MGLab: AN INTERACTIVE MULTIGRID ENVIRONMENT

James Bordner
Faisal Saied
Department of Computer Science
University of Illinois at Urbana-Champaign

SUMMARY

MGLab is a set of Matlab functions that defines an interactive environment for experimenting with multigrid algorithms. The package solves two-dimensional elliptic partial differential equations discretized using either finite differences or finite volumes, depending on the problem. Built-in problems include the Poisson equation, the Helmholtz equation, a convection-diffusion problem, and a discontinuous coefficient problem. A number of parameters controlling the multigrid V-cycle can be set using a point-and-click mechanism. The menu-based user interface also allows a choice of several Krylov subspace methods, including CG, GMRES(k), and Bi-CGSTAB, which can be used either as stand-alone solvers or as multigrid acceleration schemes. The package exploits Matlab's visualization and sparse matrix features, and has been structured to be easily extensible.

WHAT IS MGLab?

MGLab is an interactive environment based on Matlab Version 4.0 for solving elliptic partial differential equations using multigrid algorithms. A graphical user interface (GUI) enables the user to select a problem, set parameters for the multigrid V-cycle, optionally choose a Krylov subspace accelerator, and visualize the results. MGLab is written in Matlab [1] which has greatly simplified the programming, but has also led to some loss of efficiency in a few respects.

There are a number of very good introductions to multigrid methods that can be used in conjunction with MGLab including [2], [3], and [4]. The numerical treatment of elliptic partial differential equations is discussed in [5], and the finite volume method for discretizing elliptic problems is described in [6]. Some of the experiments described in [7] have been included in MGLab as demos. Some of the software that address similar issues are described in [8], [9], [10], [11], [12], [13], and [14]. The basic linear algebra concepts needed for a number of the components of MGLab, including the iterative solvers, are discussed in [15], [16], [17], [18], [19], [20], [21], [22], and [23]. Other references that may be useful background reading for MGLab users include [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], and [34].

THE GRAPHICAL USER INTERFACE

The interface between MGLab and the user is a menu structure with menu items selectable using a point-and-click mechanism. Menu items are grouped according to their function depending on whether they relate to the partial differential equation, the solver, multigrid parameters, visualization of results, or built-in demos. Top level menu choices and their sub-menus are outlined below.

MGLab

Run
Show Params
Version Info
Reset
Restart
Quit

The submenus in the MGLab top-level menu item control the basic behavior of the package, including solving the currently selected problem, displaying the currently selected parameters, and restarting MGLab with the default parameters.

Problem

Poisson
Helmholtz ▷
Convection-Diffusion ▷
Cut-Square ▷
Problem Size ▷

The submenus in the Problem top-level menu item select which partial differential equation to solve, and further submenus are available for setting problem-dependent parameters. The problem size can also be set here.

Solver

V-Cycle
CG
Bi-CGSTAB
CGS
GMRES(k) ▷
SOR ▷
Full-Multigrid
Preconditioner ▷
Stopping Criteria ▷

The submenus in the Solver top-level menu item are used to select the solver, choose a preconditioner if desired, and set the stopping criteria. The GMRES menu item has a submenu for choosing the GMRES restart parameter, and the SOR menu item has a submenu for choosing the SOR relaxation parameter.

MG-Parameters	
Number of Levels	▷
Smoother	▷
Restriction	▷
Prolongation	▷
Coarse-grid Solver	▷
Coarse-grid Operator	▷
MG Cycle	▷

The submenus in the MG-Parameters top-level menu item are used to set various multigrid parameters, including the number of grid levels, the smoother, the restriction and prolongation operators, the solver for the coarse grid problem, the method for generating the operators on the coarser grids, and the type of multigrid cycle such as the V-cycle or the W-cycle.

Visualize	
Convergence History	
Computed Solution (surf)	
Computed Solution (pcolor)	
X-Axis	▷
Y-Axis	▷

The submenus in the Visualize top-level menu item are used to view the results after solving a problem. The convergence history can be plotted, the scaling along the x and y axes for the convergence history plot can be chosen, and the numerical solution can be displayed either as a surface plot or a contour plot.

Demos
Smoothers
Fourier analysis
Truncation error

The submenus in the Demos top-level menu item select and run demonstrations that illustrate specific properties of multigrid methods, such as the behavior of different smoothers, how the errors after the coarse grid correction and after the post-smoothings in the V-cycle behave in physical and Fourier space, and how the truncation error compares with the discrete residual.

