Iterative Methods for Linear Systems. Homework #2. Numerical Methods for Differential equations. 2021-2022

December 1, 2021

1 The PCG method. Implementation notes

The Preconditioned Conjugate Gradient Method (PCG) is implemented in Matlab through the function `pcg`.

```
[x,flag,relres,iter,resvec]=pcg(A,b,tol,maxit,M1,M2,x0)
```

Input parameters.

- The SPD matrix A, the right-hand-side $\mathbf{b}$, the tolerance `tol`. Note that the exit test is on the relative residual
$$\frac{\|b - Ax\|}{\|b\|} < \texttt{tol} \ .$$
- The maximum number of iterations `maxit`.
- (Optional) The preconditioner is written as $M = M1 \cdot M2$. If for instance the incomplete Cholesky preconditioner is used then $M1 = L$ and $M2 = L'$. The PCG method solves the preconditioned system
$$M^{-1}Ax = M^{-1}b \ .$$
Application of the preconditioner to a given vector $\mathbf{r}$ is performed by solving two linear systems like
$$\mathbf{v} = M1 \backslash \mathbf{r}, \quad \mathbf{w} = M2 \backslash \mathbf{v}.$$
If $M1, M2$ are not specified then the Conjugate Gradient is used without preconditioning.
- (Optional) `x0` is the initial guess that can be specified by the user. If not specified, an all zero vector is used.

Output parameters.

`x` is the solution vector.

`flag` is an integer signaling different situations of termination of the function:

`flag`=0, convergence with a relative residual less than `tol` and a number of iterations less than `maxit`.

`flag`=1, no convergence within `maxit` iterations.

`flag`=2, the preconditioner is badly conditioned.

`flag`=3, The `pcg` function is stagnating, the residual norm is not decreasing in a number of consecutive iterations.

`flag`=4, One of the scalars computed by PCG is too small or too big to proceed.

`relres` is the relative residual computed as $\frac{\|b - Ax\|}{\|b\|}$.

`iter` is the number of iterations employed

`resvec` is a **vector** with the norm of the (absolute) residual at each iteration.

1.1 Preconditioners

- The Jacobi preconditioner is obtained by simply writing

```
M= sparse(diag(diag(A)))
```

- The triangular factor of the incomplete Cholesky preconditioner can be obtained by using the function `ichol` either without additional parameters:

```
L = ichol(A)
```

which yields the IC(0) i.e. an incomplete Cholesky factor with no fill-in, or with a record of parameters `opts` as

```
L = ichol(A,opts)
```

The most used fields of this structure are `opts.type` and `opts.droptol`. The following instructions obtain e.g. an IC preconditioner computed by discarding all elements below 10^{-2}.

```
opts.type    = 'ict'
opts.droptol  = 1e-2
```

To see the effect of a preconditioner let us consider the SPD matrix arising from Finite Difference discretization of the Laplace equation. The following code produces a $n = 10000$ matrix which discretizes the equation above on a square, computes the Incomplete Cholesky preconditioner wih no fill-in, defines a rhs corresponding to the exact solution with all ones and gives value to the other input parameters:

```
A=delsq(numgrid('S',102));
L=ichol(A);
n = size(A,1);
b=A*ones(n,1);
tol = 1.e-8;
maxit = 50;
```

Now we can invoke the `pcg` function, first without preconditioning and next with IC(0) as

```
[x,flag1,relres,iter1,resvec1]=pcg(A,b,tol,maxit);
[x,flag2,relres,iter2,resvec2]=pcg(A,b,tol,maxit,L,L');
```

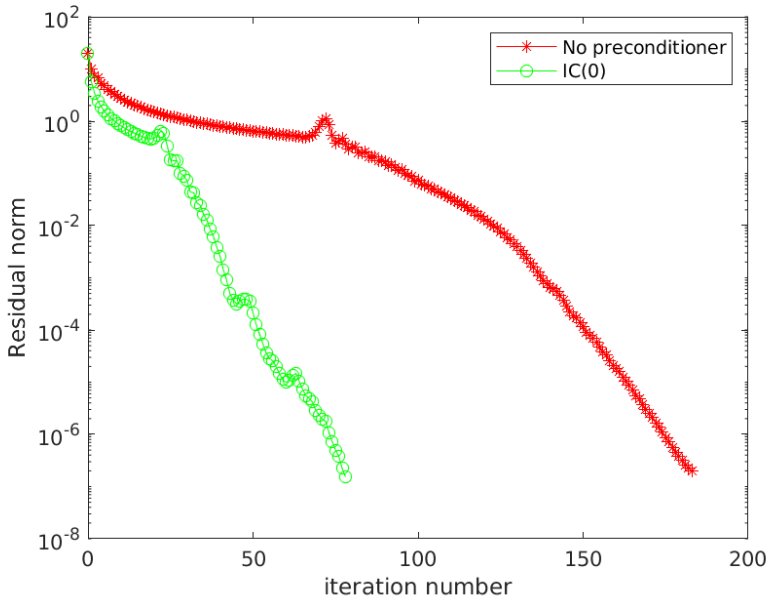

Figure 1: Residual norm vs iteration number for PCG with and without preconditioning.

