

Experiments with Higher Order Multigrid Methods

Nicole M. Burgess
Scott R. Fulton

October 1999

Abstract

This paper presents six fourth-order multigrid algorithms for solving the Poisson problem: two V-Cycle algorithms, three Full Multigrid (FMG) algorithms, and tau extrapolation. The algorithms are described and compared, using numerical experiments to investigate the truncation errors, residuals, solution errors, and computational work. All demonstrated fourth-order convergence, but the algorithms based on the compact fourth-order (Mehrstellen Verfahren) discretization produced smaller errors than tau extrapolation.

Two of the algorithms were implemented in the adaptive multigrid hurricane model MUDBAR to solve the Poisson equation relating vorticity and streamfunction (previously solved by a second-order FMG algorithm). Although fourth-order accuracy was achieved for that equation, there was little change in the overall accuracy of the model, due apparently to the continued use of a second-order discretization of the other equation (advection of vorticity).

Technical Report No. 99-02
Department of Mathematics and Computer Science
Clarkson University, Potsdam, New York

*This work was supported by the
Office of Naval Research
Marine Meteorology and Atmospheric Effects Program
Grants N00014-98-1-0103 and N00014-98-1-0368*

Contents

1	Introduction	1
2	Notation and Basic Concepts	3
2.1	The model problem	3
2.2	The V-Cycle algorithm	4
2.3	The Full Multigrid algorithm (FMG)	4
2.4	Restriction operators	5
2.5	Interpolation	5
2.6	Discretization and relaxation schemes	5
2.6.1	The 5-point stencil	5
2.6.2	The Mehrstellen Verfahren (MV) difference stencil	6
2.6.3	Tau extrapolation	7
2.7	Convergence factors and smoothing factors	8
2.8	Measuring convergence and computational work	8
3	Description of the Multigrid Algorithms	9
4	Numerical Results and Comparisons	11
4.1	Comparison of the V-Cycle algorithms	11
4.2	Comparison of the FMG algorithms	14
4.3	Numerical results from tau extrapolation	17
5	Application to the MUDBAR Hurricane Model	18
6	Conclusions	20
A	Appendix: Accuracy of the MV Discretization	21

1 Introduction

Many numerical methods have been developed to solve elliptic boundary value problems. Typically, such problems are solved by first discretizing (e.g., finite differences) and then solving the resulting system of linear equations, either directly (by Gaussian elimination) or iteratively (using a relaxation scheme such as Gauss-Seidel or SOR). Unfortunately, the amount of work required to solve for the unknowns using these methods is not proportional to the number of unknowns; it is larger. For this reason, faster solvers have been sought. One such technique is multigrid methods.

Multigrid methods are fast iterative solvers based on restricting and interpolating solutions between a sequence of nested grids. Multigrid methods can be simplified into three steps:

1. Given a grid one would like to solve the problem on, perform a few relaxation sweeps in order to smooth out the error.
2. Restrict the residual problem to a subset of the grid points, the so-called “coarse grid,” and solve the resulting coarse-grid problem.
3. Interpolate the coarse-grid solution back to the original grid, and perform a few more relaxation sweeps.

Using the same technique to solve the coarse-grid equation extends this two-grid method recursively to form a multigrid method. Thus, the coarser grids are correction grids, which accelerate the convergence of the relaxation scheme on the finest grid by efficiently reducing the smooth error components.

Multigrid methods are flexible because they can treat arbitrary regions and boundary conditions, and they do not rely on separability of equations or other special properties of the equation. Also, multigrid methods are flexible because they can be applied to a given problem in conjunction with any of the known discretization techniques, such as finite differences, finite elements, or spectral methods. Moreover, multigrid is an optimal method because it exhibits a convergence rate that is independent of the number of unknowns in the discretized system. Furthermore, the Full Multigrid (FMG) algorithm (multigrid combined with nested iteration) can solve problems to the accuracy of the truncation error in a number of operations that is proportional to the number of unknowns.

Schaffer[5] studied numerous higher-order multigrid methods. The three basic techniques Schaffer considered were (i) multigrid processing involving smoothing via higher-order difference approximations, (ii) iterated defect corrections with multigrid used as an inner loop equation solver, and (iii) tau extrapolation. He tested several choices of high-order differencing, relaxation orderings, restriction operators, and prolongation operators. After solving numerous problems with his various algorithms, Schaffer[5] concluded:

The convergence rates observed in the experiments generally stayed within the optimal rates predicted by local mode analysis . . . The Mehrstellen Verfahren discretizations were the most accurate of the discretizations considered here. The algorithms using the Mehrstellen Verfahren discretization were consistently more efficient than the corresponding algorithms using other alternatives (pp. 113-114).

This report attempts to fill in some of the details of these methods. Section 2 describes the notation and basic concepts of multigrid methods. Section 3 introduces the six fourth-order methods to be considered: two V-Cycle algorithms using the compact fourth-order Mehrstellen Verfahren (MV) discretization, three FMG algorithms using the MV discretization, and an FMG algorithm using tau extrapolation. Numerical results are summarized in section 4. The intended application of these methods is to solve the Poisson-type equation relating the streamfunction and vorticity in the adaptive multigrid hurricane model MUDBAR [2]; section 5 presents results from implementing several of these methods in that model. Conclusions are given in section 6, and the Appendix details the derivation of the truncation error of the MV discretization.

2 Notation and Basic Concepts

This section discusses the basics of multigrid methods and the various discretizations to be considered. Further details of multigrid methods can be found in the literature, e.g., [1], [4].

2.1 The model problem

As a model problem we consider the Poisson problem

$$Lu = \nabla^2 u = f \quad (1)$$

on the unit square $\Omega = (0, 1) \times (0, 1)$, with Dirichlet boundary conditions

$$u = g \quad (2)$$

on the boundary $\partial\Omega$. Here $L = \nabla^2$ is the Laplacian, f is a continuous function defined in Ω , g is a continuous function defined on $\partial\Omega$, and u is the continuous solution.

We discretize the problem on a uniform grid Ω^h covering Ω with mesh spacing $h = 1/n$, i.e.,

$$\Omega^h = \{(x_i, y_j) = (ih, jh) : i, j = 0, 1, \dots, n\} \quad (3)$$

We denote by f^h and g^h the pointwise restrictions of the continuous functions f and g onto Ω^h and $\partial\Omega^h$. Thus, $f^h(x_i, y_j) = f_{i,j} = f(x_i, y_j)$ and $g^h(x_i, y_j) = g_{i,j} = g(x_i, y_j)$. The discrete approximation to the continuous true solution u is denoted by $u^h(x_i, y_j) = u_{i,j} \approx u(x_i, y_j)$. Then the discrete Poisson problem takes the form

$$L^h u^h = F^h \quad (4)$$

on Ω^h and

$$u^h = g^h \quad (5)$$

on $\partial\Omega^h$, where L^h is a finite difference operator and F^h is a grid function related to f^h . Given an approximate solution $\tilde{u}^h \approx u^h$, the corresponding residual and error are

$$r^h = F^h - L^h \tilde{u}^h \quad (6)$$

and

$$e^h = u^h - \tilde{u}^h, \quad (7)$$

respectively; due to the linearity of L^h these satisfy

$$L^h e^h = r^h. \quad (8)$$

2.2 The V-Cycle algorithm

Brandt[1] wrote that the basic concept of multigrid methods can be demonstrated by the following simple two level multigrid cycle, which uses a fine grid Ω^h and a coarse grid Ω^{2h} :

1. Input an initial guess for \tilde{u}^h on the fine grid Ω^h ,
2. Perform relaxation on $L^h \tilde{u}^h = F^h$ to improve \tilde{u}^h ,
3. Transfer the residual r^h to the coarse grid Ω^{2h} using a fine-to-coarse grid transfer (restriction) operator I_h^{2h} ,
4. Solve the coarse grid correction equation $L^{2h} v^{2h} = I_h^{2h} r^h$,
5. Transfer the correction v^{2h} back to the fine grid Ω^h using a coarse-to-fine grid transfer (interpolation) operator I_{2h}^h , and update \tilde{u}^h by $\tilde{u}^h + I_{2h}^h v^{2h}$,
6. Perform relaxation on $L^h \tilde{u}^h = F^h$ to further improve \tilde{u}^h .