ELLIPTIC PROBLEMS

The built-in test problems in the current version of MGLab are restricted to two-dimensional elliptic partial differential equations on rectangular domains.

$$\begin{aligned} \nabla \cdot (a \nabla u) + b \cdot \nabla u + cu = f \quad \text{in} \quad \Omega, \\ u = g \quad \text{on} \quad \partial\Omega. \end{aligned} \tag{1}$$

The domain Ω is the unit square $\{(x, y) : 0 < x, y < 1\}$, and the elliptic problem is discretized using the standard 5-point stencil on a uniform mesh. Currently the test problems all have zero Dirichlet boundary conditions. The matrices are stored using Matlab's sparse storage format which ensures that matrix-vector products are efficient. Furthermore, the coarse grid problem can be solved using Matlab's built-in sparse direct solver [35] which uses graph-theoretic techniques to re-order the rows and columns of the matrix to reduce fill-in during the elimination process.

Even within the discretization and boundary condition restrictions, a number of different types of elliptic problems are possible. MGLab's test suite includes the Poisson equation, the Helmholtz equation, a convection-diffusion equation, and a discontinuous coefficient problem ("cut-square").

Poisson Equation The Poisson equation, $-\nabla^2 u = f$, is the easiest problem in MGLab to solve; the coefficient matrix of the discretized equation is both symmetric and positive definite.

Helmholtz Equation The Helmholtz equation, $-\nabla^2 u + ku = f$, is the same as the Poisson equation except for the ku term. Depending on k, this term can make the problem indefinite or complex. The parameter k can be selected by the user where $k \in \{-10, -5, -1, 0, 1, 5, 10, 10+i\}$.

Convection-Diffusion Equation The convection-diffusion equation, $-\nabla^2 u + \lambda u_x + \sigma u = f$, adds the convection term λu_x to the Helmholtz equation. This can make the coefficient matrix of the discretized problem nonsymmetric. The parameters λ and σ can be selected by the user, where $\lambda \in \{0, 10, 100, 1000\}$ and $\sigma \in \{-100, -50, 0, 5, 10, 20, 50, 100\}$.

Cut-Square Equation The cut-square equation is $-\nabla \cdot (a \nabla u) = f$, where a is a discontinuous function of x and y. Specifically, $a(x, y) = \alpha$ for $.4 \leq x, y \leq .6$, and $a(x, y) = 1$ elsewhere in Ω. The parameter α can be selected by the user where $\alpha \in \{0.001, 0.01, 0.1, 1, 10, 100, 1000\}$.

MULTIGRID PARAMETERS

MGLab is designed to solve elliptic partial differential equations using multigrid methods, with the option to embed the multigrid solver as a preconditioner in a Krylov subspace method. A number of parameters that determine the V-cycle can be set through the graphical user interface. These include the number of levels, the smoother, the number of pre- and post-smoothing sweeps, the restriction and prolongation operators, the coarse grid solver, and the type of multigrid cycle.

Number of Levels The number of grid levels can be chosen to be between 1 and 5. Note that `levels` $= 1$ corresponds to a sparse direct solver. If the chosen number of levels is too large for the current problem size it is set to the largest number possible.

Smoothers The available smoothers are weighted Jacobi, Gauss-Seidel, and Red/Black Gauss-Seidel. For the Jacobi smoother, the user can pick the weighting factor. The number of pre- and post-smoothing sweeps (ν_1, ν_2) can also be set through the Smoother submenu.

Restriction Operators The restriction operators available in MGLab are injection, half-weighting, and full-weighting. These are implemented with fairly compact code that uses Matlab's colon notation for accessing arrays.

Prolongation Operators MGLab offers bilinear and cubic interpolation as the choices for prolongation. These are currently implemented by calls to Matlab's `interp2` function.

Coarse Grid Solver The default coarse grid solver is Matlab's built-in sparse direct solver [35]. As an alternative, the user can choose to use the smoother as the coarse grid solver. This is less costly, but also less accurate.

Multigrid Cycle Although the V-cycle is the default multigrid cycle, the user can also select the W-cycle. Other cycles such as the half V-cycle or weighted V-cycle could be added easily. Full multigrid can be selected in the Solver menu.

KRYLOV SUBSPACE ACCELERATORS

The V-cycle defined through the multigrid parameters discussed previously can be used as an iterative solver on its own, or as a preconditioner for Krylov subspace methods such as CG, GMRES(k), and Bi-CGSTAB. For solving the linear system of equations $Ax = b$, these methods work with a sequence of Krylov subspaces defined by

$$K_j(r_0, A) = \mathrm{span}\{r_0, Ar_0, \ldots, A^{j-1}r_0\}. \tag{2}$$

The j-th iterate x_j is picked from

$$x_j \in x_0 + K_j(r_0, A),$$

where r_0 is the initial residual $b - Ax_0$.

