Division of Scientific Computing
Department of Information Technology
Uppsala University

NGSSC: Numerical Methods in Scientific Computing
January 24, 2012

Assignment: Iterative solution methods for linear systems of algebraic equations

Requirements: At the end of the lab session, please deliver a Matlab-gener ated report on the runs you performed. Give your comments and observations (included in the generated report or added by hand). Print the report on `pr1515`.

The task of this computer lab is to test various iterative solution methods and obtain some experience regarding their rate of convergence and how to control it.

Exercise 1 (Forward and backward Gauss-Seidel method)
Background problem: Consider the convection-diffusion problem in two dimensions

$$-\varepsilon\Delta u + v_1 u_x + v_2 u_y = f \quad (x,y) \in \Omega = [0,1]^2 \tag{1}$$

The parameter ε is positive but can be very small. The coefficients v_1 and v_2 are in general functions of x, y and may change sign being both positive and negative in the domain Ω.
Equation (1) can be discretized using different methods. These result in a discrete system of linear equations with a matrix A, which can have very different properties. It is in general nonsymmetric, but for special choice of the method and the coefficients v_1 and v_2 can be also symmetric. Two classical discretization methods are ”Central Differences” and ”Upwind”, and will be referred to below as ”cd” and ”up”. These are implemented in Matlab and obtained via a call to the routine `cd_main`. The routine is interactive and asks for three parameters:

Refinement parameter	n	determines how large the system will be. The resulting matrix will be of size $A((n-1)^2 \times (n-1)^2))$. In the numerical tests you should try $n = 6, 7, 8, 9, 10$.
Epsilon	$cdeps$	The ε parameter in equation (1). For the numerical tests ε should be taken equal to 1, 0.01 and $1e-6$.
Vector field flag	$flag$	A parameter determining v_1 and v_2

The routine then calls both the forward and backward Gauss-Seidel methods, and reports the iteration counts.

1. Load all files with names starting with `cd` and the file `GaussSeidel.m`. The latter contains an implementation of the Forward and Backward Gauss-Seidel method. Browse at least the routines `cd_main.m` and `GaussSeidel.m` and make sure you understand what are they doing.

2. Generate some test matrices of varying size, $\varepsilon = 1, 0.1, 1e-6$ and $flag = 1, 2$. To obtain the matrices and the corresponding right hand sides, call the routine `cd_main`. The matrices and the right hand side vectors are named `A_cd, b_cd, A_up, b_up`. An exact solution is also generated, named `sol`.

   Check the sparsity of the above matrices (`spy`). Are some of these symmetric (which)? Compute their complete spectra (say, for $n = 10$) by using the MATLAB function `eig`. Plot those and compare. You are advised to repeat the experiment for a larger matrix size to see how the spectrum develops.

3. Test the performance of both forward and backward Gauss-Seidel methods on both matrices `A_cd, A_up`.

   For small values of $cdeps$, one of the two methods is performing exceptionally good on the upwind matrices matrices. To give you a hint, choose the value of Epsilon to be at least 0.001 or smaller. Then run both the forward and backward Gauss-Seidel methods.

   What do you observe? Try to analyse what happens and give your arguments.

   More hints:
   - Check the spectrum of the iteration matrices, corresponding to the Forward and Backward Gauss-Seidel methods.
   - Check the values of the entries of the corresponding matrices.

4. Try to solve the matrices `A_up` using unpreconditioned `gmres`.

   (a) Consult first `help gmres`

   (b) Execute `[x,flag,relres,iter,resvec] = gmres(A_up,b_up,N,1e-6,N);` where `N=size(A_up,1);`.

   (c) Plot the residual curve (also called 'convergence history' of gmres. First try `plot(resvec)` and then `semilogy(resvec,'o')`. What do you see? How does the convergence of the unpreconditioned gmres compares with that of GS for the particular problem?

