Division of Scientific Computing, Department of Information Technology, Uppsala University

Numerical Linear Algebra

Feb-March, 2021

Hands-on: Krylov Subspace iterative solution methods: CG, GMRES, BiCG, QMR, MINRES and more

The aim of this computer lab is to get some insight on the convergence of some of the most often used Krylov subspace methods and to experience Lanczos method for estimating the extreme eigenvalues of a matrix.

Requirement: Prepare at least two questions to discuss at the next lecture.
Prepare a written report using Matlab's 'Publish' option.

A suggestion: when you plot the convergence history (the vector, containing the residual norm at each iteration (resvec)), in particular when you plot the result of several runs on one figure, use `semilogy(resvec./resvec(1))`.

Exercise 1 (Convergence of 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,fl,resnorm,it,resvec]=pcg(A,b,1e-6,size(A,1));
```

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

```
e = ones(n,1);
A = spdiags([-e 2*e -e], -1:1, n, n);
A(n,n)=1;
```

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

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.

4. *Will the convergence be faster if you choose a different initial guess?*

```
x0 = rand(n,1);
[x_it,fl,resnorm,it,resvec]=pcg(A,b,1e-6,size(A,1),[],[],x0);
```

or

```
[x_it,fl,resnorm,it,resvec]=pcg(A,b,1e-6,size(A,1),[],[],ones(n,1)-x0);
```

5. *Why CG converges so slow in this case?*

Exercise 2 (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 nonsymmetric 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?*

4. *Investigate the effect of using some incomplete LU-factorization preconditioner, given by the* `Matlab` *command* `luinc`*. What are your conclusions?*

Exercise 3 (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 some nonsymmetric matrix (matrices), for instance the convection-diffusion matrix* `A_upp`*, generated by* `cd_main` *with* `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 with a stopping tolerance* 10^{-6} *or* 10^{-8}*:*

```
[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. *Can you check the peak-plateau effects on the matrix from Task 2? What do you observe?*

Exercise 4 (The Chebyshev iteration method (Oh, again!)) *Use Lanczos method for estimating the extreme eigenvalues of a matrix.*
Test the Second Order Chebyshev iterative solution method, which works for symmetric positive definite matrices.
The details on the method are given in the appendix below.
You can use some of the following four test matrices which have different properties, however are all symmetric and positive definite. Below, s determines the size of the problem, namely, the matrix size is $n = 2^s$.

A: `A=matgen_disco(s,1);`
B: `B=matgen_disco(s,0.001);`
C: `C=matgen_anisot(s,1,1);`
D: `D=matgen_anisot(s,1,0.001);`

- `matgen_disco.m`
  The routine `matgen_disco.m` *generates a finite element stiffness matrix for the Laplace equation*

$$-\frac{\partial}{\partial x}\left(a\frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(a\frac{\partial u}{\partial y}\right) = f \tag{1}$$

  in $\Omega \equiv (0,1)^2$ *where* a *is a constant and* $a = \varepsilon \ll 1$ *in a subset* $\widetilde{\Omega} \subset \Omega$ *and* $a = 1$ *elsewhere with* $\widetilde{\Omega} \equiv \{1/4 \le x \le 3/4,\ 1/4 \le y \le 3/4\}$. *The problem is then discretized using regular isosceles triangles and piece-wise linear basis functions. The mesh-size parameter* h *is equal to* 2^{-s} *for some integer* s *and is chosen always such that there will be mesh-lines along the edges of* $\widetilde{\Omega}$

  The routine is called as `A=matgen_disco(s,a);`.

- `matgen_anisot.m`
  The routine `matgen_anisot.m` *generates a finite difference (5-point) discretization of the anisotropic Laplacian*

$$-\varepsilon_x u_{xx} - \varepsilon_y u_{yy}.$$

  The routine is called as `A=matgen_anisot(s,epsx,epsy);`, *where again mesh-size parameter* h *is equal to* 2^{-s}.

  Clearly, when both `epsx` *and* `epsy` *are equal to one, the generated matrix corresponds to* $-\Delta u = f$ *discretized with central differences.*

- `Lanczos.m`
  In order to use the Chebyshev iterative method one needs to estimate the extremal eigenvalues of A. The routine `Lanczos` *computes such* approximations. *Note, however that there is no guarantee that we will obtain a lower bound for* λ_{min} *and an upper bound for* λ_{max}*!*

  The routine is called as `[lanmin,lanmax]=Lanczos(A);` *It performs internally a number of Lanczos steps until the following criteria are met:*

$$|\lambda_1^{(k)} - \lambda_1^{(k-1)}| \leq \epsilon \text{ and } |\lambda_n^{(k)} - \lambda_n^{(k-1)}| \leq \epsilon$$

  with a default value of $\epsilon = 0.01$.

  Remark: *You can estimate the extreme eigenvalues using some other technique, for example, using Gershgorin's theorem.*

Tasks

1. *Have a look at the description of the algorithm. You have to notice the following important features of that algorithm:*

   (i) The method needs bounds for the spectrum of the matrix (a lower bound of the smallest eigenvalue and a upper bound for the largest eigenvalue).

   (ii) The method does not require any scalar products, which is an important feature when implemented in parallel, since no global communications are needed.

