

**CMSC878R/AMSC698R****Homework 1**

This is similar to the simple example given in class. The goal is to develop fast algorithms for the following matrix-vector product

$$\mathbf{v} = \mathbf{A}\mathbf{u}, \quad (1)$$

where

$$v_i = \sum_{j=1}^N u_j \cos^n(x_i - x_j), \quad j = 1, \dots, N, \quad \{x_i\} \in [0, 2\pi) \quad (2)$$

and

$$\mathbf{A} = \begin{pmatrix} 1 & \cos^n(x_1 - x_2) & \dots & \cos^n(x_1 - x_N) \\ \cos^n(x_2 - x_1) & 1 & \dots & \cos^n(x_2 - x_N) \\ \dots & \dots & \dots & \dots \\ \cos^n(x_N - x_1) & \cos^n(x_N - x_2) & \dots & 1 \end{pmatrix}, \quad \mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \\ \dots \\ u_N \end{pmatrix}, \quad \mathbf{v} = \begin{pmatrix} v_1 \\ v_2 \\ \dots \\ v_N \end{pmatrix}. \quad (3)$$

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

1. Derive a formula that can be used for developing a “fast” ( $O(N)$ ) method for arbitrary  $n$ .
2. Write a program that implements both straightforward computation based on Eq. (2) and the “Fast” method. Use Matlab. Be sure that you can vary  $x_1, \dots, x_N$ , and  $u_1, \dots, u_N$ .
3. Check that both methods produce the same results (within machine precision). Plot the absolute maximum error between the straightforward and the “Fast” method for  $n = 5$  and  $N$  varying between  $10^2$  and  $10^3$ .
4. Plot a comparison of the CPU time required by the straightforward and the “Fast” methods for  $n = 5$  and  $N$  varying between  $10^2$  and  $10^3$  for the straightforward method, and with  $N$  varying between  $10^2$  and  $10^4$  for the “Fast” method. Make the plots log-log. Also plot lines corresponding to linear and quadratic dependences of the CPU time on  $N$  and compare with your computational results.
5. Make a conclusion about the asymptotic complexity of each method.

**Hints**

1. Use the trigonometric identity

$$\cos(x_i - x_j) = \cos x_i \cos x_j + \sin x_i \sin x_j. \quad (4)$$

2. Combine with the Newton binomial theorem

$$(a + b)^n = a^n + \binom{n}{1} a^{n-1} b + \binom{n}{2} a^{n-2} b^2 + \dots + b^n = \sum_{k=0}^n \binom{n}{k} a^{n-k} b^k, \quad (5)$$

where the binomial coefficients are

$$\binom{n}{k} = \frac{n(n-1)\dots(n-k+1)}{1 \cdot 2 \cdot \dots \cdot k} = \frac{n!}{(n-k)!k!}. \quad (6)$$

3. For CPU time measurement use the Matlab function `cputime` (see Matlab help).

# 1 Solution

If  $a = \cos x_i \cos x_j$  and  $b = \sin x_i \sin x_j$  then

$$a^{n-k} b^k = (\cos^{n-k} x_i \sin^k x_i) (\cos^{n-k} x_j \sin^k x_j) = f_k^{(n)}(x_i) f_k^{(n)}(x_j), \quad (7)$$

where we introduced function

$$f_k^{(n)}(x) = \cos^{n-k} x \sin^k x, \quad (8)$$

which can be implemented as a separate supplemental function.

So from Eqs. (4) and (5) we have

$$\cos^n(x_i - x_j) = \sum_{k=0}^n \binom{n}{k} f_k^{(n)}(x_i) f_k^{(n)}(x_j). \quad (9)$$

Now substitute this into sum (2) and change the order of summation:

$$\begin{aligned} v_i &= \sum_{j=1}^N u_j \cos^n(x_i - x_j) = \sum_{j=1}^N u_j \sum_{k=0}^n \binom{n}{k} f_k^{(n)}(x_i) f_k^{(n)}(x_j) \\ &= \sum_{k=0}^n \binom{n}{k} f_k^{(n)}(x_i) \sum_{j=1}^N u_j f_k^{(n)}(x_j). \end{aligned} \quad (10)$$

Hence, the algorithm for fast summation is the following:

**Step 1** Compute and store  $n + 1$  sums

$$c_k^{(n)} = \binom{n}{k} \sum_{j=1}^N u_j f_k^{(n)}(x_j), \quad k = 0, \dots, n. \quad (11)$$

**Step 2** Compute  $N$  sums

$$v_i = \sum_{k=0}^n c_k^{(n)} f_k^{(n)}(x_i), \quad i = 1, \dots, N. \quad (12)$$

Results of computation are shown in figures below

## 1.0.1 Conclusion:

Comparing measured cpu time with linear and quadratic dependences in log-log coordinates we can see that the complexity of the direct method is close to theoretical complexity  $O(N^2)$ , while the fast method at large  $N$  is close to  $O(N)$  theoretical complexity.

