HW4, Math 228A. Multigrid Solver

Date due 11/30/2010 UC Davis, California

Fall 2010

Nasser M. Abbasi

Fall 2010 Compiled on January 4, 2020 at 11:14pm [public]

Contents

1 Problem 1

1. Write a multigrid V-cycle code to solve the Poisson equation in two dimensions on the unit square with Dirichlet boundary conditions. Use full weighting for restriction, bilinear interpolation for prolongation, and red-black Gauss-Seidel for smoothing.

   Note: If you cannot get a V-cycle code working, write a simpler code such as a 2-grid code. You can also experiment in one dimension (do not use GSRB in 1D). You may turn in one of these simplified codes for reduced credit. You should state what your code does, and use your code for the second problem of this assignment.

Figure 1: Problem 1

The multigrid V cycle algorithm was implemented in Matlab 2010a. The documented source code is included in the appendix of this problem.

For relaxation, Gauss-Seidel with red-black (GSRB) ordering was used as the default. A relax() function was written to implement this method, in addition, this function can also implement relaxation using these solvers: Jacobi, Gauss-Seidel Lex, and SOR. Selecting the relaxation method is done via an argument option. These different methods are implemented in the relax() function for future numerical experimentation. GSRB is the one used for all the solutions below as required by the problem statement and is the default method. GSRB is known to have good smoothing rates and is suitable for parallelism as well.

For mapping from the fine mesh to coarse mesh, the full weighting method is used.

For mapping from coarse mesh to fine mesh, bilinear interpolation is used.

Additional auxiliary functions are written for performing the following: finding the norm (mesh norm), finding the residue and validating dimensions of the input.

The following diagram illustrates the call flow chart for a program making a call to the V cycle algorithm, such as the program written to solve problem 2 and 3. It shows the Matlab functions used, and the interface between them. This flow diagram also shows a full multigrid solver (FMG) function, which was implemented on top of the V cycle algorithm, but was not used to generate the results in problem 2 as the problem asked to use V cycle algorithm only. FMG cycle algorithm was implemented for future reference and to study its effect on reducing number of iterations. A note on this is can be found the end of this assignment.

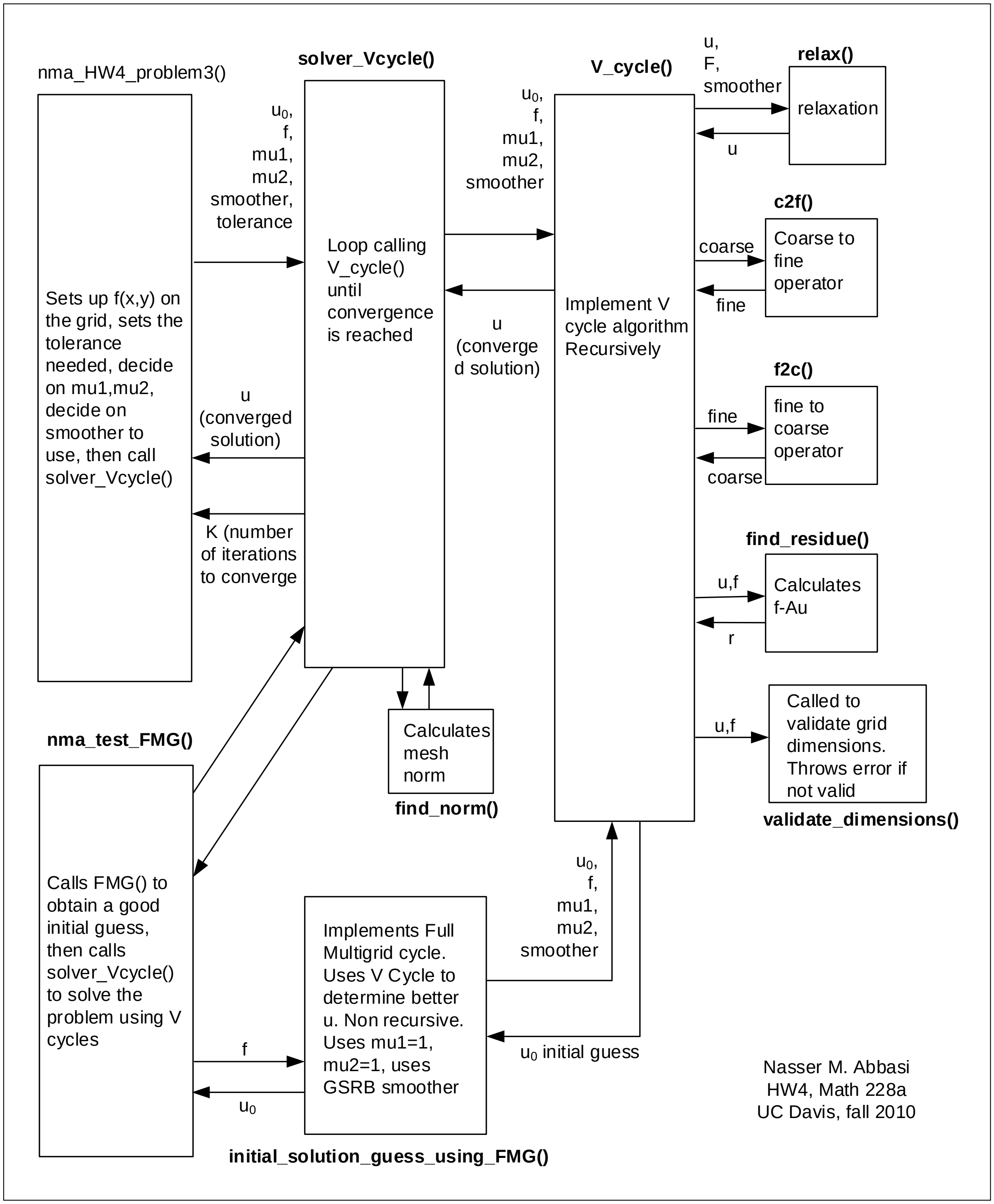

Figure 2: algorithm flow chart

1.1 Restriction and prolongation operators

The restriction operator I_h^{2h} (fine to coarse mesh) mapping uses full weighting, while the prolongation operator I_{2h}^{h} (coarse to fine mesh) uses bilinear interpolation.

For illustration, the following diagram shows applying these operators for the 1D case for a mesh of 9 points. The edge points are boundary points and in this problem (Dirichlet homogeneous boundary conditions), these will always be zero.

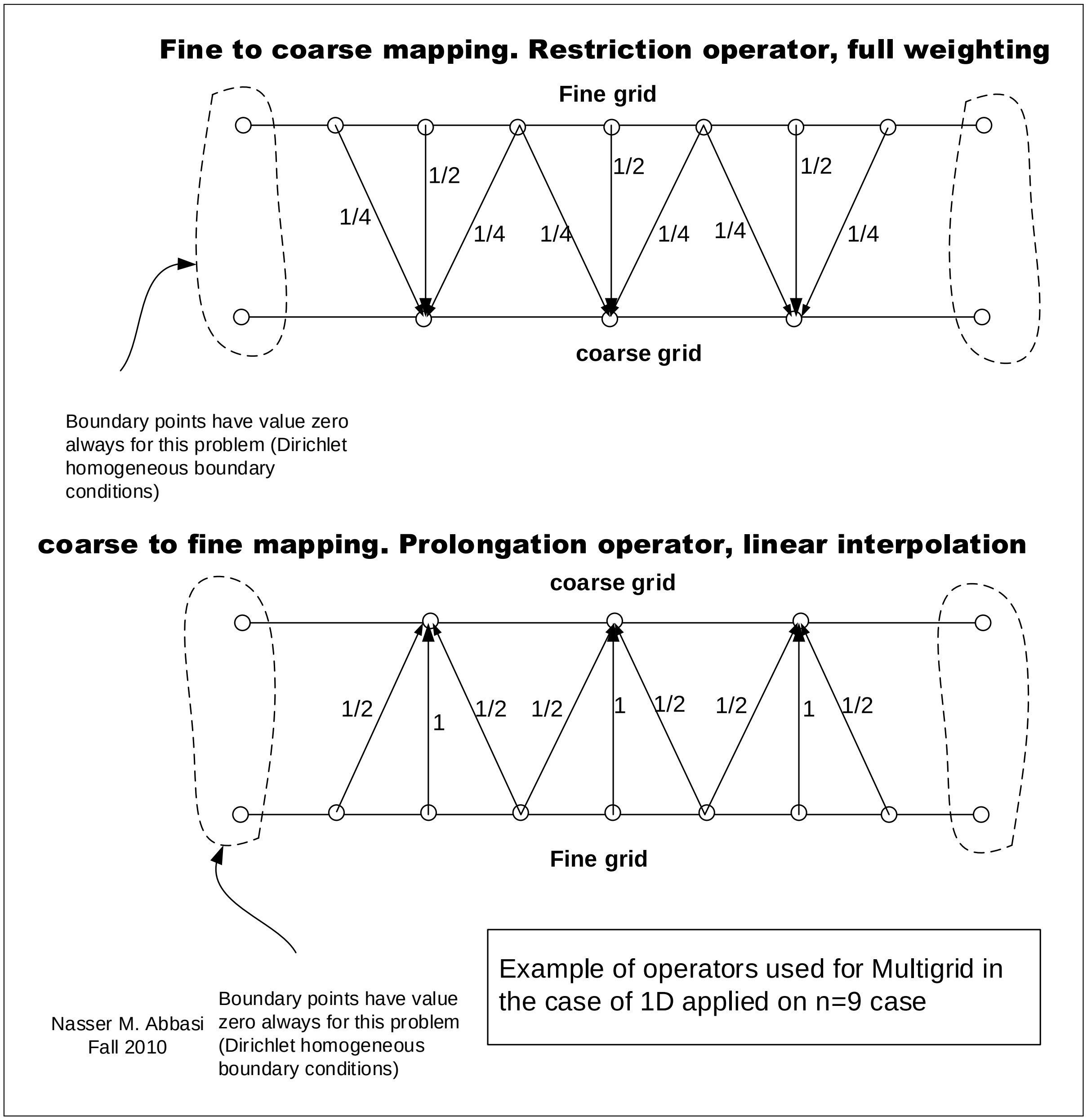

Figure 3: operator diagrams

1.2 V cycle algorithm

The multigrid V cycle algorithm is recursive in nature. The following is description of the algorithm

```
VCYCLE algorithm
----------------
input:  u, f, mu1, mu2
output: u (more accurate u)

Let n be the number of grid points of u in one dimension

IF n = 3 THEN
   find u by direct solution of 3x3 grid
ELSE
   apply mu1 smoothing on u
   residue = find residue (f-Au)
   residue = apply fine-to-coarse mapping on residue
   correction = CALL VCYCLE(ZERO,residue,mu1,mu2)
   correction = apply coarse-to-fine mapping on correction
   u = u + correction
   apply mu2 smoothing on u
END IF

RETURN u
```

.

