Technische Universität Berlin
Institut für Mathematik
Prof. Dr. Jörg Liesen
Dr. Agnieszka Międlar
André Gaul

WS 11/12
To be submitted in the lecture on 25.01.2012

Numerische Mathematik für Ingenieure II
Homework 10

Programming exercise 13: (16 points)
Consider the following algorithm for the preconditioned CG method:

Require: $A \in \mathbb{R}^{N\times N}$, $L \in \mathbb{R}^{N\times N}$ with $\det(L) \neq 0$, $b, x_0 \in \mathbb{R}^N$, $maxit \in \mathbb{N}$, $tol \in \mathbb{R}_+$
$r_0 = b - Ax_0$, $z_0 = (LL^T)^{-1}r_0$, $p_0 = z_0$ ▷ initialization
for $j = 1, 2, \ldots, maxit$ **do**
  $\gamma_{j-1} = (r_{j-1}^T z_{j-1})/(p_{j-1}^T A p_{j-1})$
  $x_j = x_{j-1} + \gamma_{j-1} p_{j-1}$ ▷ update of approximation
  $r_j = r_{j-1} - \gamma_{j-1} A p_{j-1}$ ▷ update of residual
  if $\|r_j\|/\|r_0\| < tol$ **then** ▷ test for convergence
    return x_j
  end if
  $z_j = (LL^T)^{-1} r_j$
  $\beta_{j-1} = r_j^T z_j / r_{j-1}^T z_{j-1}$
  $p_j = z_j + \beta_{j-1} p_{j-1}$ ▷ update of search direction
end for

(a) Implement the preconditioned CG method in a

`function [xj, r2u] = p13pcg(A, b, x0, maxit, tol, L).`

You can modify your implementation of the CG method from Programming Exercise 12. The vector $z_j = (LL^{\mathsf{T}})^{-1} r_j$ should be computed in two steps: first solve the linear system $Ly = r_j$; then the linear system $L^{\mathsf{T}} z_j = y$. The returned vectors `xj` and `r2u` should be as in Programming Exercise 12.

(b) Write a `function p13test()` that tests your implementation of the preconditioned CG method with the matrix A and right hand side b from data `p13fem.mat` (see ISIS website). Use the following parameters: $x_0 = (0, \ldots, 0)^T \in \mathbb{R}^N$, $maxit = 300$ and $tol = 10^{-12}$.

Compare the following preconditioners:

- No preconditioner: $L_1 =$ `speye(N)`;
- Diagonal preconditioner: $L_2 =$ `spdiags(sqrt(diag(A)),0,N,N)`
- Incomplete Cholesky decomposition (see `doc cholinc`). Notice that `cholinc` generates an upper triangular matrix, therefore you need the transposed matrix, i.e.,:

$L_3 =$`cholinc(A,1e-1)'` $\quad$ $L_5 =$ `cholinc(A,1e-3)'`
$L_4 =$`cholinc(A,1e-2)'` $\quad$ $L_6 =$ `cholinc(A,1e-4)'`

Use `semilogy` to plot the computed relative residuals for all preconditioners $L_1 - L_6$ (all 6 curves on one plot). Annotate your plot.

(c) Approximate the condition numbers of the **preconditioned matrices** (not of the preconditioners $L_1 - L_6$ itself) from (b) using the Euclidean norm (2-norm). For each square, invertible matrix B the condition number can be obtained using the relation $\kappa(B) = \|B\|\|B^{-1}\|$. Use MATLAB's `normest` and `svds` to approximate the Euclidean norm (2-norm) $\|B\|$ and the norm of the inverse matrix $\|B^{-1}\|$, respectively. The condition number can be then approximated as follows:

$$\kappa(B) \approx \texttt{normest(B)/svds(B,1,0)}$$

where $B = (L_j L_j^\mathsf{T})^{-1} A$ is the preconditioned matrix. Write a

```
function con = p13condition()
```

that returns a column vector `con` $\in \mathbb{R}^{6,1}$ with `con(j)` $\approx \kappa((L_\mathrm{j} L_\mathrm{j}^\mathsf{T})^{-1} A)$. What can you say about the relation between the value of the condition number and the speed of convergence?

(d) Repeat the experiments from (b) for the matrix and right hand side obtained from your 3D FEM implementation from Programming Exercise 11 (d), namely use your implementation of the preconditioned CG method instead of MATLAB's backslash operator. Because the system matrix in Programming Exercise 11 is nonsymmetric, we only use the symmetric part (corresponding to the non-Dirichlet nodes) for the solution of the linear system. With `b, S, M, D` and `fh` obtained from your code you can use the following code snippet to obtain the symmetric matrix `LhII`, solve for the inner nodes `xI` and reconstruct the full solution vector `xh`:

```
Lh=S+M+D;
bD=find(b); bI=find(~b);
LhII=Lh(bI,bI); LhID=Lh(bI,bD);
fD=fh(bD); fI=fh(bI);
xI=LhII\(fI - LhID*fD); % use preconditioned CG here!
xh=zeros(size(fh,1),1); xh([bD;bI])=[fD;xI];
```

This code snippet will be discussed in the upcoming tutorial. Modify your implementation of `p11eoc.m` accordingly and save it as `p13eoc.m`. What can be observed in the error and EOC for different tolerances for the CG method? Plot the convergence history in the EOC step $r = 4$ for all preconditioners with `semilogy` into one plot.