

FAST MULTIPOLE METHOD BASED FILTERING OF NON-UNIFORMLY SAMPLED DATA

Nail A. Gumerov, Ramani Duraiswami

Perceptual Interfaces & Reality Laboratory, UMIACS, University of Maryland, College Park

ABSTRACT

Non-uniform fast Fourier Transform (NUFFT) and inverse NUFFT (INUFFT) algorithms, based on the Fast Multipole Method (FMM) are developed and tested. Our algorithms are based on a novel factorization of the FFT kernel, and are implemented with attention to data structures and error analysis.

1. INTRODUCTION

Most signal processing involves the manipulation of various transforms of discretely sampled data, especially of Fourier transforms. Direct computation of a transform of N data samples requires $O(N^2)$ operations. For band-limited data that is sampled at equispaced locations, the rediscovery in 1965 by [1] of an $O(N \log N)$ algorithm, the fast Fourier transform, lead to a revolution. It is difficult to overstate its impact on science and industry.

Unfortunately, many applications require transforms between nonuniformly sampled data, and recently there has been much interest in this area, especially for various inverse scattering calculations and as a component of the computation of other transforms, e.g., on the sphere. The goal is to achieve fast ($o(N^2)$) bandwidth-preserving transforms of data. Due to reasons of space we do not discuss these here, and refer the reader to a recent review of NUFFTs [3]. We focus instead on extensions of the work of Dutt & Rokhlin [2], who used the Fast Multipole Method (FMM) to achieve a NUFFT of asymptotic complexity $O(N(\log N - \log \epsilon))$, where ϵ is the specified error.

Here, we too apply the FMM to obtain NUFFTs and INUFFTs. The major difference is in the factorization we used for the kernel function, and our efficient implementation using data structures, supported by tests for $N \simeq 10^6$. We show that for machine precision, the complexity of the FMM-based NUFFT and INUFFT is $O(N)$. We present theoretical and computational analyses of error bounds and optimal settings for the FMM.

2. FORMULATION & SOLUTION

We consider the following two problems: i) a 2π -periodic complex-valued band-limited function $f(x)$ is sampled at N equispaced points, $x_k = 2\pi k/N$, $k = 0, \dots, N-1$, so $f_k = f(x_k)$; find $g_j = f(y_j)$, $j = 0, \dots, M-1$, where $y_j \in [0, 2\pi)$ are non-equispaced points with specified accuracy ϵ ; ii) the same function is given at M non-uniform samples g_j , $j = 0, \dots, M-1$ on a non-uniform grid $\{y_j\}$; determine f_k , $k = 0, \dots, N-1$, on a uniform grid $\{x_k\}$. N determines the *bandwidth* of f , since

$$f(x) = \sum_{n=0}^{N-1} c_n e^{inx}, \quad c_n = \frac{1}{N} \sum_{k=0}^{N-1} f_k e^{-inx_k}, \quad (1)$$

We gratefully acknowledge support of NSF awards 0219681 and 0086075.

where c_n are the Fourier coefficients. The forward DFT is the transform $\{f_k\} \rightarrow \{c_n\}$, and the IDFT is transform $\{c_n\} \rightarrow \{f_k\}$. For $N = 2^L$, the transforms $\{f_k\} \rightleftharpoons \{c_n\}$ can be done in $O(N \log N)$ operations using the FFT. Therefore, if interpolation problems $\{f_k\} \rightleftharpoons \{g_j\}$ can be solved with the same efficiency, NUFFT and INUFFT, which are transforms $\{c_n\} \rightleftharpoons \{g_j\}$ will be available. Eq. (1) can be written as

$$\{g_j\} = \{K_{jk}\} \{f_k\}, \quad K_{jk} = \frac{1}{N} \sum_{n=0}^{N-1} e^{-inx_k} e^{iny_j}, \quad (2)$$

where $j = 0, \dots, M-1$; $k = 0, \dots, N-1$. It relates the vectors $\{g_j\}$ and $\{f_k\}$ via the sampled kernel matrix $\{K_{jk}\}$. The sum (2) is a geometric progression, and can be simplified as

$$K_{jk} = \frac{e^{iNy_j} - 1}{e^{i(y-x_k)} - 1} = F_j G(y_j - x_k), \quad (3a)$$

$$F_j = \frac{e^{iNy_j} - 1}{N}, \quad G(t) = \frac{1}{e^{it} - 1} = -\frac{1 + i \cot(t/2)}{2}. \quad (3b)$$