This cycle has two main parts: a smoothing part (steps 2 and 6) to reduce the high-frequency error components of \tilde{u}^h by relaxation, and a coarse-grid correction part (steps 3-5) to reduce the smooth error components of \tilde{u}^h . The coarse-grid variable v^{2h} is an approximation on the coarse grid to the fine-grid error e^h . This cycle can be repeated, resulting in an iterative method in which \tilde{u}^h converges to u^h . As mentioned above, this two-grid method can be extended recursively to a multigrid method by using the same method to solve the coarse-grid equation in step 4, thus producing a multigrid V-Cycle.

2.3 The Full Multigrid algorithm (FMG)

Using multigrid V-Cycles as an iterative solver requires an initial approximation on the finest grid Ω^h . The Full Multigrid (FMG) algorithm provides this by first performing one or more V-Cycles on the next coarser grid Ω^{2h} , and then interpolating that solution to the fine grid Ω^h to serve as the initial approximation there. This process is extended recursively back through still coarser grids, resulting in a method which starts on the coarsest grid. After solving there (e.g., by many relaxation sweeps or directly), the algorithm proceeds to each finer grid in turn, first interpolating the previous solution and then performing one or more V-Cycles. If k cycles are used for each level the method is known as the k -FMG algorithm, e.g., 1-FMG uses one cycle per level, 2-FMG uses two, etc. Note that this algorithm is not iterative: a fixed number of operations are used. However, the FMG algorithm may be followed by one or more additional V-Cycles if needed for additional convergence.

2.4 Restriction operators

Restriction operators are used to transfer the residual r^h to the next coarser grid. We consider two restriction operators: injection \hat{I}_h^{2h} and full weighting I_h^{2h} . Injection means simply copying the fine-grid values to the coarse grid at points common to both grids. Full weighting is a weighted average; the operator can be expressed explicitly as an $O(h^2)$ perturbation of the injection operator as

$$I_h^{2h} = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \hat{I}_h^{2h} + \frac{1}{16} \left\{ \frac{1}{h^2} \begin{bmatrix} 1 & 2 & 1 \\ 2 & -12 & 2 \\ 1 & 2 & 1 \end{bmatrix} \right\} h^2 \quad (9)$$

where the bracketed term is a difference approximation to the Laplacian operator [5].

2.5 Interpolation

Interpolation is used to transfer the correction from the coarse grid to the fine grid (and for the initial transfer of the solution in the FMG algorithm). We use both bilinear and bicubic interpolation in the algorithms to be tested.

2.6 Discretization and relaxation schemes

In a multigrid method, relaxation on a grid smoothes the error on the scale of that grid. While red-black ordering (also known as checkerboard ordering) is often used in multigrid methods and has some advantages, we will use lexicographic ordering for simplicity. Two different finite-difference discretizations are used as detailed below.

2.6.1 The 5-point stencil

The usual five-point discretization of the Laplacian operator is given by the difference stencil

$$L_2^h := \frac{1}{h^2} \begin{bmatrix} & & 1 \\ 1 & -4 & 1 \\ & & 1 \end{bmatrix}. \quad (10)$$

The corresponding discrete problem is

$$L_2^h u^h = \frac{1}{h^2} \begin{bmatrix} & & 1 \\ 1 & -4 & 1 \\ & & 1 \end{bmatrix} u^h = F^h, \quad (11)$$

with $F^h = f^h$ here. This discretization is second-order accurate. Solving for $u_{i,j}$ gives

$$u_{i,j} = \frac{1}{4}[(u_{i-1,j} + u_{i+1,j} + u_{i,j+1} + u_{i,j-1}) - h^2 f_{i,j}]. \quad (12)$$

Gauss-Seidel (GS) relaxation for (11) consists of using (12) to update the solution point by point, with the current values used on the right-hand side. The corresponding residual is given by

$$r_{i,j} = f_{i,j} - \frac{(\tilde{u}_{i-1,j} + \tilde{u}_{i+1,j} + \tilde{u}_{i,j-1} + \tilde{u}_{i,j+1} - 4\tilde{u}_{i,j})}{h^2}, \quad (13)$$

where \tilde{u}^h is the current approximate solution.

2.6.2 The Mehrstellen Verfahren (MV) difference stencil

In order to obtain fourth-order accuracy the compact fourth-order Mehrstellen Verfahren (MV) difference stencil

$$L_M^h := \frac{1}{6h^2} \begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix} \quad (14)$$

is used together with the weighting

$$I^h := \frac{1}{12} \begin{bmatrix} & 1 & \\ 1 & 8 & 1 \\ & 1 & \end{bmatrix} \quad (15)$$

to obtain the discrete problem in the form

$$L_M^h u^h = \frac{1}{6h^2} \begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix} u^h = \frac{1}{12} \begin{bmatrix} & 1 & \\ 1 & 8 & 1 \\ & 1 & \end{bmatrix} f^h = F^h. \quad (16)$$

This discretization is fourth-order accurate (see the Appendix). At each interior grid point, (14) implies

$$\begin{aligned} -20u_{i,j} &+ (u_{i-1,j-1} + u_{i-1,j+1} + u_{i+1,j-1} + u_{i+1,j+1}) \\ &+ 4(u_{i-1,j} + u_{i+1,j} + u_{i,j+1} + u_{i,j-1}) \\ &= \frac{h^2}{2}(8f_{i,j} + f_{i+1,j} + f_{i-1,j} + f_{i,j-1} + f_{i,j+1}) \end{aligned} \quad (17)$$

Solving for $u_{i,j}$ gives

$$\begin{aligned} u_{i,j} &= -\frac{1}{20} \left[\frac{h^2}{2}(8f_{i,j} + f_{i+1,j} + f_{i-1,j} + f_{i,j-1} + f_{i,j+1}) \right. \\ &\quad - (u_{i-1,j-1} + u_{i-1,j+1} + u_{i+1,j-1} + u_{i+1,j+1}) \\ &\quad \left. - 4(u_{i-1,j} + u_{i+1,j} + u_{i,j+1} + u_{i,j-1}) \right] \end{aligned} \quad (18)$$

Gauss-Seidel relaxation for this discretization consists of using (16) to update values point-by-point, using the current values on the right-hand side of (16). The corresponding residual is given by

$$r_{i,j} = \frac{(f_{i-1,j} + f_{i+1,j} + f_{i,j-1} + f_{i,j+1} + 8f_{i,j})}{12} - \frac{[\tilde{u}_{i-1,j-1} + \tilde{u}_{i+1,j-1} + \tilde{u}_{i+1,j+1} + \tilde{u}_{i-1,j+1} - 20\tilde{u}_{i,j} + 4(\tilde{u}_{i-1,j} + \tilde{u}_{i+1,j} + \tilde{u}_{i,j-1} + \tilde{u}_{i,j+1})]}{6h^2} \quad (19)$$

