University of Science
Faculty of Mathematics and Computer Science

Worksheet #6

Professor: *Laurence Halpern - Juliette Ryan*

Student name: *Nguyen Tu Huy*

Course: *PUF - High Performance Computing*
Due date: *December 18th, 2022*

1. Multigrid

A first tackle

Provided :

1: Multigrid-Intro : A first simple " introductionary programs to Multigrid
2: MGLAB : AN INTERACTIVE MULTIGRID ENVIRONMENT,
    a Matlab software developed by James Bordner and Faisal Saied
3: Some basic Matlab programs to solve $\Delta u = f$
4: Some papers :
    bordner_saied.pdf : a simple user's guide
    Lecture Notes by Prof. Halpern
    A multigrid tutorial

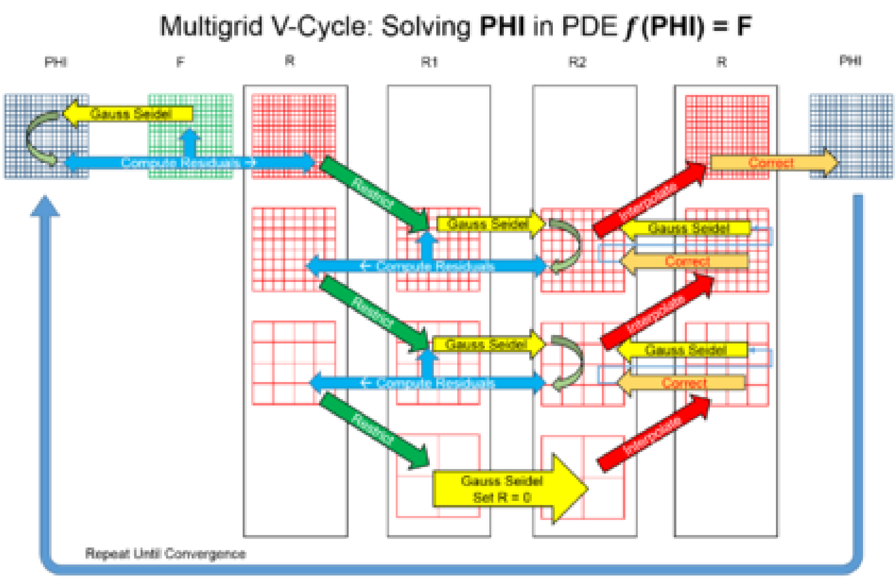


Figure 1: Multigrid Visualization

To do :

1. Get to know this solver.
2. Show that for the Poisson equation, convergence is independent of h = meshsize for a 3 level V cycle.
3. Study the influence of the different smoothers for a V cycle.
4. Study the influence of the number of iterations associated to the pre and post smoothers.
5. Do the same as above for the W cycle.
6. See the effect of a full multigrid.
7. Compare for the different mesh size provided solving the Laplace problem and a convection-diffusion problem with a restarted GMRES and a 3 level V-cycle multigrid.
8. Write a small report about your findings.

