Technische Universität München
Lehrstuhl für Informatik V
Univ.-Prof. Dr. M. Bader
Carsten Uphoff, Nikola Tchipev

SS 2018
Sheet 2P
04.05.2018

Scientific Computing II

Multigrid Methods

Exercise 1: Multigrid for Poisson Equation

We want to solve the two-dimensional finite difference approximation

$$\begin{aligned} a\frac{u_{i+1j} - 2u_{ij} + u_{i-1j}}{h_x^2} + b\frac{u_{ij+1} - 2u_{ij} + u_{ij-1}}{h_y^2} &= 0, \quad i \in \{2, ..., N_x\}, j \in \{2, ..., N_y\} \\ u_{ij} &= 0, \quad i = 1 \vee j = 1 \vee i = N_x + 1 \vee j = N_y + 1 \end{aligned} \tag{1}$$

which is defined on a unit square using a Cartesian mesh $((i-1)h_x, (j-1)h_y)$ with $h_x = 1/N_x$, $h_y = 1/N_y$[1]. The number of grid points in each direction should initially be chosen as multiple of two plus one, that is $N_x = N_y = 2^l$, $l \in \mathbb{N}$, and the size of the grid is $N_x + 1 \times N_y + 1$. The coefficients $a, b \in \mathbb{R}$ can be chosen arbitrarily. For the time being, we set them to unity, $a = b = 1$.
The following two matlab functions are provided on the webpage of this course:

- `gaussSeidel(a,b,u,rhs)` carries out one iteration of the Gauss-Seidel method. The input arguments are given by the scalar coefficients a, b from above, the two-dimensional solution $u \in \mathbb{R}^{N_x+1 \times N_y+1}$ and a right hand side $rhs \in \mathbb{R}^{N_x+1 \times N_y+1}$. The mesh size of the current solution u is determined from the number of grid points $N_x + 1$, $N_y + 1$. The function returns the updated solution of u.
- `residual(a,b,u,rhs)` evaluates the residual for the approximate solution u of Eq. (1). The function returns the residual $r \in \mathbb{R}^{N_x+1 \times N_y+1}$.

In the following, we will prepare the different subroutines required for a multigrid solver and finally put them together in the *V-cycle* algorithm.

(a) Implement a matlab function `restrict_fullweighting(r)` to perform the full-weighting restriction. The argument of the function is the residual $r \in \mathbb{R}^{N_x+1 \times N_y+1}$. The function

[1] The enumeration has been slightly adapted to stay more compatible with the MATLAB enumeration of arrays and matrix structures

should return a restricted residual $r^c \in \mathbb{R}^{\frac{N_x}{2}+1 \times \frac{N_y}{2}+1}$ which arises from

$$\begin{aligned} r^c_{ij} := & \frac{1}{4} r_{2(i-1)+1,2(j-1)+1} \\ & + \frac{1}{8}(r_{2(i-1)+1,2(j-1)} + r_{2(i-1),2(j-1)+1} + r_{2(i-1)+2,2(j-1)+1} + r_{2(i-1)+1,2(j-1)+2}) \\ & + \frac{1}{16}(r_{2(i-1),2(j-1)} + r_{2(i-1)+2,2(j-1)} + r_{2(i-1),2(j-1)+2} + r_{2(i-1)+2,2(j-1)+2}) \end{aligned} \quad (2)$$

for all inner points. The value r^c_{ij} can be considered to be zero for all points on the boundary strip.

(b) Implement a matlab function `restrict_injection(r)`. The function should return a restricted residual $r^c \in \mathbb{R}^{\frac{N_x}{2}+1 \times \frac{N_y}{2}+1}$. For the injection method, see the lecture slides. The value r^c_{ij} can be considered to be zero for all points on the boundary strip.

(c) Implement a matlab function `interpolate(e)` which prolongates the error $e \in \mathbb{R}^{\frac{N_x}{2}+1 \times \frac{N_y}{2}+1}$ to the fine grid. The interpolated error $e^f \in \mathbb{R}^{N_x+1 \times N_y+1}$ is zero on the boundary strip and defined on the inner points via bilinear interpolation as

$$e^f_{ij} := \begin{cases} \frac{1}{4}(e_{i/2,j/2} + e_{i/2+1,j/2} + e_{i/2,j/2+1} + e_{i/2+1,j/2+1}) & \text{if (i,j) is centered between 4 coarse grid points} \\ \frac{1}{2}(e_{(i+1)/2,j/2} + e_{(i+1)/2,j/2+1}) & \text{if (i,j) is centered between 2 y-aligned coarse grid points} \\ \frac{1}{2}(e_{i/2,(j+1)/2} + e_{i/2+1,(j+1)/2}) & \text{if (i,j) is centered between 2 x-aligned coarse grid points} \\ e_{(i+1)/2,(j+1)/2} & \text{if (i,j) coincides with a coarse grid point.} \end{cases} \quad (3)$$

(d) Put the algorithmic components together in a function

`vCycle(a,b,u,rhs,preSmoothing,postSmoothing,level,mgID).`

Besides the coefficients a, b, the approximate solution u and the right hand side *rhs*, the parameters *preSmoothing* and *postSmoothing* determine the number of pre- and post-smoothing Gauss-Seidel iterations. The parameter *level* corresponds to $N_x = N_y = 2^{level}$. If $level = 1$, we only have one inner grid point (and 3×3 points in total). In this case, the function `vCycle(...)` should carry out one Gauss-Seidel iteration on the current data u and return the new solution. For $level > 1$, the function should carry out the *recursive* v-cycle algorithm, see slide 8 of the multigrid lecture slides. The additional parameter $mgID$ can be used to define which restriction-interpolation combination should be used. You may introduce an additional alternative which abstains from the v-cycle and only carries out one Gauss-Seidel iteration (e.g. for testing purposes).