2.6.3 Tau extrapolation

Tau extrapolation is based on the fact that for h small enough the truncation error τ^h of the second-order discretization (11) can be expressed locally by

$$\tau^h = L_2^h u - f^h = Ah^2 + O(h^4), \quad (20)$$

where the function A depends on the points (x, y) in Ω , but is independent of h . Thus, the truncation errors τ^h and τ^{2h} can be extrapolated to yield the fourth-order approximation

$$\tau_h^{2h} = \frac{4}{3}(\tau^{2h} - I_h^{2h}\tau^h) = \tau^{2h} + O(h^4). \quad (21)$$

The method of tau extrapolation uses an approximation to the extrapolation as follows[5]. Let \tilde{u}^h be a second-order approximation to u on grid h and define

$$r^h = L_2^h \tilde{u}^h - f^h, \quad (22)$$

$$r^{2h} = L_2^{2h} \hat{I}_h^{2h} \tilde{u}^h - f^{2h}, \quad (23)$$

and

$$r_h^{2h} = \frac{4}{3}(r^{2h} - I_h^{2h}r^h), \quad (24)$$

where the solution transfer \hat{I}_h^{2h} is injection. Then combining (21) through (24) it follows that

$$\tau_h^{2h} - r_h^{2h} = \frac{4}{3}(L_2^{2h} \hat{I}_h^{2h} - I_h^{2h} L_2^h)(u - \tilde{u}^h) = O(h^4) \quad (25)$$

where the $O(h^4)$ term depends on fourth differences of the error $u - \tilde{u}^h$. Equations (21) and (25) imply that the solution \bar{u}^{2h} of the extrapolated equation

$$L_2^{2h} \bar{u}^{2h} = f^{2h} + r_h^{2h} \quad (26)$$

will be a fourth-order approximation to u on grid $2h$. Note that the notation of Brandt[1] for this extrapolation is slightly different.

2.7 Convergence factors and smoothing factors

Local mode analysis [1] gives one way to predict the performance of a multigrid method. This technique uses Fourier analysis to examine the effects of the relaxation scheme on the high frequency components of the error. The assumption of periodicity simplifies the analysis by eliminating effects due to boundary conditions; the resulting estimated convergence rates are reasonable approximations, since relaxation is a local process. On the grid Ω^h the discrete Fourier modes $E_\theta(x_j, y_k) = \exp[i(j\theta_1 + k\theta_2)]$ are visible for discrete wavenumbers $\theta = (\theta_1, \theta_2)$ with components θ_1 and θ_2 between $-\pi$ and π . Suppose the error before the sweep consists of a single Fourier mode, i.e.,

$$e^h = u^h - \tilde{u}^h = A_\theta E_\theta. \quad (27)$$

Then the error after the sweep will consist of the same mode but with a new amplitude, i.e.,

$$e^{h,new} = u^h - \tilde{u}^{h,new} = A_\theta^{new} E_\theta. \quad (28)$$

The convergence factor (per sweep) for the mode E_θ is $\mu(\theta) = |A_\theta^{new}/A_\theta|$, and the multigrid smoothing factor $\bar{\mu}$ is defined as the maximum convergence factor $\mu(\theta)$ for the high wavenumbers, i.e., those for which $\pi/2 \leq \max(|\theta_1|, |\theta_2|) \leq \pi$. Thus, the smoothing factor measures the rate at which the high wavenumber components of error are reduced by relaxation. The smoothing factor for the Gauss-Seidel relaxation (with lexicographic ordering) is $\bar{\mu} = 0.5$ for the second-order discretization (11) [1] and $\bar{\mu} = 0.464$ for the fourth-order MV discretization (16) [6].

2.8 Measuring convergence and computational work

In a convergent iterative method the residual r^h tends to zero as the solution converges, since the numerical solution \tilde{u}^h tends to the exact solution u^h of the discrete problem. However, the solution error e^h does not tend to zero, since it measures the difference between the solutions of the discrete and continuous problems. Comparing the definitions of the residual (6) and truncation error (20) we see that an appropriate measure of the convergence is the size of the residual relative to that of the truncation error. We say the problem is *solved to the level of truncation error* when $\|r^h\| \leq \|\tau^h\|$ in an appropriate norm. When this occurs, the solution error $e^h = u - \tilde{u}^h$ is typically close to its final value, i.e., $\|e^h\| \approx \|u - u^h\|$. We will use the l_∞ norm throughout, e.g.,

$$\|e^h\| = \|e^h\|_\infty = \max_{1 < i, j < n} |e_{i,j}^h|. \quad (29)$$

Measuring computational work is more problematic, since the execution time of a code depends heavily on the hardware and software used (and on the skill and approach of the programmer). Here we choose a simple approximate measure of work, counting floating point operations (flops) as estimated by the MATLAB function `flops`. Since this gives only a rough estimate of the true number of operations (and may not correlate directly with actual execution time), caution should be used in interpreting these results.

3 Description of the Multigrid Algorithms

Formulating a multigrid algorithm requires many choices: discretization scheme, relaxation method and ordering, grid transfer operators, and control algorithm (e.g., V-Cycles or FMG). Each choice may affect the accuracy or efficiency of the resulting algorithm. Schaffer[5] considered many such algorithms; here we investigate a more limited set, but in more detail.

We consider three types of V-Cycles:

- **MGV2** uses the second-order discretization L_2^h on all levels, with residuals transferred by injection or full weighting,
- **MGV4** uses the fourth-order discretization L_M^h on all levels, with residuals transferred by full weighting,
- **MGV4F** uses the fourth-order discretization L_M^h on the finest level and the second-order discretization L_2^h on all other levels, with residuals transferred by full weighting.

Each V-Cycle uses two relaxation sweeps on the downward (fine-to-coarse) branch and one sweep on the upward (coarse-to-fine) branch; all relaxation is Gauss-Seidel with lexicographic ordering. Since we will be using bicubic interpolation for the FMG initial interpolation (where the additional accuracy is needed, even with the second-order discretization), we will also use bicubic interpolation for the coarse-to-fine transfer of corrections. These V-Cycles are used as building blocks in the FMG algorithms below; the latter two—which use the fourth-order discretization—are also used as iterative solution algorithms (given an initial approximation on the finest grid, perform V-Cycles repeatedly until converged).

We also consider three types of 2-FMG algorithms:

- **FMG4F** uses two **MGV2** cycles (using injection) on each level below the finest, and then three **MGV4F** cycles on the finest level (thus using the fourth-order discretization L_M^h on the finest level only),
- **FMG4C** uses two **MGV4F** cycles on each level, in each of which “finest” is taken as “current finest”, i.e., the finest level yet reached (thus using the fourth-order discretization L_M^h on the current finest level only),
- **FMG4** uses two **MGV4** cycles on each level (thus using the fourth-order discretization L_M^h on every level).

Each of these FMG algorithms starts on an 8×8 grid and uses bicubic FMG initial interpolation (i.e., the first solution transfer to the next finer grid).