Below we list some of the important properties of the methods; details of the algorithms can be found in the references given with each method. See also [19] and [22].

CG The Conjugate Gradient method is a Krylov subspace accelerator for symmetric positive definite (SPD) systems that minimizes the A-norm of the error at each iteration. The preconditioned version (PCG) requires a symmetric positive definite preconditioner. The CG method was developed by Hestenes and Stiefel [16] and is discussed in [15].

GMRES(k) The Generalized Minimum Residual method of Saad and Schultz [18], is a direct generalization of CG to matrices that are not SPD. CG takes advantage of a three-term recurrence relation which is not available in GMRES, so the number of vectors that must be stored and the number of floating point operations performed both increase with each iteration. Because of this GMRES is typically restarted every k iterations.

CGS Conjugate Gradient Squared is a variant of the Bi-Conjugate Gradient (Bi-CG) method that, unlike Bi-CG, avoids multiplication by the transpose of the matrix

A. The CGS method was proposed by Sonneveld [21]. Unlike GMRES it is not guaranteed to minimize the 2-norm of the residual, but the number of vectors required does not increase with each iteration so CGS does not need to be restarted. The convergence behavior of CGS can be very erratic.

Bi-CGSTAB This method was introduced by van der Vorst [20], and is transpose-free like CGS, but with a more regular convergence behavior.

SOR The Successive Over-Relaxation method [17] is a stationary iterative method with a relaxation parameter ω. If $\omega = 1$ then SOR reduces to the Gauss-Seidel method.

PRECONDITIONERS

The performance of iterative methods can often be enhanced with preconditioning by premultiplying the linear system $Ax = b$ by an approximate inverse M^{-1} of A:

$$M^{-1}Ax = M^{-1}b. \tag{3}$$

The multigrid V-cycle can be used as a preconditioner. In addition, even though our emphasis is on multigrid methods, MGLab allows the V-cycle preconditioner to be replaced by something else. The current preconditioners available in MGLab for the Krylov subspace methods are the V-cycle, point Jacobi, and point Gauss-Seidel. Other preconditioners such as block Jacobi, red-black Gauss-Seidel, ILU, and SSOR could be added relatively easily.

There is a standardized interface to the preconditioner that is independent of the iterative solver. The operation $z \leftarrow M^{-1}r$ is performed by the following call:

```
z = precondition(A, r).
```

The function `precondition` accesses the parameters needed to apply the preconditioner M^{-1} which is implicitly defined in terms of A. This enhances the extensibility of MGLab in the sense that adding Matlab implementations of other iterative methods would be straightforward.

VISUALIZATION OPTIONS

MGLab exploits Matlab's powerful graphics capabilities to plot the convergence history of the solution process and to visualize the computed solution. Currently we make use of the `plot`, `surf`, `pcolor`, and `contour` commands in Matlab. The visualization options are available through the graphical user interface.

SOME COMMENTS ON THE INTERNAL STRUCTURE OF MGLab

MGLab is written entirely in Matlab. One group of functions is devoted to the user interface; these make use of Matlab's `uimenu` function. Other groups of functions implement the problem generation, algorithms, and visualization in MGLab.

MGLab makes use of Matlab's `global` mechanism. This approach leads to a considerable simplification of the programming in many situations, but carries the software engineering risk of non-transparent code and the danger of subtle bugs. We have attempted to write the higher level functions such as `sp_laplace`, `Vcycle`, `pcg`, and `precondition` in a way that does not require them to see the global variables. This results in very compact and readable code, and reduces the chance of global variables being accidentally damaged. The code for `Vcycle` is shown below to illustrate this.

The "middle level" functions such as `smooth` and `restrict` access the global workspace in a disciplined manner. Some low level functions were created expressly for the purpose of accessing the globals and returning a single value so that the globals could be hidden from the higher level functions.

```
function u_out = vCycle(level, b, u_in)

% Use the zero vector for u_in as the default

if nargin == 2,
   u_in = zeros(size(b));
end

if level == coarsest(level)
   u_out   = coarse_grid_solve(level, b);
else
   u       = smooth(level, b, u_in, 'pre');
   r       = residual(level, b, u);
   b_c     = restrict(level, r);
   u_c     = vCycle(level+1, b_c);
   correct = interpolate(level, u_c);
   u       = u + correct;
   u_out   = smooth(level, b, u, 'post');
end
```