2. Solution

2.1. How multigrid method work. By understanding *A multigrid tutorial*, multigrid method work folow Algorithm

Algorithm : Multigrid
Require: Start with x_0
1: Do some weighted-Jacobi or Gauss-Seidel iteration (pre-smoothing steps) to pre-smoothing solution error. 2: Then we have $x^{k_1} = \text{Jacobi}(A, b, x_0, \omega, N_1, tol)$ where 'Jacobi' for damped Jacobi (we can use Gauss-Seidel relaxation method) and ω is relaxation parameter for the smoother 3: Compute residual $r^{k_1} = Ax^{k_1} - b$ 4: Using restriction to retrict r^{k_1} from fine grid to coarser grid and get $r_c^{k_1}$ 5: Solve equation $A_c e_c^{k_1} = r_c^{k_1}$ by direct method $e_c^{k_1} = A_c \backslash r_c^{k_1}$ Or do another multigird cycle to get V-cycle or W-cycle. 6: Prolongation $e_c^{k_1}$ from coarse grid to e^{k_1} in fine grid. 7: Then compute $x^1 = x^{k_1} + e^{k_1}$
We end 1 cycle, we can use x^1 as x_0 and do another cycle.

```
function [u_lmax, k, flag] = mg_alg(f_lmax, lmax, matrices, u0_lmax,
    parents, ...
     isdir, nu1, nu2, mu, maxit, tol, smooth, omega)
% Multigrid algorithm.
```

```
% Can be used as standalone solver or as preconditioner for the
% Matlab pcg routine (see the pcg help for details).
%
% INPUT:   f_lmax    right hand side on the finest level
%          lmax      maximum level
%          matrices  cell array of matrices, matrices{l+1} contains
%                    the system matrix on level l, l = 0,...,lmax
%          u0_lmax   initial guess
%          parents   cell array containing the father-son relations
%          isdir     dirichlet node flags
%          nu1       number of pre-smoothing steps
%          nu2       number of post-smoothing steps
%          mu        number of recursive multigrid step calls,
%                    1 for V-cycle, 2 for W-cycle
%          maxit     maximum number of iterations
%          tol       tolerance for stopping criterion
%          smooth    type of smoother,
%                    'Jacobi' for damped Jacobi or 'SOR' for SOR
%          omega     relaxation parameter for the smoother
%
% OUTPUT:  u_lmax    solution on the finest level
%          k         number of performed iterations
%          flag      0 if the algorithm converged, 1 otherwise
%
% VERSION 1.0
% DATE 25.3.2004
% EMAIL bernd@flemisch.net

flag = 0;
for k = 1:maxit
    % perform a multigrid step on level lmax
    u_lmax = multigrid_step(f_lmax, lmax, matrices, u0_lmax, parents, ...
        isdir, nu1, nu2, mu, smooth, omega);
    % stopping criterion (relative residual):
    if (norm(matrices{lmax+1}*u_lmax - f_lmax)/norm(f_lmax) < tol)
        return;
    end
    u0_lmax = u_lmax;
end
flag = 1;
return;

function u_l = multigrid_step(f_l, l, matrices, u0_l, parents, isdir, ...
    nu1, nu2, mu, smooth, omega)
A_l = matrices{l+1};              % get system matrix of the current level
if (strcmp(smooth, 'Jacobi'))
    u_l = jacobi(A_l, f_l, u0_l, nu1, omega); % smooth nu1-times
else
    u_l = gs(A_l, f_l, u0_l, nu1, omega); % smooth nu1-times
end
d_l = f_l - A_l*u_l; % calculate defect
d_lm1 = restrict(d_l, parents{l+1}, isdir); % restrict defect

if l == 1     % solve direct on the coarsest level:
    w_lm1 = matrices{l}\d_lm1;
else          % perform mu multigrid steps
```

```
    u0_lm1 = zeros(length(d_lm1), 1);
    for k = 1:mu
        w_lm1 = multigrid_step(d_lm1, l-1, matrices, u0_lm1, parents, ...
            isdir, nu1, nu2, mu, smooth, omega);
        u0_lm1 = w_lm1;
    end
end
w_l = interpolate(w_lm1, parents{l+1});   % interpolate correction
u_l = u_l + w_l;  % add correction

if (strcmp(smooth, 'Jacobi'))
    u_l = jacobi(A_l, f_l, u_l, nu2, omega); % smooth nu2-times
else
    u_l = gs(A_l, f_l, u_l, nu2, omega);     % smooth nu2-times
end
```

2.2. Poisson equation with a 3 level V cycle. Firstly, we do on file *'mg_script'*

In *'mg_script'*, we do on general case where solve for diffusion equation

$$\operatorname{div}(a\nabla u) + cu = f$$

So to modified Poisson equation, we set $a = 1$ and $c = 0$

Consider Poisson equation

$$\begin{cases} \nabla^2 u & = f \qquad \text{on } \Omega \\ u & = 0 \qquad \text{in } \partial\Omega \end{cases}$$

