Fast Interpolation of Bandlimited Functions

Samuel F. Potter, Nail Gumerov, Ramani Duraiswami

August 8, 2016

Abstract

The nonuniform fast Fourier transform (FFT) is a class of algorithms which brackets the FFT with a pair of interpolating maps which have complexity equal or superior to the FFT (generally $O(n \log n)$, where n is the size of the argument). A set of algorithms based on the fast multipole method (FMM) providing sufficient complexity guarantees exists [5]—each such algorithm relies on a fast algorithm for the Cauchy or cotangent kernel. In this work, a recent method for computing periodic sums is applied in order to improve the speed and accuracy of these kernels, which is then applied to the nonuniform FFT. Error bounds and complexity guarantees are proved, and numerical results are presented.

Contents

1 Introduction

The discrete Fourier transform (DFT) is an orthogonal linear transformation which is a discrete analog of the continuous Fourier transform. Many problems in science and engineering yield to the methods of Fourier analysis, so much so that when the fast Fourier transform (FFT) algorithm of Cooley and Tukey was disclosed

in 1965, much of the focus of research in the field of signal processing shifted from research in analog signal processing to digital signal processing, owing to the newfound feasibility of previously speculative numerical methods [16]. Of course, this algorithm has gone on to enable the efficient solution of a myriad of problems since.

A limitation of the real FFT is that its domain consists of real vectors whose components are assumed to correspond to equally spaced samples of some underlying continuous signal; likewise, the range is comprised of coefficients of equally spaced frequency components. When presented with a nonuniformly sampled input signal or when one's goal is to determine the strength of frequency components which are unequally spaced, the usual FFT is of no use without interpolating before or after the transform. A variety of methods have been introduced to formalize an approach to computing such a nonuniform DFT or FFT.

In this work, we extend a method for computing the nonuniform FFT which makes use of the fast multipole method (FMM) [5]. In particular, making use of recent work inspired by the kernel independent FMM, we append a step to this method which takes a negligible amount of time but increases the accuracy of the result. Algorithms are derived and their complexity is analyzed, relevant error bounds are established, and numerical results are presented.

2 The Nonuniform FFT

Let $f : \mathbb{R} \to \mathbb{R}$ be a bandlimited function, and let $K \in \mathbb{N}$ be its bandlimit. If, for each k such that $0 \leq k < K$, we let $x_k = 2\pi k/K$ and $f_k = f(x_k)$, then the DFT of $f_0, \ldots, f_{K-1}$ is given by:

$$F_l = \sum_{k=0}^{K-1} f_k e^{-2\pi i k l/K} \tag{1}$$

for l such that $0 \leq l < K$. The f_k's and F_l's can be thought of as ordinates associated with corresponding equispaced ascissae. For arguments off of these regular grids of points (in the time and frequency domains, respectively), the question as to how to evaluate f naturally arises. The general class of solutions provides methods for computing what is referred to as the nonuniform DFT.

The inverse DFT formula leads us to interpret the DFT coefficients $F_0, \ldots, F_{K-1}$ as the coefficients of a trigonometric polynomial:

$$f(x) = \sum_{k=-\lfloor K/2 \rfloor}^{\lceil K/2 \rceil - 1} F_l e^{ilx} \tag{2}$$

If, for j such that $0 \leq j < J$, we define $\tilde{x}_j$ such that $0 \leq \tilde{x}_j < 2\pi$, then combining (1) with (2) evaluated at each $\tilde{x}_j$ yields:

$$\tilde{f}_j \triangleq f(\tilde{x}_j) = \frac{1}{K} \sum_{k=0}^{K-1} \left(\sum_{l=-\lfloor K/2 \rfloor}^{\lceil K/2 \rceil - 1} e^{il(\tilde{x}_j - x_k)} \right) f_k. \tag{3}$$

It is well-known and readily demonstrated that this expression can be manipulated further to yield:

$$\tilde{f}_j = \frac{\sin(K\tilde{x}_j/2)}{K} \sum_{p=-\infty}^{\infty} \sum_{k=0}^{K-1} \frac{(-1)^k f_k}{\tilde{x}_j - x_k - 2\pi p}. \tag{4}$$

A derivation of this fact can be found, for example, in [17, 5].

Equation (4) allows us to view the interpolation as a linear operator applied to the values of a Fourier series at equispaced points. We can define this operator $\boldsymbol{P} \in \mathbb{R}^{J \times K}$ by:

$$\boldsymbol{P}_{j,k} \triangleq \frac{(-1)^k \sin(K\tilde{x}_j/2)}{K} \sum_{p=-\infty}^{\infty} \frac{1}{\tilde{x}_j - x_k - 2\pi p}. \tag{5}$$

Thus, letting $\boldsymbol{f} \in \mathbb{R}^K$ and $\tilde{\boldsymbol{f}} \in \mathbb{R}^J$ be defined by $\boldsymbol{f}_k = f_k$ and $\tilde{\boldsymbol{f}}_j = \tilde{f}_j$, respectively, we can see that $\tilde{\boldsymbol{f}} = \boldsymbol{P}\boldsymbol{f}$. The expression for $\tilde{f}_j$ provided in (4) can be derived from results in the paper from which this work primarily derives [5]. The inverse interpolation formula is also given in this paper. From this formula, we have that the nominal approximate "inverse" interpolation operator $\boldsymbol{P}^{-1} \in \mathbb{R}^{K \times J}$ is defined by letting:

$$c_j \triangleq \prod_{k=0}^{K-1} \sin\left(\frac{\tilde{x}_j - x_k}{2}\right), \tag{6}$$

$$d_k \triangleq \prod_{\substack{k'=0 \\ k' \neq k}}^{K-1} \frac{1}{\sin\left(\frac{x_k - x_{k'}}{2}\right)}, \tag{7}$$

$$\boldsymbol{P}^{-1}_{k,j} \triangleq c_j \cdot d_k \cdot \sum_{p=-\infty}^{\infty} \frac{1}{\tilde{x}_j - x_k - 2\pi p}. \tag{8}$$

The corresponding formula for recovering f_k from $\tilde{f}_j$ is:

$$x_k = d_k \sum_{p=-\infty}^{\infty} \sum_{j=0}^{J-1} \frac{c_j}{\tilde{x}_j - x_k - 2\pi p}. \tag{9}$$

If we let $\boldsymbol{F}$ denote a linear operator corresponding to the DFT, e.g. an application of some FFT, then there are four operators of interest which together comprise the nonuniform FFT (Figure 1). Of course, further combinations are possible—e.g. $\boldsymbol{P}^{-\top} \circ \boldsymbol{F} \circ \boldsymbol{P}^{-1}$. As mentioned, the goal is to the design algorithms which compute $\boldsymbol{P}$ and its variations with complexity no worse than $\boldsymbol{F}$ and $\boldsymbol{F}^{-1}$. As it happens, each of (5–8) can be computed sufficiently fast. Writing:

$$\log(c_j) = \sum_{j=0}^{K-1} \log \circ \sin\left(\frac{\tilde{x}_j - x_k}{2}\right), \tag{10}$$

and similarly for d_k, allows the FMM for the $\log \circ \sin$ kernel to be applied so that these coefficients may be precomputed, although this is not our focus.

Operator	Description
$\boldsymbol{N} \triangleq \boldsymbol{P} \circ \boldsymbol{F}^{-1}$	Compute the IDFT and interpolate in the time domain.
$\boldsymbol{N}^{\top} \triangleq \boldsymbol{F}^{-1} \circ \boldsymbol{P}^{\top}$	Interpolate uniform frequencies from arbitrary ones and compute IDFT.
$\boldsymbol{N}^{-1} \triangleq \boldsymbol{F} \circ \boldsymbol{P}^{-1}$	Approximate uniformly spaced Fourier series values and compute the DFT.
$\boldsymbol{N}^{-\top} \triangleq \boldsymbol{P}^{-\top} \circ \boldsymbol{F}$	Apply the DFT and interpolate in the frequency domain.