Finally, we also consider a fourth-order tau extapolation algorithm **TAU4**, defined as follows:

1. Do 2-FMG using two **MGV2** cycles (using full weighting) on each level below the finest level and interpolate u to the finest level,
2. Do two relaxation sweeps on the finest level perform tau extrapolation to get the extrapolated coarse-grid equation (26),
3. Solve (26) using two **MGV2** cycles (using full weighting),
4. Correct the fine-grid solution via

$$\tilde{u}^h \leftarrow \tilde{u}^h + I_{2h}^h(\bar{u}^{2h} - \hat{I}_h^{2h}\tilde{u}^h), \quad (30)$$

5. Repeat steps 2–4.

Note that no fine-grid relaxation is performed after the final interpolation.

4 Numerical Results and Comparisons

This section presents the numerical results and comparisons for the six fourth-order algorithms described above. The test problem has the exact solution

$$u(x, y) = \sin(\pi x) \cos(3\pi y). \quad (31)$$

Thus, the data for the tests consists of the forcing

$$f(x, y) = Lu = u_{xx} + u_{yy} = -10\pi^2 u(x, y), \quad (32)$$

together with boundary values computed from (31). We measure the performance of each algorithm for this problem for values of the finest mesh size from $h = \frac{1}{8}$ to $h = \frac{1}{64}$ (i.e., $n = 8$ to $n = 64$ grid intervals in each direction), using the discrete l_∞ norm (29) to measure the solution error e^h , residual r^h , and/or the truncation error τ^h .

4.1 Comparison of the V-Cycle algorithms

The results of running the fourth-order V-Cycle algorithms **MGV4** and **MGV4F** are summarized in Table 1 and 2, respectively. In each case, V-Cycles were repeated until the problem was solved to the level of truncation error ($\|r^h\| \leq \|\tau^h\|$); the number of cycles required and the norms of the resulting error and residual are shown in the tables, along with the corresponding number of flops (as counted by MATLAB). Also shown is the average convergence factor μ_{avg} , defined as $(\|r_{\text{final}}\|/\|r_{\text{initial}}\|)^{1/(s-1)}$, where r_{initial} and r_{final} are the dynamic residuals associated with the first and last fine-grid sweeps, respectively, and s is the total number of sweeps executed on the finest grid. Note that the observed values of μ_{avg} compare favorably with the predicted multigrid smoothing factor $\bar{\mu} = 0.464$ for the MV discretization.

The tables show that the two V-Cycle algorithms achieve about the same final error (as they should, since they use the same discretization on the finest grid). As shown in the third column of each table, the ratio of errors $\|e^{2h}\|/\|e^h\|$ in each case is about 16 (as it should be for a fourth-order discretization). This same conclusion—that these methods exhibit fourth-order convergence—is shown graphically in Figure 1, where the slope of the curve $\|e^h\|$ as a function of h on a log-log plot is nearly 4 for both algorithms. Both algorithms were able to solve to the level of truncation error, usually in the same number of cycles (**MGV4F** took one fewer cycle than **MGV4** when $h = 1/16$). However, with the same number of cycles, **MGV4F** produced slightly smaller residuals, as shown in Figure 2.

The operations counts (flops) reported in the tables grow with decreasing h as expected, but do not reflect the fact that **MGV4** takes more operations than **MGV4F**, since the former uses the more complicated MV discretization on all grids (rather than just the finest level). Thus, we conclude that the flops count returned by MATLAB is not an accurate measure of the computational work here. A more realistic comparison can be obtained

Table 1: **MGV4** Results

h	$\ e^h\ $	ratio	cycles	$\ r^h\ $	$\ \tau^h\ $	μ_{avg}	flops
$\frac{1}{8}$	9.944(-3)	–	2	4.164(-1)	4.938(-1)	0.479	13362
$\frac{1}{16}$	4.730(-4)	21.02	4	7.826(-3)	3.426(-2)	0.547	61469
$\frac{1}{32}$	3.163(-5)	14.95	5	1.227(-3)	2.197(-3)	0.469	1556550
$\frac{1}{64}$	1.823(-6)	17.35	7	3.622(-5)	1.382(-4)	0.439	9035169

Table 2: **MGV4F** Results

h	$\ e^h\ $	ratio	cycles	$\ r^h\ $	$\ \tau^h\ $	μ_{avg}	flops
$\frac{1}{8}$	7.893(-3)	–	2	9.762(-2)	4.938(-1)	0.480	16569
$\frac{1}{16}$	4.678(-4)	16.87	3	2.541(-2)	3.426(-2)	0.447	71757
$\frac{1}{32}$	2.828(-5)	16.54	5	4.661(-4)	2.197(-3)	0.376	1608080
$\frac{1}{64}$	1.758(-6)	16.08	7	8.906(-6)	1.382(-4)	0.418	9287356

directly as follows. From the equations for relaxation and residuals in section 2.6 we find that the second-order discretization L_2^h requires 7 flops/point for relaxation and 8 flops/point for computing the residual; the corresponding counts for the fourth-order discretization L_M^h are 19 and 20 flops/point, respectively. Using these in the V-Cycle algorithm (with three relaxation sweeps per level and a large number of grid levels) and ignoring the work associated with grid transfers we obtain the following operation counts per cycle (per point on the finest grid):

$$\mathbf{MGV2} : \quad (7 \cdot 3 + 8) \left(1 + \frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \dots \right) = \frac{116}{3} \text{ flops/point} \quad (33)$$

$$\mathbf{MGV4} : \quad (19 \cdot 3 + 20) \left(1 + \frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \dots \right) = \frac{308}{3} \text{ flops/point} \quad (34)$$

$$\mathbf{MGV4F} : \quad (19 \cdot 3 + 20) + (7 \cdot 3 + 8) \left(\frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \dots \right) = \frac{260}{3} \text{ flops/point} \quad (35)$$

Thus, the **MGV4F** algorithm should take only about 85% as much work as the **MGV4** algorithm; since it also produces slightly smaller residuals for the same number of cycles, it appears to be the better of the two.

