Norwegian University of Science
and Technology
Department of Mathematics

MA2501: Numerical Methods
Spring 2016

Exercise set 1

1 Solve the two linear systems

$$\begin{aligned} 11x_1 + 10x_2 + 14x_3 &= 1, \\ 12x_1 + 11x_2 - 13x_3 &= 1, \\ 14x_1 + 13x_2 - 66x_3 &= 1, \end{aligned} \qquad \text{and} \qquad \begin{aligned} 11x_1 + \quad 10x_2 + 14x_3 &= 1, \\ 12x_1 + 11.01x_2 - 13x_3 &= 1, \\ 14x_1 + \quad 13x_2 - 66x_3 &= 1. \end{aligned}$$

Also test what happens if the right hand side of the first equation is replaced by 1.001. Try to explain the results.

2 Consider the floating point system with 3 significant digits and 2 decimal exponents, i.e. numbers have the form $\pm d_1.d_2d_3 \times 10^{d_4d_5-49}$ with $d_i \in \{0, 1, 2, \ldots, 9\}$ for $i = 1, 2, 3, 4, 5$ and $d_1 \neq 0$. We assume no tricks so we can not represent zero.

a) Prove that two different set of digits lead to two different numbers, i.e. that each machine number has a unique representation.

b) What is

- the smallest positive machine number?
- the smallest machine number strictly greater than one?
- the unit roundoff error/machine epsilon?
- the biggest possible number?

3 **Cf. Cheney & Kincaid, Exercise 1.2.54.**

It is known that

$$\pi = 4 - 8\sum_{k=1}^{\infty} \frac{1}{16k^2 - 1}.$$

Thus, replacing the infinite sum by the finite sum

$$K_n = 4 - 8\sum_{k=1}^{n} \frac{1}{16k^2 - 1}$$

can be expected to give some approximation of π.

a) Estimate the size of the approximation error $E_n := |\pi - K_n|$ in dependence of the number of terms in the sum (assuming exact calculations).[1]

[1] Note that the approximation error can be very well estimated by a certain integral.

b) Assuming you compute K_n by the iteration $K_0 := 4$, $K_{k+1} := K_k - 8/(16k^2-1)$, provide an estimate of the quality of the best possible approximation of π when using double precision. Is it possible to improve the results with a different implementation of the same formula?

c) Verify your results using MATLAB.

4 Solve the following linear systems using Gaussian elimination without pivoting or report where the algorithm fails:

a)

$$\begin{aligned} x_1 - 5x_2 + x_3 &= 7, \\ 10x_1 + 20x_3 &= 6, \\ 5x_1 - x_3 &= 4. \end{aligned}$$

b)

$$\begin{aligned} x_1 + x_2 - x_3 &= 1, \\ x_1 + x_2 + 4x_3 &= 2, \\ 2x_1 - x_2 + 2x_3 &= 3. \end{aligned}$$

c)

$$\begin{aligned} 2x_1 - 3x_2 + 2x_3 &= 5, \\ -4x_1 + 2x_2 - 6x_3 &= 14, \\ 2x_1 + 2x_2 + 4x_3 &= 8. \end{aligned}$$

d)

$$\begin{aligned} x_2 + x_3 &= 6, \\ x_1 - 2x_2 - x_3 &= 4, \\ x_1 - x_2 + x_3 &= 5. \end{aligned}$$

5 **Cf. Cheney and Kincaid, Exercise 2.2.4.**

The *Hilbert matrix* of order n is the $n \times n$ matrix with entries

$$a_{ij} = \frac{1}{i+j-1} \qquad \text{for } 1 \le i, j \le n.$$

It is a classical example of an invertible but ill-conditioned matrix.

a) Write a MATLAB program that constructs, for given $n \in \mathbb{N}$, the Hilbert matrix of order n.

b) Define a vector $b \in \mathbb{R}^n$ setting $b_i = \sum_j a_{ij}$. Then the solution of the linear system $Ax = b$ is the vector x with entries $x_i = 1$. Does this also hold numerically in the case where A is the Hilbert matrix of some moderate order (say $2 \le n \le 15$)?

MATLAB-comments

A straightforward construction of the Hilbert matrices uses a double loop for filling up its entries one at a time. One important thing to remember about MATLAB is, however, that loops are usually extremely slow and thus should be avoided whenever (sensibly) possible. A typical strategy for doing so is the *vectorization* of operations: Instead of applying an operation to single elements, one applies it simultaneously to a whole vector or an array. In this example it is, for instance, easily possible to build

up the matrix one row (or column) at a time, thus reducing the double loop to a single loop.

Some further comments:

- In this example the speed increase through vectorization will not be noticeable. In future exercises, the situation might be different, though.
- Probably the fastest and simplest way of doing the implementation is by invoking the MATLAB-command `hilb`, which produces a Hilbert matrix. Don't do that.