Technische Universität München
Lehrstuhl für Informatik V
Univ.-Prof. Dr. M. Bader
M.Sc.(hons.) Kaveh Rahnema

SS 2014
Sheet 4
19.05.2014

Scientific Computing II

Multigrid Methods

Exercise 5: Multigrid for Convection-Diffusion

Consider the one-dimensional convection-diffusion equation

$$-\epsilon u_{xx} + u_x = f(x) \tag{1}$$

and its discrete representation

$$\frac{\epsilon}{h^2}(-u_{n-1} + 2u_n - u_{n+1}) + \frac{1}{h}(u_n - u_{n-1}) = f_n, \quad n = 1, ..., N-1 \tag{2}$$

with Dirichlet conditions $u_0 = u_N = 0$ and $\epsilon \geq 0$. In this exercise, we assume a vanishing right hand side $f(x) = 0$. The solution of the differential equation is thus given by $u(x) = 0$.

(a) Show that for $\epsilon << 1$, the Jacobi method applied to Eq. (2) converges very slowly whereas the Gauss-Seidel method is expected to converge very fast. The Gauss-Seidel method is used from left to right, that is the updated values $u_1, ..., u_{n-1}$ are immediately used to compute the new value at u_n.
Hint: insert an error frequency $u_n = \sin(\pi k(nh))$ into the respective iterative scheme and analyse the behaviour for $\epsilon \to 0$.

(b) In the following programming exercise 4, we want to use a matrix-dependent restriction/ interpolation technique to set up a multigrid for solving Eq. (2), cf. slide 29 of the lecture slides (multigrid.pdf). Show that the Galerkin coarsening $A_{2h} := R_h^{2h} A_h P_{2h}^h$ applied to an arbitrary three-point stencil $[s_l \quad s_c \quad s_r]$ with $s_c = -(s_l + s_r)$ on the fine grid yields the coarse grid stencil

$$\frac{1}{s_c}[-s_l^2 \quad (s_l^2 + s_r^2) \quad -s_r^2]. \tag{3}$$

Solution:

(a) First, we formulate the respective iterative update rules:

$$\begin{array}{lrcl} \text{Jacobi:} & u_n^{new} & = & \frac{\epsilon}{2\epsilon+h} u_{n+1}^{old} + \frac{\epsilon+h}{2\epsilon+h} u_{n-1}^{old} \\ \text{Gauss-Seidel:} & u_n^{new} & = & \frac{\epsilon}{2\epsilon+h} u_{n+1}^{old} + \frac{\epsilon+h}{2\epsilon+h} u_{n-1}^{new} \end{array} \tag{4}$$

For simplified writing, we introduce $a := \frac{\epsilon}{2\epsilon+h}$ and $b := \frac{\epsilon+h}{2\epsilon+h}$. For these two coefficients, it holds

$$\begin{array}{lcll} a & & \overset{\epsilon\to 0}{\to} & 0 \\ b & & \overset{\epsilon\to 0}{\to} & 1 \\ a+b & = \quad 1 & \overset{\epsilon\to 0}{\to} & 1 \\ a-b & = \quad -\frac{h}{2\epsilon+h} & \overset{\epsilon\to 0}{\to} & -1. \end{array} \tag{5}$$

Let's start with the Jacobi method. Inserting $u_n^{old} := \sin(\pi k(nh))$ and using the theorem $\sin(\alpha+\beta) = \sin(\alpha)\cos(\beta) + \sin(\beta)\cos(\alpha)$ yields:

$$\begin{array}{rcl} u_n^{new} & = & a u_{n+1}^{old} + b u_{n-1}^{old} \\ & = & \ldots \\ & = & (a+b)\cos(\pi kh)\sin(\pi k(nh)) + (a-b)\sin(\pi kh)\cos(\pi k(nh)) \\ & \overset{\epsilon\to 0}{=} & \cos(\pi kh)\sin(\pi k(nh)) - \sin(\pi kh)\cos(\pi k(nh)) \end{array} \tag{6}$$

We observe that a frequency $\cos(\pi k(nh))$ is introduced. Due to this shift of the sine wave by $\pi/2$, we obtain the following damping factors $\cos(\pi kh)$, $\sin(\pi kh)$ for small (but fixed) mesh sizes $h = 1/N$:

$$\begin{array}{lll} & \cos(\pi kh) & \sin(\pi kh) \\ k = 1 & \approx 1 & \approx 0 \\ k = N/2 & \approx 0 & \approx 1 \\ k = N-1 & \approx -1 & \approx 0 \end{array} \tag{7}$$