Problem 2 below also has a diagram which helps understand this algorithm more. The Matlab function shown below implements the above algorithm.

1.3 Multigrid V cycle function (V_cycle.m)

This function implements one multigrid V cycle. It is recursive function

1.4 Solver using V cycle (solver_Vcycle.m)

This function is an interface to V cycle algorithm to use for solving the 2D Poisson problem. It uses V_cycle.m in a loop until convergence is reached.

1.5 Relaxation or smoother function (relax.m)

This function implements Gauss-Seidel red-black solver. It is a little longer than needed as it also implements other solvers as was mentioned in the introduction. These are added for future numerical experimentation. The

algorithm is straight forward. If the sum of the row and column index adds to an even value, then the grid point is considered a red grid point, else it is black. The red grid points are smoothed first, then the black grid points are smoothed.

1.6 Find residual function (find_residue.m)

This function is called from a number of locations to obtain the residue mesh. The residue is defined as

$$r_{i,j} = f_{i,j} - \frac{1}{h^2}\left(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{i,j}\right)$$

1.7 Find norm function (find_norm.m)

This function is called from number of locations to obtain the norm of the mesh or any 2D matrix.

1.8 Validate boundary conditions (check_all_zero_boundaries.m)

An auxiliary function used by a number of functions to validate that input has consistent boundaries for this problem.

1.9 Coarse to fine prolongation operator 2D (c2f.m)

This function is the prolongation bilinear interpolation which implements coarse to fine grid mapping on 2D grid.

1.10 Fine to coarse full weight restriction operator 2D (f2c.m)

This function is the full weight restriction operator which implements the fine grid to coarse grid mapping on 2D grid.

1.11 Validate u and f have consistent dimensions(validate_dimensions.m)

An auxiliary function used by number of other function to validate that input dimensions are consistent.

1.12 Validate grid for consistent dimensions (validate_dimensions_1.m)

An auxiliary function used by number of other function to validate that a grid dimensions are consistent.

1.13 FMG solver (initial_solution_guess_using_FMG.m)

Implements a full multigrid cycle using V cycle algorithm as building block. Used to compare effect on solution only.

1.14 Restriction operator for 1D (f2c_1D.m)

This function is the full weight restriction operator which implements the fine grid to coarse grid mapping on 1D grid

1.15 Prolongation operator for 1D (c2f_1D.m)

This function is the prolongation bilinear interpolation which implements coarse to fine grid mapping on 1D grid

2 Problem 2

2. Numerically estimate the average convergence factor,

$$\left(\frac{\|e^{(k)}\|_\infty}{\|e^{(0)}\|_\infty}\right)^{1/k},$$

for different numbers of presmoothing steps, ν_1, and postsmoothing steps, ν_2, for $\nu = \nu_1+\nu_2 \leq 4$. Be sure to use a small value of k because convergence may be reached very quickly. What test problem did you use? Report the results in a table, and discuss which choices of ν_1 and ν_2 give the most efficient solver.

Figure 4: Problem 2

The test problem used is

$$\nabla u = 0$$

with zero boundary conditions on the unit square. This has a known solution which is zero.

Initial guess is a random solution generated using matlab rand(). Hence $\|e^{(0)}\|$ has the same norm as the initial guess.

The V Cycle function written for problem 1 was used to generate the average convergence factor. For each combination of ν_1, ν_2, a table was generated which contained the following columns:

1. Cycle number.
2. The norm of the residue $\|r^{(k)}\| = \|f - Au^{(k)}\|$ after each cycle.
3. Ratio of the current residue norm to the previous residue norm $\frac{\|r^{(k+1)}\|}{\|r^{(k)}\|}$.
4. Error norm $\|e^{(k)}\| = \|u - u^{(k)}\| = \|u^{(k)}\|$ (since exact solution is zero).
5. The ratio of the current error norm to the previous error norm $\frac{\|e^{(k+1)}\|}{\|e^{(k)}\|}$.

6. The average convergence factor up to each cycle $\left(\frac{\|e^{(k)}\|_\infty}{\|e^{(0)}\|_\infty}\right)^{\left(\frac{1}{k}\right)}$.

The problem asked to generate result for $\nu \leq 4$. This solution extended this to $v \leq 8$ to use the results for future study if needed. The summary table below shows the final result for $k = 15$. Each individual table generated for each combination of ν_1, ν_2 is listed in the appendix of this problem. The function HW4_problem2() was used to generate these tables and to calculate the work done for each solver combination of ν_1, ν_2.

2.1 Average convergence factor and work unit estimates

To determine the most efficient solver, the amount of work by each solver that achieves the same convergence is determined. The solver with the least amount of work is deemed the most efficient. The total amount of work for convergence is defined as

$$\text{WORK=number of Iterations for convergence } \times \text{work per iteration} \tag{1}$$

An iteration is one full V cycle. Each V cycle contains a number of levels. The same number of levels exist on the left side of the V cycle as on the right side of the V cycle. On the left side of the V cycle, work at each level consist of the following items

1. Work to perform ν_1 number of pre-smooth operations.
2. Work needed to map to the next lower level of the grid.
3. Work needed to compute the residue.

The following diagram helps to illustrates this.

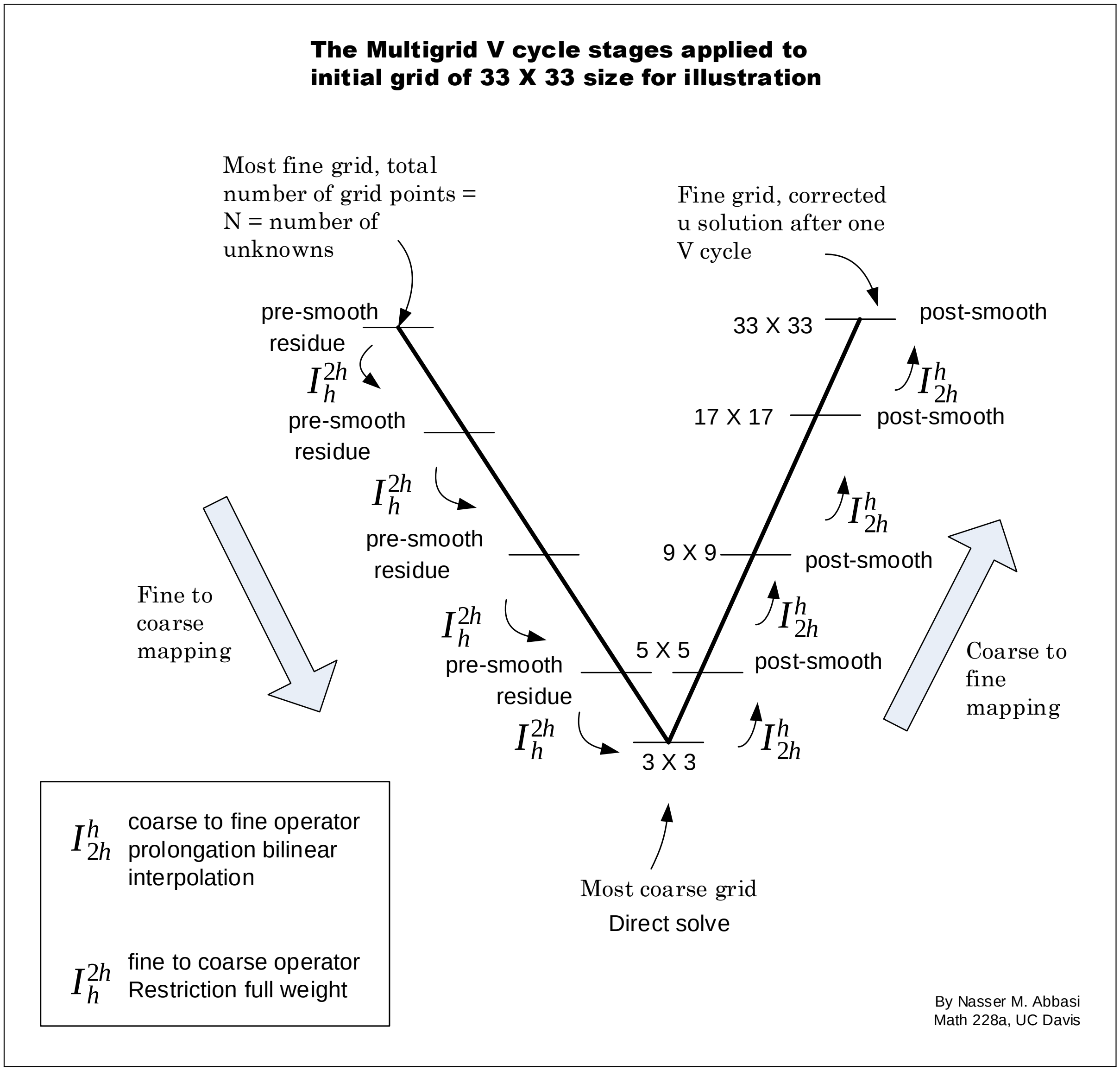

Figure 5: problem 2 v cycle shape

On the right side of the V cycle, work at each level consist of the following items

1. Work to perform ν_2 number of post-smooth operations.
2. Work needed to map to the next lower level of the grid.

At each level, the work is proportional to the size of the grid at that level. For smoothing, it is estimated that 7 flops are needed to obtain an average of each grid point. (5 additions, one multiplication, one division) based on the use of the following formula

$$u_{i,j} = \frac{1}{4}\left(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - h^2 f\right)$$

Therefore work needed for smoothing is $7 \times (v_1 + v_2) \times N$ where N is the total number of grid points at that level. (Boundary grid points are not involved in this work, but for simplicity of analysis, the total number of grid points N is used).

Work needed for finding the residual is also about $7N$. Work needed for mapping to the next grid level is about $6N$.

On the right branch of the cycle no residual calculation is required. To simplify the analysis, it is assumed that the same work is performed at each level on both sides of the V cycle.