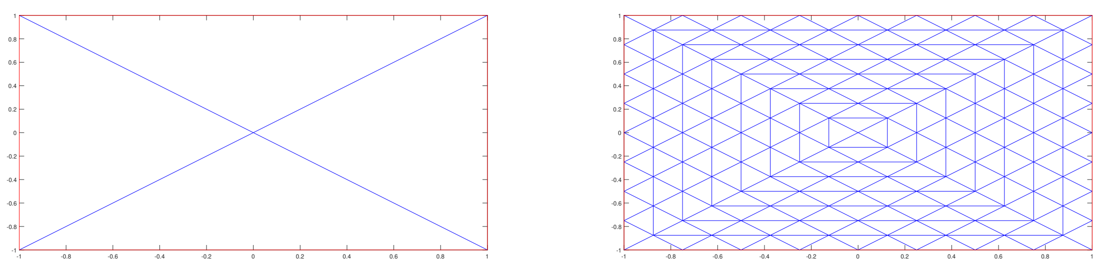
where $f = 5\pi^2 \sin(2\pi x) \times \sin(\pi y)$.

And exact solution : $u = \sin(2\pi x) \times \sin(\pi y)$

Change meshsize to see its effect to the convergence of multigrid method.

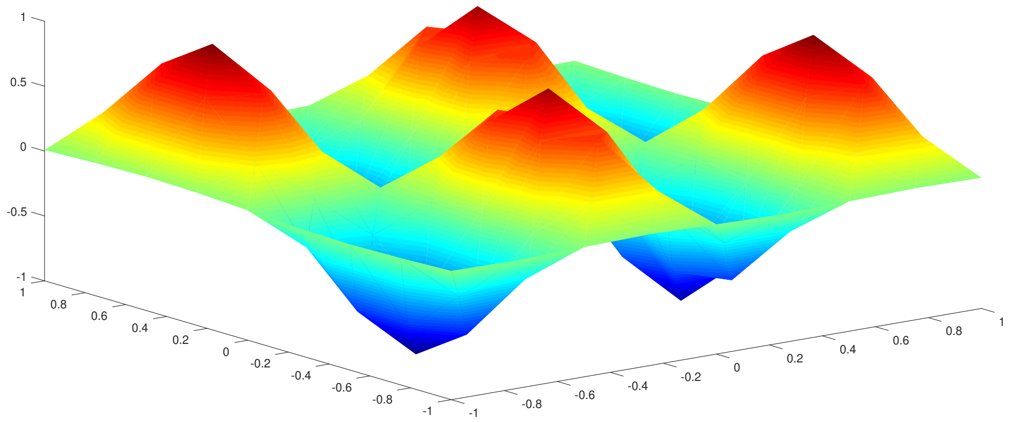
We consider the case we use: 3 level V-cycle, Jacobi-Smoother (0.70), number pre-smoothing steps = 3, post-smoothing steps = 3, tolerent = 10^{-4}