   (d) Try to speedup gmres by adding a preconditioner. Use MATLAB's incomplete LU factorization routine `luinc`. Do the following: `[L1,U1]=luinc(A_up,tau);` for `tau=1e-1`. Then execute
   `[x1,flag,relres1,iter1,resvec1] = gmres(A_up,b_up,N,1e-6,N,L1,U1);`
   Check the two gmres convergence histories on one plot:
   `semilogy(resvec,'o-'),hold,semilogy(resvec1,'rd-')`
   Note: You may wish to plot the convergence of GS on the same plot. For that reason you have to uncomment a few lines in the routine `GaussSeidel.m`.

Exercise 2 (Final termination property for the unpreconditioned CG method)

1. Consider the solution of $A\mathbf{x} = \mathbf{b}$ by the standard conjugate gradient, where

$$A = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

   The exact solution is $\hat{\mathbf{x}} = [1, 1, \cdots, 1]^T$. Starting with $\mathbf{x}^0 = [0, 0, \cdots, 0]^T$ run the unpreconditioned conjugate gradient method. You can either use your own implementation of the standard CG method or the one available in `Matlab`, called for example as

   ```
   x_it=pcg(A,b,1e-6,size(A,1));
   ```

   The matrix A can be generated in various ways. One lazy possibility is the following:

   ```
   A = 2*eye(n);
   A = A + diag(-ones(n-1,1),1);
   A = A + diag(-ones(n-1,1),-1);
   A(n,n)=1;
   ```

   Run experiments for a number of different sizes of the matrix A, say, $n = 10, 50, 100$.

2. For how many iterations does the method converges? Monitor the current CG iterates (the approximations of the solution $\mathbf{x}$ produced during each iteration. What do you observe?

3. It is very instructive to derive the exact form of the CG iterates and see how the method proceeds. One finds that after k iterations

$$\mathbf{x}^{(k)} = \left[\frac{k}{k+1}, \frac{k-1}{k+1}, \cdots, \frac{1}{k+1}, 0, \cdots, 0\right]^T$$

   for $1 \leq k \leq n-1$ and $\mathbf{x}^{(n)} = \hat{\mathbf{x}}$.

   Hence, the information travels one step at a time from left to right and it takes n steps before the last component has changed at all.

Exercise 3 (The unpreconditioned CG method converging for more than n iterations, Toeplitz matrices

Consider the matrix of size $n \geq 1$

$$A_n(a) = \begin{bmatrix} a_{\frac{1}{2}} + a_{\frac{3}{2}} & -a_{\frac{3}{2}} & & \\ -a_{\frac{3}{2}} & a_{\frac{3}{2}} + a_{\frac{5}{2}} & -a_{\frac{5}{2}} & \\ & \ddots & \ddots & \ddots \\ & & -a_{n-\frac{1}{2}} & a_{n-\frac{1}{2}} + a_{n+\frac{1}{2}} \end{bmatrix}, \quad a_t = a\left(\frac{t}{n+1}\right), \qquad (2)$$

where $a : [0, 1] \to \mathbf{R}$ is a positive function (choose now a positive function).

1. Use the Matlab file `Toeplitz_matrix.m` to build the matrix $A_n(a)$ with a fixed function a (chosen by you) and $n \geq 1$ as a parameter. For example, for $a(x) = x^2 + 1$

   ```
   A=Toeplitz_matrix(n,@(x) x^2+1);
   P=Toeplitz_matrix(n,@(x) 1);
   ```

2. Apply CG to a system $A_n(a)x = b$ (choose your data vector b) for various n ($n = 16, 32, 64, 128, \ldots$) and fixed tolerance $\epsilon = 10^{-6}$. Permit the CG to run for more iterations than the size of the matrix.
   • What do you see concerning the number of required iterations for convergence? • Can you explain the observed convergence? What could have gone wrong?

3. Consider the matrix $P_n = A_n(1)$ and apply CG to a system $A_n(a)x = b$ with preconditioner P_n (choose your data vector b) for various n ($n = 16, 32, 64, 128, \ldots$) and fixed tolerance $\epsilon = 10^{-6}$:
   • What do you see concerning the number of required iterations for convergence?

Exercise 4 (A nonsymmetric matrix; convergence of the GMRES method)

1. Consider the **non-symmetric** sparse matrix stored in `pores_2.mat`, which is the coefficient matrix resulting from a problem in oil reservoir modeling.