This decomposition was derived in [2]. It can be thought of as a modulation of a fast oscillating function $F(y)$ by a slowly changing amplitude function $G(t)$ amplitude. Evaluation of $F(y_j)$ costs $O(M)$, while evaluation of $G(t)$ can be done approximately by fast methods, such as the FMM. The transform $\{g_j\} \rightarrow \{f_k\}$ can be reduced similarly. It leads to

$$\{f_k\} = \{L_{kj}\} \{g_j\}, \quad L_{kj} = C_k G(x_k - y_j) D_j, \quad (4)$$

where $j = 0, \dots, M-1$; $k = 0, \dots, N-1$. The kernel L is decomposed as in [2], where $G(t)$ is given by Eq. (3b); and C_k and D_j are coefficients depending on $\{x_k\}$ and $\{y_j\}$

$$C_k = (-1)^k \prod_{j=0}^{N-1} \sin \frac{x_k - y_j}{2}, \quad D_j = \frac{2ie^{-iNy_j/2}}{\prod_{\substack{k=0 \\ k \neq j}}^{N-1} \sin \frac{y_j - y_k}{2}}. \quad (5)$$

These coefficients can be precomputed using the FMM. So both the forward and inverse problems can be reduced to multiplication of a matrix generated by kernel $G(t)$ by some input vector. This is the problem which is in the focus of the present paper.

2.1. Multilevel FMM

For computation of the matrix-vector product $\{G(y_j - x_k)\} \{f_k\}$ we use the Multilevel FMM (MLFMM), a description of which can be found elsewhere [6, 4]. Note that $\{x_k\}$ or $\{y_j\}$ need not be equispaced for the FMM. Both (2) and (4) use the same algorithm with source and target sets $\{x_k\}$ & $\{y_j\}$ exchanged.

Space Partitioning: The kernel $\cot \frac{y_j - x_k}{2}$ is a 2π -periodic function. We can thus make transforms $\tilde{x}_k = x_k + 2\pi n$, $n = 0, \pm 1, \dots$, which do not change the function, and keep $y_j - \tilde{x}_k$ in

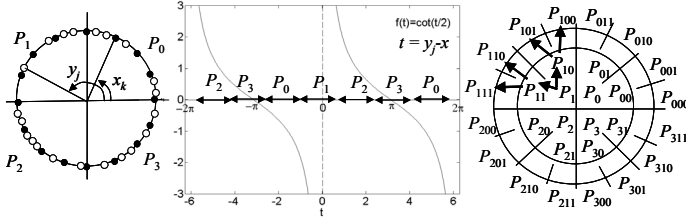


Fig. 1. On the left locations of the source, $\{x_k\}$, and target, $\{y_j\}$, points on the unit circle are shown. The circle is subdivided at level 2 into 4 arcs. In the center the behavior of the kernel function $\cot(t/2)$ is shown. At the right the hierarchical space partitioning employed in the MLFMM is illustrated. Arrows show the hierarchical parent-children relations in the binary tree.

$-\pi \leq y_j - \tilde{x}_k \leq \pi$; , i.e., y_j and x_k are points on a unit circle. Then x_k and \tilde{x}_k are identical (see Fig. 1).