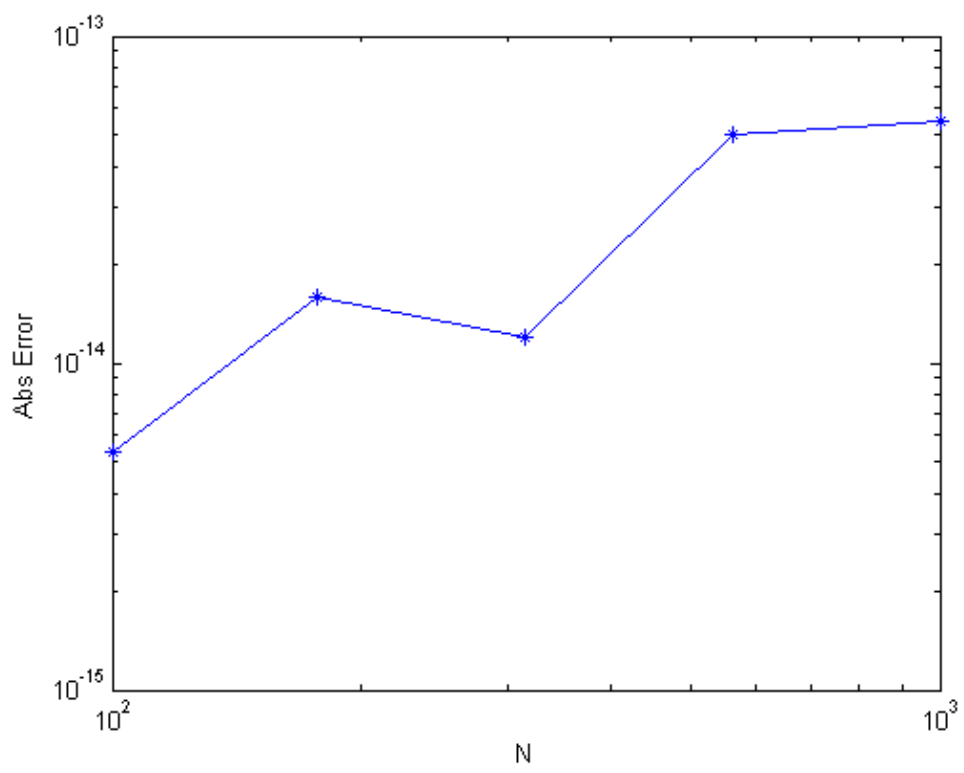


Figure 1: Absolute error between the direct and fast methods.

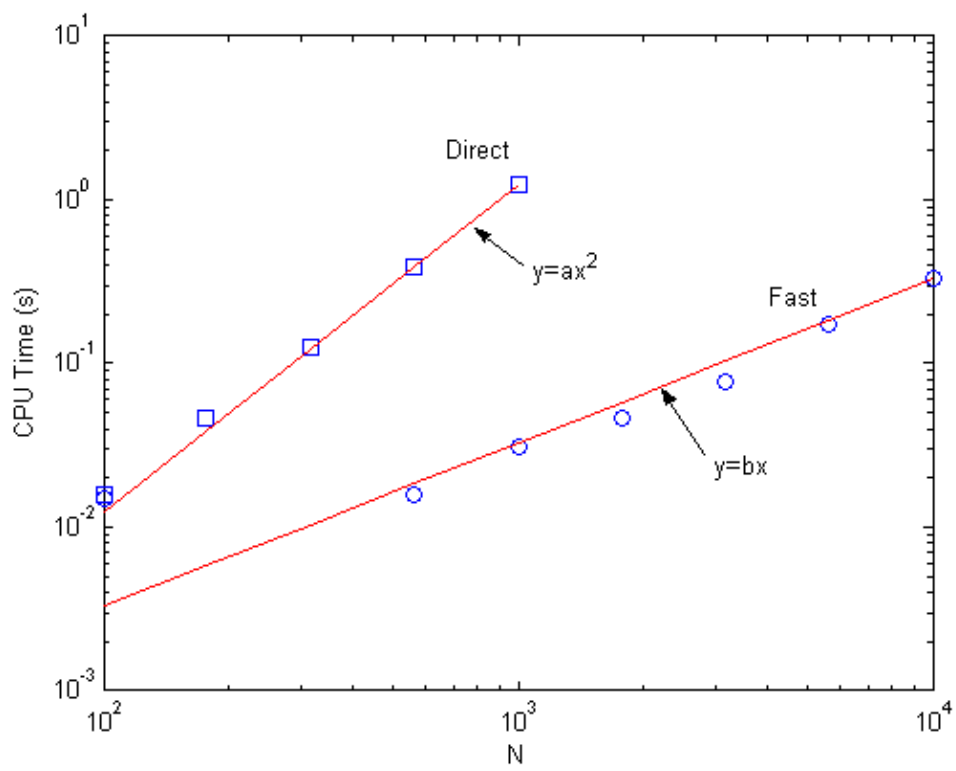


Figure 2: CPU time for direct (squares) and fast summation method using factorization (circles). The red lines show the quadratic and linear functions in log-log coordinates.