   Study the matrix - its sparsity pattern, conditioning, scaling. Is the matrix strongly non-symmetric or not? Is it well- or ill-conditioned?

2. Solve the `pores_2`-problem using the GMRES iterative method, implemented by the `Matlab` commands `gmres`.
   ```
   [x,flag,relres,iter,resvec] = gmres(A,b,restart,tol,maxit);
   ```
   As right-hand vectors use the vector stored in `pores_2_b.mat` and in `pores_2_b1.mat`, and as a stopping tolerance use `tol=1e-6`. Try different values of the restarting length, including full GMRES.

   Plot the convergence history of each run (for instance, as `semilogy(resvec)`). Compute the exact solution by the direct `Matlab` solver and compare the difference between that one and the iteratively computed solution.

3. Scale the matrix A symmetrically to unit diagonal. Run the experiments again. Does the scaling help? If 'yes', in what sense? Does diagonal scaling affect the accuracy of the computed solution in this case?

   Remark 1 *Note that the diagonal elements of A are both positive and negative.*

4. Investigate the effect of using some incomplete LU-factorization preconditioner for the scaled system, obtained using `Matlab` command `luinc`. Try, for instance, `[L,U] =`

`luinc(A,tau)` with `tau=0.1` and `tau=1e-5`. Compare the convergence and the accuracy of the computed solution. Compute the condition number of the preconditioned system. What are your conclusions?

Exercise 5 (Peaks and plateaus)

Background: In the so-called Galerkin methods, typical representatives of which are the Full Orthogonalization method (FOM), Biconjugate gradient (BICG) method and the Lanczos method, the approximation $\mathbf{x}^{(k)}$ is chosen such that the corresponding residual is orthogonal to a certain space. The methods in this class suffer some numerical instabilities and various minimum- and quasi-minimum residual methods have been developed to overcome these instabilities.
Examples of such pairs of methods are
1: GMRES and FOM
2: QMR and BICG
3: Minres and Lanczos
4: GCG-RM and GCG-OR.
The convergence behaviour of pairs of such methods shows striking similarities. One essential result shown is a relation between residuals of these pairs, called the effect of peaks and plateaus. Whenever a peak appears in the convergence plot of the orthogonal residual method, there is a plateau under it for the corresponding norm-minimizing method. Several peaks may sit on the top of what appears to be a single plateau, where there are unacceptably small improvements in the residual norm over a number of consecutive iterations. In addition, whenever the residual norm plot for the norm-minimizing method rapidly decreases, the corresponding residual norm plot for the orthogonal residual method also rapidly decreases. Thus, the corresponding residual plots appear to track each other if the arithmetic used is exact. In finite precision arithmetic one extremely high peak could lead to loss of accuracy and then the convergence plot of the orthogonal method will not meet the convergence plot of the corresponding minimal residual method at the end of the plateau under this peak.
Denote by $\mathbf{r}^{(k)}$ and $\mathbf{r}^{(k,G)}$ the residual of a minimal residual method and the residual of the corresponding Galerkin orthogonal residual method. It is shown that if the residuals are computed exactly at step k, they satisfy the following relation

$$||\mathbf{r}^{(k,G)}|| \geq \frac{||\mathbf{r}^{(k)}||}{\sqrt{1-\frac{||\mathbf{r}^{(k)}||^2}{||\mathbf{r}^{(k-1)}||^2}}}.$$

Whenever $\frac{||\mathbf{r}^{(k)}||^2}{||\mathbf{r}^{(k-1)}||^2}$ is close to unity, then $||\mathbf{r}^{(k,G)}||$ is much larger than $||\mathbf{r}^{(k)}||$.
The following test illustrates the effect of peaks and plateaus.

1. Use the same set of matrices as in Exercise 1. Generate the matrices for
   `Refinement parameter: 20`,
   `Epsilon: 1e-8` and
   `Vector field flag: 1`.

