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

Numerical Linear Algebra
Feb-March, 2021

Hands-on: Ill-conditioning, basic iterative methods, general experience with solving linear systems of equations

The aim of this computer lab is to experience matrices with various properties, ill-conditioning and issues related to the solution of linear systems using basic iterative solution methods.

Requirement: Prepare at least two questions to discuss at the next lecture.

Exercise 1 (Ill-conditioning)
Consider the problem $A\mathbf{x} = \mathbf{b}$, where

$$A\mathbf{x} = \begin{bmatrix} 0.835 & 0.667 \\ 0.333 & 0.266 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \mathbf{b}.$$

Let the right-hand vector $\mathbf{b}$ be a result of an experiment and is read from the dial of a test instrument as $b_0 = \begin{bmatrix} 0.168 \\ 0.067 \end{bmatrix}$. However, due to the small uncertainty, we have to expect that

$$0.167 \leq b_1 \leq 0.169 \quad \text{and} \quad 0.066 \leq b_2 \leq 0.068.$$

Thus, the solution for $b_1 = \begin{bmatrix} 0.167 \\ 0.068 \end{bmatrix}$ and for $b_2 = \begin{bmatrix} 0.169 \\ 0.066 \end{bmatrix}$ should be expected to be as valid as that in the first case.

1. Find the exact solution of the system for each of the above vectors $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2$.
2. Is the observed instability due to some numerical procedure, the vector $\mathbf{b}$, the matrix A or a combination of these factors?
3. Try to quantify how ill-condition the problem is. How would you do this?
4. Perturbe the system as follows

   $$\begin{bmatrix} 0.835 & 0.667 \\ 0.333 & 0.266 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0.1669995 \\ 0.066601 \end{bmatrix}$$

   Determine the exact solution and compare it with the exact solution corresponding to $\begin{bmatrix} 0.168 \\ 0.067 \end{bmatrix}$.

   On the basis of the results formulate a statement concerning the necessity for the solution of an ill-posed system to undergo a radical change for every perturbation of the original system.

Exercise 2
Consider the Hilbert matrix H of size n defined by

$$H_{i,j} = \frac{1}{i+j-1},\ 1 \leq i,j \leq n$$

The matrix is generated by the matlab command `hilb(n)`.
This is an example of a rather nasty matrix to do numerical computations with. To illustrate this we try to solve a system with a Hilber matrix and to compute its inverse.

- Solution of systems with Hilbert matrices

  1. Check the condition number for Hilbert matrices of a few different sizes.
  2. Perform the following experiments

```
for n = 2:2:16
    H = Hilbert_matrix(n);
    x = ones(n,1);
    b = H*x;
    y = H\b;
    disp(['Norm of the residual: ' num2str(norm(b-H*y))])
    disp(['Norm of the error:    ' num2str(norm(x-y))])
    figure(1), clf, plot(b-H*y)
    figure(2), clf, plot(abs(x-y))
    pause
end
```

     Note how perfect the residual looks and how bad the error is.

  What is the reason for this behaviour and what can we do to improve the situation?

- Compute the inverse of a Hilbert matrix. It turns out that entries of the inverse of a Hilbert matrix have analytical expression:

$$\widetilde{H}_{i,j} = \widetilde{H}_{j,i} = \frac{r_{ij}}{i+j-1},\ 1 \leq i,j \leq n$$

  where $\widetilde{H}_{i,j}$ is defined for $j > i$ by the following recurrences

$$\begin{cases} r_{ii} = p_i^2 \\ r_{ij} = -\dfrac{(n-j+1)(n+j-1)}{(j-1)^2} r_{1,j-1},\ j > i \end{cases} \quad \begin{cases} p_0 = n \\ p_i = -\dfrac{(n-i+1)(n-i-1)}{(i-1)^2} p_{i-1},\ 1 \leq i \leq n \end{cases}$$

  These computations are performed by the command `invhilb(n)`.

  Try the following experiment to check the limitations of the `inv` command.

```
for i = 1:20
    x = max(max(abs(invhilb(i)-inv(hilb(i)))));
    fprintf(1,'For size i = %d deviation is x = %g \n',i,x);
end
```

Exercise 3 (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 diffusion 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, resulting in discrete systems of linear equations with matrices, that can have very different properties. Due to the convectio term, the matrices are in general nonsymmetric, but for special choice of the method and the coefficients v_1 and v_2, these can be also symmetric.

Two classical discretization methods for problem (1) are "Central Differences" and "Upwind", below referred to as "cd" and "up". These are implemented in Matlab and the corresponding matrices are obtained via a call to the functions `cd_cd2d` and cd_upwd2d.
To generate both types of matrices use `cd_main`. The program is interactive and asks for three parameters:

Refinement parameter	n	determines how large the system will be. The resulting matrix will be of size $A((n-1)^2 \times (n-1)^2))$.
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 program creates the matrices A_cd and A_up, 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. 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`.