Therefore, Letting N be the number of grid points at the most fine level (the number of unknowns), the total work per cycle is found to be

$$\begin{aligned} \textit{work per V cycle} &= (7(v_1+v_2)+13)N + (7(v_1+v_2)+13)\frac{N}{4} + (7(v_1+v_2)+13)\frac{N}{16} + \cdots \\ &= (7(v_1+v_2)+13)N\left[1+\frac{1}{4}+\frac{1}{4^2}+\cdots+\frac{1}{4^{L-1}}\right] \end{aligned}$$

Where L is the number of levels. In the limit as L becomes very large, this becomes a geometric series whose sum is $\frac{(7(v_1+v_2)+13)N}{1-r}$ where $r=\frac{1}{4}$

Therefore, total amount of work from (1) becomes

$$W = M \times \frac{4}{3}(7(v_1+v_2)+13)N$$

Where M is number of iterations. Using the same tolerance ε for all solvers, $M=\frac{\log(\varepsilon)}{\log(\rho)}$. *Using the average convergence rate found as an estimate for the spectral radius* ρ, W can now be found as a function of N

$$W = \left(\frac{\log(\varepsilon)}{\log(\rho)}\right)\left(\frac{4}{3}(7(v_1+v_2)+13)N\right)$$

For the purpose of comparing the different solvers, the value of ε used is not important as long as it is the same value in all cases. Hence, for numerical computation, let $\varepsilon=10^{-6}$ and the above becomes

$$W = \left(\frac{-6}{\log(\rho)}\right)\left(\frac{4}{3}(7(v_1+v_2)+13)N\right)$$

The program written for this problem uses the above equation to calculate the work done for each solver. The result is shown below. This table shows the work done by each solver for convergence based on the same tolerance. As mentioned above, changing the tolerance value will not change the result, as the effect will be to scale all result by the same amount.

2.2 Result

$v = (v_1, v_2)$	$v_1 + v_2$	$CF = \left(\frac{\|e^{(k)}\|_\infty}{\|e^{(0)}\|_\infty}\right)^{\left(\frac{1}{k}\right)}$	work $\frac{-6}{\log(\rho)} \times \frac{4}{3}(7(v_1+v_2)+13)N$
(0,1)	1	0.374588	213 N
(0,2)	2	0.202747	177 N
(0,3)	3	0.132526	174.9 N
(0,4)	4	0.098857	182.7 N
(1,0)	1	0.323688	182.4 N
(1,1)	2	0.116811	129.4 N
(1,2)	3	0.079578	138.4 N
(1,3)	4	0.060307	150.5 N
(1,4)	5	0.049019	164.1 N
(2,0)	2	0.171973	158.5 N
(2,1)	3	0.079845	138.6 N
(2,2)	4	0.060424	150.6 N
(2,3)	5	0.049068	164.2 N
(2,4)	6	0.041573	178.4 N
(3,0)	3	0.117023	163.6 N
(3,1)	4	0.060444	150.6 N
(3,2)	5	0.049072	164.2 N
(3,3)	6	0.041575	178.4 N
(3,4)	7	0.036128	192.6 N
(4,0)	4	0.088624	174.6 N
(4,1)	5	0.049075	164.2 N
(4,2)	6	0.041575	178.4 N
(4,3)	7	0.036128	192.6 N
(4,4)	8	0.031908	206.7 N

2.3 Conclusion

From the above result, The least work was for the (1,1) solver, followed by (1,2) which had about the same as the (2,1) solver. This result shows that using $v = 2$ or $v = 3$ *is the most efficient solver* in terms of least work required.

Notice that in full multigrid, the combination which makes up the value v is important (While for the case of the 2 level multigrid, this is not the case). For example, as shown in the above table, work for solver $v = (1,2)$ was 138 N while work for solver $v = (3,0)$ was 163 N even though they both add to same total number of smooth operations $v = 3$.

2.4 Appendix. Tables for each combination of $\nu = (\nu_1, \nu_2)$

The following tables are the result of running problem 2 program on the test problem. The fields for each table are described above. The last row in each table contain the result for $k = 15$. The value of the average convergence factor used is that for $k = 15$ under the column heading convergence factor. This below is link to the text file containing the tables as they are printed by the matlab function.

Result for $v_1 = 0, v_2 = 0$

cycle	|residue|	ratio	|error|	ratio	convergence factor
1	4.197646e+004	0.000000	3.791167e+000	0.000000	0.836431
2	4.197672e+004	1.000006	3.618870e+000	0.954553	0.893542
3	4.197682e+004	1.000002	3.571746e+000	0.986978	0.923661
4	4.197685e+004	1.000001	3.557261e+000	0.995944	0.941224
5	4.197686e+004	1.000000	3.552573e+000	0.998682	0.952445
6	4.197687e+004	1.000000	3.551027e+000	0.999565	0.960141
7	4.197687e+004	1.000000	3.550514e+000	0.999856	0.965717
8	4.197687e+004	1.000000	3.550344e+000	0.999952	0.969931
9	4.197687e+004	1.000000	3.550287e+000	0.999984	0.973225
10	4.197687e+004	1.000000	3.550268e+000	0.999995	0.975870
11	4.197687e+004	1.000000	3.550262e+000	0.999998	0.978039
12	4.197687e+004	1.000000	3.550260e+000	0.999999	0.979850
13	4.197687e+004	1.000000	3.550259e+000	1.000000	0.981386
14	4.197687e+004	1.000000	3.550259e+000	1.000000	0.982704
15	4.197687e+004	1.000000	3.550259e+000	1.000000	0.983848

.

Result for $v_1 = 0, v_2 = 1$

cycle	|residue|	ratio	|error|	ratio	convergence factor
1	6.512884e+003	0.000000	1.841085e+000	0.000000	0.406191
2	1.096932e+003	0.168425	7.221287e-001	0.392230	0.399150
3	2.518487e+002	0.229594	2.950874e-001	0.408636	0.402287
4	6.501181e+001	0.258138	1.172566e-001	0.397362	0.401050
5	1.807445e+001	0.278018	4.592625e-002	0.391673	0.399157
6	5.295406e+000	0.292977	1.774863e-002	0.386459	0.397012
7	1.610302e+000	0.304094	6.756385e-003	0.380671	0.394635
8	5.023475e-001	0.311959	2.532700e-003	0.374860	0.392107
9	1.594271e-001	0.317364	9.349623e-004	0.369156	0.389488
10	5.118833e-002	0.321077	3.401773e-004	0.363841	0.386844
11	1.656749e-002	0.323658	1.221213e-004	0.358993	0.384226
12	5.392540e-003	0.325489	4.331077e-005	0.354654	0.381670
13	1.762359e-003	0.326814	1.519371e-005	0.350807	0.379202
14	5.776815e-004	0.327789	5.278592e-006	0.347420	0.376839
15	1.897765e-004	0.328514	1.818208e-006	0.344449	0.374588

.

Result for $v_1 = 0, v_2 = 2$

```
cycle  |residue|          ratio         |error|          ratio    convergence
   factor
 1   2.332824e+003      0.000000   1.130306e+000        0.000000      0.249375
 2   1.658927e+002      0.071112   2.573654e-001        0.227695      0.238289
 3   1.740638e+001      0.104926   5.746902e-002        0.223297      0.233183
 4   2.213859e+000      0.127187   1.255196e-002        0.218413      0.229399
 5   3.107344e-001      0.140359   2.675256e-003        0.213135      0.226050
 6   4.627116e-002      0.148909   5.553596e-004        0.207591      0.222863
 7   7.199976e-003      0.155604   1.123881e-004        0.202370      0.219813
 8   1.162230e-003      0.161421   2.224051e-005        0.197890      0.216945
 9   1.934709e-004      0.166465   4.319318e-006        0.194210      0.214293
10   3.300557e-005      0.170597   8.260186e-007        0.191238      0.211868
11   5.734479e-006      0.173743   1.559942e-007        0.188851      0.209664
12   1.009112e-006      0.175973   2.916007e-008        0.186930      0.207668
13   1.790692e-007      0.177452   5.405721e-009        0.185381      0.205863
14   3.194089e-008      0.178372   9.953326e-010        0.184126      0.204228
15   5.714226e-009      0.178900   1.822506e-010        0.183105      0.202747
```

.

Result for $v_1 = 0, v_2 = 3$

```
cycle  |residue|          ratio     |error|             ratio   convergence
   factor
 1   1.279733e+003      0.000000   8.246208e-001        0.000000      0.181933
 2   6.128118e+001      0.047886   1.239874e-001        0.150357      0.165393
 3   4.197852e+000      0.068502   1.807404e-002        0.145773      0.158576
 4   3.457379e-001      0.082361   2.513173e-003        0.139049      0.153451
 5   3.161771e-002      0.091450   3.385354e-004        0.134704      0.149504
 6   3.097433e-003      0.097965   4.446410e-005        0.131343      0.146311
 7   3.196538e-004      0.103200   5.723743e-006        0.128727      0.143659
 8   3.438009e-005      0.107554   7.254505e-007        0.126744      0.141427
 9   3.818351e-006      0.111063   9.087146e-008        0.125262      0.139533
10   4.343261e-007      0.113747   1.128253e-008        0.124159      0.137913
11   5.025553e-008      0.115709   1.391556e-009        0.123337      0.136520
12   5.884932e-009      0.117100   1.707747e-010        0.122722      0.135313
13   6.948378e-010      0.118071   2.087896e-011        0.122260      0.134261
14   8.250871e-011      0.118745   2.545414e-012        0.121913      0.133339
15   9.836463e-012      0.119217   3.096536e-013        0.121652      0.132526
```

.

Result for $v_1 = 0, v_2 = 4$

```
cycle  |residue|          ratio      |error|            ratio    convergence
```

```
   factor
 1   8.351773e+002      0.000000   6.609090e-001          0.000000      0.145814
 2   3.043485e+001      0.036441   7.256809e-002          0.109800      0.126532
 3   1.541666e+000      0.050655   7.850138e-003          0.108176      0.120091
 4   9.413409e-002      0.061060   8.032536e-004          0.102324      0.115379
 5   6.459266e-003      0.068618   7.934903e-005          0.098785      0.111851
 6   4.798607e-004      0.074290   7.650179e-006          0.096412      0.109116
 7   3.773372e-005      0.078635   7.251982e-007          0.094795      0.106945
 8   3.090474e-006      0.081902   6.795204e-008          0.093701      0.105192
 9   2.605052e-007      0.084293   6.317243e-009          0.092966      0.103758
10   2.240698e-008      0.086014   5.841772e-010          0.092473      0.102570
11   1.954999e-009      0.087250   5.382824e-011          0.092144      0.101575
12   1.723247e-010      0.088146   4.948089e-012          0.091924      0.100734
13   1.530346e-011      0.088806   4.541247e-013          0.091778      0.100015
14   1.366630e-012      0.089302   4.163526e-014          0.091682      0.099395
15   1.225631e-013      0.089683   3.814695e-015          0.091622      0.098857
```

.

Result for $v_1 = 1, v_2 = 0$

```
cycle  |residue|            ratio     |error|             ratio     convergence
   factor
 1   6.179909e+003      0.000000   1.360828e+000          0.000000      0.300234
 2   1.275576e+003      0.206407   4.199343e-001          0.308587      0.304382
 3   3.536744e+002      0.277266   1.341956e-001          0.319563      0.309361
 4   1.100024e+002      0.311027   4.345678e-002          0.323832      0.312917
 5   3.621869e+001      0.329254   1.417374e-002          0.326157      0.315521
 6   1.220133e+001      0.336879   4.640907e-003          0.327430      0.317475
 7   4.148812e+000      0.340030   1.522676e-003          0.328099      0.318972
 8   1.413642e+000      0.340734   5.000095e-004          0.328375      0.320132
 9   4.811293e-001      0.340347   1.642075e-004          0.328409      0.321041
10   1.633133e-001      0.339437   5.390811e-005          0.328293      0.321759
11   5.524973e-002      0.338305   1.768674e-005          0.328091      0.322330
12   1.862458e-002      0.337098   5.798496e-006          0.327844      0.322786
13   6.255900e-003      0.335895   1.899475e-006          0.327581      0.323152
14   2.094091e-003      0.334739   6.217315e-007          0.327318      0.323448
15   6.986989e-004      0.333653   2.033475e-007          0.327067      0.323688
```

.

Result for $v_1 = 1, v_2 = 1$

```
    cycle      |residue|        ratio          |error|         ratio    convergence
   factor
        1    1.030835e+003     0.000000    4.727264e-001     0.000000     0.103251
        2    5.883607e+001     0.057076    5.305423e-002     0.112230     0.107647
        3    3.801623e+000     0.064614    6.112766e-003     0.115217     0.110114
        4    2.784357e-001     0.073241    7.123738e-004     0.116539     0.111686
```

5	2.222565e-002	0.079823	8.361417e-005	0.117374	0.112801
6	1.895919e-003	0.085303	9.862335e-006	0.117951	0.113643
7	1.697849e-004	0.089553	1.167283e-006	0.118358	0.114305
8	1.581453e-005	0.093144	1.384918e-007	0.118645	0.114839
9	1.525515e-006	0.096463	1.645918e-008	0.118846	0.115277
10	1.520761e-007	0.099688	1.958413e-009	0.118986	0.115643
11	1.563620e-008	0.102818	2.332146e-010	0.119083	0.115952
12	1.653635e-009	0.105757	2.778772e-011	0.119151	0.116215
13	1.792425e-010	0.108393	3.312230e-012	0.119198	0.116442
14	1.983378e-011	0.110653	3.949174e-013	0.119230	0.116639
15	2.231664e-012	0.112518	4.709496e-014	0.119253	0.116811

.

Result for $v_1 = 1, v_2 = 2$

cycle	|**residue**|	ratio	|**error**|	ratio	convergence factor
1	3.491916e+002	0.000000	3.054693e-001	0.000000	0.066754
2	1.021899e+001	0.029265	2.334872e-002	0.076436	0.071431
3	4.299404e-001	0.042073	1.841565e-003	0.078872	0.073830
4	2.178631e-002	0.050673	1.473129e-004	0.079993	0.075325
5	1.245425e-003	0.057165	1.187207e-005	0.080591	0.076350
6	7.801009e-005	0.062637	9.606537e-007	0.080917	0.077093
7	5.235377e-006	0.067112	7.790676e-008	0.081098	0.077652
8	3.692823e-007	0.070536	6.325950e-009	0.081199	0.078087
9	2.697616e-008	0.073050	5.140286e-010	0.081257	0.078433
10	2.019954e-009	0.074879	4.178622e-011	0.081292	0.078714
11	1.539723e-010	0.076226	3.397763e-012	0.081313	0.078947
12	1.189244e-011	0.077238	2.763303e-013	0.081327	0.079143
13	9.277782e-013	0.078014	2.247589e-014	0.081337	0.079309
14	7.294259e-014	0.078621	1.828290e-015	0.081345	0.079453
15	5.769829e-015	0.079101	1.487324e-016	0.081351	0.079578

.

Result for $v_1 = 1, v_2 = 3$

cycle	|**residue**|	ratio	|**error**|	ratio	convergence factor
1	1.748348e+002	0.000000	2.273544e-001	0.000000	0.049683
2	4.020464e+000	0.022996	1.322844e-002	0.058184	0.053766
3	1.341199e-001	0.033359	7.939589e-004	0.060019	0.055774
4	5.362431e-003	0.039982	4.829726e-005	0.060831	0.056998
5	2.429919e-004	0.045314	2.956914e-006	0.061223	0.057819
6	1.205611e-005	0.049615	1.816104e-007	0.061419	0.058404
7	6.374272e-007	0.052872	1.117251e-008	0.061519	0.058839
8	3.520088e-008	0.055223	6.879119e-010	0.061572	0.059174
9	2.002565e-009	0.056890	4.237580e-011	0.061601	0.059439

cycle	|residue|	ratio	|error|	ratio	convergence factor
10	1.162928e-010	0.058072	2.611069e-012	0.061617	0.059653
11	6.852108e-012	0.058921	1.609119e-013	0.061627	0.059830
12	4.079843e-013	0.059541	9.917484e-015	0.061633	0.059978
13	2.447990e-014	0.060002	6.112860e-016	0.061637	0.060104
14	1.477344e-015	0.060349	3.767977e-017	0.061640	0.060213
15	8.954816e-017	0.060614	2.322670e-018	0.061642	0.060307

.

Result for $v_1 = 1, v_2 = 4$

cycle	|residue|	ratio	|error|	ratio	convergence factor
1	1.042637e+002	0.000000	1.827858e-001	0.000000	0.039944
2	2.005234e+000	0.019232	8.672196e-003	0.047445	0.043533
3	5.546631e-002	0.027661	4.241753e-004	0.048912	0.045257
4	1.849251e-003	0.033340	2.101081e-005	0.049533	0.046290
5	6.998200e-005	0.037843	1.046739e-006	0.049819	0.046975
6	2.890955e-006	0.041310	5.229125e-008	0.049956	0.047459
7	1.267590e-007	0.043847	2.615867e-009	0.050025	0.047818
8	5.783896e-009	0.045629	1.309520e-010	0.050061	0.048092
9	2.709911e-010	0.046853	6.558096e-012	0.050080	0.048309
10	1.292323e-011	0.047689	3.285046e-013	0.050091	0.048485
11	6.237388e-013	0.048265	1.645757e-014	0.050098	0.048629
12	3.035654e-014	0.048669	8.245760e-016	0.050103	0.048750
13	1.486172e-015	0.048957	4.131659e-017	0.050106	0.048853
14	7.307172e-017	0.049168	2.070333e-018	0.050109	0.048942
15	3.604223e-018	0.049324	1.037465e-019	0.050111	0.049019

.

Result for $v_1 = 2, v_2 = 0$

cycle	|residue|	ratio	|error|	ratio	convergence factor
1	2.655117e+003	0.000000	7.196022e-001	0.000000	0.158763
2	2.807305e+002	0.105732	1.214403e-001	0.168760	0.163685
3	4.389822e+001	0.156371	2.105498e-002	0.173377	0.166854
4	7.595636e+000	0.173028	3.666160e-003	0.174123	0.168643
5	1.345020e+000	0.177078	6.386485e-004	0.174201	0.169740
6	2.391057e-001	0.177771	1.111764e-004	0.174081	0.170456
7	4.245722e-002	0.177567	1.933186e-005	0.173885	0.170941
8	7.518789e-003	0.177091	3.357127e-006	0.173658	0.171279
9	1.327337e-003	0.176536	5.821965e-007	0.173421	0.171515
10	2.335780e-004	0.175975	1.008285e-007	0.173186	0.171682
11	4.097873e-005	0.175439	1.743931e-008	0.172960	0.171798
12	7.168928e-006	0.174943	3.012563e-009	0.172746	0.171876
13	1.250905e-006	0.174490	5.198010e-010	0.172544	0.171928
14	2.177582e-007	0.174081	8.959148e-011	0.172357	0.171958

15	3.782731e-008	0.173712	1.542621e-011	0.172184	0.171973

.

Result for $v_1 = 2, v_2 = 1$

cycle	|**residue**|	ratio	|**error**|	ratio	convergence factor
1	3.587387e+002	0.000000	3.008484e-001	0.000000	0.065744
2	1.067855e+001	0.029767	2.317076e-002	0.077018	0.071158
3	4.487287e-001	0.042021	1.842948e-003	0.079538	0.073848
4	2.256832e-002	0.050294	1.485356e-004	0.080597	0.075480
5	1.286179e-003	0.056990	1.204521e-005	0.081093	0.076571
6	8.087232e-005	0.062878	9.796869e-007	0.081334	0.077345
7	5.482293e-006	0.067789	7.979984e-008	0.081454	0.077919
8	3.922682e-007	0.071552	6.504963e-009	0.081516	0.078360
9	2.913427e-008	0.074271	5.304691e-010	0.081548	0.078708
10	2.219978e-009	0.076198	4.326818e-011	0.081566	0.078989
11	1.722107e-010	0.077573	3.529630e-012	0.081576	0.079221
12	1.353118e-011	0.078573	2.879518e-013	0.081581	0.079415
13	1.073247e-012	0.079317	2.349245e-014	0.081585	0.079580
14	8.572977e-014	0.079879	1.916672e-015	0.081587	0.079721
15	6.884988e-015	0.080310	1.563774e-016	0.081588	0.079845

.

Result for $v_1 = 2, v_2 = 2$

cycle	|**residue**|	ratio	|**error**|	ratio	convergence factor
1	1.448199e+002	0.000000	2.243955e-001	0.000000	0.049037
2	3.594191e+000	0.024818	1.315787e-002	0.058637	0.053622
3	1.246015e-001	0.034667	7.947099e-004	0.060398	0.055792
4	5.109282e-003	0.041005	4.857254e-005	0.061120	0.057079
5	2.356797e-004	0.046128	2.984395e-006	0.061442	0.057926
6	1.184050e-005	0.050240	1.838170e-007	0.061593	0.058521
7	6.316880e-007	0.053350	1.133526e-008	0.061666	0.058961
8	3.512166e-008	0.055600	6.994189e-010	0.061703	0.059297
9	2.008926e-009	0.057199	4.316969e-011	0.061722	0.059561
10	1.171965e-010	0.058338	2.664984e-012	0.061733	0.059775
11	6.933122e-012	0.059158	1.645328e-013	0.061739	0.059951
12	4.143073e-013	0.059758	1.015864e-014	0.061742	0.060098
13	2.494234e-014	0.060203	6.272422e-016	0.061745	0.060223
14	1.509931e-015	0.060537	3.872984e-017	0.061746	0.060331
15	9.179027e-017	0.060791	2.391465e-018	0.061747	0.060424

.