2. Solve the corresponding system `A_up*x=b_up` by the BICG and the QMR methods:

```
    [x_qmr,f_qmr,rres_qmr,it_qmr,rvec_qmr] = qmr(A_up,b_up,1e-6,size(A_up,1));
[x_bcg,f_bcg,rres_bcg,it_bcg,rvec_bcg] = bicg(A_up,b_up,1e-6,size(A_up,1));
```

   Plot the convergence histories of both methods on one figure. Check whether the accuracy of the solution is affected from the erratic convergence behaviour of the BICG method.

3. One can also check the peak-plateau effects on the matrix from Exercise 1. What do you observe?

Exercise 6 (Try Algebraic Multigrid)
this exercise can only be done on the Linux machines. Otherwise the `agmg` software cannot be installed.
The classical multigrid (MG) method is one of the so-called *optimal* order methods. The optimality is understood as follows:

- The method has optimal convergence property. This means that, independent of the size of the problem, respectively of the discretization parameter h, for a given stopping tolerance, the method will converge within a constant number of iterations.

- The method has optimal computational complexity. This means that the cost per one iteration is linearly proportional to the number of degrees of freedom.

Due to the above two optimality properties MG has overall computational complexity which is linearly proportional to the total number of degrees of freedom. The same properties hold for its algebraic counterpart, the Algebraic Multigrid (AMG) method.

We consider as a model problem the two-dimensional convection-diffusion-reaction (CDR) equation,

$$-\frac{\partial}{\partial x}\left(\varepsilon_1\frac{\partial u}{\partial x}\right)-\frac{\partial}{\partial y}\left(\varepsilon_2\frac{\partial u}{\partial y}\right)+b_1\frac{\partial u}{\partial x}+b_2\frac{\partial u}{\partial y}+cu=f \tag{3}$$

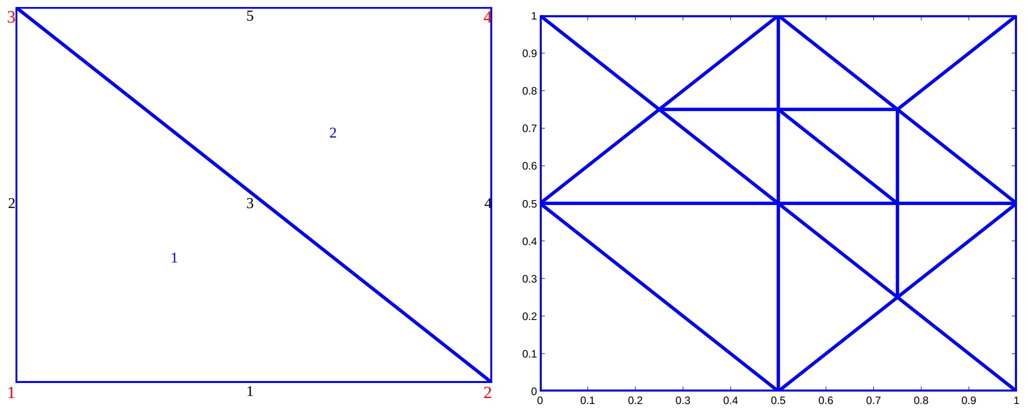
with some suitable boundary conditions.
The domain of definition is the unit square $\Omega=[0,1]^2$, discretized by some triangular mesh. The problem is discretized using standard linear finite element basis functions.

Problem (3) has three problem parameters: the diffusion coefficients $\varepsilon=[\varepsilon_1,\varepsilon_2]$, the convection vector field $\mathbf{b}=[b_1,b_2]$ and the reaction coefficient c.
By giving various values of the problem coefficients, we can simulate a series of classical problems of elliptic type:

(P1) Poisson's problem: $\varepsilon=[1,1]$, $\mathbf{b}=[0,0]$, $c=0$.

(P2) Anisotropic problem: $\varepsilon=[1,\varepsilon_2]$ or $\varepsilon=[\varepsilon_1,1]$ with $\varepsilon_i\ll 1$, $\mathbf{b}=[0,0]$, $c=0$.

(P3) Poisson's problem with variable (jumping) coefficients: let $\varepsilon_i, i=1,2$ be functions of space or constants with different values in different parts of Ω, $\mathbf{b}=[0,0]$, $c=0$.