   (ii) For a given accuracy ε*, formula (7) predicts in advance how many iterations should be performed, thus, no stopping test is needed.*

2. *Test the method and the sensitivity with respect to the accuracy of the eigenvalue bounds. Change the stopping tolerance in the Lanczos algorithm. Can Gershgorin theorem help? Use also the exact extreme eigenvalues for a comparison.*

Appendix: The Second Order Chebyshev iterative solution method

Description

Let A be a real symmetric positive definite (s.p.d.) matrix of order n. Consider the solution of the linear system $A\mathbf{x} = \mathbf{b}$ using the following iterative scheme, known as the *Second Order Chebyshev iterative solution method*:

$$\begin{array}{rl} \mathbf{x}_0 \text{ given}, & \mathbf{x}_1 = \mathbf{x}_0 + \frac{1}{2}\beta_0 \mathbf{r}_0 \\ \text{For} & k = 0, 1, \cdots \text{ until convergence} \\ & \mathbf{x}_{k+1} = \alpha_k \mathbf{x}_k + (1 - \alpha_k)\mathbf{x}_{k-1} + \beta_k \mathbf{r}_k. \\ & \mathbf{r}_k = \mathbf{b} - A\mathbf{x}_k. \end{array} \tag{2}$$

Let $\mathbf{x}^*$ be the exact solution of the above linear system. Denote by $\mathbf{e}_k = \mathbf{x}^* - \mathbf{x}_k$ the iterative error at step k. Clearly, $\mathbf{r}_k = \mathbf{b} - A\mathbf{x}_k = A(\mathbf{x}^* - \mathbf{x}_k) = A\mathbf{e}_k$.
It is seen from the recursive formula (2) that for each k, the errors satisfy a relation of the form

$$\mathbf{e}_{k+1} = \alpha_k \mathbf{e}_k + (1 - \alpha_k)\mathbf{e}_{k-1} + \beta_k A\mathbf{e}_k = Q_k(A)\mathbf{e}_0,$$

where $Q_k(\cdot)$ is some polynomial of degree k. Furthermore, the polynomials $Q_k(A)$ are related among themselves as follows:

$$Q_{k+1}(A) - \alpha_k Q_k(A) - \beta_k A Q_k(A) + (1 - \alpha_k)Q_{k-1}(A) = 0,\ k = 1, 2, \cdots \tag{3}$$

We compare the recurrence (3) with the recursive formula for the Chebyshev polynomials, namely,

$$T_0(z) = 1, \quad T_1(z) = z, \quad T_{k+1}(z) - 2T_k(z) + T_{k-1}(z) = 0. \tag{4}$$

One can easily see that for the following special choice of the method parameters

$$\alpha_k = \frac{2c\,T_k(z)}{T_{k+1}(z)} = 1 + \frac{T_{k-1}(z)}{T_{k+1}(z)}$$

and

$$\beta_k = \frac{4}{b-a} T_k(z)/T_{k+1}(z), \quad \text{where } z = \frac{b+a}{b-a}, \quad 0 < a \le \lambda_{min}(A),\ \lambda_{max}(A) \le b,$$

we get

$$Q_k(A) = \frac{T_k(z)}{T_k(z)}, \qquad Z = \frac{1}{b-a}\left[(b+a)I - 2A\right].$$

We now recall that the Chebyshev polynomials possess the following optimal approximation property - among all normalized polynomials of degree n defined in an interval $[a, b]$, the nth degree Chebyshev polynomial is the one which differs from zero least (measured in local min and max in $[a, b]$).
Therefore, for this particular polynomial $Q_k(A)$ we have (due the above approximation properties of the Chebyshev polynomials)

$$\max_z |Q_k(A)z| \le \min_{P \in \Pi_k} |P(A)z|,$$

i.e., at each step we achieve the best possible error reduction. (Here Π_k is the set of all polynomials of degree k.)
In order to use the Chebyshev iteration method we need to estimate the extreme eigenvalues of A and to determine an interval $[a, b]$ which contains the spectrum of A.
Having done that, one finds the following formulas to compute the method parameters recursively:

$$\alpha_k = \frac{a+b}{2}\beta_k, \quad \frac{1}{\beta_k} = \frac{a+b}{2} - \left(\frac{b-a}{4}\right)^2 \beta_{k-1}, \quad \beta_0 = \frac{4}{a+b}. \tag{5}$$

Note that $\alpha_k > 1, k \ge 1$.

Theorem 1 *The following results hold:*

$$\lim_{k\to\infty}\beta_k = \frac{4}{\left(\sqrt{a}+\sqrt{b}\right)^2},$$

$$\frac{\|\mathbf{e}_k\|_A}{\|\mathbf{e}_0\|_A} \le \frac{1}{T_k\left(\frac{b+a}{b-a}\right)} \le 2\frac{\sigma^k}{1+\sigma^{2k}}, \qquad \sigma = \frac{1-\sqrt{a/b}}{1+\sqrt{a/b}}.$$

It follows then that $\frac{\|\mathbf{e}_k\|_A}{\|\mathbf{e}_0\|_A} \to 0$ *monotonically.*

(For the special choice $a = \lambda_{min}(A)$*,* $b = \lambda_{max}(A)$*, we have* $\sigma = \frac{1-\sqrt{\varkappa(A)^{-1}}}{1+\sqrt{\varkappa(A)^{-1}}}$*.)*

From the above convergence rate estimate one can determine a priory the number of Chebyshev iterations needed to be performed in order to achieve an error reduction

$$\frac{\|\mathbf{e}_k\|_A}{\|\mathbf{e}_0\|_A} \le \varepsilon. \tag{6}$$

Indeed, to insure (6), it suffices to perform

$$k \ge \ln\left(\frac{1}{\varepsilon}+\sqrt{\frac{1}{\varepsilon^2}-1}\right)/\ln(\sigma^{-1}) \quad or \quad k^* = \left\lceil \frac{1}{2}\sqrt{\frac{b}{a}}\ln\frac{2}{\varepsilon}\right\rceil \quad iterations. \tag{7}$$