V-cycle Function

BUILT-IN DEMOS IN MGLab

MGLab has a working and extensible framework for adding numerical experiments. Currently, the numerical experiments supplied with MGLab are the following:

- A numerical study of the smoothing properties of weighted Jacobi, Gauss-Seidel, and Red/Black Gauss-Seidel, in physical and Fourier space [7]. The Fourier transforms are constructed out of Matlab's fast Fourier transform (fft).
- A numerical Fourier analysis of the complementary roles of the coarse grid correction and the smoother for a model problem.
- A comparison of the truncation error ("pde error") and the discrete residual [7]. This demo highlights the ability of multigrid methods to achieve truncation error accuracy very rapidly.

Figures 1 through 4 show the output of Demo 2. The intention of this demo is to give a visual sense for the different roles of the coarse grid correction and the (post-)smoothing. In this demo, we solve the Poisson problem on a 49×49 mesh by multigrid. The V-cycle parameters are as follows:

- 2 levels
- Gauss-Seidel smoothing
- $(\nu_1, \nu_2) = (0, 4)$, i. e. no pre-smoothing and 4 post-smoothing sweeps
- Half-weighting restriction
- Cubic interpolation
- The coarse grid solver is Matlab's built-in sparse Gaussian elimination

The initial guess was chosen so that the initial error had a mix of low and high frequencies:

$$e^{(0)}(x,y) = \sum_{j=1}^{4} \sin(10j\pi x)\sin(10j\pi y).$$

Figure 1 shows the initial error on the left, and the absolute values of the (scaled[1]) Fourier coefficients of the error on the right. Figure 2 shows the the error in the first V-cycle, after the coarse grid correction (top) and after the post-smoothing (bottom). In each case, the error is shown in "physical" space (left) and in Fourier space (right). Figures 3 and 4 are the same as Figures 2 except that they show the errors in the second and third iterations, respectively.

[1]The 2D sine transform was applied to the error on the mesh.

These figures show how the coarse grid correction and the smoother complement each other by reducing the low frequency and high frequency error components, respectively.

OBTAINING AND INSTALLING MGLab

MGLab V1.0 is currently available via anonymous ftp to casper.cs.yale.edu in the directory /mgnet/Codes/mglab. After the tar file is uncompressed it should be untarred in a subdirectory, ~*myname*`/matlab/MGLab` for example. To run MGLab, simply change to your MGLab directory, start up Matlab and type `MGLab`.

Comments and suggestions for improvements about the code are welcome; we plan to release future versions of MGLab that incorporate enhancements and bug fixes.

ACKNOWLEDGEMENTS

The work of the second author was supported in part by an NSF Research Initiation Award, NSF ASC 92 09502 RIA.

REFERENCES

[1] The MathWorks, Inc, Natick, Mass. *Matlab Reference Guide*, 1992.

[2] D. C. Jespersen. Multigrid methods for partial differential equations. In G. H. Golub, editor, *Studies in Numerical Analysis*, pages 270–318. The Mathematical Association of America, 1984.

[3] W. L. Briggs. *A Multigrid Tutorial.* SIAM, Philadelphia, 1988.

[4] A. Brandt. Multilevel adaptive solutions to boundary-value problems. *Math. Comp.*, 31:311–329, 1977.

[5] G. Birkhoff and R. E. Lynch. *Numerical Solution of Elliptic Problems.* SIAM, Philadelphia, 1984.

[6] R. S. Varga. *Matrix Iterative Analysis.* Prentice-Hall, New Jersey, 1962.

[7] F. Saied and M. J. Holst. Vector multigrid: An accuracy and performance study. Technical Report UIUCDCS-R-90-1636, Department of Computer Science, University of Illinois at Urbana-Champaign, 1990.

[8] R. E. Bank. *PLTMG: A Software Package for Solving Elliptic Partial Differential Equations. Users' Guide 7.0.* SIAM, Philadelphia, 1994.

[9] W. F. Mitchell. MGGHAT: Elliptic PDE software with adaptive refinement, multigrid and high order finite elements. In N. D. Melson, T. A. Manteuffel, and S. F. McCormick, editors, *Proceedings of the Sixth Copper Mountain Conference on Multigrid Methods, Part 2, April 4–9, 1993*, pages 439–448. NASA, 1993.

[10] W. Gropp and B. Smith. Simplified linear equation solvers users manual. Technical Report ANL-93/8, MCS Division, Argonne National Laboratoy, 1993.