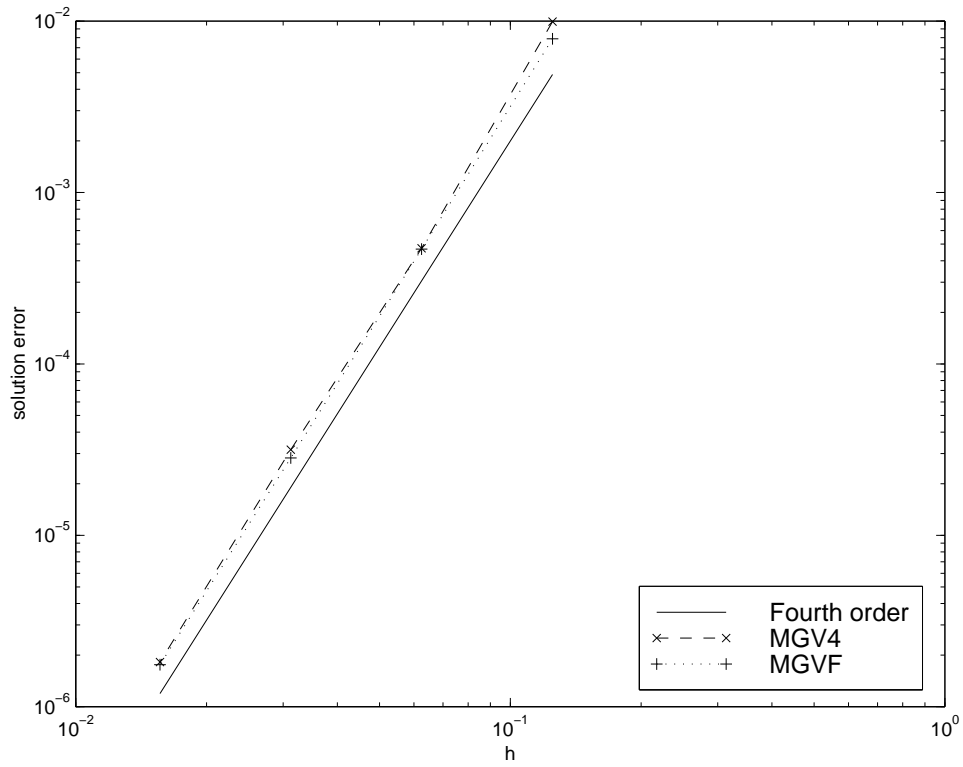


Figure 1: Solution error $\|e^h\|$ for the V-Cycle algorithms

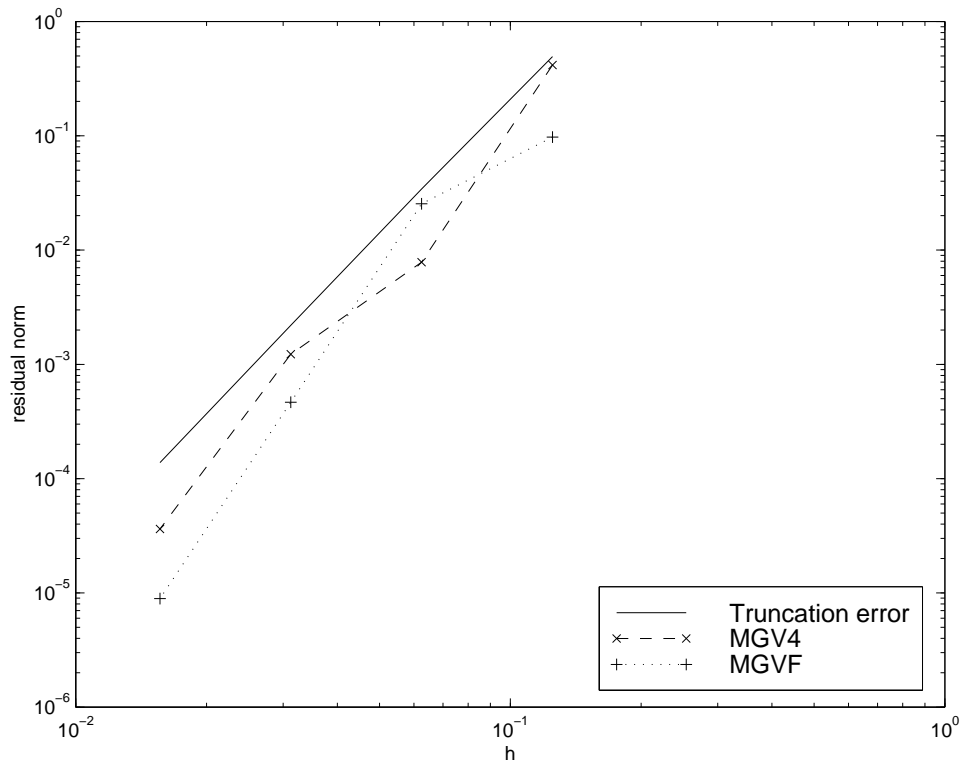


Figure 2: Residual norm $\|r^h\|$ for the V-Cycle algorithms

Table 3: **FMG4** Results

h	$\ e^h\ $	ratio	$\ r^h\ $	$\ \tau^h\ $	flops
$\frac{1}{8}$	9.944(-3)	–	4.164(-1)	4.938(-1)	26445
$\frac{1}{16}$	4.410(-4)	22.60	1.407(-2)	3.426(-2)	153720
$\frac{1}{32}$	2.609(-5)	16.87	6.403(-3)	2.197(-3)	700863
$\frac{1}{64}$	1.593(-6)	16.38	1.666(-3)	1.382(-4)	2974844

Table 4: Solution errors for **FMG4** (modified) with $h = \frac{1}{64}$

Method	$\ e^h\ $
1-FMG with 1 V-Cycle on the Finest Grid	7.584e-5
1-FMG with 2 V-Cycles on the Finest Grid	8.548e-6
2-FMG with 1 V-Cycle on the Finest Grid	7.535e-6
2-FMG with 2 V-Cycles on the Finest Grid	1.593e-6

4.2 Comparison of the FMG algorithms

Table 3 summarizes the results of running the FMG algorithm **FMG4** (which uses the fourth-order discretization L_M^h on every grid). It should be noted that while the errors are comparable to those obtained with the V-Cycle algorithms (indeed, they are a bit smaller) and exhibit the ratio $\|e^{2h}\|/\|e^h\| \approx 16$ as appropriate for fourth-order convergence, the final residual norms are not below the truncation error. Also, Table 4 shows why we chose two V-Cycles per level in the **FMG4** algorithm: with only one V-Cycle on the coarse and/or finest level, the final solution error is noticeably larger.

Table 5 summarizes the results of running the algorithm **FMG4C** (which uses the fourth-order discretization L_M^h on only the current finest grid in the FMG algorithm). The results are quite similar to those from **FMG4**.

Table 6 summarizes the results of running the algorithm **FMG4F** (which uses the fourth-order discretization L_M^h on only the finest grid in the FMG algorithm). Again, the results are quite similar to those from **FMG4**. Note, however, that in this case the algorithm uses three cycles on the finest level (instead of the two cycles used on all other levels). This was needed to obtain fourth-order convergence: Table 7 shows that with only two V-Cycles on the finest level, the convergence is only second-order. One interpretation of this result is that the initial approximation on the finest grid is better in the **FMG4C** algorithm (which produces

Table 5: **FMG4C** Results

h	$\ e^h\ $	ratio	$\ r^h\ $	$\ \tau^h\ $	flops
$\frac{1}{8}$	7.893(-3)	–	9.762(-2)	4.938(-1)	33983
$\frac{1}{16}$	4.645(-4)	16.99	2.446(-3)	3.426(-2)	184578
$\frac{1}{32}$	2.859(-5)	16.25	5.407(-3)	2.197(-3)	821741
$\frac{1}{64}$	1.713(-6)	16.69	2.054(-3)	1.382(-4)	3084562

Table 6: **FMG4F** Results

h	$\ e^h\ $	ratio	$\ r^h\ $	$\ \tau^h\ $	flops
$\frac{1}{8}$	6.784(-3)	–	4.338(-3)	4.938(-1)	41369
$\frac{1}{16}$	4.589(-4)	14.78	1.193(-3)	3.426(-2)	203620
$\frac{1}{32}$	2.868(-5)	16.00	1.146(-3)	2.197(-3)	884803
$\frac{1}{64}$	1.907(-6)	15.04	8.429(-4)	1.382(-4)	3681014

it using **MGV4F** cycles on each of the coarser levels) than in the **FMG4F** algorithm (which produces it using **MGV2** cycles on each of the coarser levels). This additional V-Cycle in the **FMG4F** algorithm adds extra work, but also results in slightly lower final residuals (but still not below the level of truncation error).

Figures 3 and 4 show some of the above results graphically. Each of the three FMG algorithms achieves fourth-order convergence, but none of them consistently solves below the level of truncation error. In summary, there is no clear winner between the three FMG algorithms considered here: all three work well and the differences in accuracy achieved are small.

