Problem (Homework 1)

Compute matrix-vector product

$$\mathbf{V} = \mathbf{A}\mathbf{U}, \qquad \#$$

or

$$v_i = u_i + \sum_{j=1}^{N} u_j (x_i - x_j)^n, \quad j = 1, \ldots, N. \qquad \#$$

where

$$\mathbf{A} = \begin{pmatrix} 1 & (x_1 - x_2)^n & (x_1 - x_3)^n & \ldots & (x_1 - x_N)^n \\ (x_2 - x_1)^n & 1 & (x_2 - x_3)^n & \ldots & (x_2 - x_N)^n \\ (x_3 - x_1)^n & (x_3 - x_2)^n & 1 & \ldots & (x_3 - x_N)^n \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ (x_N - x_1)^n & (x_N - x_2)^n & (x_N - x_3)^n & \ldots & 1 \end{pmatrix}, \quad \mathbf{U} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \ldots \\ u_N \end{pmatrix}, \quad \mathbf{V} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ \ldots \\ v_N \end{pmatrix}, \quad \#$$

and $x_1, \ldots, x_N$, $u_1, \ldots, u_N$, are given. The matrix size, $N > 0$, and the power, $n > 0$, are a given (fixed) positive integers.

Homework 1

1. Derive a formula that can be used for the "Fast" ($O(N)$) method at arbitrary n.
2. Write a code that implements both straightforward computation based on Eq. (ref: 1.1) and the "Fast" method.
3. Check that both codes produce the same (with the machine precision) results. Provide a graph of the absolute maximum error between the straightforward and "Fast" method for for $n = 5$ and N varying between 10^2 and 10^3.
4. Provide a graph that compares the CPU time required by the straightforward and the "Fast" method for $n = 5$ and N varying between 10^2 and 10^3 for straightforward and N varying between 10^2 and 10^4 for the "Fast" method.
5. Make a conclusion about the complexity of each method.

Hints (Homework 1)

1. Use the Newton binomial theorem

$$(a+b)^n = a^n + \begin{pmatrix} n \\ 1 \end{pmatrix} a^{n-1} b + \begin{pmatrix} n \\ 2 \end{pmatrix} a^{n-2} b^2 + \ldots + b^n = \sum_{k=0}^{n} \begin{pmatrix} n \\ k \end{pmatrix} a^{n-k} b^k, \qquad \#$$

where the binomial coefficients are

$$\begin{pmatrix} n \\ k \end{pmatrix} = \frac{n(n-1)\ldots(n-k+1)}{1 \cdot 2 \cdot \ldots \cdot k} = \frac{n!}{(n-k)!k!}. \quad \#$$

2. Use the language of your preference (we recommend Matlab, since it would be easy to plot the output results). Do not use libraries or standard routines for straightforward computations, but do straightforward summation exactly according Eq. (ref: 1.1). Be sure that you can vary input parameters $x_1, \ldots, x_N,\ u_1, \ldots, u_N$.

3. Take some N (e.g. $N = 10$), and random $x_1, \ldots, x_N,\ u_1, \ldots, u_N \in [0,1]$. Try for a few n. Check the difference between vectors **V** computed with the two codes and the same N, n, and $x_1, \ldots, x_N,\ u_1, \ldots, u_N$ .When the codes will produce almost the same results plot the required graph of error. The maximum absolute error is defined as

$$error = \max_{i=1,\ldots,N} \left| v_i^{straightforward} - v_i^{fast} \right|. \quad \#$$

4. For CPU time measurement use standard timer functions that allow to find the time before the routine started and after it finished, and take the time difference. In Matlab the CPU time measurements can be done with matlab function cputime (see Matlab help). Vary N in logarithmic scale. As soon as the codes are tested for consistency, you can use arbitrary $x_1, \ldots, x_N,\ u_1, \ldots, u_N \in [0,1]$ for the CPU time measurements. Provide the graph CPU time vs N in the logarithmic coordinates.

5. In logarithmic coordinates plot lines corresponding to linear and quadratic dependences of the CPU time on N and compare with your computational results.