Finally we compare the convergence profiles of the two cases by graphing in a logarithmic scale the norm of the residual vs the number of iterations:

```
semilogy(0:iter1,resvec1,'r-*',0:iter2,resvec2,'g-o');
legend('No preconditioner','IC(0)');
xlabel('iteration number');
ylabel('Residual norm');
```

obtaining the picture shown in Figure 1.

2 Nonsymmetric linear systems

The GMRES method is implemented in MATLAB and can be used with the following syntax

```
[x,flag,relres,iter,resvec] = gmres(A,b,restart,tol,maxit,M1,M2,x0)
```

For the meaning of the **input** parameters, see the description of the CG method or the MATLAB documentation obtained invoking the online help on the `gmres` function. The GMRES method needs an additional parameter, `restart` which does not indicate how many times the iteration will be restarted but indicates after how many iterations the restart will take place (the iteration and the construction of the Krylov subspace will be reinitialized every `restart` inner iterations). In this case the parameter `maxit` denotes the maximum number of **outer** iterations i.e. the number of **restarts** decreased by one.

Note. GMRES without restart can be invoked by setting to a large value (e.g. 1000, 5000) the `restart` parameter.

Regarding the preconditioner, if the ILU factorization ($A \approx LU$) is used, then `M1` is L while `M2` is U.

The meaning of the **output** parameters are the same as those of the PCG method with one exception: the output parameter `iter` is a vector with 2 components: `iter(1)` is the number of times the iteration has been started (number of restarts + 1) while `iter(2)` counts the number of iterations after the last restart. The actual number of iterations employed must be computed as

$$(\texttt{iter}(1) - 1) * \texttt{restart} + \texttt{iter}(2)$$

Note. MATLAB `gmres` implements left preconditioning. Hence vector `resvec` stores the norms of the absolute preconditioned residuals, while `relres` is $\frac{\|M^{-1}\mathbf{r}_k\|}{\|M^{-1}\mathbf{b}\|}$ and consequently the exit test performed is $\frac{\|M^{-1}\mathbf{r}_k\|}{\|M^{-1}\mathbf{b}\|} < \texttt{TOL}$.

2.1 ILU preconditioner

The ILU preconditioner can be computed using the `ilu` MATLAB function with the following syntax:

```
[L,U] = ilu(A,setup);
```

where the `setup` structure has (among the others) the following fields

```
setup.type = 'crout';
setup.droptol = 0.01;
```

If the setup parameter is not specified the command

```
[L,U] = ilu(A);
```

provides the ILU factorization with no fill-in.

3 Exercises

1. Produce your own implementation of the PCG method with Cholesky preconditioner with the following syntax:

   ```
   [x,resvec,iter]=mypcg(A,b,tol,maxit,L)
   ```

   To check the correctness, compare the output of your function with that of `pcg` on the example of the previous Section.

2. The condition number of the Poisson matrix is proportional to h^{-2} where h is the mesh discretization parameter. Verify the dependence of the number of CG iterations on $\sqrt{\kappa}(A)$ by solving 4 linear systems $A\mathbf{x} = \mathbf{b}$ with `A=delsq(numgrid('S',nx))` and $nx = 102, 202, 402, 802$.

   In particular:

   (a) Set the rhs b corresponding to the exact solution $\mathbf{x}$ with components

   $$x_i = \frac{1}{\sqrt{i}}, \quad i = 1, \ldots n.$$

   Use tol $= 10^{-8}$ and 5000 as the maximum number of iterations.

(b) Solve the four linear systems with (a) the non preconditioned CG, (b) PCG with IC(0), PCG with IC and droptol = 10^{-2} (c) and 10^{-3} (d).

(c) Produce a Table with the values of h and the number of (P)CG iterations: one row for each nx value.

3. Consider the following diagonal matrix A_1 of size $n = 10^4$ and $\kappa(A) = 10^3$, with elements

$$a_{ii} = 200 * i, \quad i = 1, 2, 3, 4, 5, \qquad \text{and} \quad a_{ii} = 1, \ i = 6, \ldots, 10^4.$$

Show theoretically that the PCG method will converge in exactly 6 iterations. Try your PCG (with tolerance 10^{-8}) with the previous matrix and a random (`rand(n,1)`) right hand side to check experimentally this property. Provide a `semilogy` plot of the residual norm vs the iteration number.