Result for $v_1 = 2, v_2 = 3$

cycle	|**residue**|	ratio	|**error**|	ratio	convergence factor
1	8.776592e+001	0.000000	1.811025e-001	0.000000	0.039576
2	1.834493e+000	0.020902	8.646274e-003	0.047742	0.043468
3	5.258689e-002	0.028666	4.246673e-004	0.049116	0.045274
4	1.790526e-003	0.034049	2.109371e-005	0.049671	0.046336
5	6.865074e-005	0.038341	1.052908e-006	0.049916	0.047030
6	2.859888e-006	0.041659	5.267602e-008	0.050029	0.047517
7	1.260987e-007	0.044092	2.638218e-009	0.050084	0.047876
8	5.776016e-009	0.045806	1.322046e-010	0.050111	0.048150
9	2.713848e-010	0.046985	6.626843e-012	0.050126	0.048365
10	1.297016e-011	0.047793	3.322268e-013	0.050133	0.048539
11	6.271114e-013	0.048350	1.665719e-014	0.050138	0.048683
12	3.056636e-014	0.048742	8.352047e-016	0.050141	0.048802
13	1.498397e-015	0.049021	4.187935e-017	0.050143	0.048904
14	7.375846e-017	0.049225	2.099994e-018	0.050144	0.048992
15	3.641929e-018	0.049376	1.053040e-019	0.050145	0.049068

.

Result for $v_1 = 2, v_2 = 4$

cycle	|**residue**|	ratio	|**error**|	ratio	convergence factor
1	5.968518e+001	0.000000	1.526966e-001	0.000000	0.033368
2	1.071291e+000	0.017949	6.187596e-003	0.040522	0.036772
3	2.628731e-002	0.024538	2.577593e-004	0.041657	0.038333
4	7.693324e-004	0.029266	1.085425e-005	0.042110	0.039244
5	2.541368e-005	0.033033	4.592064e-007	0.042307	0.039838
6	9.120864e-007	0.035890	1.946889e-008	0.042397	0.040254
7	3.457788e-008	0.037911	8.262608e-010	0.042440	0.040559
8	1.357937e-009	0.039272	3.508443e-011	0.042462	0.040792
9	5.454790e-011	0.040170	1.490141e-012	0.042473	0.040976
10	2.223580e-012	0.040764	6.330013e-014	0.042479	0.041124
11	9.153192e-014	0.041164	2.689178e-015	0.042483	0.041245
12	3.793199e-015	0.041441	1.142507e-016	0.042485	0.041347
13	1.579443e-016	0.041639	4.854164e-018	0.042487	0.041434
14	6.599536e-018	0.041784	2.062445e-019	0.042488	0.041508
15	2.764795e-019	0.041894	8.763145e-021	0.042489	0.041573

.

Result for $v_1 = 3, v_2 = 0$

cycle	|**residue**|	ratio	|**error**|	ratio	convergence factor
1	1.731895e+003	0.000000	4.772061e-001	0.000000	0.105284
2	1.332957e+002	0.076965	5.436461e-002	0.113923	0.109518

3	1.472030e+001	0.110433	6.363908e-003	0.117060	0.111977
4	1.734653e+000	0.117841	7.494297e-004	0.117762	0.113396
5	2.065908e-001	0.119096	8.847251e-005	0.118053	0.114312
6	2.463413e-002	0.119241	1.045852e-005	0.118212	0.114953
7	2.936045e-003	0.119186	1.237354e-006	0.118311	0.115427
8	3.497108e-004	0.119109	1.464715e-007	0.118375	0.115791
9	4.163098e-005	0.119044	1.734462e-008	0.118416	0.116080
10	4.953753e-006	0.118992	2.054331e-009	0.118442	0.116314
11	5.892482e-007	0.118950	2.433477e-010	0.118456	0.116507
12	7.006984e-008	0.118914	2.882724e-011	0.118461	0.116669
13	8.330007e-009	0.118881	3.414861e-012	0.118460	0.116806
14	9.900258e-010	0.118851	4.044995e-013	0.118453	0.116923
15	1.176348e-010	0.118820	4.790965e-014	0.118442	0.117023

.

Result for $v_1 = 3, v_2 = 1$

cycle	**|residue|**	ratio	**|error|**	ratio	convergence factor
1	1.806744e+002	0.000000	2.248328e-001	0.000000	0.049132
2	4.176399e+000	0.023116	1.317724e-002	0.058609	0.053662
3	1.375470e-001	0.032934	7.958156e-004	0.060393	0.055818
4	5.462156e-003	0.039711	4.864574e-005	0.061127	0.057100
5	2.477092e-004	0.045350	2.989565e-006	0.061456	0.057946
6	1.236926e-005	0.049935	1.841867e-007	0.061610	0.058541
7	6.603866e-007	0.053389	1.136149e-008	0.061685	0.058980
8	3.687804e-008	0.055843	7.012560e-010	0.061722	0.059316
9	2.122171e-009	0.057546	4.329681e-011	0.061742	0.059581
10	1.246363e-010	0.058731	2.673686e-012	0.061752	0.059795
11	7.424379e-012	0.059568	1.651230e-013	0.061759	0.059970
12	4.467406e-013	0.060172	1.019837e-014	0.061762	0.060118
13	2.707929e-014	0.060615	6.298990e-016	0.061765	0.060243
14	1.650363e-015	0.060946	3.890651e-017	0.061766	0.060350
15	1.009940e-016	0.061195	2.403156e-018	0.061767	0.060444

.

Result for $v_1 = 3, v_2 = 2$

cycle	**|residue|**	ratio	**|error|**	ratio	convergence factor
1	8.757448e+001	0.000000	1.810759e-001	0.000000	0.039570
2	1.833553e+000	0.020937	8.646696e-003	0.047752	0.043469
3	5.258255e-002	0.028678	4.247632e-004	0.049124	0.045278
4	1.791120e-003	0.034063	2.110165e-005	0.049679	0.046340
5	6.870264e-005	0.038357	1.053438e-006	0.049922	0.047035
6	2.863317e-006	0.041677	5.270844e-008	0.050035	0.047522
7	1.263069e-007	0.044112	2.640110e-009	0.050089	0.047881

8	5.788133e-009	0.045826	1.323120e-010	0.050116	0.048155
9	2.720719e-010	0.047005	6.632824e-012	0.050130	0.048370
10	1.300844e-011	0.047812	3.325556e-013	0.050138	0.048544
11	6.292148e-013	0.048370	1.667508e-014	0.050142	0.048687
12	3.068072e-014	0.048760	8.361702e-016	0.050145	0.048807
13	1.504560e-015	0.049039	4.193112e-017	0.050147	0.048909
14	7.408807e-017	0.049242	2.102755e-018	0.050148	0.048996
15	3.659443e-018	0.049393	1.054505e-019	0.050149	0.049072

.

Result for $v_1 = 3, v_2 = 3$

cycle	|**residue**|	ratio	|**error**|	ratio	convergence factor
1	5.945561e+001	0.000000	1.526825e-001	0.000000	0.033365
2	1.070297e+000	0.018002	6.187752e-003	0.040527	0.036772
3	2.627477e-002	0.024549	2.577899e-004	0.041661	0.038335
4	7.691734e-004	0.029274	1.085633e-005	0.042113	0.039246
5	2.541274e-005	0.033039	4.593210e-007	0.042309	0.039840
6	9.121526e-007	0.035894	1.947469e-008	0.042399	0.040256
7	3.458310e-008	0.037914	8.265419e-010	0.042442	0.040561
8	1.358230e-009	0.039274	3.509771e-011	0.042463	0.040794
9	5.456298e-011	0.040172	1.490758e-012	0.042475	0.040978
10	2.224324e-012	0.040766	6.332842e-014	0.042481	0.041125
11	9.156769e-014	0.041167	2.690464e-015	0.042484	0.041247
12	3.794885e-015	0.041443	1.143087e-016	0.042487	0.041349
13	1.580225e-016	0.041641	4.856765e-018	0.042488	0.041436
14	6.603117e-018	0.041786	2.063606e-019	0.042489	0.041510
15	2.766417e-019	0.041896	8.768302e-021	0.042490	0.041575

.

Result for $v_1 = 3, v_2 = 4$

cycle	|**residue**|	ratio	|**error**|	ratio	convergence factor
1	4.335915e+001	0.000000	1.321583e-001	0.000000	0.028880
2	6.849016e-001	0.015796	4.660361e-003	0.035263	0.031913
3	1.472864e-002	0.021505	1.688654e-004	0.036234	0.033293
4	3.797836e-004	0.025785	6.183105e-006	0.036616	0.034094
5	1.109514e-005	0.029214	2.274057e-007	0.036779	0.034615
6	3.520256e-007	0.031728	8.380416e-009	0.036852	0.034978
7	1.176545e-008	0.033422	3.091300e-010	0.036887	0.035245
8	4.060755e-010	0.034514	1.140828e-011	0.036904	0.035448
9	1.429858e-011	0.035212	4.211199e-013	0.036914	0.035608
10	5.099387e-013	0.035664	1.554713e-014	0.036919	0.035737
11	1.833989e-014	0.035965	5.740235e-016	0.036922	0.035843
12	6.634068e-016	0.036173	2.119492e-017	0.036923	0.035932

13	2.409612e-017	0.036322	7.826178e-019	0.036925	0.036007
14	8.778704e-019	0.036432	2.889877e-020	0.036926	0.036072
15	3.205646e-020	0.036516	1.067132e-021	0.036927	0.036128

.

Result for $v_1 = 4, v_2 = 0$

cycle	|residue|	ratio	|error|	ratio	convergence factor
1	1.305638e+003	0.000000	3.571047e-001	0.000000	0.078787
2	7.990465e+001	0.061200	3.078639e-002	0.086211	0.082415
3	6.837976e+000	0.085577	2.728251e-003	0.088619	0.084433
4	6.150176e-001	0.089941	2.433551e-004	0.089198	0.085600
5	5.555443e-002	0.090330	2.176670e-005	0.089444	0.086355
6	5.009766e-003	0.090178	1.949873e-006	0.089581	0.086885
7	4.509998e-004	0.090024	1.748412e-007	0.089668	0.087277
8	4.055972e-005	0.089933	1.568822e-008	0.089728	0.087580
9	3.645908e-006	0.089890	1.408358e-009	0.089772	0.087821
10	3.276772e-007	0.089875	1.264748e-010	0.089803	0.088017
11	2.945016e-008	0.089876	1.136069e-011	0.089826	0.088180
12	2.647044e-009	0.089882	1.020666e-012	0.089842	0.088317
13	2.379448e-010	0.089891	9.171024e-014	0.089853	0.088434
14	2.139097e-011	0.089899	8.241165e-015	0.089861	0.088536
15	1.923168e-012	0.089906	7.405975e-016	0.089866	0.088624

.