[11] W. Gropp and B. Smith. User's manual for KSP data-structure-neutral codes implementing Krylov space methods. Technical Report ANL-93/30, MCS Division, Argonne National Laboratoy, 1993.

[12] J. R. Rice and R. F. Boisvert. *Solving Elliptic Problems using ELLPACK*. Springer-Verlag, New York, 1985.

[13] B. Smith. Extensible PDE solvers package user's manual. Technical Report ANL-94/40, MCS Division, Argonne National Laboratoy, 1994.

[14] Y. Saad. SPARSKIT: A basic toolkit for sparse matrix computations. Technical Report 1029, CSRD, University of Illinois at Urbana-Champaign, 1990.

[15] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, 2nd edition, 1989.

[16] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Stand.*, 49:409–435, 1952.

[17] D. M. Young. *Iterative Solution of Large Linear Systems*. Academic Press, New York, 1971.

[18] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7(3):856–869, 1986.

[19] R. W. Freund, G. H. Golub, and N. Nachtigal. Iterative solution of linear systems. *Acta Numerica*, 1:57–100, 1991.

[20] H. A. van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of non-symmetric problems. *SIAM J. Sci. Stat. Comput.*, 13:631–645, 1992.

[21] P. Sonneveld. CGS: A fast Lanczos-type solver for nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 10:36–52, 1989.

[22] R. Barrett et al. *Templates for the Solution of Linear Syatems: Building blocks for iterative methods*. SIAM Publications, Philadelphia, 1993.

[23] W. Hackbusch. *Iterative Lösung grosser schwachbesetzter Gleichungssysteme.* Teubner, Stuttgart, 1993.

[24] W. Hackbusch. *Elliptic Differential Equations.* Springer-Verlag, New York, 1992.

[25] A. Brandt. Multigrid solvers on parallel computers. In M. H. Schultz, editor, *Elliptic Problem Solvers.* Academic Press, New York, 1981.

[26] J. E. Dendy, Jr. Black box multigrid. *J. Comp. Phys.*, 48:366–386, 1982.

[27] H. Foerster, K. Stueben, and U. Trottenberg. Non-standard multigrid techniques using checkered relaxation and intermediate grids. In M. H. Schultz, editor, *Elliptic Problem Solvers*, New York, 1981. Academic Press.

[28] W. Hackbusch and U. Trottenberg. *Multi-grid Methods.* Springer-Verlag, Berlin, 1982.

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

[30] M. J. Holst and F. Saied. Parallel performance of some multigrid solvers for three-dimensional parabolic equations. Technical Report UIUCDCS-R-91-1697, Dept. of Computer Science, University of Illinois at Urbana-Champaign, 1991.

[31] M. Holst and F. Saied. Multigrid solution of the Poisson-Boltzmann equation. *J. Comput. Chem.*, 14(1):105–113, 1993.

[32] M. Holst, R. Kozack, F. Saied, and S. Subramaniam. Treatment of electrostatic effects in proteins: Multigrid-based Newton iterative method for solution of the full nonlinear Poisson-Boltzmann equation. *Proteins: Structure, Function, and Genetics*, 18(3):231–245, 1994.

[33] S. McCormick, editor. *Multigrid Methods.* SIAM, Philadelphia, 1987.

[34] F. Saied and M. J. Holst. Multigrid methods for computational acoustics on vector and parallel computers. In R. L. Lau and D. Lee, editors, *Proceedings of the Third IMACS Symposium on Computational Acoustics, held at Harvard University*, pages 71–80, 1991.

[35] J. R. Gilbert, C. Moler, and R. Schreiber. Sparse matrices in MATLAB: Design and Implementation. *SIAM J. Mat. Anal. Appl.*, 13:333–356, 1992.

