MATH 561: Numerical Analysis I

Instructor: Prof. Wolfgang Bangerth
`bangerth@colostate.edu`

Homework assignment 1 – due 1/31/2017

Problem 1 (Continuous vs. discrete). Functions $f(x)$ are usually defined over an entire domain $x \in I = (a, b) \subset \mathbb{R}$ and – if interesting – take values in an image $f(I) \subset \mathbb{R}$. The domain and, typically also the image, are sets with infinitely many elements. On the other hand, computers can only represent numbers using a finite number of bits, most often as 32-bit (`float`, or `REAL*4`) or 64-bit (`double`, or `REAL*8`) IEEE floating point numbers, which store numbers in the form $\pm m2^e$, where $0 \leq m < 1$ is the mantissa

$$m = b_1 2^{-1} + b_2 2^{-2} + b_3 2^{-3} + \cdots + b_M 2^{-M} \tag{1}$$

and e is the exponent and has the form

$$e = \pm(u_0 2^0 + u_1 2^1 + u_2 2^2 + u_3 2^3 + \cdots + u_E 2^E). \tag{2}$$

The coefficients b_i, u_i are single-bit numbers, i.e., either 0 or 1. In the binary system, floating point numbers can therefore be written as $\pm 0.b_1 b_2 b_3 \ldots \times 2^{\pm u_E u_{E-1} u_{E-2} \ldots u_0}$. The total number of bits needed for the representation are M bits for the mantissa, $E + 1$ bits for the exponent, and 2 bits for the two signs.

Obviously, not all elements of I and $f(I)$ can be represented. Write a short program to find

a) an approximation to the smallest and largest positive numbers that can be represented in `float` and `double` precision;

b) a reasonably close approximation to the smallest `float` and `double` floating point number you can add to 1 such that the result is different from 1.

c) In exact arithmetic, the system of linear equations

$$\begin{aligned} x_1 + x_2 &= 2, \\ x_1 + 10^{20} x_2 &= 1 + 10^{20} \end{aligned}$$

has the solution $x_1 = x_2 = 1$. Are there corresponding (double precision) floating point numbers for x_1, x_2 that when plugged into the left hand side of the equations yields the exact values on the right hand side? If so, which? If not, is this a problem?

(20 points)

Problem 2 (Floating point vs real numbers). Let ε be the smallest floating point number in double precision such that in computer arithmetic $1+\varepsilon \neq 1$ (you determined ε in Problem 1b). What are the floating point values of $(1+\frac{\varepsilon}{2})-1$, $1+(\frac{\varepsilon}{2}-1)$, and $(1-1)+\frac{\varepsilon}{2}$? In what important way do exact and floating point arithmetic therefore differ? **(10 points)**

Problem 3 (Associativity of addition). In exact arithmetic, the partial sums

$$S_N = \sum_{k=1}^{N} \frac{1}{k}$$

diverge as $N \to \infty$. Write a program that keeps adding $1/k$ in single precision arithmetic (`float`, `REAL*4`) until the sum stays exactly the same. How can this happen?

As a second exercise, consider the following reformulation of the problem: in exact arithmetic, the order in which we add up the numbers $1/k$ does not matter. Check what happens if your program computes the partial sums in groups of 10 terms at a time as follows:

$$S_{10N} = \sum_{j=0}^{N-1} \left(\sum_{k=1}^{10} \frac{1}{10j+k} \right)$$

where the terms in parentheses are added up first, before they are added to the global sum. Perform the outer summation until the value of the sum does not change anymore. Compare the result to what you got previously. Explain.

(15 points)

Problem 4 (Taylor series). Derive the first four terms and integral remainder term of the Taylor series of

a) $f(x) = \sin x$ when expanded around $x_0 = 0$;

b) $f(x) = x \sin x$ when expanded around $x_0 = \pi/2$;

c) $f(x) = 4(x-3)^2(x+2)$ when expanded around $x_0 = 1$. What happened to the remainder term and what does this mean for the accuracy of the Taylor expansion with only four terms?

d) $f(x) = x^x$ when expanded around $x_0 = 1$. (Note: You will first have to figure out how to differentiate $f(x)$. Use the identity $a^b = e^{b \ln a}$.)

You may use a computer algebra system like Maple to compute derivatives of $f(x)$, but not to generate the entire Taylor series. **(10 points)**

Problem 5 (Taylor series). Many important functions such as the sine cannot be computed in a simple way, i.e. with only the four basic operations plus, minus, multiplication and division. However, they can be approximated with these operations.

a) Graph the first eight Taylor approximations of $f(x) = \sin x$ over the interval $[0, 2\pi]$ when expanded around zero, i.e.

$$\begin{aligned} f_1(x) &= f(0) + f'(0)x, \\ f_2(x) &= f(0) + f'(0)x + \tfrac{1}{2}f''(0)x^2, \\ f_3(x) &= f(0) + f'(0)x + \tfrac{1}{2}f''(0)x^2 + \tfrac{1}{3!}f'''(0)x^3, \end{aligned}$$

etc. What do you observe? What does this mean for the approximation of $f(2\pi)$?

b) How large is the maximal error (i.e., the maximum of the difference between $f_k(x)$ and $f(x) = \sin x$) of each of the approximations above on the interval $[0, 2\pi]$.

c) Write a program to experimentally determine the number of terms you need to approximate $f(2\pi) = 0$ to an accuracy of 10^{-4} and 10^{-12}.

(15 points)

Problem 6 (Gaussian elimination). Solve (on paper, showing the individual steps) the following system of linear equations using Gaussian elimination:

$$\begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}.$$

Verify that your result is correct.

(The matrix in this example is the so-called Hilbert matrix, with entries $H_{ij} = \frac{1}{i+j-1}$. It has a number of nasty properties that make it a good testcase for matrix algorithms, see `http://en.wikipedia.org/wiki/Hilbert_matrix`.)

(10 points)

Problem 7 (Gaussian elimination for matrix inversion). Write a computer function that takes a general $n \times n$ matrix A as input and computes its inverse A^{-1} as output. Implement the algorithm by hand, i.e., you shouldn't just call the Matlab function that computes the inverse.

Apply this function to compute, numerically, the inverse of the matrix of the previous problem. Do the same for Hilbert matrices of sizes 10×10, 100×100 and 200×200. In each case, verify numerically that $AA^{-1} = I$ where I is the identity matrix. What do you observe? **(15 points)**

Problem 8 (Gaussian elimination). Using Gaussian elimination, it is simple to solve the following problem

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

One would eliminate the occurrence of x_1 in the second equation by subtracting the first from the second equation, arriving at a diagonal matrix.

Describe what happens if the system instead looked like this:

$$\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix}.$$

This is of course the same system as before, we have just rotated the order of the three equations. Does the algorithm still work? If not, propose a remedy.

(5 points)