In the case of circular topology, hierarchical space partitioning with a binary tree with l_{\max} levels, leads to a division of the circle by 2^l equal arcs. In this hierarchy, parent-children relations are defined as in a binary tree, but the concept of neighbors and neighborhoods is modified. At level 2 we subdivide $[0, 2\pi)$ into 4 pieces: $P_0 : [0, \pi/2)$, $P_1 : [\pi/2, \pi)$, $P_2 : [\pi, 3\pi/2)$, $P_3 : [3\pi/2, 2\pi)$, where P_2 and P_0 are neighbors of P_3 and they together form a neighborhood of P_3 , which we denote as $\Omega_1(P_3)$. We also denote P_1 as $\Omega_2(P_3)$, i.e. the piece outside the neighborhood of P_3 .

Kernel Expansions: Level $l = 2$ requires special consideration. Here we separate the sum over all the sources as

$$h(y_j) = \sum_{x_k \in \Omega_1(P_n)} f_k \cot \frac{y_j - x_k}{2} + \sum_{x_k \in \Omega_2(P_n)} f_k \cot \frac{y_j - x_k}{2}, \quad (6)$$

$y_j \in P_n$; $n = 0, \dots, 3$. The first sum can be computed using:

$$\cot \frac{t}{2} = \frac{2}{t} - 2 \sum_{m=1}^{\infty} \frac{|B_{2m}|}{(2m)!} t^{2m-1}, \quad |t| < 2\pi, \quad (7)$$

where B_m are the Bernoulli numbers [5]. When applied here, we note that for $x_k \in \Omega_1(P_n)$ we have $-\pi \leq y_j - \tilde{x}_k \leq \pi$, so for $t = y_j - \tilde{x}_k$ we have $|t| \leq \pi$. This provides fast convergence in Eq. (7), which otherwise blows up as $y_j - x_k \rightarrow \pm 2\pi$.

For the second sum, we introduce $t = y_j - \hat{x}_k$, where $\hat{x}_k = x_k \pm \pi$ is a point opposite x_k . We have

$$\cot \frac{y_j - x_k}{2} = -\tan \frac{t}{2} = -2 \sum_{m=1}^{\infty} \frac{(2^{2m} - 1) |B_{2m}|}{(2m)!} t^{2m-1}, \quad (8)$$

$|t| < \pi$. Note that as $t \rightarrow \pm\pi$ the series blows up. However, since $x_k \in \Omega_2(P_n)$ we have $\hat{x}_k \in P_n$. Due to both y_j and \hat{x}_k belong to the same P_n , whose length is $\pi/2$, we have $-\pi/2 \leq y_j - \hat{x}_k \leq \pi/2$, or $|t| \leq \pi/2$. This provides a fast convergence of the sum.

There are thus two parts to the kernel function $\cot(t/2)$: singular $(2/t)$, and regular. Fast summation with the singular kernel can be performed using the MLFMM, while the regular part can be truncated and factorized for all points in the respective neighborhoods at level 2. The MLFMM for kernel $1/t$ has been well studied, optimized, and error bounds are established [4]. We just modified our software in [4] with a circular data structure.

Truncation Errors: Consider the truncation error for sum (7) when q first terms of the expansion are used to approximate the infinite sum. We note that the Bernoulli numbers satisfy [5]

$$|B_{2m}| < \frac{2(2m)!}{(2\pi)^{2m}} \frac{1}{1 - 2^{1-2m}}, \quad m = 1, 2, \dots \quad (9)$$

This shows that the truncated part of the series can be majorated by the geometric progression. Using $|t| \leq \pi$, the error bound is

$$|\epsilon_q^{(1)}| = \left| \sum_{m=q+1}^{\infty} \frac{|B_{2m}| t^{2m-1}}{(2m)!} \right| < \frac{2^{1-2q}}{3\pi(1 - 2^{-2q-1})} = \epsilon_q. \quad (10)$$

The truncation error, $\epsilon_q^{(2)}$, in (8) can be bounded similarly, using $|t| \leq \pi/2$: $|\epsilon_q^{(2)}| < 2\epsilon_q$. The estimates are for a single source-evaluation pair. For $3N/4$ sources located in the neighborhood of point y_j , and $N/4$ outside it, the total maximum absolute error in computation of g_j (with $|f_k| \leq 1$) is

$$|\epsilon_q^{reg}| = \left| \sum_{k=0}^{N-1} (K_{jk} - K_{jk}^{(q)}) f_k \right| < \frac{2}{N} \left(\frac{3N\epsilon_q}{4} + \frac{2N\epsilon_q}{4} \right) = \frac{5\epsilon_q}{2}.$$

