Prof. Dr. I. F. Sbalzarini
TU Dresden, 01187 Dresden, Germany

Solution 1

Release: 26.10.2020
Due: 02.11.2020

Question 1: Floating point representation

Consider a computer with floating point representation $(B = 2, l = 4, k = 2)$, where B : base, and l : number of digits of mantissa and k : length of exponent

a) What is the largest and smallest (normalized) positive number that can be represented?

$x_{max} = (0.1111)_2 \cdot 2^3 = 7.5$
$x_{min} = (0.10)_2 \cdot 2^{-3} = 0.0625$

b) How many numbers can be represented approximately ?

$(2^3) \cdot (2^2) \cdot (2^2) + 1 = 2^7 + 1$

c) Repeat analysis a) and b) for a calculator with floating point representation given by $(B = 10,\ l = 10,\ k = 2)$

$x_{max} = (0.9999999999)_{10} \cdot 10^{(9\cdot10^0+9\cdot10^1)} = 0.9999999999 \cdot 10^{99}$
$x_{min} = (0.1000000000)_{10} \cdot 10^{-(9\cdot10^0+9\cdot10^1)} = 0.1 \cdot 10^{-99}$

numbers representable $\approx 10^9 \cdot 10^2 \cdot 2^2 = 4 \cdot 10^{11} + 1$

Question 2: Machine precision

a) Write a code to determine the machine precision of MATLAB and compare the result with the output of the MATLAB command **eps**.

MATLAB **eps** $= 2.2204 \cdot 10^{-16}$

```
epsilon = 1.0;
while (1.0 + 0.5 * epsilon) != 1.0:
    epsilon = 0.5 * epsilon
```

b) What is the relation between the worst relative rounding error of a floating point representation (B, l, k), and the machine precision ? What floating point precision does MATLAB use, according to the **eps** command ?

Answer:

relative error $\leq \frac{B^{1-l}}{2}$, where l is the number of digits in the mantissa and B is the base.

For base $B = 2$ and $l = 52$, $\frac{B^{1-l}}{2} = 2.2204 \cdot 10^{-16}$
Since MATLAB uses **IEEE 754 - 2008**, **eps** would correspond to double precision (**binary64**)

Question 3: Condition number

a) Find the condition numbers of the following functions and also comment on the well and ill-conditioned regions

$$f_1(x) = \sin(x)$$
$$\kappa_H = \left|\frac{x f_1'(x)}{f_1(x)}\right| = |x\cot(x)|$$

For $x \approx n\pi, n \in \mathbb{Z}\backslash\{0\}$, we have high condition number.

$$f_2(x, y) = x - y$$
$$\kappa_{Hx} = \left|\frac{x\frac{\partial f_2(x,y)}{\partial x}}{f_2(x, y)}\right| = \left|\frac{x}{x-y}\right|, \kappa_{Hy} = \left|\frac{x\frac{\partial f_2(x,y)}{\partial y}}{f_2(x, y)}\right| = \left|\frac{y}{x-y}\right|$$

For x $\approx$ y, we have high condition number

$$f_3(x, y) = x \cdot y$$
$$\kappa_{Hx} = \left|\frac{x \cdot y}{x \cdot y}\right| = 1, \ \ \kappa_{Hy} = \left|\frac{x \cdot y}{x \cdot y}\right| = 1$$

b) Calculate the condition number of the Matrix C and verify your solution with MATLAB command **cond(C)** / Python's **numpy.linalg.cond(C)**

$$\mathrm{C} = \begin{bmatrix} 1 & 1 & 2 \\ -1 & 2 & 1 \\ 0 & -1 & 1 \end{bmatrix}$$

The eigen-values of $C^T C$ are 1,4,9. The Condition number $\kappa(C) = \frac{\sqrt{\lambda_{max}}}{\sqrt{\lambda_{min}}} = 3$

c) Given a function $f(x) = \sqrt{1+x} - 1$

i) Determine the condition number near $x = 0$ and see if the step-wise evaluation (1) $y = 1 + x$, (2) $z = \sqrt{y}$ and (3) $f(x) = z + 1$ is well conditioned and stable.

Condition number $\kappa_f(x) = \frac{\sqrt{1+x}+1}{2\sqrt{1+x}}$ so $\kappa_f(0) = 1$.
To compute this in a computer would need above three steps. Step (1) and (2) are well-conditioned (condition number 0 and 1/2), while Step (3) is ill-conditioned. Thus this algorithm is unstable.

ii) Can you restructure the problem to make it well conditioned in all evaluation steps ?

We can re-formulate the function as $f(x) = \frac{x}{\sqrt{1+x}+1}$ by multiplying the original function by $\frac{\sqrt{1+x}+1}{\sqrt{1+x}+1}$. This would result in an algorithm which is stable in all evaluation steps.

Question 4: Roundoff and Extinction

Calculate the sum

$$s := \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + ... - \frac{4}{19999} = \sum_{k=1}^{k=10000} \frac{4 \cdot (-1)^{k+1}}{2k-1}$$

in two ways in MATLAB

i) from right to left

ii) from left to right

Explain the difference from the 11th decimal place and compare the exact result π for 20 decimal points i.e.

$$\pi \approx 3.14159265358979323846$$

<u>Answer</u>:

In finite-precision arithmetic, addition is only commutative, but not associative and distributive. It is always best to sum numbers from small to large.