Introduction to Supercomputing

TMA4280 · Problem set 1

Exercise 1. Exercise 1.1 in the lecture notes. Consider the maximum and minimum numbers defined in single precision (i.e. using 32 bits):

$$V_{\max} = 1 \cdot 2^{254-127} \cdot 2 \approx 3.40 \cdot 10^{38} \tag{1}$$

$$V_{\min} = 1 \cdot 2^{1-127} \cdot 1 \quad \approx 1.17 \cdot 10^{-38} \tag{2}$$

How many digits should we include in each of these numbers when written out?

Exercise 2. Exercise 1.2 in the lecture notes. Find the binary floating point representation of the decimal number 4.25 in single precision.

Exercise 3. Exercise 1.3 in the lecture notes. How many decimal digits of accuracy does a double precision floating point number have?

Exercise 4. Exercise 1.4 in the lecture notes. An integer is typically represented using 32 bits. One bit is used to represent the sign. Hence, the possible integers will be in the range $\pm 2^{31} \approx \pm 2 \cdot 10^9$ If an algorithm includes a loop which needs to be done n times (e.g., a Monte Carlo simulation), n needs to be less than approximately 10^9. This may seem sufficient, however, there could be situations when this limitation could become an issue. Propose one or several ways around this limitation.

Exercise 5. Exercise 1.5 in the lecture notes. Let c be a scalar (a floating point number), let $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$ be vectors, each comprising n floating point numbers, and let $\boldsymbol{A}$ be an $n \times n$ matrix. How many floating point operations does it take to perform the following basic linear algebra operations?

$$\boldsymbol{z} = \boldsymbol{x} + c\boldsymbol{y}, \qquad \boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$$

Exercise 6. Exercise 1.6 in the lecture notes. Let $\boldsymbol{A}$ be an $n \times n$ matrix, and $\boldsymbol{x}$ and $\boldsymbol{b}$ be two vectors of length n. Assume that we want to solve the linear system of equations

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

using Gaussian elimination. Assume further that the matrix $\boldsymbol{A}$ is dense, meaning that we need to store all the n^2 entries in the matrix.

What is (approximately) the largest equation system we can solve (i.e., the largest number of n we can use) and still be able to fit the whole problem in the main memory, which we assume is 1 GB?

Exercise 7. In the lecture we found that adding a small number to a large number can cause problems when the relative difference between the numbers exceed the accuracy of the floating point representation.

With this in mind, suggest an algorithm for summing a list of numbers that is more accurate than doing it "naively".

Exercise 8. *Matrix-Vector product*. Implement a C/C++ program that computes $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$.

$$\boldsymbol{A} = \begin{pmatrix} 0.3 & 0.4 & 0.3 \\ 0.7 & 0.1 & 0.2 \\ 0.5 & 0.5 & 0.0 \end{pmatrix}, \qquad \boldsymbol{x} = \begin{pmatrix} 1.0 \\ 1.0 \\ 1.0 \end{pmatrix}.$$

Two versions possible.

Exercise 9. *Vector sum*. Implement a C/C++ program that creates three real vectors $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$, one real number α and computes $\boldsymbol{z} = \alpha\boldsymbol{x} + \boldsymbol{y}$ for given $\boldsymbol{x}$ and $\boldsymbol{y}$ (double-precision). First use a fixed-size of 10 then implement a version with dynamic size.

Exercise 10. *Dot product*. Implement a C/C++ program that creates two real vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N$ and one real number α and computes $\alpha = \sum_{i=1}^{N} \boldsymbol{x}_i \boldsymbol{y}_i$ for given $\boldsymbol{x}$ and $\boldsymbol{y}$ (double-precision).