Table 7: Solution errors for **FMG4F** with only two V-Cycles on the finest level

h	$\ e^h\ $	ratio
$\frac{1}{8}$	7.893(-3)	–
$\frac{1}{16}$	5.447(-4)	14.49
$\frac{1}{32}$	7.592(-5)	7.17
$\frac{1}{64}$	1.582(-5)	4.80

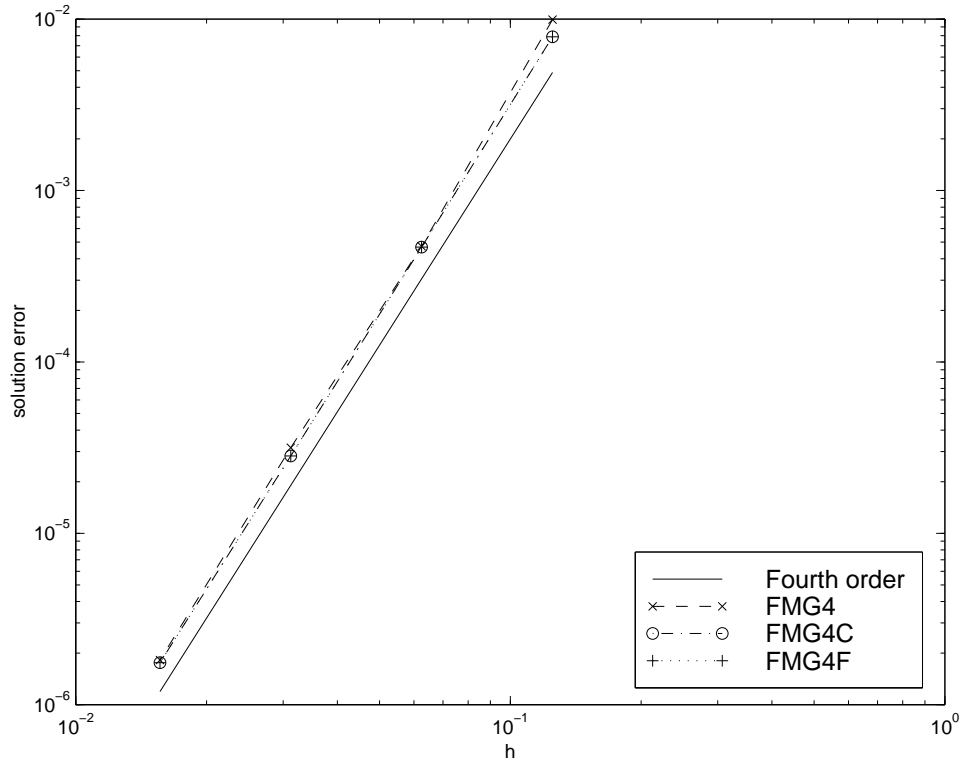


Figure 3: Solution error $\|e^h\|$ for the FMG algorithms

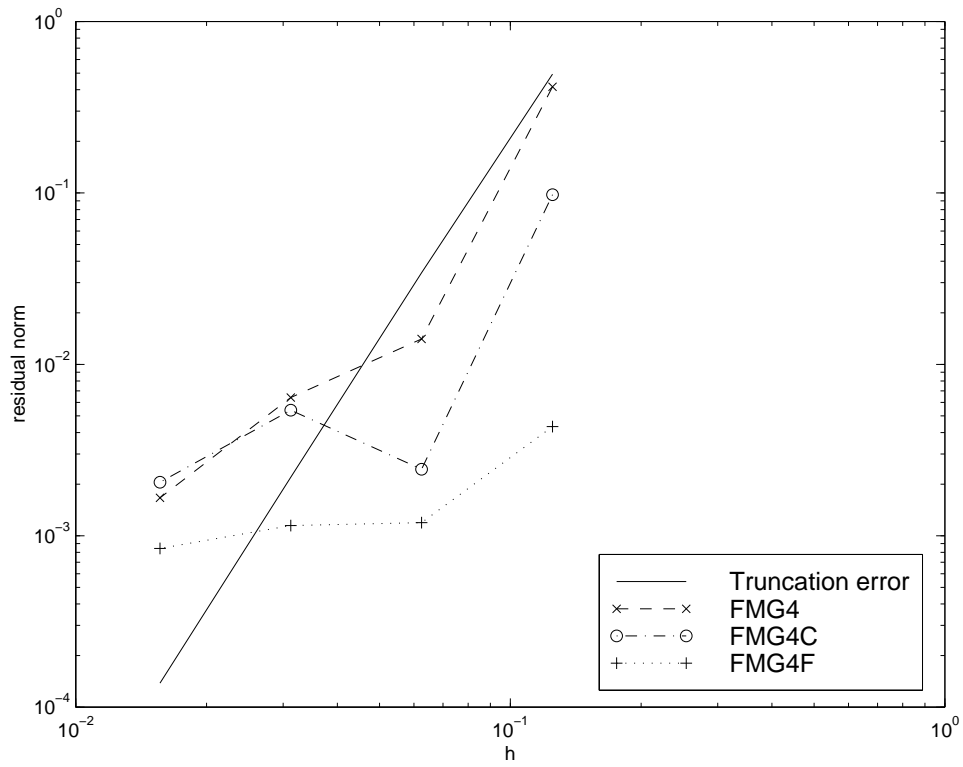


Figure 4: Residual norm $\|r^h\|$ for the FMG algorithms

Table 8: **TAU4** Results

h	$\ e^h\ $	ratio
$\frac{1}{16}$	2.366(-3)	–
$\frac{1}{32}$	1.685(-4)	14.04
$\frac{1}{64}$	1.059(-5)	15.90

4.3 Numerical results from tau extrapolation

Table 8 summarizes the results of running the fourth-order tau extrapolation algorithm **TAU4**. This method successfully achieves fourth-order convergence, but the accuracy achieved using the tau discretization is less than that from the MV discretization, as reported by Schaffer[5] (the errors are nearly an order of magnitude larger). In addition, obtaining this fourth-order convergence depends on doing two tau-correction cycles on the finest grid; with only one such cycle (steps 2–4 in the **TAU4** algorithm) the convergence is only second-order. Due to this extra complexity and the reduced accuracy of the resulting solution, tau extrapolation will not be considered for application to the adaptive multigrid hurricane model MUDBAR in the next section.

5 Application to the MUDBAR Hurricane Model

The MUDBAR hurricane model[2, 3] is a barotropic numerical model designed for studying hurricane dynamics and tropical cyclone track prediction. The model solves the nondivergent barotropic vorticity equation

$$\frac{\partial \zeta}{\partial t} + \frac{\partial(\zeta, \psi)}{\partial(x, y)} + \beta \frac{\partial \psi}{\partial x} = 0 \quad (36)$$

for the vorticity ζ , and the Poisson problem

$$\nabla^2 \psi = \zeta \quad (37)$$

for the streamfunction ψ , using an adaptive multigrid method to refine the mesh around the vortex as it moves and evolves. The model is formulated on a rectangle, with Dirichlet boundary conditions for the Poisson problem (37). The original version of the model used second-order centered finite differences to discretize both equations. Improved accuracy could presumably be obtained by using fourth-order discretizations for both equations. What improvement might be possible by solving just the Poisson problem (37) using a fourth-order discretization?