The error of the MLFMM with kernel $1/t$ has been estimated in [4]. This result applied to our case in a computational domain of size $3\pi/2$, and for $3N/4$ sources is

$$|\epsilon_p^{MLFMM}| < \frac{5}{\pi} \cdot 2^{l_{\max}} 3^{-p}, \quad (11)$$

where l_{\max} is the maximum level of space subdivision and p is the truncation number used in the MLFMM. Therefore the total truncation error for the present method can be estimated as

$$\epsilon \lesssim \frac{5}{3\pi} \left(4^{-q} + 2^{l_{\max}} 3^{1-p} \right). \quad (12)$$

By requiring that the error in the singular and regular parts be the same in (12) we relate p and q , and obtain a combined bound as

$$q \gtrsim \frac{1}{2} \log_2 \frac{3\pi}{10\epsilon}, \quad p \gtrsim \log_3 \frac{3\pi}{10\epsilon} + \frac{l_{\max}}{\log_2 3} + 1. \quad (13)$$

Complexity and Optimizations: For fast computations of the products involving q -truncated sums (7) and (8) we factorize powers t^{2m-1} using the Newton binomial expansion near the center, $x_c^{(n)}$, of the segment P_n containing y_j . Substituting this expansion into Eqs (7) and (8) and further into Eq. (6), we obtain after changing the order of summation:

$$h(y_j) = 2 \sum_{x_k \in \Omega_1(P_n)} \frac{f_k}{y_j - \tilde{x}_k} - \sum_{l=0}^{2q-1} \frac{d_l}{l!} (y_j - x_c^{(n)})^l, \quad (14)$$

$$d_l = \sum_{m=[l/2]+1}^q \frac{|B_{2m}|}{m} \left[\alpha_{2m-l-1}^{(1)} + (2^{2m} - 1) \alpha_{2m-l-1}^{(2)} \right],$$

$$\alpha_l^{(s)} = \frac{1}{l!} \sum_{x_k \in \Omega_s(P_n)} f_k (x_c^{(n)} - \tilde{x}_k)^l, \quad s = 1, 2.$$

For $2q \ll \min(N, M)$ the second sum in Eq. (14) is $O(2q(N+M))$. For the complexity of the MLFMM in the present case we have [4]

$$C_{FMM} = O \left(p(N+M) + \frac{3NM}{2^{l_{\max}}} + \frac{6P}{2^{-l_{\max}}} \right), \quad (15)$$

where $P(p)$ is the cost of a single translation for truncation p .

The total cost of the fast Fourier interpolation algorithm (FFIA) can be estimated, and minimized by selection of l_{\max} for given error (12). A simplified estimate assuming that p and q change slower than $2^{l_{\max}}$, yields

$$l_{\max}^{(opt)} \sim \frac{1}{2} \log_2 \frac{NM}{2P}, \quad (16)$$

and the total complexity of the optimized algorithm will be

$$C_{FFIA}^{(opt)} = O\left((N+M)(p+2q) + 6[2NMP(p)]^{1/2}\right). \quad (17)$$

For $N \sim M$, this yields $C_{FFIA}^{(opt)} = O(N(\log N + \log \epsilon^{-1}))$.

3. NUMERICAL STUDY

The FFIA was implemented and tested on a 933 MHz Pentium III Xeon PC with 1GB RAM. in complex double precision. Tests were performed for $M = N = 2^L$ varying in $2^3 - 2^{20}$ for various l_{\max} , p , and q . In all cases, y_j , $j = 0, \dots, M-1$ were randomly distributed in $[0, 2\pi)$ for the transform $\{f_k\} \rightarrow \{g_j\}$ and were within 10% perturbation near the centers $\{x_k\}$ for the transform $\{g_j\} \rightarrow \{f_k\}$. We found that the errors for the latter depend substantially on the distribution of $\{y_j\}$, while for $\{f_k\} \rightarrow \{g_j\}$ there was no such dependence. We thus focused on the transform $\{f_k\} \rightarrow \{g_j\}$, since the error analysis for the other case should be combined with a more rigorous study of non-uniformity errors and errors in computation of coefficients (5).

