc_log` option.

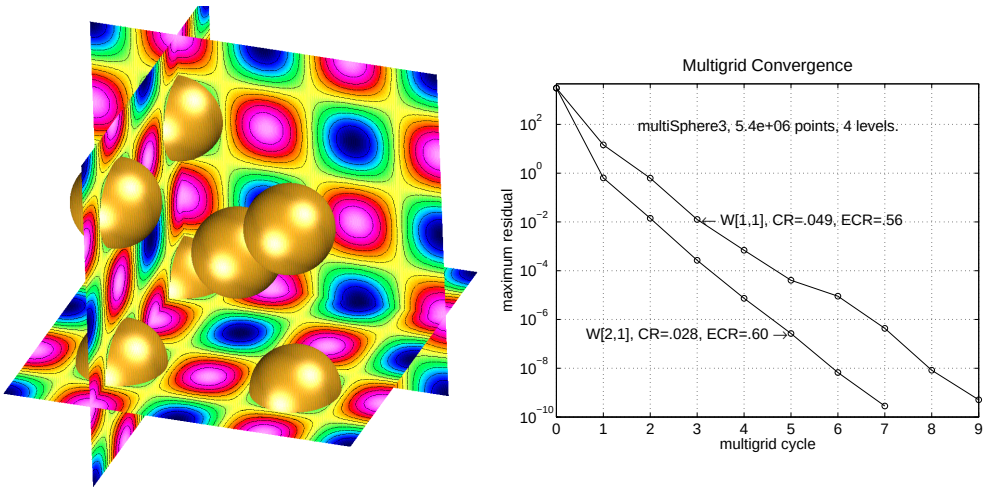


FIG. 6.5. *Left: computed solution for multiple spheres in a box. Right: convergence history with a full multigrid cycle.*

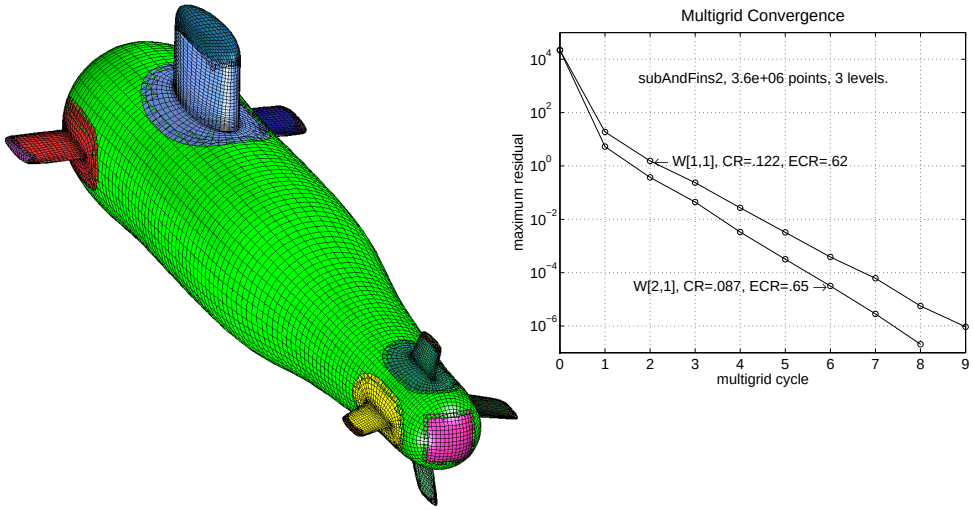


FIG. 6.6. *Left: Grid for a submarine-in-a-box, Right: convergence history.*

uses approximately 7 times less storage. As the grids become finer, Ogmg should be expected to have an even larger advantage in speed.

All the results presented in this manuscript were performed on a Linux desktop with a 2.2GHz Xeon processor and 2 GBytes of memory.