(e) Use your implementations to solve the 2D Poisson problem. Initialise the solution u according to the function $u(x, y) = \sin(\pi x)\sin(\pi y)$ on the unit square. Use two pre- and two post-smoothing steps and iterate until the maximum norm of the residual has

reached an accuracy of 10^{-5}. Run the simulation for different grid levels $l \in \{2, 3, 4, 5, 6\}$ and both full-weighting restriction or injection. How many iteration steps do you need for each grid resolution/ solver scheme? How many iteration steps do you need if you use pure Gauss-Seidel iterations to solve the Poisson problem? Create error plots (iteration steps vs. maximum norm of residual) for both multigrid schemes and the Gauss-Seidel scheme for grid level $l = 5$, assuming a tolerance $TOL = 1e - 12$.

$l \rightarrow N_x = N_y = 2^l$	2	3	4	5	6
full-weighting	4	6	7	7	7
injection	3	5	5	5	5
Gauss-Seidel	22	93	375	1503	6015

Table 1: V-cycle iterations required to converge ($TOL < 1e-5$) for different grid sizes, restriction methods, and the pure Gauss-Seidel scheme in comparison.

Solution:

(a) See restrict_fullweighting.m.

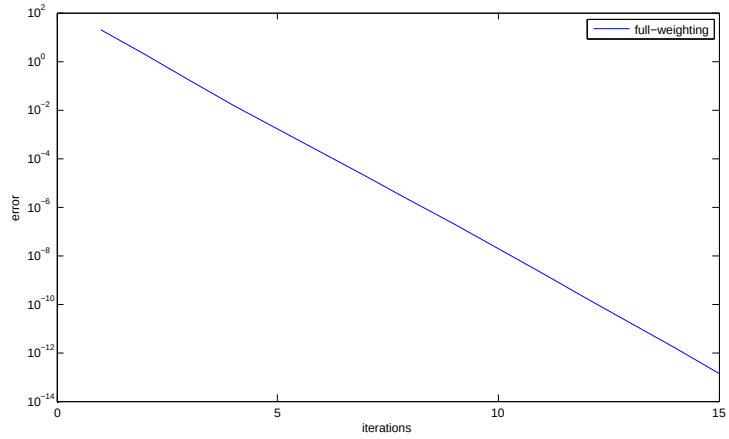
(b) See restrict_injection.m.

(c) See interpolate.m.

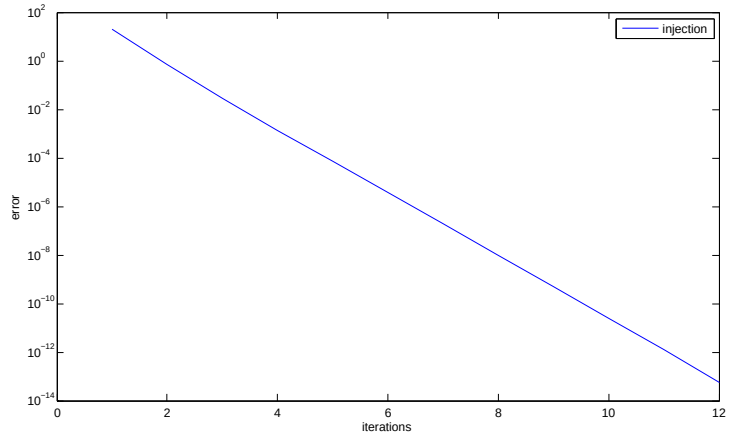
(d) See vCycle.m.

(e) The number of iteration steps for the different restriction methods, the pure Gauss-Seidel algorithm and different grid levels are shown in Tab. 1. For the solving of the 2D problem, see exercise3.m. From the table, we see that the number of v-cycle iterations stays constant for increasing numbers of grid levels. In contrast, the number of Gauss-Seidel iterations increases by a factor of four when halving the mesh size. We can thus observe that the convergence of the multigrid scheme is *independent from the grid resolution* whereas the pure Gauss-Seidel scheme is not. The convergence plots are shown in Fig. 1. Due to the constant decrease rate of the error per iteration, a linear curve needs to evolve in the semi-log graph. From the slope of the curve, we can further deduce the convergence speed of the respective method and compare it to the theoretical predictions. For example, consider the Gauss-Seidel plot. Since we start with a very low frequency as initial guess, we expect a very low convergence rate. The residual drops from ~ 20 to 10^{-12} in 3172 iterations. We thus obtain a convergence rate

