Technische Universität München
Lehrstuhl für Informatik V
Dr. T. Neckel, Dr. D. Pflüger

SS 2011
Sheet 9
18.7.2011

Algorithms of Scientific Computing (Algorithmen des Wissenschaftlichen Rechnens)

Multigrid

This week we want to solve Laplace's Equation (1) on the one-dimensional unit interval $\Omega = (0; 1)$ using Jacobi and multigrid methods.

$$-u''(x) = 0\,, \qquad x \in \Omega \tag{1}$$
$$u(x) = 0\,, \qquad x \in \partial\Omega \tag{2}$$

The trivial solution obviously is the constant function

$$u(x) = 0\,, \qquad x \in \partial\Omega \cup \Omega.$$

We will use this knowledge to draw conclusions concerning the decay of the error.

Hint: The source code used in the tutorials is in the Python source file *worksheet9.py*.

Exercise 1: The Jacobi Method

Let $A\mathbf{x} = \mathbf{b}$ a system of linear equations with n-dimensional vectors $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$ and matrix $A \in \mathbb{R}^{n\times n}$. In each iteration of the Jacobi Method the new approximation $\mathbf{x}^{(n+1)}$ to the solution $\mathbf{x}$ is computed according to the rule

$$x_i^{(n+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j\neq i} a_{ij} x_j^{(n)} \right), \qquad i \in \{0, 1, \ldots, n-1\}$$

The vector $\mathbf{r} = \mathbf{b} - A\mathbf{x}$ is called the *residual*, and it usually serves as indicator for the quality of the approximation.

(i) Fill in the body of the function *residual* computing the residual for a given vector $\mathbf{x}^{(n)}$.

(ii) Fill in the missing parts in the function *jacobi* applying one Jacobi smoothing step to the current approximation $\mathbf{x}^{(n)}$.

(iii) Use your implementation of the Jacobi smoother to compute and plot the solution to equation (1). Try different initial guesses $\mathbf{x}^{(n)}$ from the set of eigenvectors of the matrix A. They can be computed via the function *eigenvector*.

What do you notice when comparing the low and high frequency error components?

Exercise 2: The Multigrid Method

Now we use the Multigrid Method to solve $A\mathbf{x} = \mathbf{b}$. In our implementation the Jacobi Method from above will serve us as smoother.

(i) Fill in the bodies of functions *prolongation* and *restriction* refining resp. coarsening an approximated solution vector when moving from one refinement level to another.

(ii) Fill in the missing parts in the recursive function *multigrid*.

(iii) Determine a set of parameters for *multigrid* such that a call to this function triggers a classical Jacobi smoothing step.

(iv) Run further experiments for different initial guesses. Try the set of eigenvectors of A, linear combinations of these, as well as vectors with random components.

Compare the results to those of the Jacobi Method.