**Exercises for the lecture**

# Fundamentals of Simulation Methods

**WS 2016/17**

Lecturers: Frauke Gräter and Rüdiger Pakmor

**Exercise sheet 6** ***(due date: Dec 1, 2016, 11:59pm)***

**Multigrid solver for linear systems**

## 1) Galerkin coarse grid approximation (15 points)

Consider the one-dimensional problem

$$\frac{\partial^2 \Phi}{\partial x^2} = 4\pi G \rho(x), \tag{1}$$

which we want to solve using a multigrid accelerated iterative method on a (periodic) grid of size $N = 8$ with spacing $h$. The problem can be rephrased in the form

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{2}$$

where $\boldsymbol{x} = \Phi$ and $\boldsymbol{b} = 4\pi G\rho h^2$ are vectors of size $N$ and $\mathbf{A}$ is an $N \times N$ matrix. Given the operator

$$\mathbf{A^{(h)}} = \begin{pmatrix} -2 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 \\ & & & \dots & & & & \\ 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & -2 \end{pmatrix} \tag{3}$$

on the finest grid level ($N = 8$), find the operator $\mathbf{A^{(2h)}}$ on the next coarser grid level ($N = 4$, spacing $2h$). In order to do so, carry out the following steps:

a) Find the prolongation and restriction operators

$$\mathbf{I}^{\mathbf{(h)}}_{\mathbf{(2h)}} : x^{(h)}_{2i} = x^{(2h)}_i, x^{(h)}_{2i+1} = \frac{1}{2}\left(x^{(2h)}_i + x^{(2h)}_{i+1}\right) \tag{4}$$

$$\mathbf{I}^{\mathbf{(2h)}}_{\mathbf{(h)}} : x^{(2h)}_i = \frac{1}{4}\left(x^{(h)}_{2i-1} + 2x^{(h)}_{2i} + x^{(h)}_{2i+1}\right) \tag{5}$$

in matrix form. *Hint:* $\mathbf{I}^{\mathbf{(h)}}_{\mathbf{(2h)}}\mathbf{x^{(h)}} = \mathbf{x^{(2h)}}$ and vice versa.

b) Now calculate the operator on the coarse grid using the Galerkin coarse grid approximation: $\mathbf{A^{(2h)}} = \mathbf{I}^{\mathbf{(2h)}}_{\mathbf{(h)}}\mathbf{A^{(h)}}\mathbf{I}^{\mathbf{(h)}}_{\mathbf{(2h)}}$.

c) Compare the result to an operator obtained by direct discretization on the coarse grid.

## 2) Iteratively solving Poisson's equation (35 points)

(a) Suppose we want to solve for the gravitational potential $\Phi$ on a *periodic* mesh on a square-shaped domain $[0, L]^2$ (with $L = 1.0$ for definiteness) with $N \times N$ cells, for the mass density distribution

$$\rho(\mathbf{r}) = \rho_0 \, \mathrm{e}^{-\frac{\mathbf{r}^2}{2\eta^2}}, \tag{6}$$

where $\eta = 0.1L$ is the spread of the mass spike, and $\rho_0 = 10$. Discretization of the Poisson equation gives an iteration rule where the left-hand side gives the updated values in terms of the old values on the right-hand side, in the form

$$x_{i,j}^{(n+1)} = \frac{1}{4}(x_{i-1,j}^{(n)} + x_{i+1,j}^{(n)} + x_{i,j-1}^{(n)} + x_{i,j+1}^{(n)} - b_{i,j}). \tag{7}$$

where $x_{i,j} = \Phi_{i,j}$ and $b_{i,j} = 4\pi G h^2 \rho_{i,j} - \bar{b}$. This can be readily used for Jacobi or Gauss-Seidel iteration schemes. Write a function

```
jacobi_step(x[ ], b[ ], N)
```

that replaces the input array `x[ ]` with a correspondingly updated array after one iteration step.