Result for $v_1 = 4, v_2 = 1$

cycle	|residue|	ratio	|error|	ratio	convergence factor
1	1.093773e+002	0.000000	1.812869e-001	0.000000	0.039616
2	2.098783e+000	0.019188	8.652527e-003	0.047728	0.043484
3	5.704834e-002	0.027182	4.249640e-004	0.049114	0.045285
4	1.882553e-003	0.032999	2.111018e-005	0.049675	0.046345
5	7.106054e-005	0.037747	1.053863e-006	0.049922	0.047039
6	2.943509e-006	0.041423	5.273130e-008	0.050036	0.047526
7	1.297051e-007	0.044065	2.641368e-009	0.050091	0.047884
8	5.950199e-009	0.045875	1.323812e-010	0.050118	0.048158
9	2.802074e-010	0.047092	6.636618e-012	0.050133	0.048373
10	1.342545e-011	0.047913	3.327617e-013	0.050140	0.048547
11	6.507751e-013	0.048473	1.668619e-014	0.050145	0.048690
12	3.179931e-014	0.048864	8.367653e-016	0.050147	0.048810
13	1.562661e-015	0.049141	4.196279e-017	0.050149	0.048912
14	7.710595e-017	0.049343	2.104432e-018	0.050150	0.048999
15	3.816096e-018	0.049492	1.055388e-019	0.050151	0.049075

.

Result for $v_1 = 4, v_2 = 2$

```
cycle  |residue|          ratio       |error|            ratio     convergence
   factor
 1   5.956504e+001        0.000000   1.526861e-001         0.000000    0.033366
 2   1.070895e+000        0.017979   6.187900e-003         0.040527    0.036773
 3   2.628769e-002        0.024547   2.577967e-004         0.041661    0.038335
 4   7.695512e-004        0.029274   1.085667e-005         0.042113    0.039246
 5   2.542668e-005        0.033041   4.593380e-007         0.042309    0.039841
 6   9.127487e-007        0.035897   1.947552e-008         0.042399    0.040256
 7   3.460995e-008        0.037918   8.265816e-010         0.042442    0.040561
 8   1.359449e-009        0.039279   3.509957e-011         0.042464    0.040794
 9   5.461792e-011        0.040177   1.490844e-012         0.042475    0.040978
10   2.226781e-012        0.040770   6.333235e-014         0.042481    0.041126
11   9.167659e-014        0.041170   2.690642e-015         0.042484    0.041247
12   3.799678e-015        0.041447   1.143166e-016         0.042487    0.041349
13   1.582320e-016        0.041644   4.857121e-018         0.042488    0.041436
14   6.612229e-018        0.041788   2.063764e-019         0.042489    0.041510
15   2.770358e-019        0.041897   8.769001e-021         0.042490    0.041575
```

.

Result for $v_1 = 4, v_2 = 3$

```
cycle   |residue|          ratio      |error|             ratio      convergence
   factor
 1   4.335295e+001       0.000000   1.321581e-001         0.000000      0.028880
 2   6.848903e-001       0.015798   4.660374e-003         0.035264      0.031913
 3   1.472868e-002       0.021505   1.688664e-004         0.036235      0.033293
 4   3.797890e-004       0.025786   6.183161e-006         0.036616      0.034094
 5   1.109540e-005       0.029215   2.274083e-007         0.036779      0.034615
 6   3.520365e-007       0.031728   8.380531e-009         0.036852      0.034978
 7   1.176590e-008       0.033422   3.091348e-010         0.036887      0.035245
 8   4.060938e-010       0.034514   1.140848e-011         0.036905      0.035448
 9   1.429932e-011       0.035212   4.211278e-013         0.036914      0.035608
10   5.099674e-013       0.035664   1.554744e-014         0.036919      0.035737
11   1.834100e-014       0.035965   5.740358e-016         0.036922      0.035843
12   6.634495e-016       0.036173   2.119540e-017         0.036923      0.035932
13   2.409774e-017       0.036322   7.826365e-019         0.036925      0.036007
14   8.779315e-019       0.036432   2.889949e-020         0.036926      0.036072
15   3.205874e-020       0.036516   1.067159e-021         0.036927      0.036128
```

.

Result for $v_1 = 4, v_2 = 4$

```
cycle   |residue|          ratio      |error|               ratio    convergence
```

```
   factor
 1   3.318582e+001      0.000000   1.163430e-001         0.000000     0.025424
 2   4.677808e-001      0.014096   3.627130e-003         0.031176     0.028154
 3   8.965233e-003      0.019165   1.161573e-004         0.032025     0.029389
 4   2.073972e-004      0.023134   3.758056e-006         0.032353     0.030104
 5   5.448802e-006      0.026272   1.221050e-007         0.032492     0.030567
 6   1.551551e-007      0.028475   3.974903e-009         0.032553     0.030889
 7   4.638434e-009      0.029895   1.295102e-010         0.032582     0.031125
 8   1.427642e-010      0.030779   4.221525e-012         0.032596     0.031306
 9   4.472442e-012      0.031327   1.376361e-013         0.032603     0.031447
10   1.416727e-013      0.031677   4.487966e-015         0.032607     0.031561
11   4.520389e-015      0.031907   1.463521e-016         0.032610     0.031655
12   1.449490e-016      0.032066   4.772759e-018         0.032611     0.031734
13   4.664276e-018      0.032179   1.556520e-019         0.032613     0.031801
14   1.504811e-019      0.032262   5.076339e-021         0.032613     0.031858
15   4.864498e-021      0.032326   1.655598e-022         0.032614     0.031908
```

.

3 Problem 3

3. Use your V-cycle code to solve

$$\Delta u = -\exp\left(-(x-0.25)^2 - (y-0.6)^2\right)$$

on the unit square $(0,1)\times(0,1)$ with homogeneous Dirichlet boundary conditions using a grid spacing of 2^{-7}. How many steps of pre and postsmoothing did you use? What tolerance did you use? How many cycles did it take to converge? Compare the amount of work needed to reach convergence with your solvers from Homework 3 taking into account how much work is involved in a V-cycle.

Figure 6: problem 3

This problem was solved using multigrid V cycle method. The following is the solution found

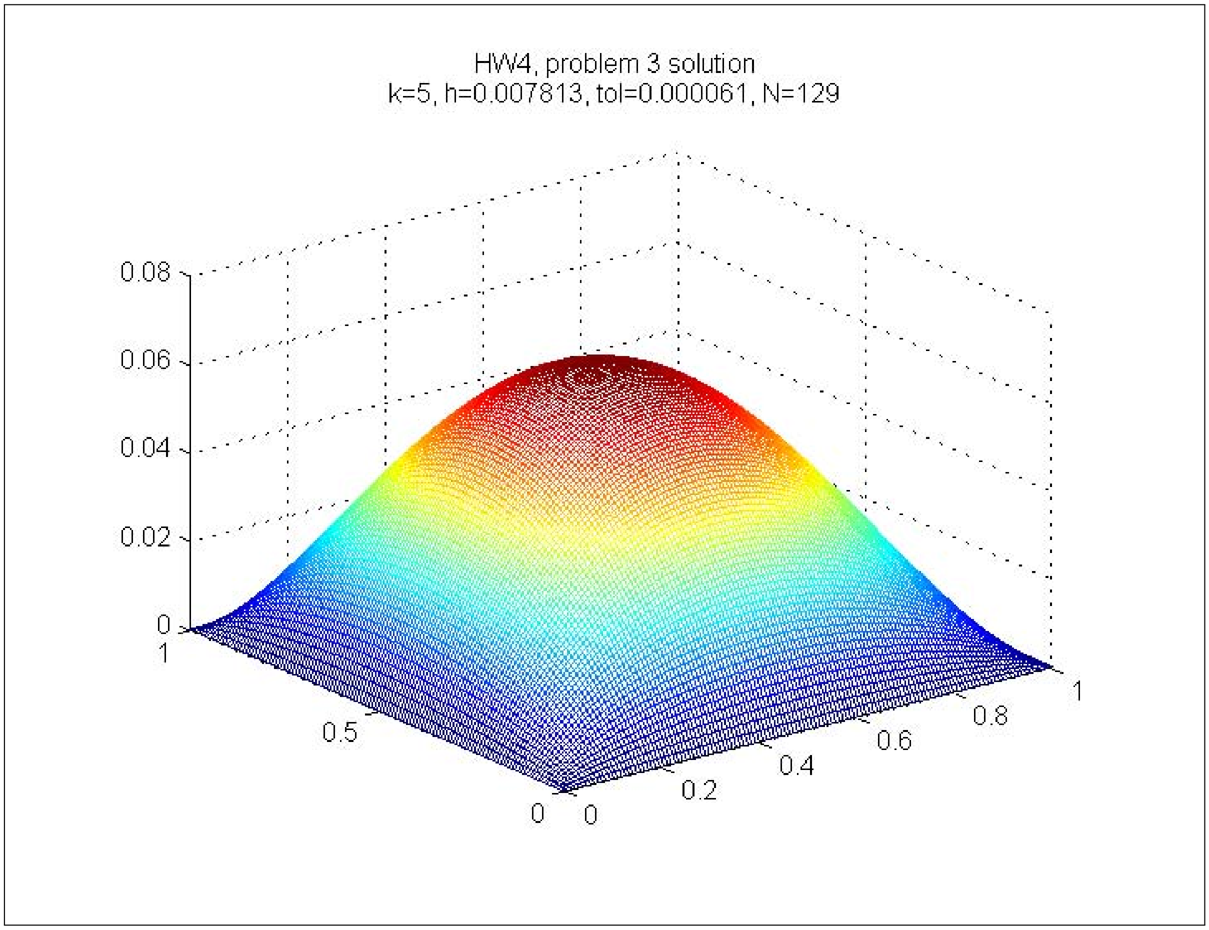

Figure 7: problem 3 solutions found earlier

Tolerance used is $h^2 = 0.000061$, the number of grid points along one dimension is $n = 129$, the spacing is 2^{-7}.