a. maximum initial element diameter = 2



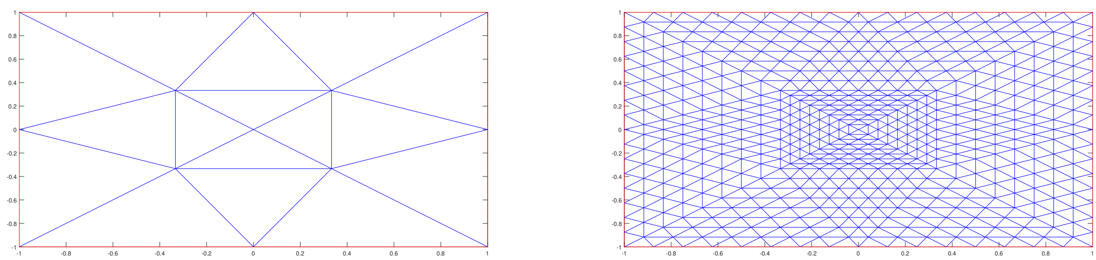
(a) Initial mesh FE (b) Final mesh

Figure 2: Compare initial Mesh FE and Mesh after using Multigrid method



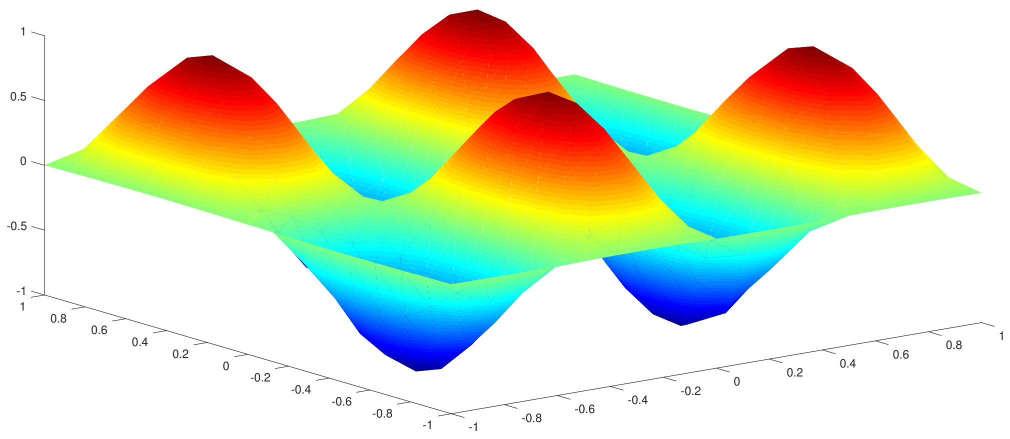
Level	number of degrees of freedom	iter	error Multigrid	error direct	time MG	time direct
1	13	2	2.01e+00	2.01e+00	1.08e-03	9.86e-05
2	41	2	1.08e+00	1.08e+00	7.76e-04	1.20e-04
3	145	4	4.73e-01	4.73e-01	7.96e-04	2.36e-04

b. maximum initial element diameter = 1



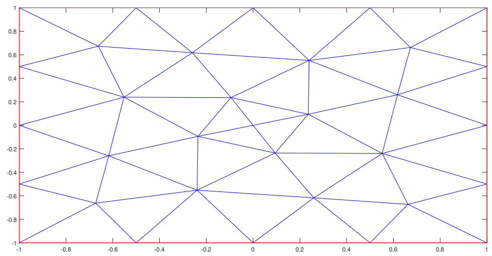
(a) Initial mesh FE (b) Final mesh

Figure 3: Compare initial Mesh FE and Mesh after using Multigrid method

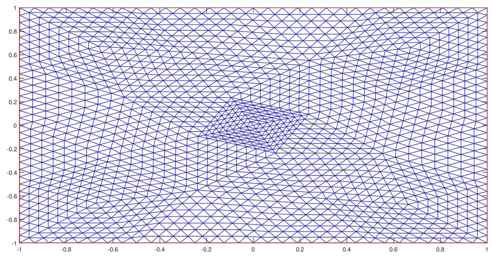


Level	number of degrees of freedom	iter	error Multigrid	error direct	time MG	time direct
1	41	2	9.10e-01	9.10e-01	1.24e-03	8.30e-05
2	145	4	2.94e-01	2.94e-01	1.22e-03	2.58e-04
3	545	4	1.30e-01	1.29e-01	1.19e-03	1.02e-03