Figure 1: the various nonuniform FFT operators.

3 The Fast Multipole Method

The acceleration of the interpolation steps in the preceding section relies on a sufficient accurate and optimized fast multipole method (FMM) algorithm. The FMM is an algorithm which computes matrix-vector products of a certain form in sub-quadratic time [10]. Typically, the FMM is developed for a particular radial basis function of a certain dimensionality. In this work, we provide an optimized one-dimensional FMM for the Cauchy kernel:

$$\Phi(y, x) = \frac{1}{y - x}. \tag{11}$$

Implementing an FMM requires the derivation of regular and singular factorizations the kernel function (so-called S-factorizations and R-factorizations), along with derivations of translation operators which reexpand these factorizations: singular-to-singular, singular-to-regular, and regular-to-regular translation operators [12].

The R and S-factorizations of the Cauchy kernel are derived in a straightforward manner from the Taylor expansion of Φ. For the S-factorization, we fix x, y, and x_* such that $|x - x_*| < |y - x_*|$ and define $b_m(x, x_*) = (x - x_*)^m$ and $S_m(y - x_*) = (y - x_*)^{-m-1}$. The point x_* is referred to as the expansion center. Then, the S-factorization of Φ is given by:

$$\Phi(y, x) = \sum_{m=0}^{\infty} b_m(x, x_*) S_m(y - x_*). \tag{12}$$

Very similarly, for the R-factorization, we fix x, y, and x_* such that $|y - x_*| < |x - x_*|$ and define $a_m(x, x_*) = -(x - x_*)^{-m-1}$ and $R_m(y - x_*)^m = (y - x_*)^m$, giving us:

$$\Phi(y, x) = \sum_{m=0}^{\infty} a_m(x, x_*) R_m(y - x_*). \tag{13}$$

These expansions of Φ are used in the implementation of our FMM, as well as in our discussion of error bounds.

The FMM involves evaluating a function which is comprised of a sum of weighted kernels at a set of target points. In doing so, the computational domain is decomposed using a spatial data structure (e.g. a binary tree or octree). The algorithm first expands each weighted kernel using an S-factorization, then applies the translation operators according to the spatial decomposition (Figure 3). We denote the singular-to-singular translation matrix by $\mathbf{S}|\mathbf{S}$. Represented as an infinite matrix, its entries are given by:

$$(\mathbf{S}|\mathbf{S})_{n,m} \triangleq \frac{(-1)^{n-m} n! \delta^{n-m}}{(n-m)! m!}, \tag{14}$$

where δ is the translation vector between the two expansion centers. That is, if the resultant S-factorization is to be expanded about x'_*, then $\delta = x_* - x'_*$. The corresponding S-reexpansion is:

$$S_m(y - x_*) = \sum_{n=0}^{\infty} (\mathbf{S}|\mathbf{S})_{n,m} S_n(y - x'_*). \tag{15}$$

Such a reexpansion requires that $|\delta| < |y - x'_*|$. More details can be found in the appendix, but the expressions for the singular-to-regular translation operator $\mathbf{S}|\mathbf{R}$ and the regular-to-regular translation operator $\mathbf{R}|\mathbf{R}$ are given by:

$$(\mathbf{S}|\mathbf{R})_{n,m} \triangleq \frac{(-1)^n (m+n)!}{m! n! \delta^{m+n+1}}, \tag{16}$$

and:

$$(\mathbf{R}|\mathbf{R})_{n,m} = \begin{cases} \frac{m! \delta^{m-n}}{(m-n)! n!} & \text{if } n \leq m, \\ 0 & \text{otherwise,} \end{cases} \tag{17}$$

respectively.

Error Analysis

Lemma 1. With $x_* \in \mathbb{R}$, and $x, y, r, R \in \mathbb{R}$ such that:

$$|x - x_*| < r < R < |y - x_*|, \tag{18}$$

approximating $\Phi(y, x)$ with a P term S-expansion is bounded by:

$$|\varepsilon_S^{(P)}| < \frac{1}{R-r}\left(\frac{r}{R}\right)^P.$$

Proof. The P term S-expansion of $\Phi(y, x)$ can be written:

$$\Phi(y, x) = \varepsilon_S^{(P)} + \frac{1}{y - x_*}\sum_{m=0}^{P-1}\left(\frac{x - x_*}{y - x_*}\right)^m, \tag{19}$$

where $\varepsilon_S^{(P)}$ is evaluated to be:

$$\varepsilon_S^{(P)} = \frac{1}{y - x}\left(\frac{x - x_*}{y - x_*}\right)^P, \tag{20}$$

by evaluating the corresponding geometric series and invoking (18). Making use of (20) lets us bound:

$$|\varepsilon_S^{(P)}| = |x - y|^{-1}\left|\frac{x - x_*}{y - x_*}\right|^P < \frac{1}{R-r}\left(\frac{r}{R}\right)^P, \tag{21}$$

where the inequality follows from (18). □

Lemma 2. **TODO:** **S|S** reexpansion error.

Lemma 3. **TODO:** **S|R** reexpansion error.

Lemma 4. **TODO:** **R|R** reexpansion error.

4 Periodic Summation

The FMM sees frequent application in dynamics problems such as the n-body problem. For problems with periodic boundary conditions, the FMM on its own is not sufficient. Others algorithms are used instead—for example, Ewald summation, accelerated using the FFT [6]. Recent work inspired by kernel-independent FMM algorithms has resulted in a method for computing periodic sums which is ideally suited for use with the FMM [13]. Our method makes use of a simplified and accelerated version of this approach. We briefly derive the necessary results as they pertain to our problem.

The periodic summation method uses an FMM and least squares fitting to evaluate a sum which is a sum of weighted kernels which periodically repeat in some infinite computational domain. In our case, we are interested in computing the part of (4) which matches this description—namely, we define:

$$\phi(y) \triangleq \sum_{p=-\infty}^{\infty}\sum_{k=0}^{K-1}\frac{(-1)^k f_k}{y - x_k - 2\pi p}, \tag{22}$$

where we have ingored the constant factor. Next, in order to apply the method, we select some region of interest, a larger neighborhood which contains it, and decompose ϕ into a term due to the neighborhood and one due to its complement. Our region of interest is the interval $[0, 2\pi)$. We fix $n \in \mathbb{N}$ and consider it to be an integer "neighborhood radius" and define:

$$\mathcal{P} \triangleq \{-n, -n+1, \ldots, n\}. \tag{23}$$

Then, letting:

$$\phi_{\text{near}}(y) \triangleq \sum_{p\in\mathcal{P}} \sum_{k=0}^{K-1} \frac{(-1)^k f_k}{y - x_k - 2\pi p}, \tag{24}$$

$$\phi_{\text{far}}(y) \triangleq \sum_{p\notin\mathcal{P}} \sum_{k=0}^{K-1} \frac{(-1)^k f_k}{y - x_k - 2\pi p}, \tag{25}$$

we can decompose ϕ as:

$$\phi(y) = \phi_{\text{near}}(y) + \phi_{\text{far}}(y). \tag{26}$$

With $x_* = \pi$, we have that when y satisfies $0 \leq y < 2\pi$, the R-factorization:

$$\frac{1}{y - x_k - 2\pi p} = \sum_{m=0}^{\infty} a_m(x_k + 2\pi p, \pi) R_m(y - \pi) \tag{27}$$

is valid. Making use of (27), we can define, for each $m = 0, 1, \ldots$:

$$c_m \triangleq \sum_{p\notin\mathcal{P}} \sum_{k=0}^{K-1} (-1)^k f_k a_m(x_k + 2\pi p, \pi), \tag{28}$$

which allows us to define:

$$\phi_{\text{far}}(y) = \sum_{m=0}^{\infty} c_m R_m(y - \pi). \tag{29}$$

Reexpressing ϕ_{far} in terms of the R_m basis functions of the R-factorization is the crucial step that enables the method.