To answer this question, we implemented the **FMG4F** and **FMG4C** algorithms in the model, retaining the second-order discretization of (36). The test problem consisted of the motion of a vortex as described in [2], using a single uniform grid (with underlying coarse grids for the multigrid solver). The accuracy of the model was measured by the track error (l_2 norm of the position error relative to a high-resolution reference run) at $t = 24$ h, $t = 48$ h, $t = 72$ h, and averaged over the 72 h model run. The computational work was measured by the CPU time required on a SUN Ultra10 workstation.

Table 9 presents results from a set of experiments using the mesh size $h = 32$ km (128×128 grid intervals), and Table 10 presents analogous results using the mesh size $h = 16$ km (256×256 grid intervals). Results are given for various algorithms, listed in order of increasing CPU time. The baseline is the standard second-order 1-FMG method **FMG2** of the original model: second-order discretization L_2^h , Gauss-Seidel relaxation with red-black ordering, full weighting of residuals, bilinear interpolation of corrections, and bicubic initial FMG interpolation. Other algorithms considered include **FMG4F** and **FMG4C**, modified by using either the 1-FMG or 2-FMG algorithm and/or additional V-Cycles on the finest grid level.

The most notable feature of these results is the similarity of the errors: each algorithm gives essentially the same accuracy. Thus, increasing the discretization accuracy for the Poisson equation (37) alone is not enough: to increase the overall accuracy of the model, it will be necessary to use a higher-order discretization of the vorticity equation (36) as well.

Table 9: Results of MUDBAR on grid with $h = 32$ km

Method	Additional V-Cycles	track error (km)				CPU time (s)
		$t = 24$ h	$t = 48$ h	$t = 72$ h	mean	
FMG2	–	53.970	24.622	52.336	40.123	18
FMG4F with 1-FMG	–	54.693	24.788	51.275	39.802	20
FMG4C with 1-FMG	–	55.640	25.952	52.242	40.297	25
FMG2	1	54.875	24.014	50.368	39.514	28
FMG4F with 2-FMG	–	55.122	24.279	50.577	39.695	33
FMG4F with 2-FMG	1	55.115	24.253	50.561	39.687	43
FMG4F with 2-FMG	2	55.116	24.253	50.561	39.686	53
FMG4F with 2-FMG	3	55.116	24.253	50.561	39.686	64

Table 10: Results of MUDBAR on grid with $h = 16$ km

Method	Additional V-Cycles	track error (km)				CPU time (s)
		$t = 24$ h	$t = 48$ h	$t = 72$ h	mean	
FMG2	–	4.113	4.413	11.714	7.651	162
FMG4F with 1-FMG	–	4.137	4.457	11.675	7.666	207
FMG4C with 1-FMG	–	4.304	4.673	11.709	7.750	216
FMG2	1	4.196	4.518	11.680	7.692	286
FMG4F with 2-FMG	–	4.217	4.551	11.687	7.704	350

6 Conclusions

We have considered six fourth-order multigrid methods for the Poisson problem. All achieved fourth-order convergence for the test problem. The five algorithms based on the fourth-order compact Mehrstellen Verfahren (MV) discretization produced solution errors which were nearly an order of magnitude smaller than those produced by the fourth-order tau extrapolation algorithm. The MV algorithms are also somewhat simpler to implement. Observed smoothing rates for relaxation with the MV discretization closely matched those predicted by the theoretical multigrid smoothing factor.

None of the algorithms solved to the level of truncation error (residual norm less than truncation error norm) with a fixed number of cycles: the two V-Cycle algorithms required an increasing number of cycles with decreasing mesh size, and the three FMG algorithms required extra V-Cycles (one required an extra V-Cycle to achieve fourth-order convergence). Thus, while fourth-order accuracy may indeed be achieved using multigrid methods in a variety of ways, we have not yet found a specific fixed algorithm which guarantees solution to the level of truncation error as does the standard 1-FMG algorithm for second-order discretizations.

Using fourth-order multigrid algorithms based on the MV discretization in the adaptive multigrid hurricane model MUDBAR produced fourth-order accuracy for the Poisson equation relating streamfunction and vorticity. However, the overall accuracy of the model showed negligible improvement. We attribute this to the fact that the other model equation (for advection of vorticity) retained a second-order discretization; improvement in overall accuracy will require implementing a fourth-order discretization for that equation as well.

References

- [1] Brandt, A., 1977: Multi-level adaptive solutions to boundary value problems. *Math. Comp.*, **33**, pp 333–390.
- [2] Fulton, S. R., 1997: A comparison of multilevel adaptive methods for hurricane track prediction. *Electronic Transactions on Numerical Analysis*, **6**, 120–132. <http://etna.mcs.kent.edu/>.
- [3] Fulton, S. R., 1999: An adaptive multigrid barotropic model for tropical cyclone track prediction. To be submitted to *Mon. Wea. Rev.*.
- [4] Fulton, S.R., P.E. Ciesielski, and W.H. Schubert; Multigrid methods for elliptic problems: a review, *Mon. Wea. Rev.*, **114**, 943–959.
- [5] Schaffer, S., 1984: Higher-Order Multi-Grid Methods. *Math. Comput.*, **43**, 89–115.
- [6] Taft, R., 1987: A comparison of two multigrid methods for solving the Poisson problem. Atmos. Sci. Paper No. 415, Dept. of Atmospheric Science, Colorado State University.

A Appendix: Accuracy of the MV Discretization

We seek the truncation error of the MV discretization, that is,

$$\tau^h(u) := L_M^h u - I^h f, \quad (38)$$

where $f = Lu = u_{xx} + u_{yy}$ and the difference and averaging operators are given by

$$L_M^h u := \frac{1}{6h^2} \begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix} u \quad (39)$$

and

$$I^h f := \frac{1}{12} \begin{bmatrix} & 1 & \\ 1 & 8 & 1 \\ & 1 & \end{bmatrix} f \quad (40)$$

as in section 2.6.2. To show that the discretization is fourth-order, we will expand L_M^h and I^h using Taylor series, with the assumption that $u \in C^6(\Omega)$ and thus $f \in C^4(\Omega)$.

For the Taylor expansion of the difference operator L_M^h we rewrite (39) as

$$L_M^h u = \frac{1}{12h^2} \left\{ \begin{bmatrix} 1 & -2 & 1 \\ 10 & -20 & 10 \\ 1 & -2 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 10 & 1 \\ -2 & -20 & -2 \\ 1 & 10 & 1 \end{bmatrix} \right\} u = (J_y^h D_{xx}^h + J_x^h D_{yy}^h) u, \quad (41)$$

where we have introduced the one-dimensional difference operators

$$D_{xx}^h u := \frac{1}{h^2} \begin{bmatrix} 1 & -2 & 1 \end{bmatrix} u, \quad D_{yy}^h u := \frac{1}{h^2} \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} u \quad (42)$$

and averaging operators

$$J_y^h v := \frac{1}{12} \begin{bmatrix} 1 \\ 10 \\ 1 \end{bmatrix} v, \quad J_x^h v := \frac{1}{12} \begin{bmatrix} 1 & 10 & 1 \end{bmatrix} v. \quad (43)$$