c. maximum initial element diameter = 0.5

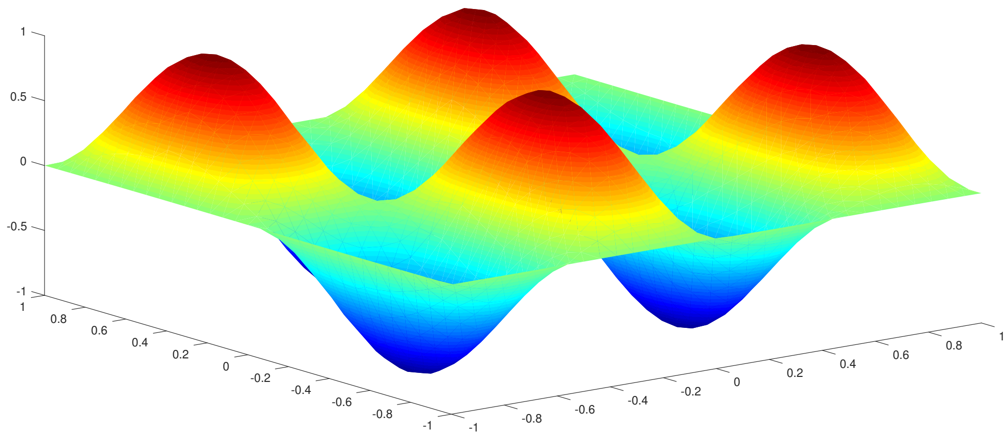


(a) Initial mesh FE



(b) Final mesh

Figure 4: Compare initial Mesh FE and Mesh after using Multigrid method



Level	number of degrees of freedom	iter	error Multigrid	error direct	time MG	time direct
1	113	3	3.73e-01	3.73e-01	5.47e-03	2.56e-04
2	417	4	1.63e-01	1.63e-01	2.69e-03	7.08e-04
3	1601	5	7.80e-02	7.80e-02	3.17e-03	3.66e-03

2.3. The effect of smoother method. For smoother method, we use *'damped Jacobi'* or *'relaxation Gauss-Seidel'* method.

We consider the case we use: 3 level V-cycle, number pre-smoothing steps = 6, post-smoothing steps = 6, tolerent = 10^{-4}

- **Jacobi smoother**

$\omega = 0.5$

Level	Number of degrees of freedom	iter	Error Multigrid	Error direct	Time MG	Time direct
1	113	2	3.73e-01	3.73e-01	5.61e-03	2.52e-04
2	417	3	1.64e-01	1.63e-01	3.33e-03	1.58e-03
3	1601	4	7.80e-02	7.80e-02	3.62e-03	3.26e-03

$\omega = 0.7$

Level	Number of degrees of freedom	iter	Error Multigrid	Error direct	Time MG	Time direct
1	457	4	1.81e-01	1.81e-01	7.74e-03	8.36e-04
2	1761	5	8.71e-02	8.70e-02	5.59e-03	3.82e-03
3	6913	5	4.30e-02	4.28e-02	1.60e-02	2.20e-02

- **Gauss Seidel smoother**

$\omega = 0.5$

Level	Number of degrees of freedom	iter	Error Multigrid	Error direct	Time MG	Time direct
1	113	2	3.73e-01	3.73e-01	5.16e-03	5.39e-04
2	417	3	1.64e-01	1.63e-01	4.78e-03	1.06e-03
3	1601	4	7.80e-02	7.80e-02	9.85e-03	3.99e-03

$\omega = 0.7$

Level	Number of degrees of freedom	iter	Error Multigrid	Error direct	Time MG	Time direct
1	113	2	3.73e-01	3.73e-01	5.65e-03	2.42e-04
2	417	3	1.63e-01	1.63e-01	3.59e-03	1.06e-03
3	1601	3	7.82e-02	7.80e-02	7.32e-03	3.23e-03

2.4. The pre and post smoothers. We see the influence of the number of iterations associated to the pre and post smoothers.

We change number of pre-smoothing

Number pre	Number post	levels	iteration
3	3	1	3
		2	4
		3	5
6	3	1	3
		2	4
		3	4
9	3	1	2
		2	3
		3	4

We change number of post-smoothing

Number pre	Number post	levels	iteration
3	3	1	3
		2	4
		3	5
3	6	1	3
		2	4
		3	4
3	9	1	2
		2	3
		3	3

Commend: Increase number of pre-smoothers and post-smoothers decrease number of iteration.