V cycle method converged in 5 iterations. The number of pre-smooth is 1 and the number of post smooth is 1. These are selected since in problem 2 it was found they lead to the most efficient solver.

Hence, the amount of work done

$$W = \left(\frac{\log(\varepsilon)}{\log(\rho)}\right)\left(\frac{3}{4}\left(7\left(v_1 + v_2\right) + 13\right)N\right)$$

From the table in problem 2, $\rho = 0.116811$ for the solver (1,1), and given that $N = (n-2)^2 = 127^2 = 16129$, hence the above becomes

$$\begin{aligned} W &= \left(\frac{\log_{10}(0.000061)}{\log_{10}(0.116811)}\right)\left(\frac{3}{4}\left(7(1+1)+13\right)16129\right) \\ &= 1.4762 \times 10^6 \text{ operations} \end{aligned}$$

The above is compared with the solvers used in HW3, for same tolerance as above and same h, from HW3, the results were the following

method	Number of iterations
Jacobi	31702
Gauss-Seidel	15852
SOR	306

To compare work between all methods, it is required to find the work per iteration for the Jacobi, GS and SOR.

Work per iteration in these methods required only one smooth operation, and one calculation for the residue. No mapping between different grid sizes was needed. Hence, assuming about 13 flops to calculate the averaging and residue per one grid point, work per iteration for the above solver becomes

$$\textit{Work Per iteration} = (6N + 7N) = 13N$$

where N is the total number of grid points which is 16129 in this example.

Therefore, total work can be found for all the methods, including the multigrid solver. The following table summarizes the result

method	Number of iterations	W (flops)
Jacobi	31702	$31702 \times 13 \times 16129 = 6.6472 \times 10^9$
Gauss-Seidel	15852	$15852 \times 13 \times 16129 = 3.3238 \times 10^9$
SOR	306	$306 \times 13 \times 16129 = 6.4161 \times 10^7$
Multigrid V Cycle	5	1.4762×10^6

Using the multigrid as a base measure, and normalizing other solvers relative to it, the above becomes

method	work
Jacobi	$\frac{6.6472\times10^9}{1.4762\times10^6} = 4502.9$
Gauss-Seidel	$\frac{3.3238\times10^9}{1.4762\times10^6} = 2251.6$
SOR	$\frac{6.4161\times10^7}{1.4762\times10^6} = 43.464$
Multigrid V Cycle	1

The above shows clearly that Multigrid is the most efficient solver. SOR required about 44 times as much work, GS over 2251 more work, and Jacobi about 4500 more work.

4 Note on using Full Multigrid cycle (FMG) to improve convergence

It was found that using FMG to determine a better initial guess solution before initiating the V cycle algorithm resulted in about 40% reduction in the number of iterations needed to convergence by the V cycle algorithm.

The following is a plot of one of the tests performed showing the difference in number of iterations needed to converge. All other parameters are kept the same. This shows that with FMG cycle, convergence reached in 4 iterations, while without FMG, it was reached in 7 cycles. The cost of the FMG cycle itself was not taken into account. It is estimated that the FMG correction cycle adds about $\frac{1}{2}$ the cost of one V cycle to the total cost.

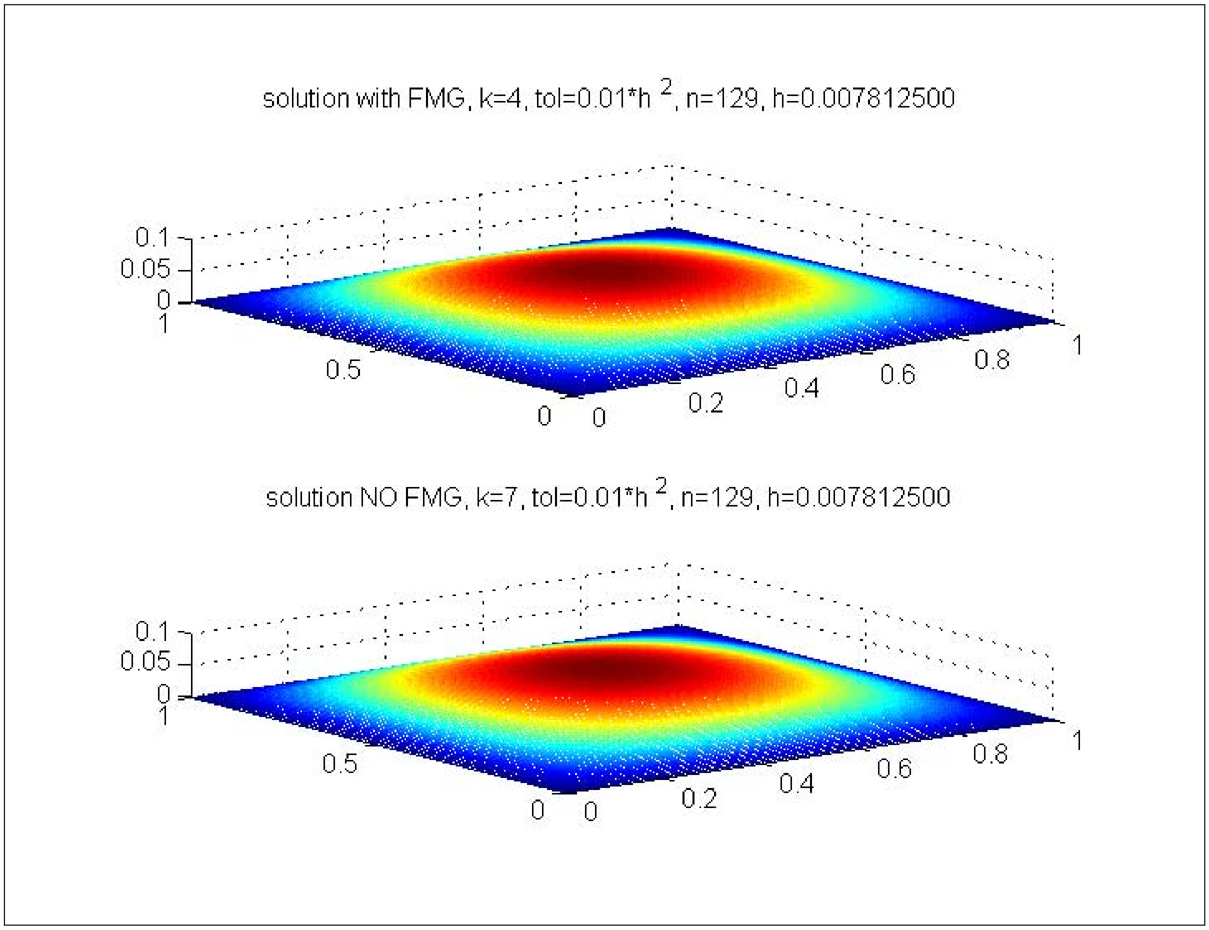

Figure 8: compare with FMG

The following is a small function written to compare convergence when using FMG and without using FMG which generated the above result. (see code on web page)

5 References

[1] Thor Gjesdal, Analysis of a new Red-Black ordering for Gauss-Seidel smoothing in cell-centered multigrid. ref no. CMR-93-A200007, November 1993.

[2] William L. Briggs, A multigrid tutorial, W-7405-Eng-48.

[3] Robert Guy, Lecture notes, Math 228a, Fall 2010. Mathematics dept. UC Davis.

[4] Jim Demmel, Lecture notes, CS267, 1996. UC Berkeley.

[5] P. Wesseling, A survey of Fourier smoothing analysis results. ISNM 98, Multigrid methods II page 105-106.

