Chapter 2

A Brief Introduction to the Fast Multipole Method

In many applications, one needs to evaluate a sequence of sums in the following form:

\[(2.0.1)\]
\[v(x_i) = \sum_{j=1, j \neq i}^{N} u_j K(x_i, x_j), \quad i = 1 : N.\]

A direct evaluation obviously requires \(O(N^2)\) operations. The fast multipole method is a method that can calculate these sums with \(O(N \log N)\) operations. We will illustrate the main idea of this type of method by a simple example.

Denote:

\[K = (K(x_i, x_j)) \in R^{N \times N}, \quad u = (u_i) \in R^N, \quad v = (v(x_j)) \in R^N.\]

The sums in (2.0.1) are equivalent to

\[v = K u\]

namely \(v\) is the multiplication of the dense matrix \(K\) with the vector \(u\).

### 2.1 A special case

The summation problem (2.0.1) requires only \(O(pN)\) operations if \(K\) is a sum of a few \(p\) of separable functions, namely

\[(2.1.2)\]
\[K(x, y) = \sum_{k=1}^{p} f_k(x)g_k(y).\]
In this case, the matrix $K$ is of rank, at most, $p$, namely

$$K = \sum_{k=1}^{p} a_k b_k^T$$

where

$$a_k = (f(x_i)), \quad b_k = (g(x_i)).$$

It follows that

$$v = K u = \sum_{k=1}^{p} a_k (b_k^T u)$$

which obviously only requires $O(pN)$ operations.

The main idea of the fast multipole method, simply speaking, is to transform a general kernel function $K(x, y)$ into sums of separable functions.

### 2.2 Illustration by an example

In this section, we will illustrate the main idea behind the fast multipole method using a special example. We consider the following very special kernel function

$$(2.2.3) \quad K(x, y) = \frac{1}{(x - y)^2}$$

Let us try to evaluate the sums in (2.0.1) with the above defined $K$ for $x_i \in [0, 1], i = 1 : N$.

One main task is to somehow separate the variables $x$ and $y$ in the form of (2.1.2). The technique we will use here is the Taylor expansion.

**Lemma 2.2.4** Given $\delta \in (0, 1)$ and $z \in [0, 1)$, then for any $x, y \in [0, 1)$ satisfying

$$|x - z| \leq \delta,$$  

we have

$$\frac{1}{(x - y)^2} = K_p(x, y) + R_p,$$

where

$$K_p(x, y) = \sum_{k=0}^{p-1} (k + 1) \frac{(x - z)^k}{(y - z)^{k+2}}$$

$$|R_p| \leq (p + 1)\delta^p.$$
Proof.

\[
\frac{1}{(x-y)^2} = \frac{1}{(y-z-(x-z))^2} = \frac{1}{(y-z)^2} \left(1 - \frac{x-z}{y-z}\right)^2.
\]

The desired result then follows by the well-known Taylor expansion

\[
\frac{1}{(1-t)^2} = \sum_{k=0}^{p-1} (k+1)t^k + R_p, \quad \text{for } |t| \leq \delta < 1
\]

where the remainder \( R_p \) satisfies (2.2.7).

Lemma 2.2.4 means that the variables \( x \) and \( y \) in the kernal function \( K(x, y) \) can be separated as long as \( x \) and \( y \) are far apart, namely satisfying (2.2.5) for some \( z \in [0,1) \). Let us illustrate this with two simple cases.

1. For \( y \in [0, 1/4) \), (2.2.5) is satisfied with
   (a) \( x \in [1/2, 1) \) with \( z = 3/4 \) and \( \delta = 1/2 \).

2. For \( y \in [0, 1/8) \), then (2.2.5) is satisfied with
   (a) \( x \in [1/4, 1/2) \) with \( z = 3/8 \) and \( \delta = 1/2 \), and
   (b) \( x \in [1/2, 1) \) with \( z = 3/4 \) and \( \delta = 2/5 < 1/2 \).

With the above two examples and their obvious generalizations, we have the following important observations.

1. If we divide \([0,1)\) into 4 (case 1 above) or 8 (case 2) (equal) subintervals, then variables \( x \) and \( y \) can be separated as long as they do not belong to the same or neighboring subintervals.

2. Given a subinterval for \( y \), say \([0, 1/8)\), the size of separable subintervals for \( x \) can get larger when it gets further away from \( y \).

Therefore, if \([0,1)\) is divided into, say, dyadic subintervals, a separation of variable for the function \( K(x, y) \) can be realized for all subintervals except when \( x \) and \( y \) belonging to the same or immediate neighboring subintervals. But if we have sufficiently many subintervals so that each subinterval only contains a few points \( x_i \), then a direct computation on each interval together with its neighbors is inexpensive. Consequently, the overall computational complexity of the resulting algorithm is considerably smaller than \( O(N^2) \). (It can be estimated that is at most of order \( O(pN \log N) \) while \( p \), according to (2.2.7), is of order \( O(|\log \epsilon|) \) for a desired accuracy \( \epsilon \)).