The periodic summation method involves using the FMM to compute the neighborhood term ϕ_{near} and then using the FMM and least squares collocation to compute ϕ_{far}. In particular, if we choose y such that $-2\pi n \leq y < 0$ or $2\pi \leq y < 2\pi(n+1)$, then there exists some p such that $\tilde{y} = y - 2\pi p$ satisfies $0 \leq \tilde{y} < 2\pi$. Since $\phi(y) = \phi(\tilde{y})$, we have:

$$\phi_{\text{near}}(y) - \phi_{\text{near}}(\tilde{y}) = \phi_{\text{far}}(\tilde{y}) - \phi_{\text{far}}(y) = \sum_{m=0}^{\infty} c_m (R_m(y - \pi) - R_m(\tilde{y} - \pi)). \tag{30}$$

Fixing $q \in \mathbb{N}$, we let $y_0, \ldots, y_{q-1}$ and $\tilde{y}_0, \ldots, \tilde{y}_{q-1}$ satisfy the same properties as y and $\tilde{y}$. Defining the matrices $\boldsymbol{\phi} \in \mathbb{R}^L, \boldsymbol{c} \in \mathbb{R}^\infty$, and $\boldsymbol{R} \in \mathbb{R}^{L\times\infty}$ by:

$$\boldsymbol{\phi}_l = \phi_{\text{near}}(y_l) - \phi_{\text{near}}(\tilde{y}_l), \tag{31}$$

$$\boldsymbol{c}_m = c_m, \tag{32}$$

$$\boldsymbol{R}_{l,m} = R_m(y_l - \pi) - R_m(\tilde{y}_l - \pi), \tag{33}$$

the coefficients c_m can be recovered by solving the corresponding least squares problems, e.g. by computing each c_m from $\boldsymbol{c} \approx \boldsymbol{R}^\dagger \boldsymbol{\phi}$.

There are two practical considerations in all of this. First, we only compute and apply a finite portion of $\boldsymbol{R}$, although the full matrix will be used in our error analysis. Second, since $\boldsymbol{R}_{l,0} = 0$ for each $l \in \mathbb{N}$, this column can be ignored. As a result, the question as to what to do about c_0 arises—this is addressed in the corresponding section. The resulting submatrix can be dealt with by making use of a convenient matrix decomposition, also addressed in a later section.

Necessary Conditions

The periodic summation method does not converge if a few basic necessary conditions are not met. In particular, when we compute ϕ_{far}, we approximate it by:

$$\phi_{\text{far}}(y) = \varepsilon_{\text{far}}^{(P)} + \sum_{m=0}^{P-1} c_m R_m(y-\pi), \tag{34}$$

where the error term is given by:

$$\varepsilon_{\text{far}}^{(P)} \triangleq \sum_{m=P}^{\infty} c_m R_m(y-\pi). \tag{35}$$

Then, for the method to converge, we require $\varepsilon_{\text{far}}^{(P)}$ and each c_m to converge. The following section establishes these necessary conditions.

Lemma 5. The coefficients c_m for $m = 0, \ldots, P-1$ are finite.

Proof. To show that the coefficients c_m converge, we consider the cases $m = 0$ and $m > 0$ separately. For $m = 0$, we observe that:

$$c_0 = \sum_{k=0}^{K-1} (-1)^k f_k \sum_{p=n+1}^{\infty} \frac{2(\pi - x_k)}{(x_k-\pi)^2 - 4\pi^2 p^2} = \frac{1}{2\pi^2} \sum_{k=0}^{K-1} (-1)^k f_k \cdot (x_k - \pi) \sum_{p=n+1}^{\infty} \frac{1}{p^2 - \left(\frac{x_k-\pi}{2\pi}\right)^2}.$$

We can see that this quantity is finite by an application of the integral test. Specifically, for each $k = 0, \ldots, K$, we have:

$$\int_{n+1}^{\infty} \frac{dt}{t^2 - \left(\frac{x_k-\pi}{2\pi}\right)^2} = \frac{2\pi}{x_k} \operatorname{arctanh}\left(\frac{x_k - \pi}{2\pi(n+1)}\right),$$

when:

$$\left(\frac{x_k-\pi}{2\pi}\right)^2 \leq (n+1)^2,$$

which is the case, since $n \in \mathbb{N}$ and $0 \leq x_k < 2\pi$ for $k = 0, \ldots, K-1$. For c_m where $m > 0$, we can see that the coefficient is finite by again applying the integral test and evaluating the corresponding integral. □

Next, we deal with the error due to the truncation of ϕ_{far}, as in (35).

Lemma 6. The error term for the approximation to $\phi_{\text{far}}(y)$ given by $\varepsilon_{\text{far}}^{(P)}$ is finite and bounded as follows:

$$|\varepsilon_{\text{far}}^{(P)}| \leq -\frac{||\boldsymbol{f}||_1}{2\pi} \left(\operatorname{Ei}\left(P\log(2n+3)\right) + \operatorname{Ei}\left(P\log(2n+1)\right)\right).$$

Proof. First, for each y such that $0 \leq y < 2\pi$, we have that:

$$|\varepsilon_{\text{far}}^{(P)}| \leq \sum_{k=0}^{K-1} |f_k| \sum_{m=P}^{\infty} \sum_{p \notin \mathcal{P}} \left| \frac{(y-\pi)^m}{x_k + 2\pi p - \pi} \right|. \tag{36}$$

Then, we have that $|y - \pi| \leq \pi$, and that $|x_k - 2\pi p - \pi| \geq \pi|2p-1|$. This gives us:

$$\left| \frac{(y-\pi)^m}{x_k + 2\pi p - \pi} \right| \leq \frac{1}{\pi|2p-1|}. \tag{37}$$

From (36) and (37), we have that:

$$|\varepsilon_{\text{far}}^{(P)}| \leq \frac{1}{\pi} \sum_{k=0}^{K-1} |f_k| \sum_{p=n+1}^{\infty} \left(\sum_{m=P}^{\infty} \frac{1}{(2p-1)^{m+1}} + \sum_{m=P}^{\infty} \frac{1}{(2p+1)^{m+1}} \right). \tag{38}$$

Next, since $p > n \geq 1$, we have that the two inner series in (38) are positive and decreasing for each m. Applying the integral test to (38) lets us bound:

$$\sum_{m=P}^{\infty} \frac{1}{(2p \pm 1)^{m+1}} \leq \int_{P}^{\infty} \frac{dm}{(2p \pm 1)^{m+1}} = \frac{(2p \pm 1)^{-P-1}}{\log(2p \pm 1)}. \tag{39}$$

Next, we consider the series in p. Again, since the terms of the series are positive and decreasing, we can bound using the integral test as follows:

$$\sum_{p=n+1}^{\infty} \frac{(2p \pm 1)^{-P-1}}{\log(2p \pm 1)} \leq \int_{p=n+1}^{\infty} \frac{(2p \pm 1)^{-P-1}}{\log(2p \pm 1)} dp. \tag{40}$$

We make the following changes of variable: first we let $x = 2p \pm 1$, and then $y = p \log(x)$. This allows us to write:

$$\int_{p=n+1}^{\infty} \frac{(2p \pm 1)^{-P-1}}{\log(2p \pm 1)} dp = \int_{2(n+1)\mp 1}^{\infty} \frac{dx}{x^{P+1} \log(x)} = \frac{1}{2} \int_{P \log(2(n+1)\mp 1)} \frac{dy}{y e^{y}}. \tag{41}$$

The final integral is the negative of the exponential integral function Ei evaluated at $-P \log(2(n+1) \mp 1)$. Thus, combining (40) and (41), we have:

$$\sum_{p=n+1}^{\infty} \frac{(2p \pm 1)^{-P-1}}{\log(2p \pm 1)} \leq -\frac{1}{2} \operatorname{Ei}\left(-P \log(2(n+1) \mp 1)\right). \tag{42}$$

Finally, combining (38), (39), and (42) lets us write:

$$|\varepsilon_{\text{far}}^{(P)}| \leq -\frac{1}{2\pi} \left(\operatorname{Ei}\left(-P \log(2n+3)\right) + \operatorname{Ei}\left(-P \log(2n+1)\right) \right) \sum_{k=0}^{K-1} |f_k|, \tag{43}$$

which completes the proof. □

Theorem 1. The necessary conditions for the convergence of the periodic summation are established by Lemmas 5 and 6.