2.5. W - cycle. Compare difference of V -cycle and W - cycle

	W-cycle			**V-cycle**		
level	iter	error MG	time MG	iter	error MG	time MG
1	2	3.73e-01	5.25e-03	2	3.73e-01	6.37e-03
2	3	1.63e-01	2.30e-03	3	1.64e-01	1.88e-03
3	3	7.81e-02	5.05e-03	4	7.80e-02	5.35e-03

Table 1: Comparison of V-cycle and W-cycle

2.6. Multigrid is used as preconditioner. We use *pcg function* to use Multigrid as a preconditioner

	Multigrid			**Multigrid with preconditioning**		
level	iter	error MG	time MG	iter	error MG	time MG
1	2	3.73e-01	3.54e-03	2	3.73e-01	4.80e-03
2	3	1.64e-01	1.68e-03	3	1.63e-01	3.19e-03
3	4	7.80e-02	2.81e-03	3	7.80e-02	5.93e-03

Table 2: Comparison of Multigrid and Multigrid use as preconditioner

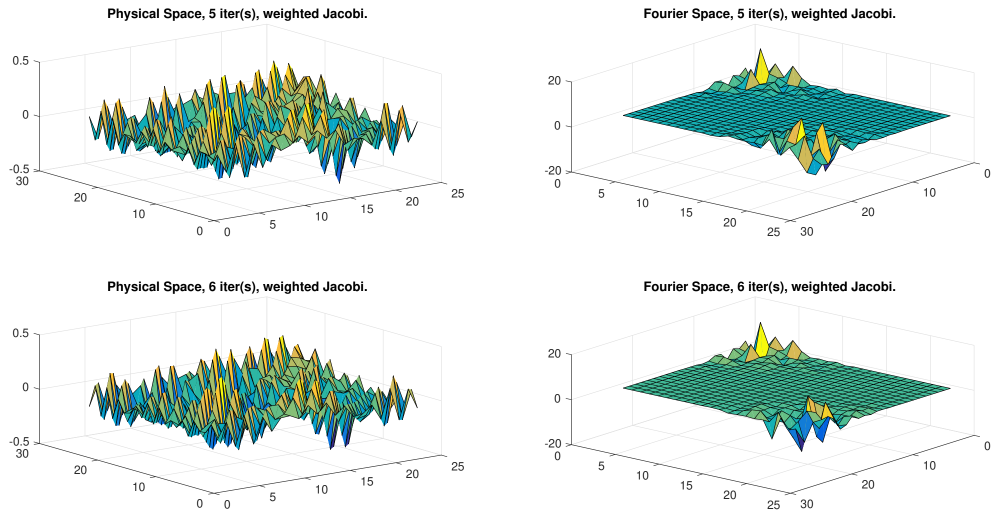
3. Study on MGLAB

MGLab are restricted to two dimensional elliptic partial differential equations on rectangular domains

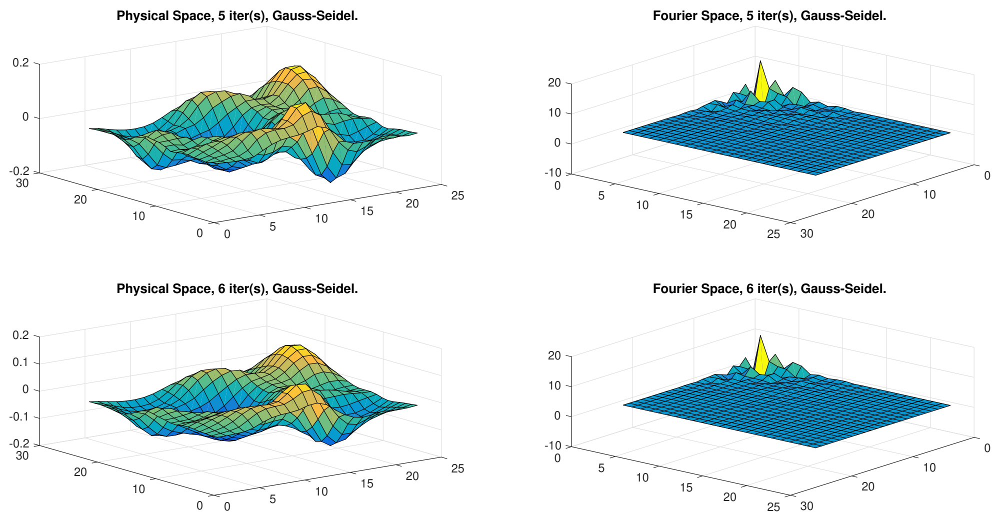
$$\begin{cases} \nabla \cdot (a\nabla u) + b \cdot \nabla u + cu &= f \qquad \text{on } \Omega \\ u &= 0 \qquad \text{in } \partial\Omega \end{cases}$$