4. An SPD matrix can be obtained e.g. via the command

```
A=gallery('wathen',100,100);
```

Solve with PCG a linear system with A as the coefficient matrix and b the rhs corresponding to a random vector solution. Compare the non preconditioned GC with Jacobi and IC(0) preconditioners.

5. (a) Implement the GMRES method (**without restart**) as a MATLAB function following the Algorithm in the slides.

The syntax of the function should be the following:

```
function [x,iter,resvec,flag] = mygmres(A,b,tol,maxit,x0)
```

where the output parameters are

- `x` is the solution vector
- `iter` the number of iterations employed
- `resvec` the vector with the norm of the residuals
- `flag` a variable signaling breakdown (= 0: canonical termination, = −1: breakdown has occurred).

The input parameters are the coefficient matrix, the right hand side, the tolerance, the maximum number of iterations and the initial guess vector.

(b) Download (from Moodle) the matrix `mat13041.rig`, which is stored in coordinate format. Solve the linear system $A\mathbf{x} = \mathbf{b}$ with $\mathbf{b}$ corresponding to an exact solution with components $x_i = \frac{1}{\sqrt{i}}$. Use your GMRES implementation with `tol` = 10^{-10}, `itmax` = 550, and `x0` the all zero vector. Plot the residual norm vs the number of iterations in a `semilogy` profile.

6. (a) Implement the preconditioned GMRES method (**without restart**) as a MATLAB function with the following syntax

```
function [x,iter,resvec,flag] = myprecgmres(A,b,tol,maxit,x0,L,U)
```

Note. Use the following exit test:

$$\|(LU)^{-1}\mathbf{r}_k\| \ < \ TOL \cdot \|(LU)^{-1}\mathbf{b}\| \qquad \text{LEFT}$$

(b) Using the same matrix `mat13041.rig` as in the previous exercise, compute the ILU preconditioner with drop tolerance 0.1.

Then solve the linear system $A\mathbf{x} = \mathbf{b}$ with $\mathbf{b}$ corresponding to an exact solution with components $x_i = \frac{1}{\sqrt{i}}$. Use your preconditioned GMRES implementation with `tol` $= 10^{-10}$, `itmax` $= 550$, and `x0` the all zero vector. Display the final computed residual (last component of `resvec`) and the "true" final residual norm (computed as $\|\mathbf{b} - A\mathbf{x}\|$).

(c) Solve the same linear system as in 3.(b) using the MATLAB `gmres` function.

(d) Create a table comparing the results obtained with MATLAB `gmres` with those obtained with your own implementation in 3.(b). The number of iterations may differ by 2–3 at most between the two implementations.

7. Using the same matrix as in the previous exercise, solve the linear system $A\mathbf{x} = \mathbf{b}$ with $\mathbf{b}$ corresponding to an exact solution `x = 1./sqrt(1:n)`. Use the GMRES method[1] with `tol` $= 10^{-12}$, `restart` $= 10, 20, 30, 50$ and the ILU preconditioner with `droptol` $= 10^{-2}$. For each of the four values of restart, display the number of total iterations, the relative residual at convergence, and the CPU time employed.

   Provide the convergence profiles for the four values of restart in the same figure.

8. Download matrix `ML_laplace.mtx` from the Moodle platform. This matrix is in coordinate format. To trasform it to Matlab format use the following commands

```
A = load('ML_laplace.mtx');
A = spconvert(A);
```

   Solve the linear system with this matrix as coefficient matrix and a right hand side computed so that the exact solution is the vector of all ones. Keeping the *restart* parameter fixed to 50 use different drop tolerances for the ILU preconditioner (`droptol` $= 2 \cdot 10^{-2}, 10^{-2}, 3 \cdot 10^{-3}, 10^{-3}, 10^{-4}, 10^{-5}$). For each of the six runs display on a table: the drop tolerance, the number of iterations, the CPU time needed for computing the preconditioner (tprec), the CPU time needed for GMRES[1] to solve the system (tsol), the total CPU (tprec+tsol), the final residual norm and the **density** ρ of the preconditioner[2].

   Plot the convergence profile of the six runs on the same figure (use the legend command to clearly distinguish the different curves).

Upload on the Moodle page an archive with all the source files (e.g. M-files in Matlab) and a report containing the answer to theoretical questions, a brief description of the methods, some comments to the results and all the outputs. The report should be prefrerably written in LaTeXand named with your surname and name (e.g. `riemann_bernhard_HW2.pdf`).

[1] with the MATLAB implementation

[2] defined as $\rho = \dfrac{nnz(L) + nnz(U) - n}{nnz(A)}$, where n is the order of the matrix. Note that the nonzero number of a sparse matrix A is computed in MATLAB with `nnz(A)`.