The error bound developed in Lemma 6 provides a useful threshold criterion for choosing the neighborhood radius n and truncation number P parameters (see Figure 2).

5 Computation of c_0

The first column of the fitting matrix $\boldsymbol{R}$ is zero, which prevents the preceding least squares fitting procedure from recovering c_0. For some problems, this does not present a problem, but for this one it does, and a method of computing c_0 is necessary. As it happens, approximating c_0 from:

$$c_0 = \sum_{p \notin \mathcal{P}} \sum_{k=0}^{K-1} \frac{(-1)^k f_k}{x_k + 2\pi(p-1)} \tag{44}$$

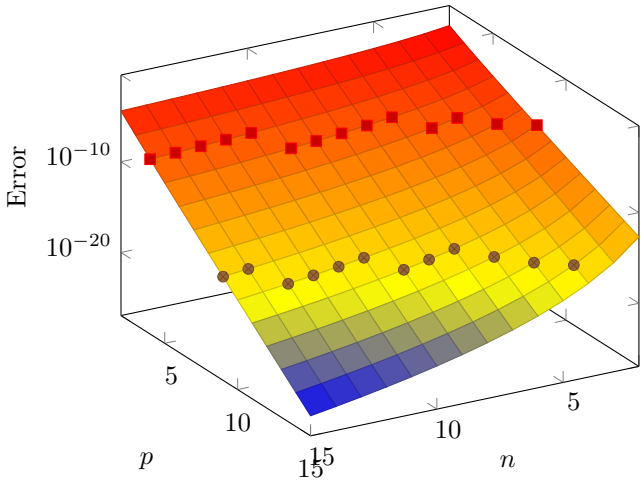

Figure 2: the bound for $|\varepsilon_{\text{far}}^{(P)}|$ from Lemma 6. The two sets of labeled values indicate points such that the error bound is below 10^{-7} and 10^{-15}. The ℓ_1 norm in the bound is taken to be unity.

is unrealistic due to slow convergence—the series is similar to the alternating harmonic series, for instance. Other methods for more general problems have been suggested [13].

In our case, we find that, for each $y \in \mathbb{R}$:

$$c_0 = \frac{1}{K} \sum_{k=0}^{K-1} \phi_{\text{near}}(y - x_k). \tag{45}$$

Using the FMM, we are able to approximately compute c_0 without compromising the overall complexity of the algorithm. What's more, by merging each $y - x_k$ into the set of target points and using $y - x_k$ for the algorithm's checkpoints, c_0 can be estimated without compromising the performance of the implementation. An added benefit of this approach is that, since c_0 computed this way does not depend on the set of target points, it can be precomputed so that the cost of its computation is amortized over repeated invocations.

As a matter of practical importance, we also note that ϕ_{near} as computed by the FMM is most accurate at the abscissae $(2k+1)\pi/K$, for $k = 0, \ldots, K-1$. An implementation of the algorithm presented in this paper, then, should select $y = \pi/K$ when computing c_0 in the above fashion.

If the overhead of computing the FMM at K points is too much, another approach which provides comparable (in our experiments, indistinguishable) accuracy is by bounding the series defining c_0 in the same fashion as in the proof of convergence of c_0 in the proof of the necessary condition for the periodic summation method. Specifically, we have that c_0 is approximately given by:

$$c_0 \approx \frac{1}{2\pi} \sum_{k=0}^{K-1} (-1)^k f_k \cdot \left(\operatorname{arctanh}\left(\frac{x_k - \pi}{2\pi n} \right) + \operatorname{arctanh}\left(\frac{x_k - \pi}{2\pi(n+1)} \right) \right). \tag{46}$$

This quantity can be computed in $O(K)$ time, and in our numerical experiments we found that it gave results as accurate as the results of the previous method. Because of this, this is the method used in our implementation.

6 Fitting Matrix

The least squares fitting matrix $\boldsymbol{R}$ from (33) is given by:

$$\boldsymbol{R} = \begin{bmatrix} R_1(y_0 - \pi) - R_1(\tilde{y}_0 - \pi) & \cdots & R_P(y_0 - \pi) - R_P(\tilde{y}_0 - \pi) \\ \vdots & \ddots & \vdots \\ R_1(y_{q-1} - \pi) - R_1(\tilde{y}_{q-1} - \pi) & \cdots & R_P(y_{q-1} - \pi - R_P(\tilde{y}_{q-1} - \pi) \end{bmatrix} \tag{47}$$

If we define $z_l = y_l - \pi$ and $\tilde{z}_l = \tilde{y}_l - \pi = y_l - 2\pi p$, we have that $\boldsymbol{R}$ can also be written:

$$\boldsymbol{R}_{l,m} = z_l^m - (z_l - 2\pi p)^m. \tag{48}$$

This requires us to choose the same p for each $\tilde{y}_l$. In a later section, we will see that this does not present a problem.

Some algebra yields that this matrix can be decomposed as the product of a Vandermonde matrix in $z_0, \ldots, z_{q-1}$ and an upper triangular matrix. In particular, letting:

$$\boldsymbol{V} \triangleq \begin{bmatrix} 1 & z_0 & z_0^2 & \cdots & z_0^{P-1} \\ 1 & z_1 & z_1^2 & \cdots & z_1^{P-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & z_{q-1} & z_{q-1}^2 & \cdots & z_{q-1}^{P-1} \end{bmatrix} \tag{49}$$

denote the Vandermonde matrix, and:

$$\boldsymbol{W}_{i,j} \triangleq \begin{cases} (-2\pi p)^{j-i+1} \binom{j}{j-i+1}, & \text{if } i \leq j, \\ 0, & \text{if } i > j, \end{cases} \tag{50}$$

where $\boldsymbol{W} \in \mathbb{R}^{P \times P}$, we have that $\boldsymbol{R} = \boldsymbol{V}\boldsymbol{W}$. The appendix contains a proof of this as well as some of $\boldsymbol{W}$ explicitly tabulated.

The properties of the Vandermonde matrix $\boldsymbol{V}$ are well understood. One property, the fact that $\boldsymbol{V}$ is—generally speaking—extremely poorly conditioned, will be considered in the next section. In this section, we make use of a well-known factorization of $\boldsymbol{V}^{-1}$ which will aid us in developing a fast and easily implemented algorithm for computing the fitting coefficients [18]. In particular, we have that there exists a lower triangular matrix $\boldsymbol{L}^{-1}$ and an upper triangular matrix $\boldsymbol{U}^{-1}$ such that $\boldsymbol{V}^{-1} = \boldsymbol{U}^{-1}\boldsymbol{L}^{-1}$ and such that the lower triangular factor is given by:

$$\boldsymbol{L}^{-1}_{i,j} = \begin{cases} 1, & \text{if } i = 1 = j, \\ 0, & \text{if } i < j, \\ \prod_{k=1, k\neq j}^{i} \frac{1}{z_j - z_k}, & \text{otherwise,} \end{cases} \tag{51}$$

and the upper triangular factor by:

$$\boldsymbol{U}^{-1}_{i,j} = \begin{cases} 1, & i = j, \\ 0, & j = 1, i > 1, \\ \boldsymbol{U}^{-1}_{i-1,j-1} - \boldsymbol{U}^{-1}_{i,j-1} z_{j-1}, & \text{otherwise,} \end{cases} \tag{52}$$

where $\boldsymbol{U}^{-1}_{0,j} = 0$. Our algorithm for computing the coefficients c_m, then, is based on writing $\boldsymbol{R}\boldsymbol{c} = \boldsymbol{\phi}$ as $\boldsymbol{W}\boldsymbol{c} = \boldsymbol{U}^{-1}\boldsymbol{L}^{-1}\boldsymbol{\phi}$. The right-hand side can be computed efficiently due to the simple and regular form of the matrices which allows $\boldsymbol{c}$ to be computed using an efficient back-substitution.