7. Conclusions. A multigrid algorithm for the solution of elliptic boundary value problems on two- and three-dimensional overlapping grids has been described. The approach has been implemented in the Ogmg solver. A new procedure for automatically generating the coarse grid levels and the connecting interpolation information has been presented. A composite smoothing operator that automatically adjusts the number of sub-smooths on each component grid is combined with a procedure for locally smoothing the defect near interpolation boundaries. Convergence rates are improved through the use of operator averaging and over-relaxed Red-Black smoothers.

					CPU time (s)			storage
Solver	grid	pts	its	$\|\|res\|\|_\infty$	total	setup	solve	reals/pt
Ogmg V[1,1] FMG	cic	1.1e6	7	5.7e-10	3.34	.537	2.8	4.6
biCG-stab, ILU(5)	cic	1.1e6	144	8.9e-9	152	35.	117	53.5
gmres ILU(5)	cic	1.1e6	435	1.0e-8	271	35	236	65.0
biCG-stab, ILU(0)	cic	1.1e6	554	8.6e-9	342	32	310	33.3
gmres ILU(0)	cic	1.1e6	2657	8.9e-9	1135	33	1102	49.0
Ogmg V[1,1] FMG	elb	2.0e6	10	3.3e-10	21.5	4.52	17.0	9.9
biCG-stab, ILU(2)	elb	2.0e6	46	4.1e-10	222.	106	116	70.3
biCG-stab, ILU(0)	elb	2.0e6	113	3.7e-10	264.	77.	187	41.6
gmres(20), ILU(0)	elb	2.0e6	218	5.2e-10	306.	70.	236	56.5

TABLE 6.3

A comparison of the multigrid solver Ogmg to some Krylov based solvers. The cic grid is a two-dimensional circle-in-a-channel, the elb grid is an ellipsoid-in-a-box.

Performance was enhanced using optimizations for predefined equations and Cartesian component grids. Numerical results in two and three space dimensions demonstrate that very good multigrid convergence rates can be obtained. For overlapping grids dominated by Cartesian component grids, the results approach the "text-book" convergence rates of single Cartesian grids. A comparison to some good Krylov-based iterative solvers showed the multigrid solver can be much faster and use significantly less memory.

REFERENCES

[1] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, *The portable extensible toolkit for scientific computation*, Tech. Rep. http://www.mcs.anl.gov/petsc/petsc.html, Argonne National Laboratory, 1999.

[2] A. Brandt, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31 (1977), pp. 333–390.

[3] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, SIAM, 2000.

[4] D. L. Brown, G. S. Chesshire, W. D. Henshaw, and D. J. Quinlan, *Overture: An object oriented software system for solving partial differential equations in serial and parallel environments*, in Proceedings of the Eighth SIAM Conference on Parallel Processing for Scientific Computing, 1997.

[5] D. L. Brown, W. D. Henshaw, and D. J. Quinlan, *Overture: An object oriented framework for solving partial differential equations*, in Scientific Computing in Object-Oriented Parallel Environments, Springer Lecture Notes in Computer Science, **1343**, 1997, pp. 177–194.

[6] P. G. Buning, I. T. Chiu, S. Obayashi, Y. M. Rizk, and J. L. Steger, *Numerical simulation of the integrated space shuttle vehicle in ascent*, paper 88-4359-CP, AIAA, 1988.

[7] G. Chesshire and W. Henshaw, *Composite overlapping meshes for the solution of partial differential equations*, J. Comp. Phys., 90 (1990), pp. 1–64.

[8] P. Fast, *Dynamics of Interfaces in non-Newtonian Hele-Shaw flow*, PhD thesis, New York University, Courant Institute of Mathematical Sciences, 1999.

[9] W. Hackbusch, *Multi-Grid Methods and Applications*, Springer-Verlag, Berlin, 1985.

[10] W. Henshaw, *Part II: Composite Overlapping Grid Techniques*, PhD thesis, Dept. of Applied Mathematics, California Institute of Technology, 1985.