2. Run `cd_main` for problem sizes $n = 51, 101, 201$, $\varepsilon = 1, 10^{-6}$ and $flag = 1, 2$. Take account for the number of iterations required, how this increase with size and how it depends on ε.

   You may study the matrices - sparsity patterns, are these symmetric or not. Compute their complete spectra for a few not very large sizes n by using the MATLAB function `eig(full(A))`. Plot those and see how the spectrum develops when n increases.

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

   For $flag = 1$ and $\varepsilon = 10^{-6}$, one of the two methods is performing exceptionally good on the upwind matrices matrices. Why is this so?

   Do you observe the same behaviour for $flag = 2$?

   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`.

4. Perform the following test: Uncomment lines 39-54, where the central difference matrix is used and run `cd_main` for $n = 51$, $flag = 1$ and $\varepsilon = 5\,10^{-3}$ and then for $n = 51$, $flag = 1$ and $\varepsilon = 4\,10^{-3}$.

   What do you see? Explain, please.

Exercise 4 (PMHSS) Consider the Preconditioned Modified Hermitian Skew-Hermitian Splitting (PMHSS) method ([1])
The PMHSS method can be used to solve the complex linear system

$$\boldsymbol{C}\boldsymbol{z} = \boldsymbol{h}, \tag{2}$$

where $\boldsymbol{C} = A + iB$, $\boldsymbol{z} = \mathbf{x} + i\mathbf{y}$ and $\boldsymbol{h} = \mathbf{f} + i\mathbf{g}$. The complex system can be rewritten in a matrix form

$$\begin{bmatrix} A & -B \\ B & A \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix} .$$

The PMHSS method takes the form of a stationary iterative method,

$$\begin{aligned} (\alpha V + A)\mathbf{x}^{k+1/2} &= (\alpha V - iB)\mathbf{x}^k + \mathbf{b} \\ (\alpha V + B)\mathbf{x}^{k+1} &= (\alpha V + iA)\mathbf{x}^{k+1/2} - i\mathbf{b} \end{aligned} \tag{3}$$

where α is a given positive parameter, V a prescribed symmetric positive definite matrix and i is the imaginary unit. We assume that A is symmetric positive devinite (spd) and B is symmetric positive semi-definite (spsd). It follows that both $\alpha V + A$ and $\alpha V + B$ are spd. For $V = I$ the method reduces to the modified Hermitian and skew-Hermitian splitting method, presented in [2]. One possible choice of V is A.
The method in (3) can be written in a compact form as

$$\mathbf{x}^{k+1} = \mathcal{L}(V;\alpha)\mathbf{x}^k + \mathcal{R}(V;\alpha)\mathbf{b}\,, \quad k = 0, 1, 2, \ldots \tag{4}$$

where the iteration matrix, $\mathcal{L}(V;\alpha)$, has the form

$$\mathcal{L}(V;\alpha) = (\alpha V + B)^{-1}(\alpha V + iA)(\alpha V + A)^{-1}(\alpha V - iB)$$

and

$$\mathcal{R}(V;\alpha) = (1 - i)\alpha(\alpha V + B)^{-1}V(\alpha V + A)^{-1}\,.$$

Tasks:

- Based on the form (3), formulate PMHSS for $\alpha = 1$ and $V = A$.
- Implement the PMHSS algorithm in Matlab.
  Hint: the method requires only one solution with the matrix $A + B$ and some vector operations. Use Matlab's $\backslash$ command to solve with $A + B$. Choose the initial approximation to be $\mathbf{x}^0 = \mathbf{0} + i\mathbf{0}$ and compute the right-hand side vector as `b=random_complex(n);`, where n is the size of the system.
- Load the files (one after another)

  ```
  Laplace2D_FEM_1089.mat
  Laplace2D_FEM_4225.mat
  Laplace2D_FEM_16641.mat
  Laplace2D_FEM_66049.mat
  Laplace2D_FEM_263169.mat
  ```

  The files contain two matrices K and M of sizes 1089, 4225, 16641, 66049 and 263169, a finite element stiffness (discrete Laplacian) and a mass matrix, correspondingly.
- Run your implementation for the complex matrix $A + i\,B$, where $A = K$ and $B = \omega M$, where ω could be $0.1, 1, 100$, for instance. Time the execution. Check how much time it takes for the direct solver to solve the original complex system. Compare.

Exercise 5 ('Impossible' matrices)
Load the matrices `PF_54.mat` and `PF_186.mat`. These correspond a numerical model of a real-life problem to two consequtive refinements of a model of the so-called 'wetting' phenomenon.
Study the matrices by all means you know - spectrum, condition number, scaling...
This is just to give you an indication how difficult real-life problems could be.

References

[1] Bai, Z.-Z., Benzi, M., Chen, F.: On preconditioned MHSS iteration methods for complex symmetric linear systems, *Numer. Alg.*, 56 (2011), 297–317.

[2] Bai, Z.-Z., Benzi, M., Chen, F.: Modified HSS iteration methods for a class of complex symmetric linear systems, Computing, 87(2010), 93-111.