Thus, each time when a sine-wave is damped efficiently, the cosine is not damped and vice versa. We thus expect a slow convergence of the Jacobi method in this case.
Next, we consider the Gauss-Seidel method. Since we need to take the updated values immediately into account during the current iteration, we consider the updates of u_1, u_2,...:

$$\begin{array}{rcl} u_1^{new} = \ldots = a u_2^{old} \\ u_2^{new} = \ldots = a(u_3^{old} + b u_2^{old}) \\ \vdots \\ u_n^{new} = \ldots = a \sum_{i=2}^{n+1} b^{n+1-i} u_i^{old} \end{array} \tag{8}$$

For a very small parameter ϵ and a fixed number of grid points, we see that the Gauss-

Seidel method will decrease the error in all components u_n:

$$\begin{aligned}
|u_n^{new}| &= \left| a \sum_{i=2}^{n+1} b^{n+1-i} u_i^{old} \right| \\
&\overset{\text{triangle inequ.}}{\leq} a \sum_{i=2}^{n+1} b^{n+1-i} \left| u_i^{old} \right| \\
&\overset{|u_i^{old}| = |\sin(\pi k (ih))| \leq 1}{\leq} a \sum_{i=2}^{n+1} b^{n+1-i} \\
&= a \sum_{i=0}^{n-1} b^i \\
&= \underbrace{a}_{\overset{\epsilon \to 0}{\longrightarrow} 0} \underbrace{\frac{b^n - 1}{b-1}}_{\overset{\epsilon \to 0}{\longrightarrow} n} \longrightarrow 0
\end{aligned} \tag{9}$$

The fraction $\frac{b^n-1}{b-1}$ tends to n in the latter equation due to l'Hôpital's rule:

$$\lim_{\epsilon \to 0} \frac{b^n - 1}{b-1} = \lim_{\epsilon \to 0} \frac{\left(\frac{\epsilon+h}{2\epsilon+h}\right)^n - 1}{\frac{\epsilon+h}{2\epsilon+h} - 1} = \lim_{\epsilon \to 0} n \left(\frac{\epsilon+h}{2\epsilon+h}\right)^{n-1} = n \tag{10}$$

Due to the dependency in n, we also see that the error is reduced less in those components u_n which are located closer to the right side of the boundary.

(b) The operators A_h, R_h^{2h} and P_{2h}^h are given by

$$A_h := \begin{pmatrix} s_c & s_r & 0 & 0 & 0 & 0 & 0 \\ s_l & s_c & s_r & 0 & 0 & 0 & 0 \\ 0 & s_l & s_c & s_r & 0 & 0 & 0 \\ 0 & 0 & s_l & s_c & s_r & 0 & 0 \\ 0 & 0 & 0 & s_l & s_c & s_r & 0 \\ 0 & 0 & 0 & 0 & s_l & s_c & s_r \\ 0 & 0 & 0 & 0 & 0 & s_l & s_c \end{pmatrix}, P_{2h}^h := \begin{pmatrix} -\frac{s_l}{s_c} & 0 & 0 \\ 1 & 0 & 0 \\ -\frac{s_r}{s_c} & -\frac{s_l}{s_c} & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{s_r}{s_c} & -\frac{s_l}{s_c} \\ 0 & 0 & 1 \\ 0 & 0 & -\frac{s_r}{s_c} \end{pmatrix}, R_h^{2h} = P_{2h}^{h^\top}, \tag{11}$$

cf. lecture slides (for the restriction and prolongation operator). Evaluating $A_{2h} := R_h^{2h} A_h P_{2h}^h$ yields:

$$A_{2h} := \begin{pmatrix} s_c - \frac{2 s_l s_r}{s_c} & -\frac{s_r^2}{s_c} & 0 \\ -\frac{s_l^2}{s_c} & s_c - \frac{2 s_l s_r}{s_c} & -\frac{s_r^2}{s_c} \\ 0 & -\frac{s_l^2}{s_c} & s_c - \frac{2 s_l s_r}{s_c} \end{pmatrix}. \tag{12}$$

It is enough to consider the second row of A_{2h}. The left and the right entry are already in agreement with what needs to be shown. For the entry in the center, we can insert the requirement $s_c = -(s_l + s_r)$ and obtain:

$$s_c - \frac{2 s_l s_r}{s_c} = \frac{s_c^2 - 2 s_l s_r}{s_c} = \frac{(s_l + s_r)^2 - 2 s_l s_r}{s_c} = \frac{s_l^2 + s_r^2}{s_c}. \tag{13}$$

Programming Exercise 4: Multigrid for Convection-Diffusion

Consider the one-dimensional convection-diffusion equation from Eq. (1) and its discrete counterpart Eq. (2). In this exercise, we assume a right hand side $f(x) = 2x - (2\epsilon + 1)$ and a respective discrete representation $f_n = 2(nh) - (2\epsilon + 1)$. The number of grid points is given by $N := 2^l + 1, l \in \mathbb{N}$, and the mesh size is $h := 1/N$, respectively. In the following, we want to solve the discrete system from Eq. (2) by a multigrid method implemented in Matlab. A Gauss-Seidel smoother is provided on the webpage ($\rightarrow$ smooth.m).