From Taylor's theorem we have the expansions

$$\begin{aligned} u(x-h, y) &= u(x, y) - hu^{(1,0)}(x, y) + \frac{h^2}{2!} u^{(2,0)}(x, y) - \frac{h^3}{3!} u^{(3,0)}(x, y) \\ &+ \frac{h^4}{4!} u^{(4,0)}(x, y) - \frac{h^5}{5!} u^{(5,0)}(x, y) + \frac{h^6}{6!} u^{(6,0)}(\hat{x}^-, y) \end{aligned} \quad (44)$$

for some $\hat{x}^- \in (x-h, x)$, and

$$\begin{aligned} u(x+h, y) &= u(x, y) + hu^{(1,0)}(x, y) + \frac{h^2}{2!} u^{(2,0)}(x, y) + \frac{h^3}{3!} u^{(3,0)}(x, y) \\ &+ \frac{h^4}{4!} u^{(4,0)}(x, y) + \frac{h^5}{5!} u^{(5,0)}(x, y) + \frac{h^6}{6!} u^{(6,0)}(\hat{x}^+, y) \end{aligned} \quad (45)$$

for some $\hat{x}^+ \in (x, x+h)$. Here we are using the notation $u^{(m,n)} = \partial^{m+n}u/\partial x^m \partial y^n$ for partial derivatives. Substituting (44) and (45) into (42) and using the Intermediate Value Theorem to combine the remainder terms yields

$$D_{xx}^h u(x, y) = u_{xx}(x, y) + \frac{h^2}{12} u^{(4,0)}(x, y) + \frac{h^4}{360} u^{(6,0)}(\hat{x}, y) \quad (46)$$

for some $\hat{x} \in (x-h, x+h)$. Similarly,

$$D_{yy}^h u(x, y) = u_{yy}(x, y) + \frac{h^2}{12} u^{(0,4)}(x, y) + \frac{h^4}{360} u^{(0,6)}(x, \hat{y}) \quad (47)$$

for some $\hat{y} \in (y-h, y+h)$. Since $u \in C^6(\Omega)$, we have $u^{(6,0)}(\hat{x}, y) \rightarrow u^{(6,0)}(x, y)$ and $u^{(0,6)}(x, \hat{y}) \rightarrow u^{(0,6)}(x, y)$ as $h \rightarrow 0$, so the last terms in (46) and (47) are $O(h^4)$.

Analogous calculations for the generic one-dimensional averaging operators

$$A_x^h v := \frac{1}{2+a} \begin{bmatrix} 1 & a & 1 \end{bmatrix} v, \quad A_y^h v := \frac{1}{2+a} \begin{bmatrix} 1 \\ a \\ 1 \end{bmatrix} v, \quad (48)$$

for any function $v \in C^4(\Omega)$ and any positive number a yields

$$A_x^h v(x, y) = v(x, y) + \frac{h^2}{2+a} v^{(2,0)}(x, y) + \frac{h^4}{12(2+a)} v^{(4,0)}(\hat{x}, y) \quad (49)$$

for some $\hat{x} \in (x-h, x+h)$ and

$$A_y^h v(x, y) = v(x, y) + \frac{h^2}{2+a} v^{(0,2)}(x, y) + \frac{h^4}{12(2+a)} v^{(0,4)}(x, \hat{y}) \quad (50)$$

for some $\hat{y} \in (y-h, y+h)$. With the value $a = 10$ we see that (43) reduces to

$$J_x^h v(x, y) = v(x, y) + \frac{h^2}{12} v^{(2,0)}(x, y) + \frac{h^4}{144} v^{(4,0)}(\hat{x}, y) \quad (51)$$

and

$$J_y^h v(x, y) = v(x, y) + \frac{h^2}{12} v^{(0,2)}(x, y) + \frac{h^4}{144} v^{(0,4)}(x, \hat{y}) \quad (52)$$

for some \hat{x} and \hat{y} [not necessarily the same as in (46) and (47)]. Analogous expansions hold with one or two less terms if $v \in C^2(\Omega)$ or $v \in C(\Omega)$. Using (46), (47), (51), and (52) in (41) then yields the expansion

$$\begin{aligned} L_M^h u = u_{xx} + u_{yy} &+ \frac{h^2}{12} \left[u^{(4,0)} + 2u^{(2,2)} + u^{(0,4)} \right] + \frac{h^4}{72} \left[u^{(4,2)}(\hat{x}_1, \hat{y}_1) + u^{(2,4)}(\hat{x}_2, \hat{y}_2) \right] \\ &+ \frac{h^4}{360} \left[u^{(6,0)}(\hat{x}_3, \hat{y}_3) + u^{(0,6)}(\hat{x}_4, \hat{y}_4) \right], \end{aligned} \quad (53)$$

where u and its derivatives are evaluated at (x, y) except where noted.

For the Taylor expansion of the averaging operator I^h we rewrite (40) as

$$I^h f = \frac{1}{2} \left\{ \frac{1}{6} \begin{bmatrix} 0 & 0 & 0 \\ 1 & 4 & 1 \\ 0 & 0 & 0 \end{bmatrix} + \frac{1}{6} \begin{bmatrix} 0 & 1 & 0 \\ 0 & 4 & 0 \\ 0 & 1 & 0 \end{bmatrix} \right\} u = \frac{1}{2} (I_x^h + I_y^h) f, \quad (54)$$

where we have introduced the one-dimensional averaging operators

$$I_x^h f := \frac{1}{6} \begin{bmatrix} 1 & 4 & 1 \end{bmatrix} f, \quad I_y^h f := \frac{1}{6} \begin{bmatrix} 1 \\ 4 \\ 1 \end{bmatrix} f. \quad (55)$$

Using (49) and (50) with the value $a = 4$ we obtain the expansions

$$I_x^h f(x, y) = f(x, y) + \frac{h^2}{6} f^{(2,0)}(x, y) + \frac{h^4}{72} f^{(4,0)}(\hat{x}, y) \quad (56)$$

and

$$I_y^h f(x, y) = f(x, y) + \frac{h^2}{6} f^{(0,2)}(x, y) + \frac{h^4}{72} f^{(0,4)}(x, \hat{y}) \quad (57)$$

for some $\hat{x} \in (x - h, x + h)$ and $\hat{y} \in (y - h, y + h)$. Using these expansions in (54) yields

$$I^h f = f + \frac{h^2}{12} [f^{(2,0)} + f^{(0,2)}] + \frac{h^4}{144} [f^{(4,0)}(\hat{x}, y) + f^{(0,4)}(x, \hat{y})], \quad (58)$$

where f and its derivatives are evaluated at (x, y) except where noted.

Finally, combining (53) and (58) with (38) and using $f = u_{xx} + u_{yy}$ we obtain the truncation error

$$\begin{aligned} \tau^h(u) &= \frac{h^4}{720} \left\{ 10 [u^{(4,2)}(\hat{x}_1, \hat{y}_1) + u^{(2,4)}(\hat{x}_2, \hat{y}_2)] + 2 [u^{(6,0)}(\hat{x}_3, \hat{y}_3) + u^{(0,6)}(\hat{x}_4, \hat{y}_4)] \right. \\ &\quad \left. - 5 [f^{(4,0)}(\hat{x}_5, y) + f^{(0,4)}(x, \hat{y}_6)] \right\} \\ &= \frac{h^4}{720} \left\{ 5 [u^{(4,2)}(x, y) + u^{(2,4)}(x, y)] - 3 [u^{(6,0)}(x, y) + u^{(0,6)}(x, y)] \right\} + o(h^4), \quad (59) \end{aligned}$$

showing that the MV discretization is indeed fourth-order accurate.