7 Main Algorithm

Our main algorithms apply $\boldsymbol{P}$ and $\boldsymbol{P}^{-1}$ by evaluating (4) using the periodic summation method as applied to (22). Evaluating $\boldsymbol{P}^{\top}$ and $\boldsymbol{P}^{-\top}$ can be done by interchanging arguments.

Algorithm 1. Compute $\phi(\tilde{x}_j)$ for each j.

1. Evaluate $\phi_{\text{near}}(\tilde{x}_j)$ for each j using the FMM.
2. Compute $\boldsymbol{\phi}$ using the FMM and the checkpoint sets $\{y_l\}$ and $\{\tilde{y}_l\}$.
3. Form $\boldsymbol{U}^{-1}\boldsymbol{L}^{-1}\boldsymbol{\phi}$.
4. Solve $\boldsymbol{W}\boldsymbol{c} = \boldsymbol{U}^{-1}\boldsymbol{L}^{-1}\boldsymbol{\phi}$ for $\boldsymbol{c}$ using back substitution.
5. Compute c_0 **TODO:** *by some method.*
6. Evaluate $\phi_{\text{far}}(\tilde{x}_j)$ for each j using $\boldsymbol{c}$ and (29).
7. Compute $\phi(\tilde{x}_j) = \phi_{\text{near}}(\tilde{x}_j) + \phi_{\text{far}}(\tilde{x}_j)$ for each j.

Algorithm 2. Compute $\tilde{f}_j$ for each j (apply $\boldsymbol{P}$ to $\boldsymbol{f}$).

1. Evaluate $\phi(\tilde{x}_j)$ for each j using Algorithm 1.
2. Compute each $\tilde{f}_j = \sin(K\tilde{x}_j/2)\phi(\tilde{x}_j)/2$.

Algorithm 3. Compute f_k for each k (apply $\boldsymbol{P}^{-1}$ to $\tilde{\boldsymbol{f}}$).

1. Precompute c_j for each j using the FMM.
2. Precompute d_k for each k using the FMM.
3. Evaluate (9) for each k using the precomputed coefficients and Algorithm 1.

8 Complexity

In this section, we come up with a rough estimate of the cost of the MLFMM and an asymptotic estimate of its cost. The FMM used in our implementation consists of several different parts, which can be considered independently:

- The **S**|**S**, **S**|**R**, and **R**|**R** translations.
- The construction of S factorizations for the finest boxes.
- Direct evaluation.
- Indirect evaluation.

We will consider each of these items in turn. Important quantities related to the algorithm are as follows:

L	The maximum FMM depth.
K	The bandlimit of the function f.
n	The integer neighborhood radius.
Kn	The number of source points for the FMM (due to periodic tiling).
J	The number of evaluation points (interpolation nodes $\tilde{x}_j$).
P	The truncation number of the kernel approximation.

We also note that in our estimate of the complexity, we assume that the source and evaluation points are distributed uniformly. We note that assumption of uniformity, while generally useful, can be improved upon for the interpolation nodes. In particular, the integer neighborhood radius n plays a larger role: if we ignore the checkpoints, the evaluation points only occur in the range $[\frac{n}{2n+1}, \frac{n+1}{2n+1})$. If the FMM is implemented to take advantage of this, the number of translation operators that must be applied can be significantly reduced. The implementation used in the current work does not take advantage of this optimization.

Applying the $\mathbf{S}|\mathbf{S}$, $\mathbf{S}|\mathbf{R}$, and $\mathbf{R}|\mathbf{R}$ translation operators takes approximately $O(P^2)$ operations, as applying these operators corresponds to multiplication of a vector in $\mathbb{C}^P$ by a matrix in $\mathbb{C}^{P\times P}$. With the parameter L chosen, a union of four perfect binary trees of height $L-1$ is constructed—this data structure will be referred to as a translation hierarchy (see Figure 3 for a graphical depiction of the case of $L=4$). For each subinterval at the finest level of subdivision, the FMM involves a hierarchical sequence of translations such that the function can be efficiently evaluated in this subinterval using an R expansion. We will not dwell on the details of this, as they can readily be found elsewhere [12, 10].

To estimate the cost of the one-dimensional FMM, we start by determining the cost of the translation operators. As the translation hierarchy consists of four perfect binary trees of height $L-1$, since one $\mathbf{S}|\mathbf{S}$ and one $\mathbf{R}|\mathbf{R}$ translation operator is computed for each edge in this tree, and since there are $2^{L-1}-2$ edges in a perfect binary tree of the given height, the combined cost of the $\mathbf{S}|\mathbf{S}$ and $\mathbf{R}|\mathbf{R}$ translation operators is $8(2^{L-1}-2)O(P^2) = O(2^L P^2)$. As for the $\mathbf{S}|\mathbf{R}$ translation operators, we note for each node in the translation hierarchy, at most (and in most cases) three $\mathbf{S}|\mathbf{R}$ operators are computed (cf. Figure 3). Since there are $4(2^{L-1}-1)$ nodes in the translation hierarchy, the cost of applying the $\mathbf{S}|\mathbf{R}$ operators is $12(2^{L-1}-1)O(P^2) = O(2^L P^2)$. Then, the overall cost of applying the translation operators is $O(2^L P^2)$.

The cost of the rest of the phases of the algorithm is straightforward to estimate. The construction of the S-factorizations is asymptotic in the number of source points and the truncation number, so the corresponding cost is $O(KPn)$. Indirect evaluation at a single point is $O(P)$ (evaluating a polynomial of degree P), and the entire indirect evaluation phase takes place at J points, so the overall cost is $O(JP)$. Finally, the direct evaluation cost is proportional to the number of points in the 1-cell neighborhood of each cell containing a target point times the number of target points—since the number of source points that align with target points is K, the overall cost is proportional to $2^{-L}KJ$.

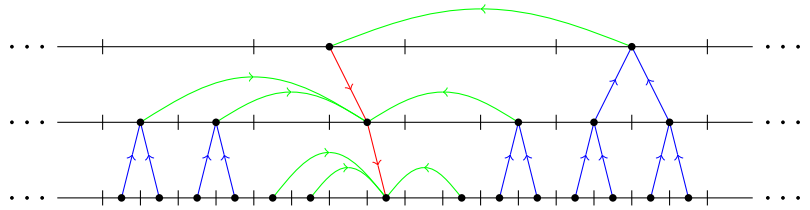

Figure 3: an illustration of the translation phases of the one-dimensional FMM.

9 Numerical Results

To evaluate our method, we chose several real-valued bandlimited test functions to use in comparisons made with existing software libraries that can be used to solve the same problem. We chose four, well-known and canonical trigonometric series: the triangle, square, semicircle, and saw waves. For a given bandlimit K,

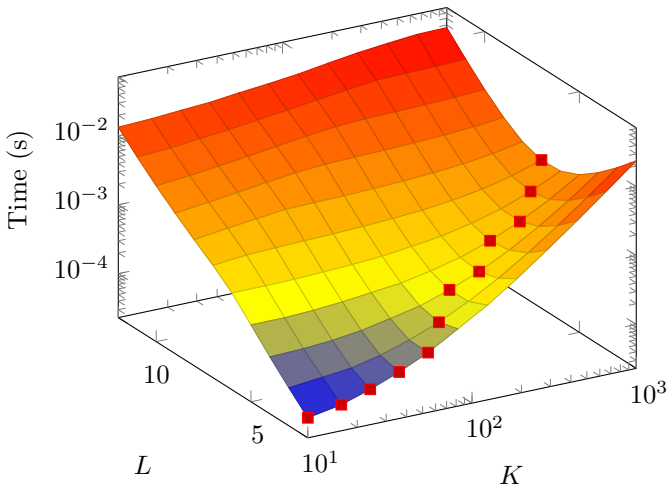

Figure 4: test

approximations to each of these are given by:

$$\text{Triangle}(x) = \sum_{k=-\lfloor K/2 \rfloor}^{\lceil K/2 \rceil - 1} e^{ikx} \cdot \begin{cases} 0 & \text{if } k \bmod 2 \equiv 0 \\ \frac{i(-1)^{(k+1)/2}}{\pi^2 k^2} & \text{otherwise} \end{cases} \tag{53}$$

$$\text{Square}(x) = \sum_{k=-\lfloor K/2 \rfloor}^{\lceil K/2 \rceil - 1} e^{ikx} \cdot \begin{cases} 0 & \text{if } k \bmod 2 \equiv 0 \\ \frac{-2i}{\pi k} & \text{otherwise} \end{cases} \tag{54}$$

$$\text{Semicircle}(x) = \sum_{k=-\lfloor K/2 \rfloor}^{\lceil K/2 \rceil - 1} e^{ikx} \cdot \begin{cases} \pi^2/4 & \text{if } k = 0 \\ \frac{\pi(-1)^k J_1(\pi k)}{2k} & \text{otherwise} \end{cases} \tag{55}$$

$$\text{Sawtooth}(x) = \sum_{k=-\lfloor K/2 \rfloor}^{\lceil K/2 \rceil - 1} e^{ikx} \cdot \begin{cases} 0 & \text{if } k = 0 \\ \frac{i(-1)^k}{\pi k} & \text{otherwise} \end{cases} \tag{56}$$

where J_1 is a Bessel function of the first kind. Plots of several approximations of these trigonometric series are included for reference (see Figure 5).

The software libraries used in comparisons include NFFT3 [15], NUFFT [8, 9], and IRT [7]. The first library consists of a large suite of methods which have a similar flavor as to the nonuniform FFT. The second library is a highly optimized library based on analysis of the nonuniform FFT initially carried out in [4], and refined in [8, 9] and in other articles by the same authors. The third library represents a separate approach based on min-max interpolation.

Our first numerical test consists of—for a varying choice of bandlimit K—evaluating the nonuniform DFT directly from (3), using each of the implementation provided by each library and Algorithm 2 to evaluate the nonuniform DFT, and plotting the corresponding ℓ_∞ error between each result and the groundtruth result evaluated using (3) (see Figure 6). The evaluation points for each experiment were chosen uniformly at random from the interval $[0, 2\pi)$. The algorithm used in the NUFFT library involves a fast Gaussian gridding approach, and the interface provides an error parameter which is gauranteed to be attained (with machine epsilon being the default). Because of this, we did not include it in this test.

The next test was a comparison of the runtimes of the different algorithms for the same choice of parameters as in the first experiment (Figure 7). As a baseline, we included timings of the inverse FFT, as provided by Python's numpy library.

Our timing test confirms our basic hypotheses regarding the speed of the algorithm. As with all FMM-

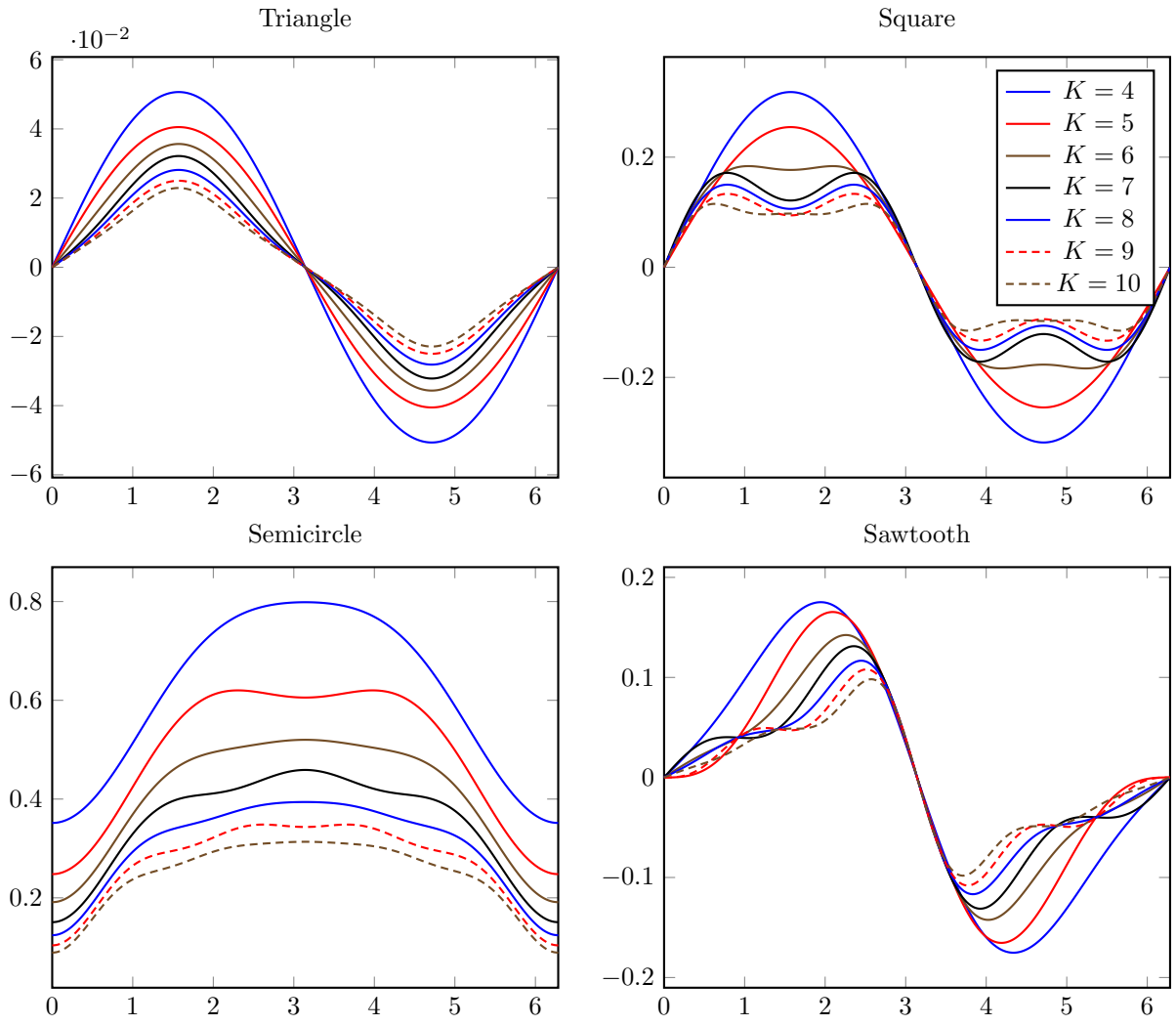

Figure 5: test

based algorithm, the scaling of the algorithm is such that beyond a certain point, the algorithm "breaks even" and is beaten by a direct method. However, for problem sizes below this threshold, we see substantial speedups.

The error plots are more intriguing. Although Algorithm 2 cannot compete with the implementation provided by NUFFT without appealing to a high truncation number P and neighborhood radius n, for reasonably small values, the algorithm attains the same performance as the highly-tuned NFFT and soundly outperforms what is provided by IRT. Of particular importance, we can see that for our choice of test functions, IRT and NUFFT exhibit different accuracy regimes. In particular, we can see that for Triangle and Square, both Algorithm 2 and NFFT are significantly more accurate than IRT, and that for some problem sizes NFFT attains machine precision, and for others NFFT and Algorithm 2 are nearly as accurate as one another. However, Semicircle and Sawtooth provide exceptions to this rule: in the Semicircle test, NFFT and IRT perform equally poorly, with Algorithm 2 providing much more accurate results; on the other hand, the Sawtooth test sees NFFT computing results exactly as accurate as NFFT. Overall, throughout all of the tests, Algorithm 2 does not surprising provide machine precision, as NFFT occasionally seems to, but

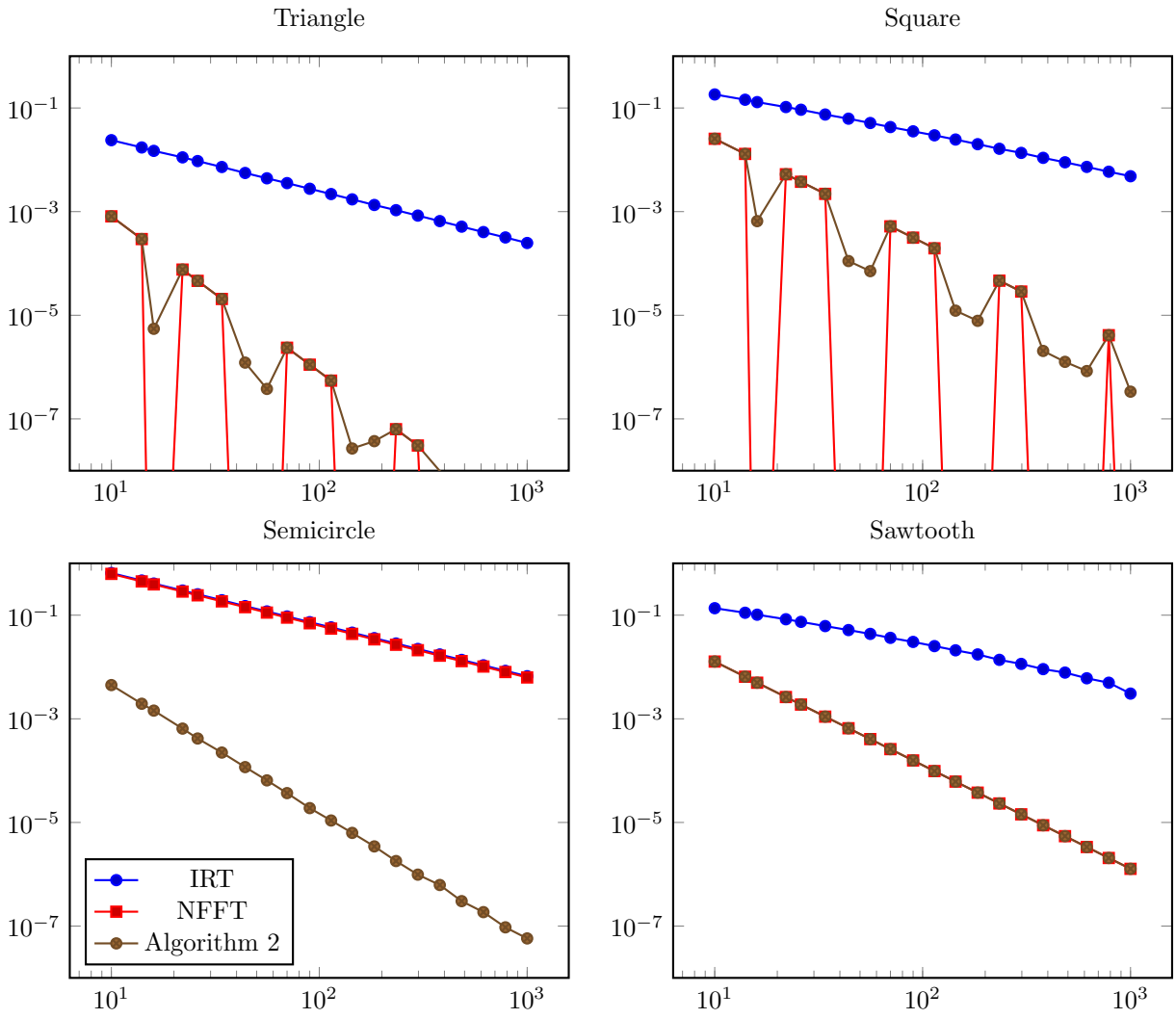

Figure 6: test

does provide consistently more accurate results than either NFFT or IRT.

10 Future Work

In this work, we extend and further develop an existing method for computing the nonuniform FFT, as originally analyzed in [5]. In the original analysis, the method is mostly passed over in favor of the algorithms derived in [4]. As demonstrated by the highly successful NUFFT library, those methods are indeed useful and powerful. Our work applies results derived in [13] to make the method presented in [5] significantly more precise, for negligible added runtime cost (in asymptotic terms, but also in terms of a real implementation). The work establishes the viability of the FMM as applied to the nonuniform FFT, laying the ground for future developments. In particular, as the core of the algorithm makes use of a generic 1D FMM, any improvements to such an FMM could profitably be exploited by the NUFFT. Such improvements may be application dependent. Additionally, the FMM implementation used in our numerical experiments did not

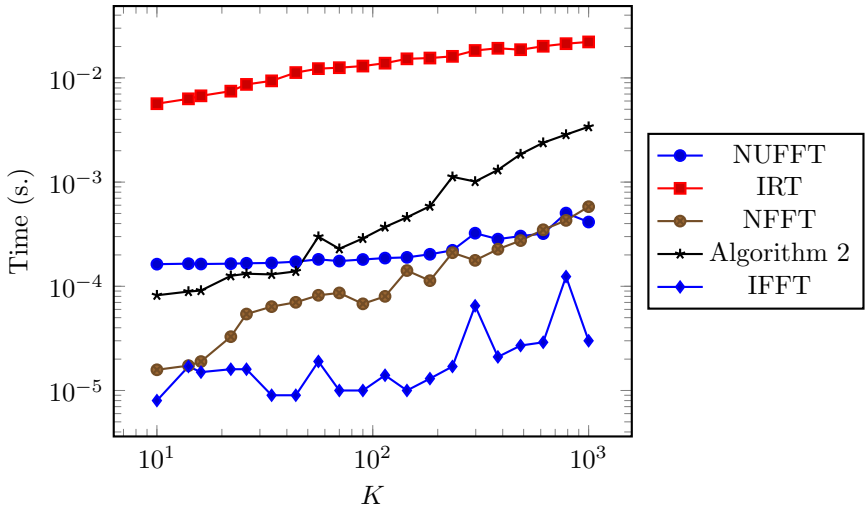

Figure 7: test

take advantage of the structure of the target points—doing so could reduce the number of translations required by the algorithm significantly.

Due to the low communication costs of the FMM, it is well-suited to parallel implementation, and parallel CPU, GPU [11], heterogeneous CPU/GPU [14], and distributed implementations exist and have been investigated to varying degrees. Our initial implementation was single-threaded and comparisons were done with other libraries that were compiled to run in a single thread.

With respect to parallel implementations of the FMM, most development has been around "uniform" FMMs—those whose underlying translation trees are computed to a uniform depth. Research has gone into so-called adaptive FMMs (e.g. [12, 2]), but parallel adaptive FMMs have not been research as intensively, due to the recursive nature of the adaptive FMM. However, as many applications of the nonuniform FFT involve fixed sets of target points, precomputation of the translation trees of adaptive FMMs may prove to be a good match for the parallel nonuniform FFT. Some examples of such applications include non-Cartesian MRI, which employs nonuniform grids in order to speed scanning patterns. Other such applications exist—in audio signal processing, the constant-Q transform requires a fixed, logarithmic grid of target points for a given frequency ratio [1].