(a) Show that the analytical solution to the problem from Eq. (1) is given by $u(x) = x^2 - x$.

(b) Implement a function `restrict(stencil,residual)` which constructs a matrix-dependent restriction (see slide 29 of multigrid.pdf from the lecture). The vector `stencil`$= [s_l \ s_c \ s_r] \in \mathbb{R}^3$ corresponds to the three-point stencil on the fine grid, `residual` to the residual vector of the current grid level. The function should return the coarsened residual vector.

(c) Implement a function `interpolate(stencil,eCoarse)` which interpolates the error `eCoarse` from the coarse grid to the fine grid. The interpolation should be carried out based on the matrix-dependent interpolation rule as presented in the multigrid slides (cf. slide 29). The stencil `stencil` should correspond to the three-point stencil on the fine grid. The function should return the interpolated error vector.

(d) Implement a function `computeCoarseGridStencil(stencil)` which computes and returns the coarse grid stencil for a given fine grid stencil `stencil`. The function should again be based on the matrix-dependent restriction/ prolongation from the multigrid slides.

(e) Put the steps together into a function `wCycle(u,rhs,level,stencil)` which implements the w-cycle algorithm. The vector `u` corresponds to the current solution and should also be returned by this function. The right hand side of the linear problem is given by `rhs`, `level` corresponds to the current grid level and `stencil` denotes the three-point stencil for the current grid level.
Test your implementation by

- using a two-grid algorithm (e.g. remove the recursive call in wCycle(...) and use a sufficient number of smoothing steps on the coarse grid)
- comparing your results with results obtained from pure Gauss-Seidel solving
- comparing your results with the exact solution.

(f) Use the w-cycle implementation to solve the discrete system from Eq. (2) for $\epsilon = 1, 0.1, 0.01$. The simulation should stop when the maximum norm of the residual drops below a tolerance $tol = 1e - 10$. Measure the number of w-cycle iterations for each simulation setup and initial grid levels $l = 4, 6, 8, 10$. Besides, plot the numerical solution as well as the error $e = u - u^{analytic}$ where $u^{analytic}$ denotes the discrete representation of $u(x) = x^2 - x$. What do you observe?

	$l = 4$	$l = 6$	$l = 8$	$l = 10$
$\epsilon = 1$	7	7	7	7
$\epsilon = 0.1$	6	7	7	7
$\epsilon = 0.01$	5	7	7	7

Table 1: Number of iterations of the w-cycle algorithm for different grid resolutions and diffusion levels ϵ.

Solution:

(a) Differentiating the function $u(x) = x^2 - x$ and inserting the results for the first and second derivative into the convection-diffusion equation from Eq. (1) shows the respective statement.

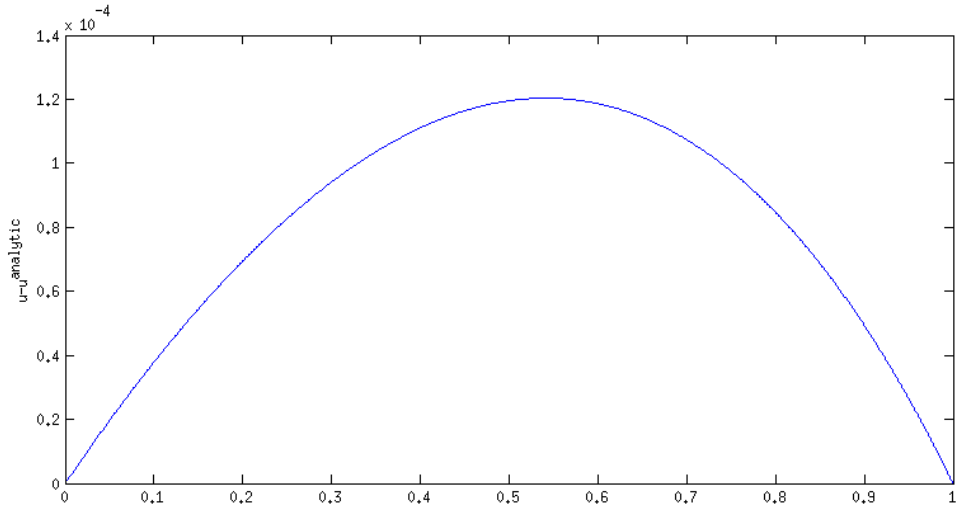
(b) See restrict.m.