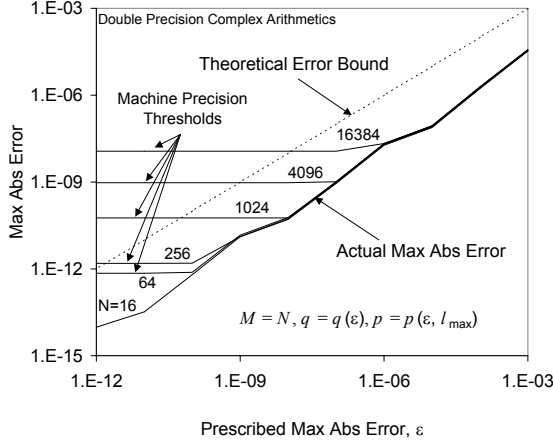


Fig. 2. Dependences of ϵ_a , for the FFIA solution on the prescribed ϵ for various N , indicated near the curves.

Error analysis

We performed several error tests to ensure that the algorithm worked properly and results were consistent with theory. First, we took the analytical solution $f(x) \equiv 1$, and varied l_{\max} from 3 to $L = \log_2 N$, and prescribed $\epsilon = 10^{-12}, \dots, 10^{-3}$; q and p were found from (13). Results of the tests are in Fig. 2. The actual error was measured as $\epsilon_a = \max_j |g_j - 1|$ and depends on N, l_{\max}, ϵ . We observed that ϵ_a just slightly depends on l_{\max} , consistent with (12). Another theoretical prediction provided by Eq. (12) is that the error should not depend on N . Fig. 2 shows this for $\epsilon \gg \epsilon_{th}(N)$, where $\epsilon_{th}(N)$ is a threshold error, which we refer as “machine precision error”; ϵ_a was much smaller than ϵ .

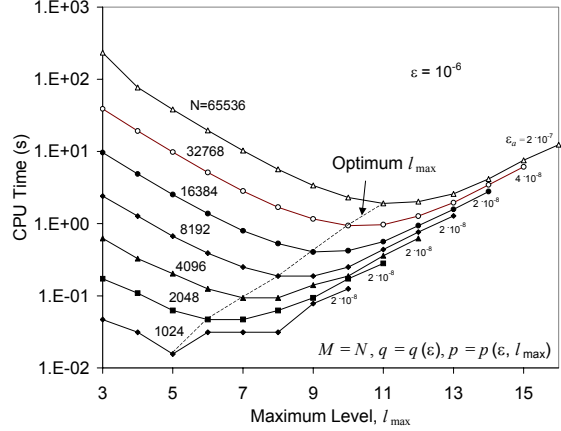


Fig. 3. Dependence of the CPU time on l_{\max} in the MLFMM for prescribed $\epsilon = 10^{-6}$ for different N . The small numbers near the right end of the curves show ϵ_a .

To validate that ϵ_{th} is really related to the machine precision arithmetic, we performed a test, when $\{g_j\}$ was computed by the “exact” straightforward method, i.e. by multiplication of matrix $\{K_{jk}\}$ computed using Eq. (3a) by vector $\{f_k\}$. The results of this test yield dependence $\epsilon(N)$, which is consistent with $\epsilon_{th}(N)$ obtained in experiments using the “approximate” fast method. We also compared the maximum absolute difference between the results obtained using the straightforward method and the FFIA for constant and random $f_k \in [0, 1]$. In both cases for $\epsilon = 10^{-12}$ this error was smaller than $\epsilon_{th}(N)$.