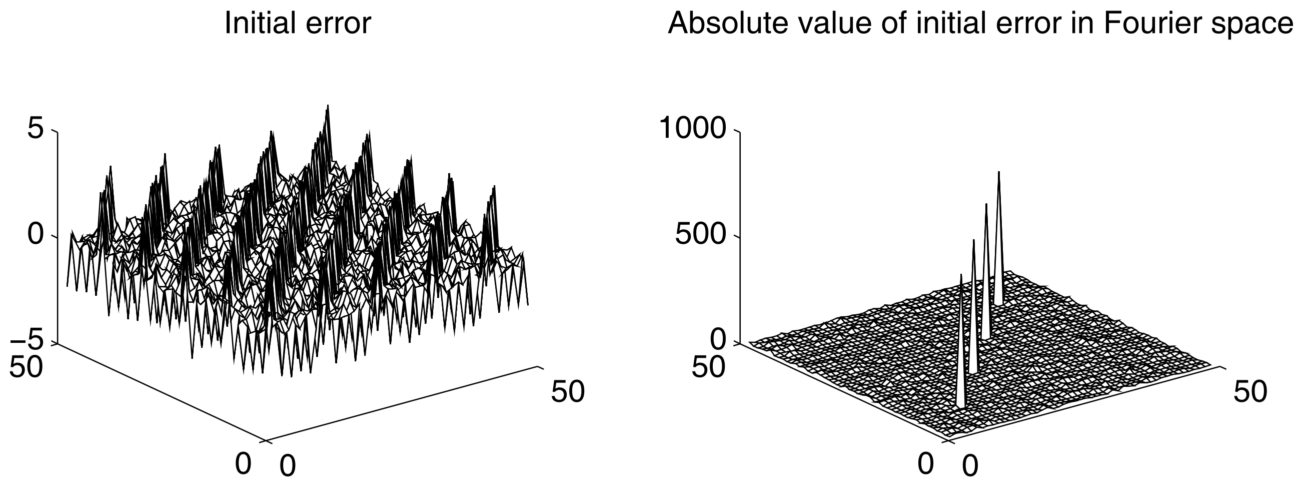

Figure 1: Output from Demo 2. Initial Error for Poisson's Equation on a 49×49 grid. The two-grid algorithm was used, with Gauss-Seidel Smoothing with $(\nu_1, \nu_2) = (0, 4)$, half-weighting and cubic interpolation. The error is shown on the left, and the 2D sine transform of the error on the right.