(c) See interpolate.m.

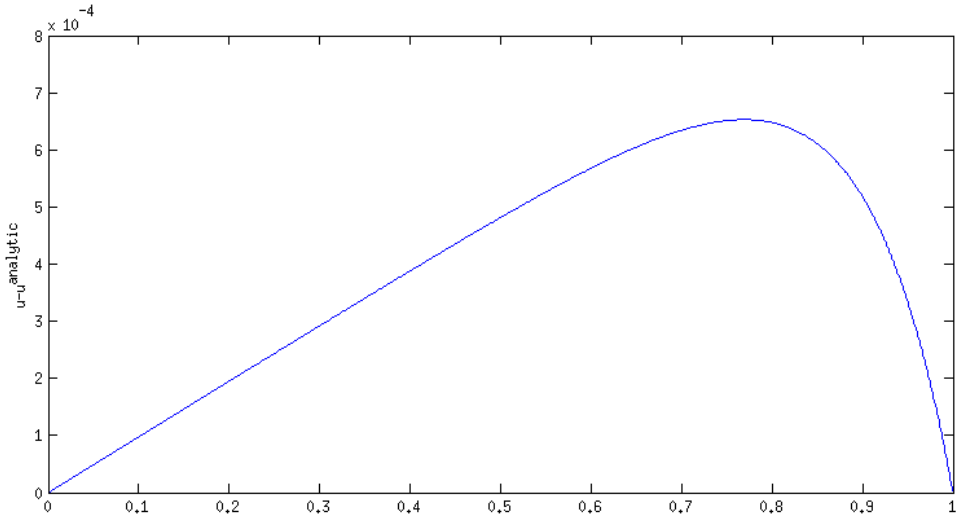
(d) See computeCoarseGridStencil.m.

(e) See wCycle.m.

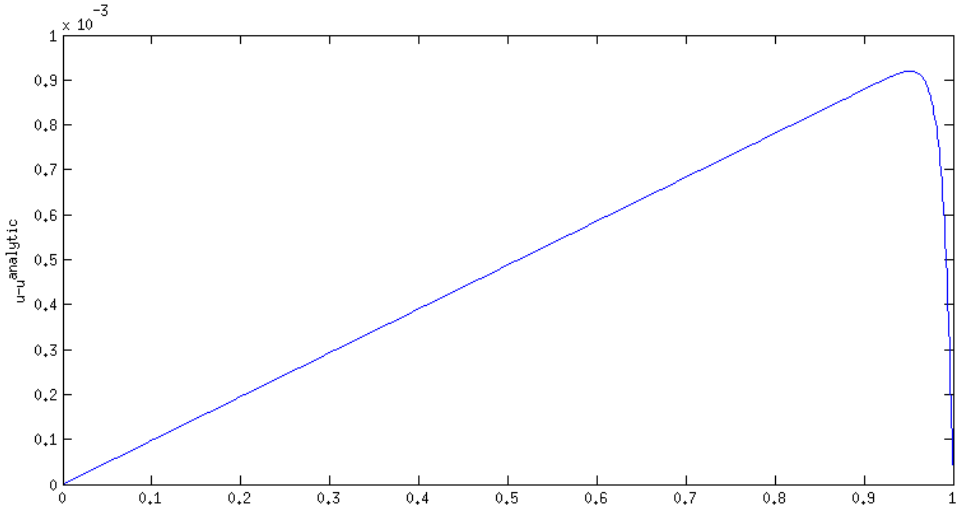
(f) See conDiff.m. The results for different values of ϵ and grid levels l are shown in Tab. **??** ($tol = 1e - 10$). The number of iterations stays—similar to the multigrid scheme for the Poisson equation from the previous worksheet—approx. constant, even for higher grid resolutions.
Exemplary error plots $u - u^{analytic}$ are shown in Fig. **??**. The error increases towards the middle of the domain, but it is shifted to the right for increasing values of ϵ, that is for decreasing diffusion and increasing convection. Since we do not have any error on the Dirichlet boundaries, the error must be bigger in the inner part of the domain. Besides, due to the convection, the error is "transported" and increased from left to right.



(a)



(b)



(c)

Figure 1: Plot of error $u - u^{analytic}$ over space. (a) $\epsilon = 1, l = 10$. (b) $\epsilon = 0.1, l = 10$. (c) $\epsilon = 0.01, l = 10$.