We can thus conclude that the theoretical error or machine precision bound the FFIA error. It makes no sense to specify an error below the precision, and this provides a result on the asymptotic complexity of the MLFMM. Indeed, estimation $C_{FFIA}^{(opt)} = O(N \log N)$ does not take into account the growth of $\epsilon_{th}(N)$ with N due to roundoff errors, while assumes that computations are done with arbitrary precision. However, this happens only for $\epsilon \gg \epsilon_{th}(N)$. If the computational task is formulated as “solve the problem with machine precision” then the complexity will be $O(N)$.

Optimization: We conducted optimization tests in two basic settings for error control. The first case we considered is minimization of the CPU time for fixed prescribed error $\epsilon \geq \epsilon_{th}(N)$. The second case is minimization of the CPU time for computations with maximum available machine precision. In the first case the range of N was limited by condition $\epsilon \geq \epsilon_{th}(N)$ and in the second case N was limited by machine resources available.

Fig. 3 illustrates dependence of the CPU time on l_{\max} at fixed $\epsilon = 10^{-6}$ and different N . Truncation numbers q and p were selected using (13). The actual error of computations ϵ_a was estimated by comparison with analytical solution, $f \equiv 1$ and shown on Fig. 3.

All curves on Fig. 3 clearly show that there exist an optimum maximum level of the space subdivision, $l_{\max}^{(opt)}$, for the MLFMM. The CPU time for computations with $l_{\max} = l_{\max}^{(opt)}$ and non-optimized l_{\max} can differ substantially. An empirical correlation is $l_{\max}^{(opt)} = \log_2 N - l_*(\epsilon)$, where $l_*(\epsilon)$ depends on the algorithm implementation (in our case $l_*(\epsilon) = 5$, see Fig. 3).

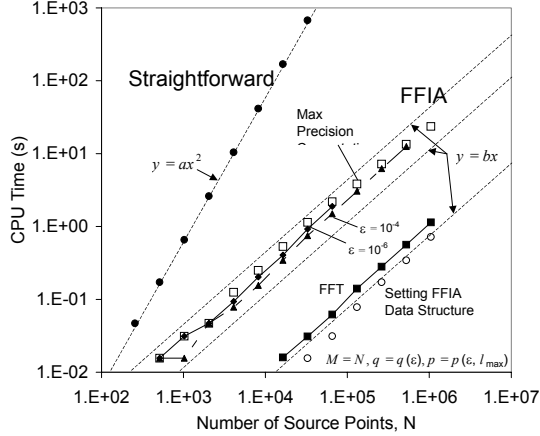


Fig. 4. Dependences of the CPU time on N , for the straightforward method (dark circles), and present method (FFIA) with prescribed accuracy (dark diamonds and triangles) and with the machine precision (light squares). The light circles show CPU time for setting the data structure. The dark squares show the CPU time required for FFT on the same machine.

In the second case we imposed experimentally measured machine precision threshold $\epsilon_{th}(N)$ as the prescribed accuracy ϵ . In this case we found that the same dependence $l_{max}^{(opt)}(N)$ describes $l_{max}^{(opt)}(L)$ well. We also noted that the actual error of computations ϵ_a was equal to $\epsilon_{th}(N)$ for $\epsilon = \epsilon_{th}(N)$.

Speed of Computations: To evaluate the algorithm we measured CPU time for the straightforward method, and the optimized FFIA for computations with fixed $\epsilon \geq \epsilon_{th}(N)$ and machine precision, $\epsilon = \epsilon_{th}(N)$. It is noticeable that the MLFMM procedures consist of two steps: setting the data structure, the step which needs to be performed when the target data set $\{y_j\}$ changes, and evaluation, which is executed each time the input $\{f_k\}$ changes. Fig. 4 shows that the computational cost for setting the data structure grows as $O(N)$ and is much smaller than the cost of the evaluation step. In case the interpolation is performed for INUFFT, an execution of the IFFT is required for the transform $\{c_n\} \rightarrow \{f_k\}$. Fig. 4 shows that the cost of this step is low compared to the FFIA evaluation step and is slightly above that of setting of the data structure.