(a) (b)

Figure 1:

(P4) Singularly perturbed convection-diffusion problem: $\varepsilon_1 = \varepsilon_2 \ll 1$, $\mathbf{b}$ nonzero.

(P5) Parabolic-type problems: $c > 0$. The coefficient c can be related to the inverse of the time-step when discretizing a parabolic problem with FEM in space and with some implicit time-stepping method.

(P$_\infty$) Any other choice of problem parameters could also be of interest, however we limit ourselves to the cases (P1)-(P5).

The resulting system matrix is of the form $K = L(\varepsilon) + C(\mathbf{b}) + cM$, where L, C and M are the discrete divergence, convection and mass matrices, correspondingly.

Software needed and adjustments:

1. Download all files from the directory.
2. Download and install `agmg` from `http://homepages.ulb.ac.be/~ynotay/`.
3. In the file `Main_CDR.m` correct the paths for the MG and AMG directory.

The main routine to use is `Main_CDR.m`. You can edit it and give various values to the problem parameters. The program reads a predefined coarse mesh and then refines it a given number of times. The two predefined meshes are shown in Figure 1. Then it asks for a number of refinements and generates the corresponding matrix of the linear system to be solved.

The major task is to see the robustness and scalability of multilevel methods of Multigrid type. Robustness is understood as (near) independence of problem parameters and problem size. These methods are shown to have computational cost per iteration, linear in the number of degrees of freedom and (nearly) optimal convergence rate, i.e., the number of iterations stay (nearly) constant with the problem size (for a given set of problem parameters). The word 'near' reflects

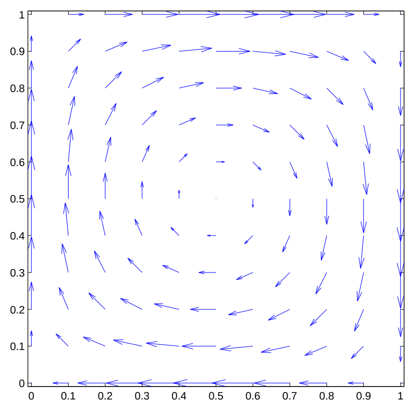


Figure 2:

that when one uses a V-cycle, as in the provided solvers, a slight growth in the iteration counts might be observed when the problem size is increased.
The above study can be made by performing the following subtasks.

(T1) Check the suitability of `agmg` to solve strongly anisotropic problems. Choose $\varepsilon = [1, 10^{-s}]$, $s = 2, 3, 4$, $\mathbf{b} = [0, 0]$, $c = 0$. How do the iteration counts behave? Check the spectrum of the original matrix and you will see that there is a whole cluster of very small eigenvalues, and the unpreconditioned CG will have a slow convergence rate.

(T3) Consider a problem with jumping coefficients. Change the coarse mesh generating routine from `Square_ConvDiff` to `Square_disco_hier(jmp)`. The code assumes that in the subregion $0.5 \leq x, y \leq 0.75$ the diffusion coefficient is equal to `jmp` and is taken to be one in the rest of the domain. By varying `jmp` one tests the robustness of the solver with respect to discontinuous coefficients.

(T4) Consider heat conduction in a space domain, where also a convection is present. Choose $\varepsilon = [1, 1]$, `flag=1`, and vary $c = 10^s$, $s = 0, 1, 2, 3, 4$. What happens to the matrix for large values of c?

(T5) Check the suitability of `agmg` for solving singularly perturbed convection-diffusion problems. Choose $\varepsilon_1 = \varepsilon_2 = 10^{-s}$, $s = 2, 3$, `flag=1`, $c = 0$. The variable `flag` determines which vector field to be used. The one suggested is of the form, shown in Figure 2.

Remark: Testing solvers for singularly perturbed convection-diffusion problems, where typically layers occur in the solution, is a difficult task as it requires suitable discretizations and adaptive meshes to relate h with ε in order to resolve those layers. Here, the boundary conditions are Dirichlet all over and, therefore, no layers are present.

Disclaimer: The code is not much optimized wrt matrix generation and above 7 levels of refinement it feels rather slow. Any input on that or other issues related to the lab will be very highly appreciated. Thank you! Maya Neytcheva