Technische Universität München
Lehrstuhl für Informatik V
Univ.-Prof. Dr. M. Bader
Dipl.-Tech. Math. P. Neumann
Dipl.-Inf. W. Eckhardt

SS 2013
Sheet 6
10.06.2013

Scientific Computing II

Krylov Methods continued

Programming Exercise 5: Steepest Descent

We have to solve the discretised three-dimensional Poisson equation on a unit cube with homogeneous Dirichlet boundary conditions (cf. sheet 1):

$$\begin{aligned} \Delta u &= f \text{ in }]0;1[^3, \\ u &= 0 \text{ at } \partial]0;1[^3. \end{aligned}$$

The Laplacian is discretised in a regular cartesian grid by the 7-point-stencil

$$\Delta u(i \cdot h, j \cdot h, k \cdot h) \approx \frac{u_{i-1,j,k} + u_{i,j-1,k} + u_{i,j,k-1} - 6u_{i,j,k} + u_{i+1,j,k} + u_{i,j+1,k} + u_{i,j,k+1}}{h^2}.$$

(a) Implement a matlab function performing one steepest descent iteration for the given system in dependence on the grid resolution per coordinate direction N, the right hand side b, and the current solution approximation u.

Hint: Use the given function `residual_vec` to compute the current residual vector.

(b) If you solve the given system (using the given `main`-function) with the steepest descent method for different grid resolutions N you will notice that you are always done with one iteration. Thus, for this example, the first steepest descent direction directly leads to the solution. To see that this doesn't hold in general, change the right-hand side to one (constant function) and try to solve the system again. Record the resulting runtimes and numbers of iterations in the following tabular:

N	runtime	# iterations
7		
15		
31		

(c) Can you give a relation between N and the number of iterations required with the steepest descent in the form $O(N^p)$?

(d) Give the overall costs of the steepest descent method for the given system in the form $O(N^q)$. Do you know solvers that have the same complexity?

Solution:

(a) `MATLAB/SOLUTION/steepest_descent.m`

(b)

N	runtime	# iterations
7	0.2830 sec	135
15	11.0191 sec	576
31	6 min 28.2837 sec	2358

(c) For each doubling of N, the number of iterations is multiplied by a factor of four, thus the number of iterations is $O\left(N^2\right)$.

(d)
- costs per iteration (from sheet 5, Exercise **6 e)**):
$$O\left(N^3\right).$$
- number of iterations:
$$O\left(N^2\right).$$
- total costs:
$$O\left(N^5\right).$$

Solvers with the same complexity: Jacobi, Gauss-Seidel (but with a lower constant before N^5).

Programming Exercise 6: Conjugate Gradients Method and Preconditioning

We have to solve the discretised three-dimensional Poisson equation on a unit square with homgeneous Dirichlet boundary conditions (see exercise 1) and constant right-hand side one. The Laplacian is discretised by the known 7-point-stencil.

(a) Recapitulate the formulation of the PCG method so that it can be used with an arbitrary symmetric positive definite matrix M as preconditioner!
How do you have to change it to be able to use a given Matlab-solver?

(b) Implement a matlab function performing the preconditioning in the context if the preconditioned conjugate gradient method for the given system in dependence on the grid resolution per coordinate direction N, the current residual and the type of preconditioner (0 = no preconditioning, 1 =Jacobi or 2 =Gauss-Seidel as shown in the lecture).

(c) Implement a matlab function performing one iteration of the preconditioned conjugate gradient method for the given system in dependence on the grid resolution per coordinate direction N and the current solution approximation u.

(d) Solve the given system with the help of the given `main`-program using the pcg method for different grid resolutions N without preconditioning and with both preconditioners. Record the resulting runtimes and numbers of iterations in the following tabular:

N	t (CG)	t (Jacobi)	t (Gauss-Seidel)	# it (CG)	# it (Jacobi)	# it (Gauss-Seidel)
7						
27						
39						
56						
127						

(e) Compare your results to the conjugate gradient method without preconditioning and the optimal convergence of a multigrid method.

Solution:

(a) The standard formulation of PCG reads

for it = 1, 2, ...

$$
\begin{aligned}
a &= \vec{r}^T M^{-1} \vec{r} \\
b &= \vec{p}^T A \vec{p} \\
\vec{u} &= \vec{u} + \tfrac{a}{b} \vec{p} \\
\vec{r} &= \vec{r} - \tfrac{a}{b} A \vec{p} \\
b &= \vec{r}^T M^{-1} \vec{r} \\
\vec{p} &= M^{-1} \vec{r} - \tfrac{b}{a} \vec{p}
\end{aligned}
$$

The inverse of the preconditioner M is applied three times. In order to use a given solver (and to avoid the repeated applition), we compute $M^{-1} \cdot \vec{r}$ only once we know the new residual and store the result in a vector $\vec{v}$:

for it = 1, 2, ...

$$
\begin{aligned}
a &= \vec{r}^T \vec{v} \\
b &= \vec{p}^T A \vec{p} \\
\vec{u} &= \vec{u} + \tfrac{a}{b}\vec{p} \\
\vec{r} &= \vec{r} - \tfrac{a}{b} A\vec{p} = \overrightarrow{rhs} - A\vec{u} \\
\vec{v} &= \text{solver\_iterations}(\vec{0}, A, \vec{r}) \\
b &= \vec{r}^T \vec{v} \\
\vec{p} &= \vec{v} - \tfrac{b}{a}\vec{p}
\end{aligned}
$$

(b) `MATLAB/SOLUTION/precond.m`

(c) `MATLAB/SOLUTION/pcg.m`

(d)

N	t (CG)	t (Jacobi)	t (Gauss-Seidel)	# it (CG)	# it (Jacobi)	# it (Gauss-Seidel)
7		0.16 sec	0.17 sec	13	13	14
15		0.35 sec	0.28 sec	27	27	23
31		1.45 sec	1.30 sec	56	56	39
63		23.3 sec	18.1 sec	116	116	70
127		376 sec	270 sec	237	237	132

(e)

- same number of iterations for the original conjugate gradient method and the Jacobi-preconditioned conjugate gradient method,
- better convergence for the Gauss-Seidel-preconditioned conjugate gradient method,
- no optimal convergence (= constant number of iterations) for both variants