Fig. 4 also shows that the FFIA far outperforms the straightforward method even for small data sets ($N \sim 10^2$) and the difference gains orders of magnitude for larger N . This is clear, since the straightforward method is scaled as $O(N^2)$, while the FFIA for maximum precision is scaled as $O(N)$. It is interesting to see that for fixed prescribed error ϵ , such as $\epsilon = 10^{-6}$ and $\epsilon = 10^{-4}$ shown in the figure, the CPU time required for computations deviates from the linear dependence $O(N)$ at larger N . This is consistent with the estimation of the algorithm complexity $O(N \log N)$ valid for $\epsilon \gg \epsilon_{th}(N)$. In contrast to this behavior the curve corresponding to computations with the maximum precision available, $\epsilon = \epsilon_{th}(N)$, shows deviation from the linear dependence in the *opposite* direction at larger N . The explanation of this effect comes from the error analysis provided above and also clear from Fig. 5, which illustrates the dependences of the truncation numbers on N . While q for fixed ϵ stays constant and p increases $a+b \log N$

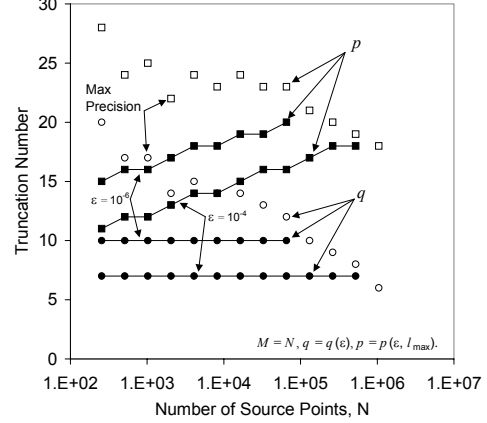


Fig. 5. Truncation numbers, q (circles) and p (squares), vs the N . Dark circles and squares show dependence of computation with fixed ϵ . Light circles and squares show the threshold dependences $q_{th}(N)$, $p_{th}(N)$ obtained using Eq. (13) for $\epsilon = \epsilon_{th}(N)$.

a different behavior of the bounding values p_{th} and q_{th} for maximum precision computations occurs. Both the truncation numbers, p_{th} and q_{th} , are bounded by a global constant. Moreover, they have a tendency to decay at larger N and $p_{th}, q_{th} \sim a - b \log N$, where a and b some constants.

4. CONCLUSIONS

We developed and tested a fast algorithm for matrix-vector multiplication with a kernel appropriate for the forward and inverse interpolation problems of band limited functions, which can be used for the forward and inverse Fourier Transforms for non-equispaced data. The algorithm far outperforms straightforward methods. Our results stay within the specified error bounds or maximum available machine accuracy. The algorithm asymptotically scales as $O(N)$ for computations with fixed maximum machine precision and as $O(N \log N + N \log \epsilon^{-1})$ otherwise. Furthermore, the current algorithm is parallelizable.

5. REFERENCES

- [1] J.W. Cooley, J. W. Tukey. "An algorithm for the machine calculation of complex Fourier series," Math Comp. 19, 297-301, 1965.
- [2] A. Dutt, V. Rokhlin. "Fast Fourier transforms for nonequipped data II," J. Comp. Harm. Anal., 2, 85-100, 1995.
- [3] D. Potts, G. Steidl, M. Tasche. "Fast Fourier transforms for nonequipped data: A tutorial," In: Modern Sampling Theory: Math and Applications, Birkhäuser, 253 - 274, 2001.
- [4] N.A. Gumerov, R. Duraiswami, E.A. Borovikov. "Data structures, optimal parameters, and complexity results for generalized multilevel fast multipole methods in d dimensions," UMIACS-TR-2003-28, 2003 (submitted to J. Comp. Phys.)
- [5] M. Abramowitz and I. A. Stegun (eds.). Handbook of Mathematical Functions, U. S. Government Printing Office, 1964.
- [6] L. Greengard, The Rapid Evaluation Of Potential Fields in Particle Systems, MIT Press, Cambridge, 1988.