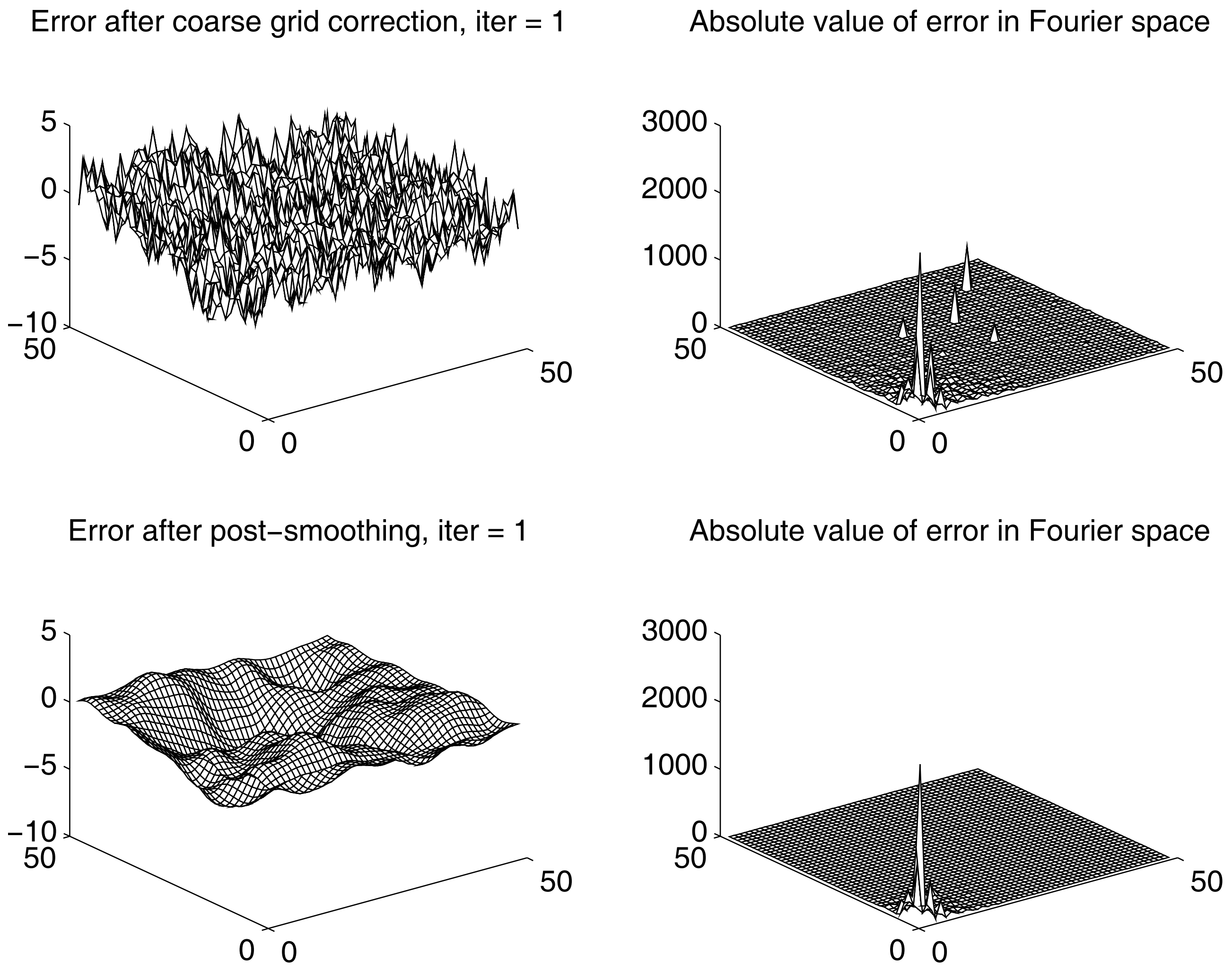

Figure 2: Output from Demo 2. Error in the first V-cycle, after the coarse grid correction (top) and after the post-smoothing (bottom). As in Figure 1, the error in physical space is shown on the left and in Fourier space on the right.

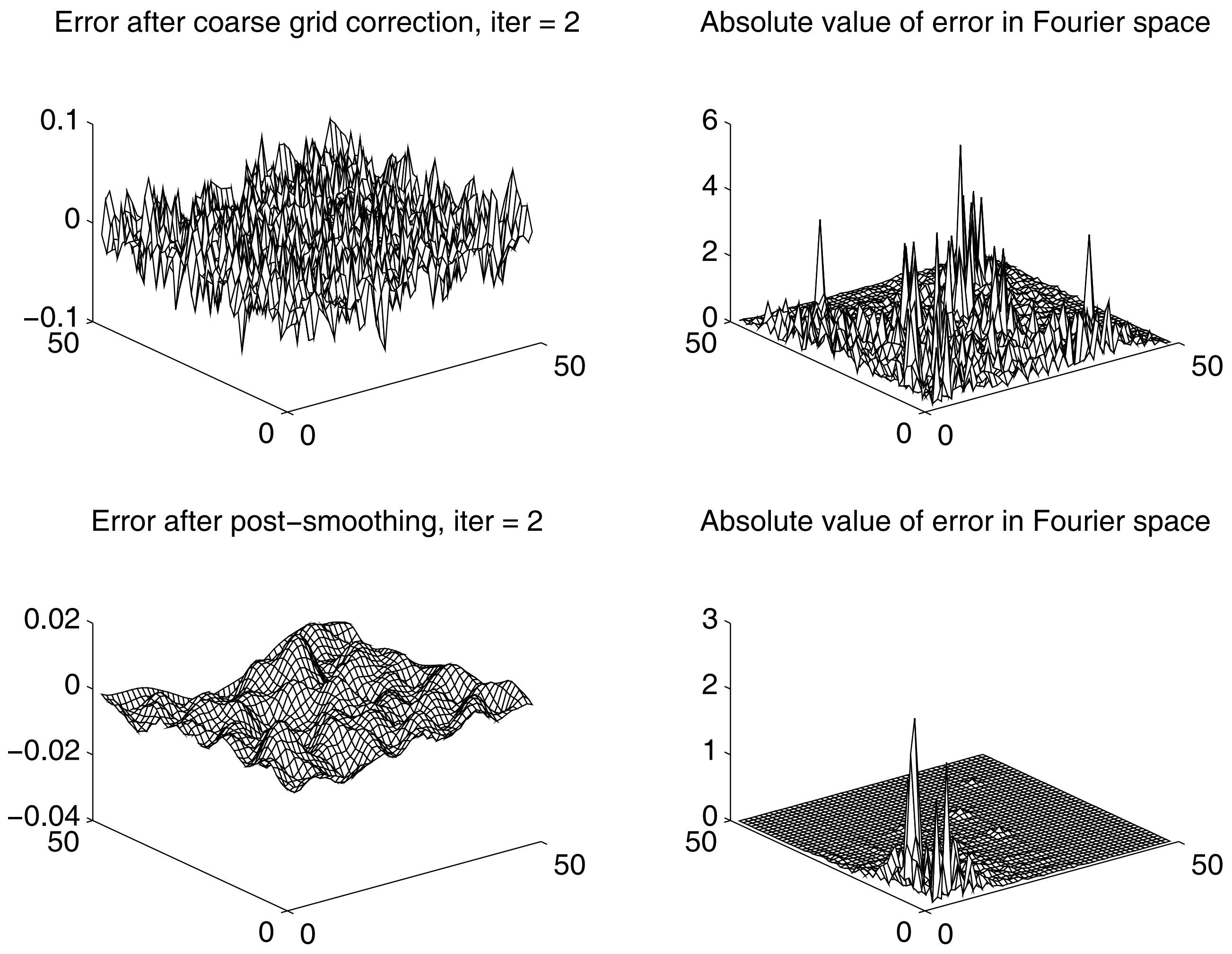

Figure 3: Output from Demo 2. Same as Figure 2, for the second V-cycle.