(b) To characterize the current error in the solution, write a function that calculates the residual

$$r_{i,j} = b_{i,j} - (\mathbf{A}\mathbf{x})_{i,j}. \tag{8}$$

Also, write a function that computes the norm

$$S = \left[ \sum_{j,k} r_{j,k}^2 \right]^{1/2} \tag{9}$$

of this vector and returns it. We can use $S$ as a simple measure of the overall error.

(c) Now specialize to the case of $N = 256$ and write a routine that suitably initializes the density field, with the density peak placed in the center of the box. Set the initial guess for the potential to $\mathbf{x} = 0$. Carefully think about how to initialize the array $\mathbf{b}$ (for a periodic system, equation 2 is only solvable for $\langle \mathbf{b} \rangle = 0$ because the matrix $\mathbf{A}$ is singular, therefore $\bar{b}$ has to be chosen appropriately). Now add a loop that calls the Jacobi iteration $N_{\mathrm{steps}} = 2000$ times, and after each step, determines the norm of the residual. You may use the C- or Python template on the Moodle site and fill in the missing parts in order to reduce the coding work, or write everything from scratch if you prefer. Make a plot of the decay of the log of this residual as a function of step number. What do you expect to get for $\sum_{i,j} \Phi$?

(d) Now produce a second version of your program in which the Jacobi iteration is replaced by Gauss-Seidel iteration where new values for elements of $\mathbf{x}$ are used as soon as they become available, replacing any old value in the array.

(e) Change the Gauss-Seidel scheme to a red-black Gauss-Seidel iteration, i.e. you first update the red cells in a chess-board pattern overlaid over the mesh, then the black cells in a second pass. Produce a common plot of the decay rates of the residual for $N_{\text{steps}}$ steps with the three iteration variants considered thus far. Interpret the results. The red-black scheme result will perhaps disappoint you at first – what's going on?

(f) We now solve the problem with multigrid acceleration. First, write a function that carries out a restriction step of a 2D mesh with dimension $N \times N$ (where $N$ is a power of 2) onto a coarser mesh with dimension $N/2 \times N/2$. Make each point in the coarser mesh a weighted average of neighboring points, according to the stencil

$$\begin{bmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \end{bmatrix}. \tag{10}$$

Give this function a calling signature

```
do_restrict(N, fine[ ], NN, coarse[ ])
```

(only `NN = N/2` is allowed when calling this function).

(g) Next, write a function that carries out a prolongation step of a 2D mesh of dimension $N \times N$ onto a finer mesh of dimension $2N \times 2N$. Each point in the coarser mesh is additively injected into the finer mesh with weights

$$\begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{bmatrix}. \tag{11}$$

Note that points of the fine mesh that see a coarse point with weight 1/4 will get contributions from 4 coarse points in total, similarly for the points with weight 1/2, while the fine-mesh point with weight 1 simply inherits the value of one of the coarse mesh points. Give this function a calling signature

```
do_prolong(NN, coarse[ ], N, fine[ ])
```

(again, only `NN = N/2` is allowed when calling this function).

(h) Now write a function

```
do_v_cycle(x[ ], b[ ], N)
```

that carries out a V-cycle multigrid iteration on the current solution vector. The steps of the function should be

1. Do a Gauss-Seidel step.
2. Calculate the residuum.
3. Restrict the residuum to a coarser mesh $N' = N/2$, and scale it by a factor of 4 to effectively take care of the $1/h \to 1/(2h)$ scaling of the differential operator (whose explicit form we leave invariant, for simplicity).

4. Call `do_v_cycle` recursively for the coarser mesh, with a zero error vector as starting guess for `x`, and the coarsened residual for `b`.
5. Now prolong the returned error vector to the fine mesh $N$.
6. Add this to the current solution vector on the fine mesh.
7. Do another Gauss-Seidel step.

Steps 2-6 are only done provided $N > 4$. To simplify the coding, you may use one of the code templates provided on the Moodle site.

(i) Finally, solve the original problem by repeatedly calling your V-cycle iteration. Again, plot the residual as a function of the number of steps taken, and compare with the results obtained earlier for plain Jacobi and Gauss-Seidel iteration.