The domain Ω is the unit square $\{(x, y) : 0 < x, y < 1\}$

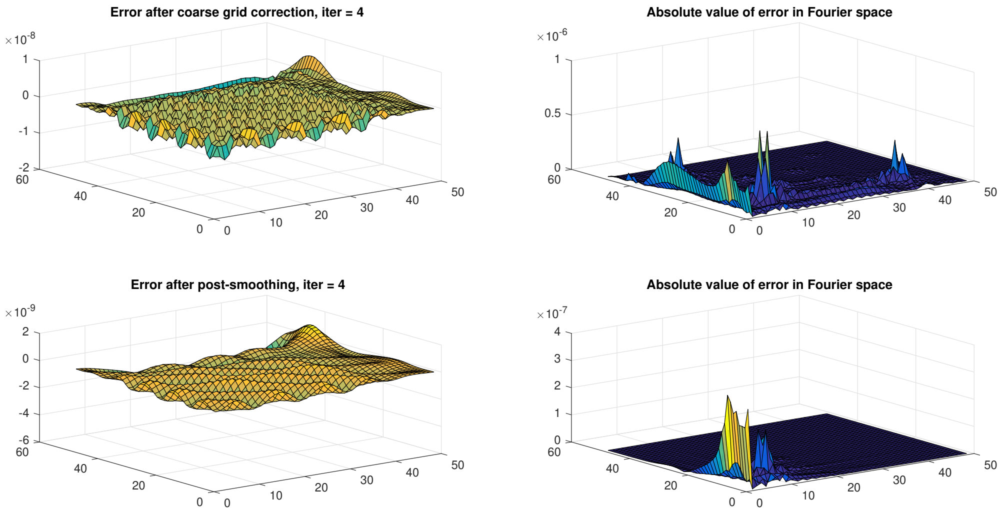
- Consider using weighted Jacobi smoother



- Consider using weighted Gauss-Seidel smoother (Left solution, Right Error)



- Error



If we run MGLab.m

Problem
Poisson Equation
Problem size = (15, 15)
Solver
V-Cycle
Preconditioner
Stopping Criteria
(precon) Residual Tolerance = 1e-04
Iteration Limit = 5
MG Parameters
Number of Levels = 3
Smoother = Weighted Jacobi (w=0.95)
Pre-smoothings = 1
Post-smoothings = 1
Restriction = Half Weighting
Prolongation = Cubic
Coarse-grid Solver = Sparse GE
MG Cycle Type = V-Cycle

Result : V - cycle
1: iteration 0 residual 4.01894
2: iteration 1 residual 0.421197
3: iteration 2 residual 0.0585156
4: iteration 3 residual 0.00995454
5: iteration 4 residual 0.00188439
6: iteration 5 residual 0.000417033
rho = 0.177387.
Relative residual = 0.000417033

And if we consider Diffusion Convection equation

```
        Problem
   Convection-Diffusion Equation
       lambda =    -10
       sigma  =      5
Problem size = ( 15, 15)
Solver
   FMG
Preconditioner
Stopping Criteria
   (precon) Residual Tolerance = 1e-04
   Iteration Limit =    5
MG Parameters
   Number of Levels = 4
   Smoother = Gauss-Seidel
   Pre-smoothings =  3
   Post-smoothings =  3
   Restriction = Half Weighting
   Prolongation = Cubic
   Coarse-grid Solver = Sparse GE
   MG Cycle Type = V-Cycle
```