Hands-on 1: Ill-conditioning

Exercise 1 (Ill-conditioned linear systems)

Definition 1 A system of linear equations is said to be ill-conditioned when some small perturbations in the system can produce relatively large changes in the exact solution. Otherwise, the system is said to be well-conditioned.

Consider the problem $Ax = b$, where

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

Let the right-hand vector $b$ be a result of an experiment and is read from the dial of a test instrument. Assume we know that the tolerance within which the dial can be read is $\pm 0.001$.

Assume that the values of the components are read as $b_0 = \begin{bmatrix} 0.168 \\ 0.067 \end{bmatrix}$. However, due to the small uncertainty, we have to expect that $0.167 \leq b_1 \leq 0.169$ and $0.066 \leq b_2 \leq 0.068$.

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 $b_0, b_1, b_2$.

2. Is the observed instability due to some numerical procedure, the vector $b$, the matrix $A$ or a combination of these factors?

3. Try to quantify how ill-condition the problem is. How would you do this?

Remark 1 One can demonstrate the reason for a $2 \times 2$-system to be ill-conditioned as follows. Geometrically, two equations in two unknowns define two straight lines. The point of intersection of those lines defines the solution. An ill-conditioned system represents two lines which are almost parallel. (It is advisable to try to plot the two lines corresponding to the considered system.)
Exercise 2
Consider the system in (1) once more,

\[
\begin{bmatrix}
0.835 & 0.667 \\
0.333 & 0.266 \\
\end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0.168 \\ 0.067 \end{bmatrix}
\]

and imagine that you somehow have computed a solution \( x = [-666, 834] \). Try to "check the error" by substituting the solution in the equations and to see what is the error. In other words, compute the residual \( r = Ax - b \).

1. Is the residual small or big? Is the residual a reliable source of information whether the computed solution accurate or not and in which cases?

2. How can we check a computed solution for accuracy?

Exercise 3
Determine the exact solution of the following system

\[
\begin{bmatrix}
8 & 5 & 2 \\
21 & 19 & 16 \\
39 & 48 & 53 \\
\end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 15 \\ 56 \\ 140 \end{bmatrix}.
\]

Now change 15 to 14 in the first equation and again solve the system with exact arithmetic. Is the system ill-conditioned?

Exercise 4 (Effect of matrix scaling)

Remark 2 (Practical scaling strategies)
1. When scaling, choose units which are natural to the underlying problem and do not distort the proportion between the problem components, such as matrix entries and entries of the right-hand vector. Column scaling can cause such a distortion.

2. Row-scale the system \( Ax = b \) so that the coefficient of maximum magnitude in each row is equal to 1.

3. When the matrix is symmetric, it is better to scale symmetrically, for example as \( \tilde{A} = DAD \), where \( D \) is diagonal \( D = \{1/\sqrt{a_{ii}}, i = 1, \cdots, n\} \).

Consider the following system of equations:

\[
\begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}
\]

Matrix of the above type is referred to as a Hilbert matrix.
(a) First convert the coefficients to 3-digit floating point numbers and the use 3-digit arithmetic with partial pivoting but no scaling to compute the solution.

(b) Again use a 3-digit arithmetic, but row-scale the coefficients (after converting them to 3-digit floating point numbers), and then use partial pivoting to compute the solution.

(c) Proceed as in [(b)] but this time row-scale the coefficients before each elimination step.

(d) Use exact arithmetic to determine the solution and compare the result with those obtained in (a), (b) and (c).

**Hint:** One should somehow simulate 3-digit arithmetic in Matlab. One limited possibility is to replace a value v by \( w = \text{floor}(v \times 10^3) \times 10^{-3} \). The latter will give the following approximation: \( v = 0.123333456 \), \( w = 0.1230 \).

**Exercise 5 (Feel the sparse-dense difference)**
Create a 100 × 100 random matrix that has about 5% nonzero entries:
\[ S = \text{sprand}(100, 100, .05); \]
Convert it to full matrix
\[ F = \text{full}(S); \]
Then check:
\[
\begin{align*}
\text{tic}, S*S; & \text{toc} \\
\text{tic}, F*F; & \text{toc}
\end{align*}
\]

**Exercise 6 (The effect of fill-in)**
MATLAB provides a nonsymmetric sparse demonstration matrix west0479 of dimension 479×479, which can be accessed after executing load west0479. Let us use an abbreviate name \( A = \text{west0479}; \)

1. Compute the sparse LU factorization, checking the time it takes to perform it, for example \( \text{tic}, [L, U, P] = \text{lu}(A), \text{toc}. \)

   Look at \text{spy} plots of \( A \), \( PA \), \( L+U \) and make note of the number of nonzero elements in the triangular factors.

2. Modify \( A \) by applying a random column permutation to it. Then repeat (1).

   \[
   \begin{align*}
   \% \text{ random column permutation} \\
   \text{col} & = \text{randperm}(479); \\
   \text{AP} & = A(:, \text{col}); \\
   \text{spy}(\text{AP})
   \end{align*}
   \]

3. Modify \( A \) by applying the column minimum degree ordering to the columns. Repeat (1).
4. Compare the above results.

5. Generate a random right-hand-side vector \( \mathbf{b}=\text{randn}(479,1) \). Solve the system as \( \mathbf{x}=\mathbf{A}\backslash\mathbf{b} \). Compare the solution time with that spent to factorize the matrix.

**Exercise 7 (‘Hard-to-handle’ matrices)**

Load the matrices Bone\_matr\_2169.mat and Bone\_matr\_127524.mat. These originate from a numerical model of a trabecular bone tissue. A small sample of bone tissue is scanned using a high resolution CT scanner and the 3D geometry of the bone tissue is reconstructed. Then, using the linear elasticity model and the finite element method, we arrive at linear systems to be solved, to determine whether or not the bone can withstand a certain load.

The smaller matrix corresponds to a tiny piece of bone with a geometry as shown in Figure 1.

The files contain the corresponding stiffness and mass matrices \( \mathbf{K} \) and \( \mathbf{M} \). Study the matrices \( \mathbf{K} \) by all means you know - structure, spectrum, condition number, scaling, ...

Create a right-hand-side (rhs) vector, containing zeros except for its first element, which should be taken as 1. Try to solve with the matrices \( \mathbf{K} \) and the above rhs (of suitable size) using

- a direct method

- the conjugate gradient method, provided in Matlab

\[
\text{[sol\_it,flag,relres,iter,resvec]} = \text{pcg}(\mathbf{K},\mathbf{rhs},1e-6,10000);
\]

**Note 1:** The file Bone\_matr\_127524.mat is large. The size of the matrices in this case is 127 524. This, however, is considered as very small. The 'bone' problems of practical interest are of size several hundreds of millions. The largest problem of this type, solved by now is over 500 000 000 degrees of freedom on a machine with over 4000 processors.
Used literature: