FAST INTERPOLATION OF BANDLIMITED FUNCTIONS

Samuel F. Potter[⋆†] *Nail A. Gumerov*[⋆] *Ramani Duraiswami*[⋆†]

[⋆] Perceptual Interfaces and Reality Lab
[⋆] University of Maryland Institute for Advanced Computer Studies
[†] University of Maryland Department of Computer Science

ABSTRACT

The nonuniform fast Fourier transform comprises a set of algorithms which approximately interpolate the usual discrete Fourier transform in the time and/or frequency domain. A nonuniform fast Fourier transform based on the fast multipole method was previously developed but passed over in favor of other approaches [1, 2]. This work extends a recent method for computing a periodized fast multipole method [3] with an adaptive algorithm that reduces the required number of multipole-to-local and local-to-local translations by an order of magnitude. This combination improves the speed and accuracy of the original algorithm, and results in an algorithm that is competitive with other nonuniform fast Fourier transforms. Numerical experiments are carried out comparing our implementation with others, demonstrating its viability.

Index Terms— nonuniform FFT, adaptive FMMs, constant Q transforms, periodization, nonuniform sampling

1. INTRODUCTION

The nonuniform Fourier transform (NUFFT) finds use in a plethora of applications—within signal processing, a primary motivation for our work is the family of constant Q transforms [4], which are frequently used in real-time audio analysis, where they are often computed for a train of short blocks with lengths on the order of 64–1024 samples, similar to the short-time Fourier transform (STFT). Several accelerated constant Q transforms exist [5, 6]. The constant Q transform is a particular nonuniform DFT, and the work presented here can be applied to it with little modification.

The NUFFT approximately interpolates the discrete Fourier transform (DFT) in $O(n \log n)$ time, where n is the problem size. The original papers on the NUFFT present two separate algorithms, one based on an analysis of the Gaussian [1], and another using the fast multipole method, or FMM [2, 7]. Since then, accelerated methods based on min-max interpolation [8], fast Gaussian gridding [9], and other gridding methods [10] have been developed and are in widespread use.

In this work, we extend the original FMM-based method for the NUFFT [2, 7] by employing a periodized FMM [3] and developing an adaptive algorithm which accelerates its computation. With these improvements, our single-threaded NUFFT is competitive with existing NUFFT implementations and runs exceptionally fast on the small problems required by the constant Q transform. It also has steps which can be precomputed, amortizing their cost.

The structure of the paper is as follows: we first summarize results from the original NUFFT research, then outline relevant details regarding the FMM. Following this, we discuss periodizing the FMM and show that the original procedure can be simplified and accelerated when computing the NUFFT. Our adaptive FMM algorithm is presented in Section 5, alongside empirical results that indicate its suitability for the NUFFT. Finally, we discuss our numerical experiments comparing our implementation with others and assessing its applicability to the constant Q transform.

2. THE NONUNIFORM FAST FOURIER TRANSFORM

Let $f : \mathbb{R} \to \mathbb{C}$ be a bandlimited function, and let $K \in \mathbb{N}$ be its bandlimit. Then, for $J \in \mathbb{N}$ and j such that $0 \leq j < J$, if we define $\tilde{x}_j$ such that $0 \leq \tilde{x}_j < 2\pi$, $x_k = 2\pi k/K$, and $f_k = f(x_k)$, we can use the DFT to interpolate f exactly at $\tilde{x}_j$:

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

It is well-known [2, 11] that this expression is equivalent to:

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

which describes a linear operator that performs the interpolation step of the NUFFT. Since the outer summation in p periodizes the sum of weighted kernels, this operator can be approximated using a periodized FMM, which is the focus of this work.

Although the operator described here interpolates a bandlimited function, the same operator can be used with the FFT to compute the forward constant Q transform. The original NUFFT research describes the full complement of algorithms more completely [2]. Although our focus is only on two of the original FMM-based NUFFT transforms, the work here can be applied to the other two, and will be pursued in a later work.

3. THE CAUCHY KERNEL FAST MULTIPOLE METHOD

There are multiple FMMs for the so-called Cauchy kernel [7, 12], given by $\Phi(y, x) = (y - x)^{-1}$. The implementation of an FMM requires the derivation of regular (R-) and singular (S-) factorizations of the kernel function, as well as translation operators for these factorizations [13]. We use a straightforward implementation based on single variable Taylor series expansions of Φ [12].

For the S-factorization, we fix x, y, and x_* with $|x - x_*| < |y - x_*|$ 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:

$$\Phi(y, x) = \sum_{m=0}^{\infty} b_m(x, x_*) S_m(y - x_*). \tag{3}$$

For the R-factorization, we fix x, y, and x_* with $|y - x_*| < |x - x_*|$ and define $a_m(x, x_*) = -(x - x_*)^{-m-1}$ and R_m by $R_m(y - x_*)^m = (y - x_*)^m$. Then, the R-factorization is:

$$\Phi(y, x) = \sum_{m=0}^{\infty} a_m(x, x_*) R_m(y - x_*). \tag{4}$$

The FMM uses linear translation operators to repeatedly reexpand a grid of S-expansions, converting them to R-expansions for evaluation. The three translation operators are denoted by $\mathbf{S}|\mathbf{S}$, $\mathbf{S}|\mathbf{R}$, and $\mathbf{R}|\mathbf{R}$—these operators are used to switch between factorizations, e.g. $\mathbf{S}|\mathbf{R}$ reexpands an S factorization as an R factorization. The $\mathbf{S}|\mathbf{S}$ operator is given by:

$$(\mathbf{S}|\mathbf{S})_{n,m} = \begin{cases} \frac{(-1)^{n-m} n! \delta^{n-m}}{(n-m)! m!} & \text{if } n \geq m, \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

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 $\mathbf{S}|\mathbf{R}$ and $\mathbf{R}|\mathbf{R}$ operators are given by:

$$(\mathbf{S}|\mathbf{R})_{n,m} = \frac{(-1)^n (m+n)!}{m! n! \delta^{m+n+1}}, \tag{6}$$

$$(\mathbf{R}|\mathbf{R})_{n,m} = \begin{cases} \frac{m! \delta^{m-n}}{(m-n)! n!} & \text{if } n \leq m, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

More details concerning this FMM, including error and complexity analysis, are available [12].

Two parameters are required by the FMM. First, an integer $P > 0$ is selected: this is the truncation number, and is the point at which each $S-$ and $R-$ factorization and translation operator are truncated (e.g., see (13)). Second, the FMM makes use of a tree data structure (such as a binary tree, quadtree, or octree): the depth of this tree is $L \geq 2$.

4. PERIODIC SUMMATION

We are interested in computing the factor of (2) which matches this description. In order to do so, we employ a periodized black box summation method [3] which is particularly well-suited to the FMM. We define:

$$\phi(y) = \sum_{p=-\infty}^{\infty} \sum_{k=0}^{K-1} \frac{(-1)^k f_k}{y - x_k - 2\pi p}. \tag{8}$$

To apply the method, we select an interval 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 define $\mathcal{P} = \{-n, -n+1, \ldots, n\}$. Letting:

$$\phi_{\text{near}}(y) = \sum_{p \in \mathcal{P}} \sum_{k=0}^{K-1} \frac{(-1)^k f_k}{y - x_k - 2\pi p}, \tag{9}$$

$$\phi_{\text{far}}(y) = \sum_{p \notin \mathcal{P}} \sum_{k=0}^{K-1} \frac{(-1)^k f_k}{y - x_k - 2\pi p}, \tag{10}$$

we can write $\phi(y) = \phi_{\text{near}}(y) + \phi_{\text{far}}(y)$. With $x_* = \pi$, if y satisfies $0 \leq y < 2\pi$, then, using (4):

$$\frac{1}{y - x_k - 2\pi p} = \sum_{m=0}^{\infty} a_m(x_k + 2\pi p, \pi) R_m(y - \pi). \tag{11}$$

From (11), we define for each $m \in \{0, 1, \ldots\}$:

$$c_m = \sum_{p \notin \mathcal{P}} \sum_{k=0}^{K-1} (-1)^k f_k a_m(x_k + 2\pi p, \pi), \tag{12}$$

which allows us to write $\phi_{\text{far}}(y) = \sum_{m=0}^{\infty} c_m R_m(y - \pi)$. This expansion of ϕ_{far} in terms of the R_m basis is the crucial step that enables the method.

The original periodic summation algorithm uses the FMM to compute ϕ_{near} and solves a linear least squares problem to compute ϕ_{far}. This approach allows the c_m coefficients to be computed in a manner which avoids dealing with them directly. However, c_0 is not recovered by this method and must be handled separately. The original algorithm was designed for problems where c_0 can be ignored, but this is not case for the NUFFT. Because of this, and since values of c_m for $m > 0$ are small and can be bounded away or estimated, we deal with the coefficients directly and avoid the least squares problem. The next two sections bound these coefficients and discuss their approximation.

4.1. Necessary Conditions

The periodic summation method does not converge if a few basic necessary conditions are not met. We approximate ϕ_{far} by:

$$\phi_{\text{far}}(y) = \varepsilon_{\text{far}}^{(P)} + \sum_{m=0}^{P-1} c_m R_m(y - \pi). \tag{13}$$

For the method to converge, we require $\varepsilon_{\text{far}}^{(P)}$ and each c_m to converge. The following lemmas establish this.

Lemma 1. The coefficient c_0 is finite.

Proof. A bit of algebra yields:

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

Then, for each $k = 0, \ldots, K$, we have:

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

when $(x_k - \pi)^2 / 4\pi^2 \leq (n+1)^2$. This holds, since $n \in \mathbb{N}$ and $0 \leq x_k < 2\pi$ for $k = 0, \ldots, K-1$. □

Lemma 2. The coefficients c_m for $m > 0$ are finite.

Proof. From (12), we have:

$$c_m = -\sum_{k=0}^{K-1} (-1)^k f_k \sum_{p=n+1}^{\infty} \frac{1}{(x_k + 2\pi p - \pi)^{m+1}}. \tag{14}$$

Then, since $\int_{n+1}^{\infty} (x_k + 2\pi t - \pi)^{-m-1} dt < \infty$ for each $m > 0$ and $n > 0$, we have that c_m converges for each $m > 0$. □

Lemma 3. The error term for the approximation to $\phi_{\text{far}}(y)$ given by $\varepsilon_{\text{far}}^{(P)}$ is finite and satisfies:

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

where Ei is the exponential integral [14].

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

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

since $|y - \pi| \leq \pi$, and $|x_k - 2\pi p - \pi| \geq \pi|2p-1|$. We can bound each side of the inner summation over p as:

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

Since each of these terms are positive and decreasing, we have:

$$\sum_{p=n+1}^{\infty} \frac{(2p \pm 1)^{-P-1}}{\log(2p \pm 1)} \leq \int_{p=n+1}^{\infty} \frac{(2p \pm 1)^{-P-1}}{\log(2p \pm 1)} dp$$
$$= \frac{1}{2} \int_{P \log(2(n+1) \mp 1)} \frac{dy}{ye^y},$$

where the equality follows from the change of variables $x = 2p \pm 1$ and $y = p \log x$. The final integral is the negative of the exponential integral function Ei evaluated at $-P \log(2(n+1) \mp 1)$. Combining our results so far, we have:

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

Using the monotonicity of Ei gives us our result. □

Lemmas 1–3 ensure that the periodic summation method works. Additionally, the error bound developed in Lemma 3 provides a useful criterion for choosing the n and P parameters.

4.2. Computation of Fitting Coefficients

We can use the bounds in Lemmas 1 and 2 to estimate c_m for $m \geq 0$. From Lemma 1, we note that c_0 is approximately given by:

$$c_0 \approx \frac{1}{4\pi} \sum_{k=0}^{K-1} (-1)^k f_k \log\left(\frac{\alpha + x_k}{\gamma - x_k} \cdot \frac{\beta + x_k}{\alpha - x_k}\right), \qquad (15)$$

where $\alpha = (2n+1)\pi$, $\beta = (2n-1)\pi$, and $\gamma = (2n+3)\pi$. From Lemma 2, we have:

$$c_m \approx \frac{-1}{2m} \sum_{k=0}^{K-1} (-1)^k f_k \left(\frac{1}{x_k + \alpha} + \frac{1}{x_k + \beta}\right). \qquad (16)$$

Altogether, these approximations can be evaluated in $O(PK)$ time, and used to improve the accuracy of our method by approximately imposing periodic boundary conditions.