Finally, the strategy of our implementation is to take advantage of the well-developed and existing research surrounding the FMM. Common kernels have been researched intensively, and the Cauchy kernel is no exception. While our implementation takes advantage of the straightforward (but perhaps somewhat naïve) R-expansions and S-expansions that come by way of the Taylor series expansion of the Cauchy kernel, other methods will be investigated—for instance, a Chebyshev expansion-based approach is presented in [3] which has tighter error bounds.

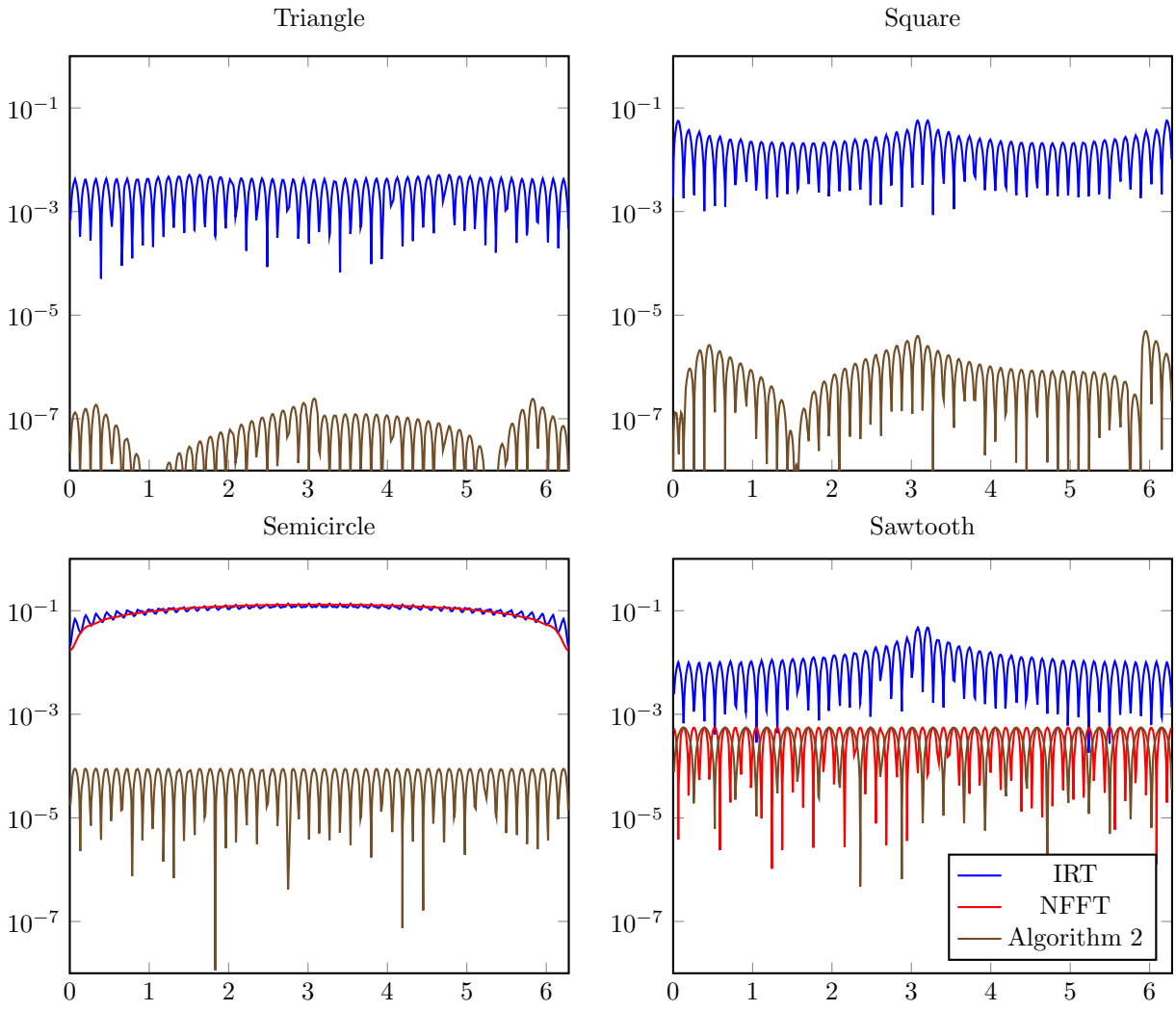

Figure 8: test

A Appendix

A bit of the matrix $\boldsymbol{U}$ is depicted as follows:

$$\boldsymbol{W} = -2\pi p \begin{bmatrix} 1 & -2p\pi & 4p^2\pi^2 & -8p^3\pi^3 & 16p^4\pi^4 & -32p^5\pi^5 & \cdots \\ & 2 & -6p\pi & 16p^2\pi^2 & -40p^3\pi^3 & 96p^4\pi^4 & \cdots \\ & & 3 & -12p\pi & 40p^2\pi^2 & -120p^3\pi^3 & \cdots \\ & & & 4 & -20p\pi & 80p^2\pi^2 & \cdots \\ & & & & 5 & -30p\pi & \cdots \\ & & & & & 6 & \cdots \\ & & & & & & \ddots \end{bmatrix} \tag{57}$$

B References

[1] J. C. Brown. Calculation of a constant Q spectral transform. *J. Acoust. Soc. Am.*, 89(1):425–434, 1991.

[2] J. Carrier, L. Greengard, and V. Rokhlin. A fast adaptive multipole algorithm for particle simulations. *SIAM J. Sci. Stat. Comput.*, 9(4):669–686, 1988.

[3] A. Dutt, M. Gu, and V. Rokhlin. Fast algorithms for polynomial interpolation, integration, and differentiation. *SIAM J. Numer. Anal.*, 33(5):1689–1711, 1996.

[4] A. Dutt and V. Rokhlin. Fast Fourier Transforms for Nonequispaced Data. *SIAM J. Sci. Comput.*, 14:1368–1393, 1993.

[5] A. Dutt and V. Rokhlin. Fast fourier transforms for nonequispaced data, II. *Applied and Computational Harmonic Analysis*, pages 85–100, 1995.

[6] P. P. Ewald. Die Berechnung optischer und elektrostatischer Gitterpotentiale. *Ann. Phys.*, 369(3):253–287, 1921.

[7] J. A. Fessler and B. P. Sutton. Nonuniform Fast Fourier Transforms Using Min-Max Interpolation. *IEEE Transactions on Signal Processing*, 51(2):560–574, February 2003.

[8] L. Greengard and J.-Y. Lee. Accelerating the Nonuniform Fast Fourier Transform. *SIAM Review*, 46(3):443–454, 2004.

[9] L. Greengard and J.-Y. Lee. The type 3 nonuniform FFT and its applications. *J. Comput. Phys.*, 206:1–5, June 2005.

[10] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Comput. Phys.*, 73, December 1987.

[11] N. Gumerov and R. Duraiswami. Fast multipole methods on graphics processors. *J. Comput. Phys.*, 227(18):8290–8313, 2008.

[12] N. A. Gumerov and R. Duraiswami. *Fast Multipole Methods for the Helmholtz Equation in Three Dimensions*. Elsevier, 2004.

[13] N. A. Gumerov and R. Duraiswami. A method to compute periodic sums, 2014.

[14] Q. Hu. *Scalable Fast Multipole Method on Heterogeneous Architecture*. PhD thesis, University of Maryland, 2013.

[15] J. Keiner, S. Kunis, and D. Potts. Using NFFT 3 - a software library for various nonequispaced fast fourier transforms. *ACM Trans. Math. Software*, 36:1–30, 2009.

[16] A. V. Oppenheim and R. W. Schafer. *Digital Signal Processing*. Prentice Hall, 1st edition, 1975.

[17] T. Schanze. Sinc interplation of discrete periodic signals. *IEEE Transactions on Signal Processing*, 43(6):1502–1503, 1995.

[18] R. L. Turner. Inverse of the Vandermonde matrix with applications. Technical report, NASA, 1966.