[11] ———, *Ogen: An overlapping grid generator for Overture*, Research Report UCRL-MA-132237, Lawrence Livermore National Laboratory, 1998.

[12] W. Henshaw and G. Chesshire, *Multigrid on composite meshes*, SIAM J. Sci. Stat. Comput., 8 (1987), pp. 914–923.

[13] M. Hinatsu and J. Ferziger, *Numerical computation of unsteady incompressible flow in complex geometry using a composite multigrid technique*, International Journal for Numerical Methods in Fluids, 13 (1991), pp. 971–997.

[14] D. Jespersen, T. Pulliam, and P. Buning, *Recent enhancements to overflow*, paper 97-0644,

AIAA, 1997.
[15] R. A. Johnson and D. M. Belk, *Multigrid approach to overset grid communication*, AIAA J., 33 (1995), pp. 2305–2308.
[16] C. Kiris, D. Kwak, S. Rogers, and I. Chang, *Computational approach for probing the flow through artificial heart devices*, ASME J. of Biofluidmechanical Engineering, 119 (1997), pp. 452–460.
[17] B. Kreiss, *Construction of a curvilinear grid*, SIAM J. of Sci. Stat. Comput., 4 (1983), pp. 270–279.
[18] S. McCormick, *Multilevel Adaptive Methods for Partial Differential Equations*, Frontiers in Applied Mathematics 6. SIAM, Philadelphia, 1989.
[19] C. Perng and R. Street, *A coupled multigrid-domain-splitting technique for simulating incompressible flows in geometrically complex domains*, International Journal for Numerical Methods in Fluids, 13 (1991), pp. 269–286.
[20] B. Smith, P. B. rstad, and W. Gropp, *Domain Decomposition*, Cambridge University Press, Cambridge, 1996.
[21] G. Starius, *Composite mesh difference methods for elliptic and boundary value problems*, Numer. Math., 28 (1977), pp. 243–258.
[22] ———, *On composite mesh difference methods for hyperbolic differential equations*, Numer. Math., 35 (1980), pp. 241–255.
[23] J. L. Steger and J. A. Benek, *On the use of composite grid schemes in computational aerodynamics*, Computer Methods in Applied Mechanics and Engineering, 64 (1987), pp. 301–320.
[24] K. Stüben and U. Trottenberg, *Multigrid methods: Fundamental algorithms, model problem analysis and applications*, in Multigrid Methods, W. Hackbusch and U. Trottenberg, eds., Springer-Verlag, 1982, pp. 1–176.
[25] U. Trottenberg, C. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, London, 2001.
[26] J. Y. Tu and L. Fuchs, *Overlapping grids and multigrid methods for the three-dimensional unsteady flow calculations in IC engines*, International Journal for Numerical Methods in Fluids, 15 (1992), pp. 693–714.
[27] ———, *Calculation of flows using three-dimensional overlapping grids and multigrid methods*, International Journal for Numerical Methods in Engineering, 38 (1995), pp. 259–282.
[28] P. Wesseling, *An Introduction To Multigrid Methods*, Jonh Wiley & Sons, New York, 1991.
[29] R. Wienands, *Lfa00_2d_scalar*, Tech. Rep. http://www.mgnet.org/mgnet-codes-wienands.html, GMD - Institute for Algorithms and Scientific Computing (SCAI), 2000.
[30] R. Wienands and C. Oosterlee, *On three-grid Fourier analysis for multigrid*, SIAM J. Sci. Comput., 28 (2001), pp. 651–671.
[31] I. Yavneh, *On red-black SOR smoothing in multigrid*, SIAM J. Sci. Comput., 17 (1996), pp. 180–192.
[32] Y. Zang and R. Street, *A composite multigrid method for calculating unsteady incompressible flows in geometrically complex domains*, International Journal for Numerical Methods in Fluids, 20 (1995), pp. 341–361.