6 Source code listing

```
function nma_build_HW4()

list = dir('*.m');

if isempty(list)
    fprintf('no matlab files found\n');
    return
end

for i=1:length(list)
    name=list(i).name;
    fprintf('processing %s\n',name)
    p0 = fdep(list(i).name,'-q');
    [pathstr, name_of_matlab_function, ext] = fileparts(name);

    %make a zip file of the m file and any of its dependency
    p1=dir([name_of_matlab_function '.fig']);
    if length(p1)==1
        files_to_zip =[p1(1).name;p0.fun];
    else
        files_to_zip =p0.fun;
    end

    zip([name_of_matlab_function '.zip'],files_to_zip)

end

end
```

```
%-------------------------------------------------------
% This function is the prologation operator for 1D
% it takes 1D coarse grid of spacing 2h and generate 1D fine
% grid of spacing h by linear interpolation
%
% INPUT:
%  c the coarse 1D grid, spacing 2h
% OUTPUT:
%  f the fine 1D grid, spacing h
```

```
%
% EXAMPLE:
% nma_c2f_1D( [0 1 0] )
%         0    0.5000    1.0000    0.5000         0
%
% Nasser M. Abbasi
% Math 228a, UC Davis fall 2010
function  f = nma_c2f_1D( c )
if ndims(c) > 2
   error('nma_c2f_1D:: input number of dimensions too large');
end

[n,nCol]  = size(c(:));
if nCol>1
   error('nma_c2f_1D::input must be a vector');
end

valid_grid_points = log2(n-1);
if  round(valid_grid_points) ~= valid_grid_points
   error('nma_c2f_1D:: invalid number of grid points value');
end

if n < 3
   error('length of coarse grid must be at least 3');
end

fine_n      = 2*(n-2) + 3;
f           = zeros(fine_n ,1);
f(1:2:end) = c;
indx        = 2:2:fine_n;
f(indx)     = ( f(indx-1) + f(indx+1) )/2;
f           = f(:)';
end
```

```
%-------------------------------------------
% function for debugging
%
function nma_DEBUG(msg)
debug = false;
if debug

    fprintf(msg);
    fprintf('\n');
end
end
```

```
%--------------------------------------------------------
% This function is the restriction full weight operator
% for mapping 1D fine grid to 1D coarse grid
%
% INPUT:
%   f the fine 1D grid, spacing h
% OUTPUT:
%   c the corase 1D grid, spacing 2h
%
% EXAMPLE
%nma_c2f_1D( [0 1 0] )
%         0    0.5000    1.0000    0.5000         0
%nma_f2c_1D(ans)
%         0    0.7500         0
%
% Nasser M. Abbasi
% Math 228a, UC Davis fall 2010

function c = nma_f2c_1D( f )
if ndims(f) > 2
   error('nma_f2c_1D:: input number of dimensions too large');
end

[n,nCol]  = size(f(:));
if nCol>1
   error('nma_f2c_1D::input must be a vector');
end

valid_grid_points = log2(n-1);
if  round(valid_grid_points) ~= valid_grid_points
   error('nma_f2c_1D:: invalid number of grid points value');
end

if n < 5
   error('length of fine grid cant be smaller than 5');
end

if 2*floor(n/2) == n
   error('length of fine grid must be odd number');
end

c           = zeros((n+1)/2,1);
indx        = 3:2:n-2;
c(2:end-1) = (1/4)*f(indx-1) + (1/2)*f(indx) + (1/4)*f(indx+1);
c           = c(:)';
end
```

```
function  nma_HW4_residue_animation

myforce = @(X,Y) -exp( -(X-0.25).^2 - (Y-0.6).^2 ); % RHS of PDE
N = 6;         % number of levels
n = 2^N+1;     % total number of grid points, always odd number
h = 1/(n-1);  % mesh spacing
[X,Y] = meshgrid(0:h:1,0:h:1);  % for plotting solution

% use a problem whose solution is known Lu=0, since B.C. are zero
f     = myforce(X,Y);   %  RHS of the problem
u     = 0.*X;  % initial guess of solution, use random

%-- select number of cycle to run. choose number not too large
number_of_cycles =10;

METHOD  = 1;  % GSRB
mu1=2; mu2=1;
k=0;


for i = 1:number_of_cycles

   k = k+1;

   u = nma_V_cycle(u,f,mu1,mu2,METHOD);  %--- CALL V CYCLE
   the_residue  = nma_find_residue(u,f);

   mesh(X,Y, u);  drawnow; title(sprintf('k=%d\n',k));
   pause(.01);
end
end
```

```
%-----------------------------------------
%This function perform full mutligrid cycle
%to determine a good initial guess to use
%for initial solution to the V cycle interative
%solver
%
% INPUT
%   f: 2D grid, the force on the orginal fine grid
% OUTPUT:
%   u: estimated solution on the fine grid obtained
%      by running FMG cycle once
%
% Example use: see nma_test_FMG.m
%
```

```
% by Nasser M. Abbasi
% Math 228a, UC Davis, Fall 2010

function u = nma_initial_solution_guess_using_FMG( f )

nma_validate_dimensions_1(f); %asserts dimensions make sense
   [n,~] = size(f);
   if n<3
      error('initial_solution_guess_using_FMG:: size of f too small');
   end

   mu1 = 1;
   mu2 = 1;
   smoother = 1; %GSRB

   k = 3;
   while k <= n
      fk = f;
      [current_size,~] = size(fk);
      while current_size ~= k
            fk = nma_f2c(fk);
            [current_size,~] = size(fk);
      end

      if k == 3
         u      = zeros(3,3);
         u(2,2) = -0.25*(0.5)^2*fk(2,2);
      else
        u = nma_c2f(u);
        u = nma_V_cycle(u,fk,mu1,mu2,smoother);
      end
      k = 2*k - 1;
   end

end
```

```
%--------------------------------------------------------
% This function solves problem 2 in HW4
%
%LOGIC
%
%  LOOP over all combinations of mu1 and mu2
%       run V cycle 15 times on the problem
%       record the result in table, which contains
%       average convergence factor
%  END LOOP
%
```

```
% By Nasser M. Abbasi
% Math 228A, UC Davis
function nma_math_228_fall_2010_HW4_problem2

N = 6;         % number of levels
n = 2^N+1;     % total number of grid points, always odd number
h = 1/(n-1);   % mesh spacing

[X,Y] = meshgrid(0:h:1,0:h:1);
% use a problem whose solution is known Lu=0, since B.C. are zero

%force = @(X,Y) -2* ( (1-6*X.^2).*Y.^2.*(1-Y.^2)+(1-6*Y.^2).*X.^2.*(1-X.^2)); % RHS of PDE
%force = @(X,Y) -exp( -(X-0.25).^2 - (Y-0.6).^2 );  % RHS of PDE
%exact = @(X,Y) (X.^4-X.^2).*(Y.^4-Y.^2);
%f = force(X,Y);

table = zeros(20,6);
f     = zeros(n,n);   %  RHS of the problem
exact = zeros(n,n);   %  exact solution

initial_guess = rand(n,n);  % initial guess of solution, use random
initial_guess(:,1)=0; initial_guess(:,end)=0; %initialize B.C. to zero
initial_guess(1,:)=0; initial_guess(end,:)=0;

%-- select mu1 and mu2, these are the number of pre-smooth and
%-- number of post smooth to be used inside the V cycle function
mu1_values = 0:2;
mu2_values = 0:2;

%-- select number of cycle to run. choose number not too large
number_cycles = 10;

%-- select where to send the result, either to stdout or to a file
%fileID = fopen('table_result_work','w');
fileID  = 1;
METHOD  = 1;  % GSRB

for i = 1:length(mu1_values)

    mu1 = mu1_values(i);
    for j = 1:length(mu2_values)
        mu2   = mu2_values(j);

        table = run_V_cycle(initial_guess, ...
            exact,...
            f,...
            mu1,...
```

```
            mu2,...
            number_cycles,...
            h,...
            METHOD);

        print_table(table,mu1,mu2,fileID);
    end
end
%fclose(fileID);
end

%-------------
% This function runs the V cycle for k times, recording
% the average convergence rate at each step k. In the end
% it returns table containing the results found
%
%
function table = run_V_cycle(u,exact,f,mu1,mu2,number_of_cycles,h,METHOD)

%-- initialization of data and storage
table = zeros(number_of_cycles,7);  %initialize table for result collection

last_error_norm    = 0;
last_residue_norm  = 0;
initial_error_norm = nma_find_norm(u-exact);
k=0;
[X,Y] = meshgrid(0:h:1,0:h:1);  % for plotting solution

for i = 1:number_of_cycles

    k = k+1;

    u = nma_V_cycle(u,f,mu1,mu2,METHOD);  %--- CALL V CYCLE

    e = exact-u ;
    norm_error   = nma_find_norm(e);
    the_residue  = nma_find_residue(u,f);
    norm_residue = nma_find_norm(the_residue);

    % fill in table for analysis
    table(k,1) = k;
    table(k,2) = norm_residue;
    table(k,4) = norm_error;
    table(k,6) = (norm_error/initial_error_norm)^(1/k);

    if k > 1
        table(k,3) = norm_residue/last_residue_norm;
```

```
        table(k,5) = norm_error/last_error_norm;
        table(k,7) = -6/log10(table(k,6))*(3/4)*(7*(mu1+mu2)+13);
    end

    last_error_norm   = norm_error;
    last_residue_norm = norm_residue;

    mesh(X,Y, u);  drawnow; title(sprintf('k=%d\n',k));
end
end
%--------------------------------------------
% This function is called by HW4 problem 2 solver
% to format the results and print it. THe results
% are included in the final report.
%
% this also makes plots of the data
%
function print_table(table,mu1,mu2,fileID)

fprintf(fileID,'mu1=%d, mu2=%d\n',mu1,mu2);

% figure;
% subplot(5,1,1);
% plot(table(:,3),'-o');
% title('residual ratio');
%
% subplot(5,1,2);
% plot(table(:,5),'-o');
% title('error ratio');
%
% subplot(5,1,3);
% plot(table(:,2),'-o');
% title('residual norm');
%
% subplot(5,1,4);
% plot(table(:,4),'-o');
% title('error norm');
%
% subplot(5,1,5);
% plot(table(:,6),'-o');
% title('RAO(M)');

titles={'V-cycle','|residue|','ratio','|error|','ratio','C.F','work'};
wid    = 16;
fms    = {'d','.6e','.6f','.6e','.6f','.6f','.1f'};

nma_format_matrix(titles,table,wid,fms,fileID,true);
```

```

end
```

```
%-------------------------------------------------------
% This function solves problem 3 in HW4
% It uses the code generated in problem 1, which implements
% V cycle.
% By Nasser M. Abbasi
% Math 228A, UC Davis

function nma_math_228_fall_2010_HW4_problem3

%----------- INITIALIZATION SECTION -------------
myforce = @(X,Y) -exp( -(X-0.25).^2 - (Y-0.6).^2 ); % RHS of PDE
h       = 2^-7;          % mesh spacing
n       = 1/h+1;         % total number of grid points, always odd number
tol     = 1*h^2;         % tolerance used
[X,Y]   = meshgrid(0:h:1,0:h:1); % coordinates
f       = myforce(X,Y); % evaluate the force on the grid
u       = zeros(n,n);   % initial guess of solution
mu1     = 1;             % mu1 number of pre-smooth
mu2     = 1;             % mu2 number of post-smooth
METHOD  = 1;             % relaxation method for multigrid: Gause-Seidel B/R

[k, u ] = nma_solver_Vcycle(u,f,mu1,mu2,METHOD,tol,false);

[X,Y] = meshgrid(0:h:1,0:h:1); % coordinates
mesh(X,Y, u);  drawnow; title(sprintf ...
   ('HW4, problem 3 solution\nk=%d, h=%f, tol=%f, N=%d\n',k,h,tol,n));

end
```

```
%---------------------------------------
%This function compares the performance of multigrid
% with and without a FMG initial cycle by solving HW3 poisson
% problem and comparing the number of iterations needed
% to converge when using V cycle.
%
% by Nasser M. Abbasi
% Math 228a, UC Davis, Fall 2010
function nma_math_228_fall_2010_HW4_test_FMG()
%----------- INITIALIZATION SECTION -------------
myforce = @(X,Y) -exp( -(X-0.25).^2 - (Y-0.6).^2 ); % RHS of PDE
h       = 2^-7;          % mesh spacing
n       = 1/h+1;         % total number of grid points, always odd number
tol     = 0.01*h^2;      % tolerance used
```

```
[X,Y]    = meshgrid(0:h:1,0:h:1); % coordinates
f        = myforce(X,Y); % evaluate the force on the grid
mu1      = 1;            % mu1 number of pre-smooth
mu2      = 1;            % mu2 number of post-smooth
METHOD   = 1;            % relaxation method for multigrid: Gause-Seidel B/R


%------------ LOGIC SECTION -------------------

% Solve by doing initial FMG correction
u     = nma_initial_solution_guess_using_FMG(f);
[k,u] = nma_solver_Vcycle(u,f,mu1,mu2,METHOD,tol,false);
subplot(2,1,1);
mesh(X,Y, u);
title(sprintf('solution with FMG, k=%d, tol=%s, n=%d, h=%0.9f\n',k, ...
             '0.01*h^2',n,h));

% Solve without doing an initial FMG correction
u     = zeros(n);
[k,u] = nma_solver_Vcycle(u,f,mu1,mu2,METHOD,tol,false);
subplot(2,1,2);
mesh(X,Y, u);
title(sprintf('solution NO FMG, k=%d, tol=%s, n=%d, h=%0.9f\n',...
              k,'0.01*h^2',n,h));

end
```