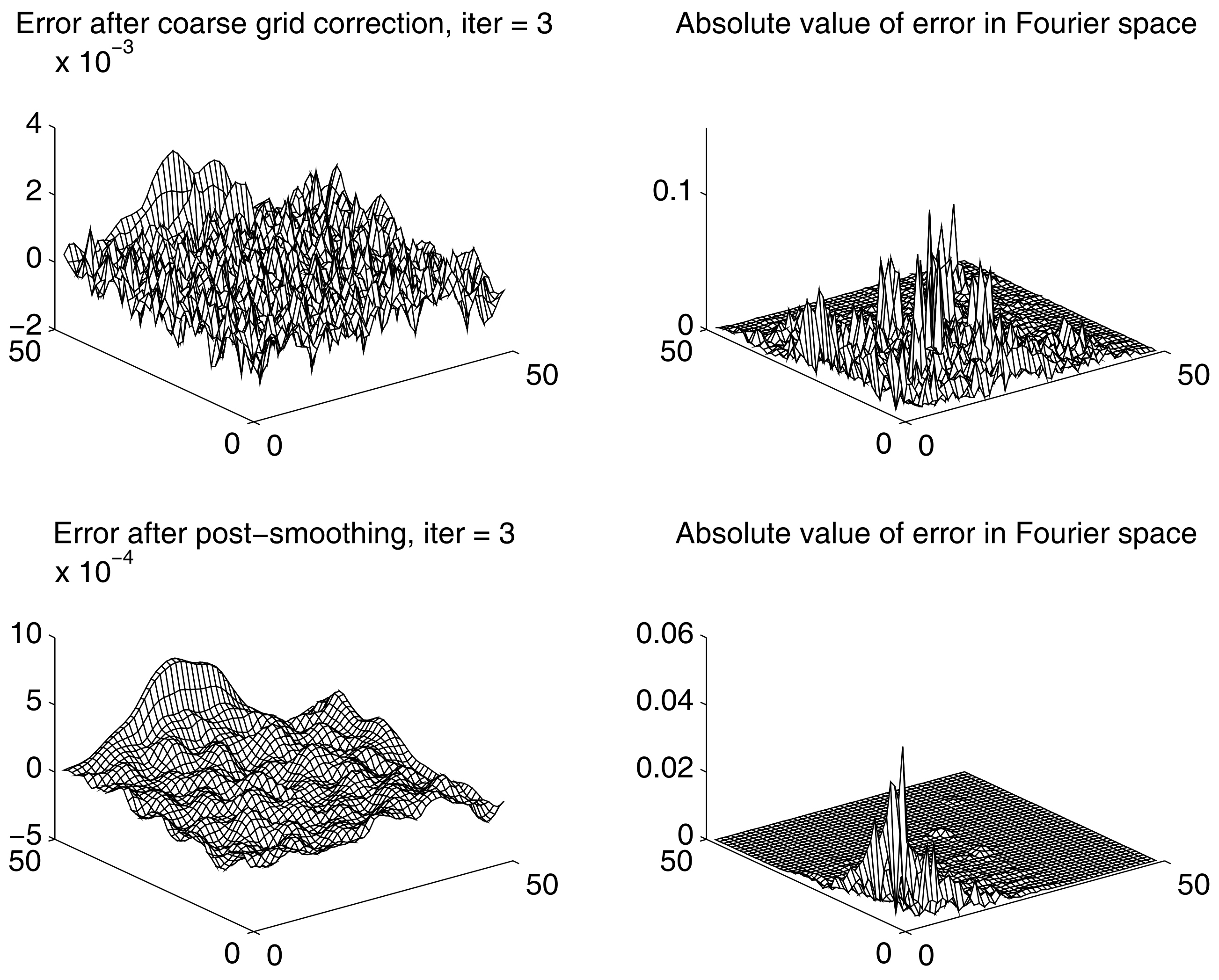

Figure 4: Output from Demo 2. Same as Figure 2, for the third V-cycle.