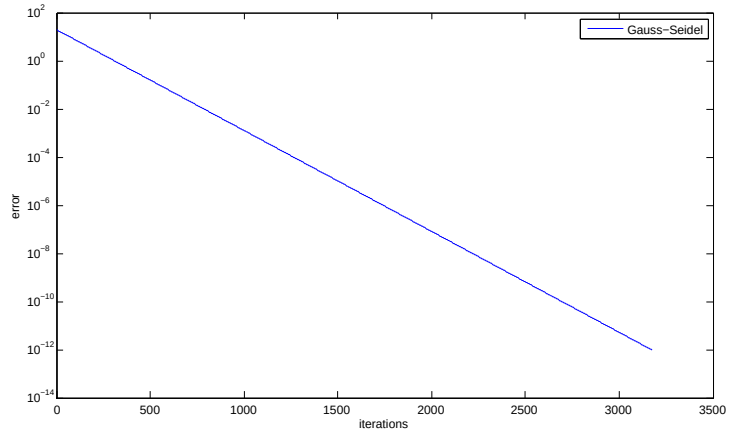
$$\gamma^{3172} = 20 \cdot 10^{-12} \Leftrightarrow \gamma \approx 0.99. \tag{4}$$



(a)



(b)



(c)

Figure 1: Decrease of the maximum norm of the residual over several iterations of different linear solver schemes. (a) Multigrid using full-weighting for restriction and bilinear interpolation. (b) Multigrid using injection and bilinear interpolation. (c) Gauss-Seidel solver.

Exercise 2: Multigrid for Anisotropic Poisson Equation

So far, we considered the coefficients $a = b = 1$. In the following, we want to consider the case where $a \gg b$ and study the respective behaviour of the multigrid scheme. For this purpose, we set $a = 1$, $b = 1e-4$ and obtain an anisotropic Poisson-like equation.

(a) Use your matlab code (using the settings $TOL = 1e-5$, pre-/post-smoothing=2 steps etc.) to solve the anisotropic Poisson problem. How many iteration steps do you need now for the different v-cycle schemes and the pure Gauss-Seidel method?

(b) Implement new functions `restrict_semicoarsening(r)` and `interpolate_semi(e)` which allow for a semi-coarsening and semi-prolongation of the residual along x-direction:

- `restrict_semicoarsening(r)` restricts a residual $r \in \mathbb{R}^{N_x+1 \times N_y+1}$ only along x-direction and returns the restricted residual $r^c \in \mathbb{R}^{\frac{N_x}{2}+1 \times N_y+1}$. The restriction rule reads

$$r^c_{ij} := \frac{1}{4}(r_{2(i-1)+1,j} + r_{2(i-1),j}) + \frac{1}{2} r_{2(i-1)+1,j}. \tag{5}$$

- `interpolate_semi(e)` prolongates an error $e \in \mathbb{R}^{\frac{N_x}{2}+1 \times N_y+1}$ to a finer grid, resulting in a prolongated error $e^f \in \mathbb{R}^{N_x+1 \times N_y+1}$. The interpolation rules read

$$e^f_{ij} := \begin{cases} \frac{1}{2}(e_{i/2,j} + e_{i/2+1,j}) & \text{if (i,j) is centered between 2 x-aligned coarse grid points} \\ e_{(i+1)/2,j} & \text{if (i,j) coincides with a coarse grid point.} \end{cases} \tag{6}$$

Incorporate the new coarsening and interpolation schemes into the function `vCycle(...)`. The coarsest grid contains more than one inner grid point in case of semi-coarsening; what do you have to change in your current v-cycle implementation so that you obtain convergence? Measure the number of iteration steps for different grid levels. What do you observe?

$l \rightarrow N_x = N_y = 2^l$	2	3	4	5	6
full-weighting	5	17	59	195	585
injection	5	18	64	212	644
semi-coarsening	4	5	6	6	6
Gauss-Seidel	21	89	357	1431	5727

Table 2: V-cycle iterations required to converge ($TOL < 1e-5$) for different grid sizes, restriction methods, and the pure Gauss-Seidel scheme in comparison. The considered problem is an anisotropic Poisson equation with coefficients $a = 1$, $b = 1e-4$.

Solution:

(a) The results are shown in Tab. 2 for $a = 1$, $b = 1e-4$. We can see that the v-cycle scheme cannot solve the problem with a constant number of iterations for any grid level anymore. Instead, the number of iterations increases by a factor 3-4 when halving the mesh size. This is similar to the slow convergence of the Gauss-Seidel scheme.

(b) See restrict_semicoarsening.m and interpolate_semi.m. Since there are more inner points on the coarsest grid level, a single Gauss-Seidel sweep over the data is not sufficient anymore. For the considered test cases, setting the number of Gauss-Seidel iterations to 10000 is (more than) enough to obtain a valid coarse grid solution. The results for the semi-coarsening approach are also listed in Tab. 2. A constant number of iteration steps for the v-cycle is recovered.