1 Problem (Homework 3/4)

Compute the matrix-vector product

$$\mathbf{v} = \mathbf{\Phi u}, \tag{1}$$

or the sum

$$v_j = \sum_{i=1}^{N} \Phi_{ji} u_i, \quad j = 1, ..., M, \tag{2}$$

with absolute error $\epsilon < 10^{-6}$, where

$$\mathbf{\Phi} = \begin{pmatrix} \Phi_{11} & \Phi_{12} & ... & \Phi_{1N} \\ \Phi_{21} & \Phi_{22} & ... & \Phi_{2N} \\ ... & ... & ... & ... \\ \Phi_{M1} & \Phi_{M2} & ... & \Phi_{MN} \end{pmatrix}, \quad \mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ ... \\ u_N \end{pmatrix}, \quad \mathbf{v} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ ... \\ v_M \end{pmatrix}, \tag{3}$$

$$\Phi_{ji} = \frac{1}{y_j - x_i}, \quad i = 1, ..., N, \quad j = 1, ..., M.$$

and $x_1, ..., x_N$ are random points uniformly distributed on [0,1], $M = N - 1$, and each y_j is located between the closest x_i's on each side, $j = 1, ..., N - 1$ using optimized version of Pre-FMM that uses R-expansions near the centers of the target boxes.

Homework 3 (Developing supporting programs and theoretical estimations)

1. Create a $O(N \log N)$ method to create the data structure for use with the Pre-FMM, which employs space partitioning with K boxes (segments). It is suggested to use a bookmarking method (see Appendix). Other methods of bookkeeping are OK if they have complexity $O(N \log N)$). Test your program to be sure that bookkeeping works properly.
2. Read the tasks for Homework 4. Draw a sketch of the Pre-FMM algorithm, assuming that you need to perform tasks of Homework 4. Identify supporting programs that you will need to use.
3. Write supporting programs, which allow you to determine necessary indeces for the sources and targets for arbitrary N and K and box centers.
4. Evaluate the truncation number, $p(K, N)$, that should provide the specified accuracy as a function of the number of boxes K and the size of the problem, N. N will vary in the tests between 10^2 and 10^4 and you can use some simplifications and find some p which should be sufficient to provide the necessary accuracy for this range.
5. Evaluate theoretically the optimal number of boxes $K_{opt}(N)$ based on the obtained evaluations of p for specified accuracy.
6. Write a program which provides you the local R-expansion coefficients for a given target box (or target box center) and a source. Verify its accuracy by evaluating the function directly and via the expansion.

Homework 4 (Developing the main routine, optimization, and tests)

1. Using Lecture #6 write a program that implements both straightforward multiplication based on Eq. (2) and Pre-FMM that uses local R-expansions.
2. Provide a graph of the absolute maximum error between the straightforward method and the Pre-FMM for $N = 10^3$, K varying between 10 and 100, and p from your theoretical evaluations. Compare the results with your evaluations of the accuracy. You may find that practically the theoretical p can be substantially reduced to stay within the specified error bounds. In this case you may (or may not) reduce p and use experimental values to proceed further.

3. Provide a dependence of the CPU time required by the Pre-FMM as a function of K for $N = 10^3$ ($10 < K < 100$). Determine K_{opt} experimentally and compare with the theoretical evaluations (use actual p). Scale $K_{opt}(N)$ for computations with varying N. Plot your scaled function $K_{opt}(N)$.

4. Provide a graph of actual error (between the standard and the fast method with $K = K_{opt}(N)$) for N varying between 10^2 and 10^3 and the truncation number used.

5. Provide a graph that compares the CPU time required by the straightforward method and the Pre-FMM for N varying between 10^2 and 10^3 for straightforward and N varying between 10^2 and 10^4 for the optimized Pre-FMM. Compare results with theoretical complexities of the algorithms.

6. Find the "break-even" point (i.e. N at which the "Fast" method requires the same CPU time as the straightforward method) for your implementation.

1.1 Appendix: One of the ways to make a data structure for 1D-FMM

We have two sets of points $\mathbb{X}$ and $\mathbb{Y}$ (sources and evaluation points). Assume that these arrays are sorted. (Use the Matlab *sort* function to sort $\mathbb{X}$ after it is generated and find $\mathbb{Y}$ as points between sources, so $\mathbb{Y}$ is also automatically sorted). You can then use "*bookmarks*" to have fast determination of all sources or evaluation points, belonging to a particular box.

The idea is the following: Because $\mathbb{X}$ is ordered you can create arrays $BookmarkLeft$ and $BookmarkRight$ (in the current problem you may find that one array is sufficient, but you may use two, for easier and faster search). These arrays contain bookmarks which show the bounds of $\mathbb{X}$ indices as shown in the table below. To generate such a bookmark table you need only $O(N)$ operations (one pass through your ordered data) and it takes memory of order $O(K)$.

BoxIndex	1	2	...	k	...	K
Set	$x_{n_0},...,x_{n_1},$	$x_{n_1+1},...,x_{n_2},$	...	$x_{n_{k-1}+1},...,x_{n_k},$	...	$x_{n_{K-1}+1},...,x_{n_K},$
BookmarkLeft	n_0	n_1+1	...	$n_{k-1}+1$	...	$n_{K-1}+1$
BookmarkRight	n_1	n_2	...	n_k	...	n_K

If it appears that the Box with index k is empty you may put $BookmarkLeft(k)$ and $BookmarkRight(k)$ equal to zero. You also may create an array of $NonEmptyBoxes$ where you store consequently the non-empty box indexes k only, so it looks like $\{k_1,...,k_L\}$, where $k_1,...,k_L$ are indices of the nonempty-boxes. In this case you will skip possible empty boxes and your program will be faster (if such boxes exist). A similar bookmark table should be also created for $\mathbb{Y}$ and non-empty evaluation box indexes can be stored in an array.

Write a Matlab function $SetDataStructure(Set, K)$,which returns you arrays $BookmarkLeft$, $BookmarkRight$, and $NonEmptyBoxes$.

Call this function twice with $Set = \mathbb{X}$ and $Set = \mathbb{Y}$.

Then the procedure of going over non-empty boxes with respect to evaluation points looks like

```
$for \quad k = EvaluationonNonEmptyBoxes$
    $for\ j = EvaluationBookmarkLeft(k) : EvaluationBookmarkRight(k)$
        $y = Y(j)$; % get the evaluation point coordinate in this box;
        [Required operations with $y$];
    $end$;
$end$;
```

Based on this idea you can also create your program which for given Evaluation Box runs through appropriate source boxes.