5. AN ADAPTIVE FAST MULTIPOLE METHOD

Adaptive algorithms for the FMM have been proposed; we describe our own contribution here. Definitions of FMM-specific jargon can be looked up in standard sources [13, 15]. When computing ϕ_{near} in Procedure 3, source points correspond to an equispaced grid of K points between $-2\pi n$ and $2\pi(n+1)$. Target points only occur within the interval $[0, 2\pi)$. Our adaptive FMM builds a set of "translation stencils": data structures that indicate whether or not a given translation needs to be computed.

Table 1. The number of $\mathbf{S}|\mathbf{R}$ translations required by Procedure 2.

L	4	6	8	10	12
Max. # $\mathbf{S}\|\mathbf{R}$	93	381	1533	6141	24573
n	# $\mathbf{S}\|\mathbf{R}$				
1	34	136	526	2068	8218
2	27	90	327	1254	4947
3	27	72	246	915	3552
4	18	60	192	708	2766
5	18	48	162	582	2268
6	18	48	141	507	1923
7	18	48	132	450	1686
8	18	39	117	390	1482

Procedure 1 Recursively mark FMM translation stencils.

1: Initialize $\mathbf{S}|\mathbf{S}$, $\mathbf{S}|\mathbf{R}$, and $\mathbf{R}|\mathbf{R}$ translation stencils.
2: **for all** nodes in the current node's E4 neighborhood[1] **do**
3: **if** the arc from the node in the E4 neighborhood to the current node hasn't been marked in the $\mathbf{S}|\mathbf{R}$ translation stencil **then**
4: Mark the corresponding entry in the translation stencil.
5: Mark all $\mathbf{S}|\mathbf{S}$ translations beneath the node in the E4 neighborhood.
6: **end if**
7: **if** the current node is not at the highest level of the translation hierarchy **then**
8: Mark the current node in the $\mathbf{R}|\mathbf{R}$ translation stencil.
9: Recursively run Procedure 1 at the current node's parent.
10: **end if**
11: **end for**

Applying $\mathbf{S}|\mathbf{R}$ operators is the slowest part of the FMM. The reasons are twofold: three times as many $\mathbf{S}|\mathbf{R}$ operators as $\mathbf{R}|\mathbf{R}$ or $\mathbf{S}|\mathbf{S}$ operators are applied, and the $\mathbf{S}|\mathbf{R}$ operator for this kernel is a dense matrix (the others are triangular—see Equations 5 and 7). Asymptotically, $O(2^L)$ operators are applied, although L is usually small—e.g., $L = 4$. Our adaptive FMM dramatically reduces the number of $\mathbf{S}|\mathbf{R}$ and $\mathbf{R}|\mathbf{R}$ translations needed (see Table 1 for the number of $\mathbf{S}|\mathbf{R}$ operators—we note that the number of $\mathbf{S}|\mathbf{R}$ translations remains approximately three times the number of $\mathbf{R}|\mathbf{R}$). There are large improvements: with $n = 1$, only one third the number of $\mathbf{S}|\mathbf{R}$ and $\mathbf{R}|\mathbf{R}$ operators are required. The maximum reduction achieved is even more significant—for $L = 12$ and $n = 8$, only **6.03%** the number of $\mathbf{S}|\mathbf{R}$ operators are required. Our adaptive algorithm is straightforward:

Procedure 2 Adaptive FMM.

1: Run Procedure 1.
2: Run the usual FMM algorithm, but only compute translations which have been marked by Procedure 1.

Using Procedure 2 and our estimates for each c_m, we can compute the NUFFT. There is some question of choosing L optimally—this is covered elsewhere [12, 13], and corresponds to solving a simple optimization problem derived from an analysis of the algorithm.

[1] The E4 neighborhood [12, 13] is also called an interaction list [15].

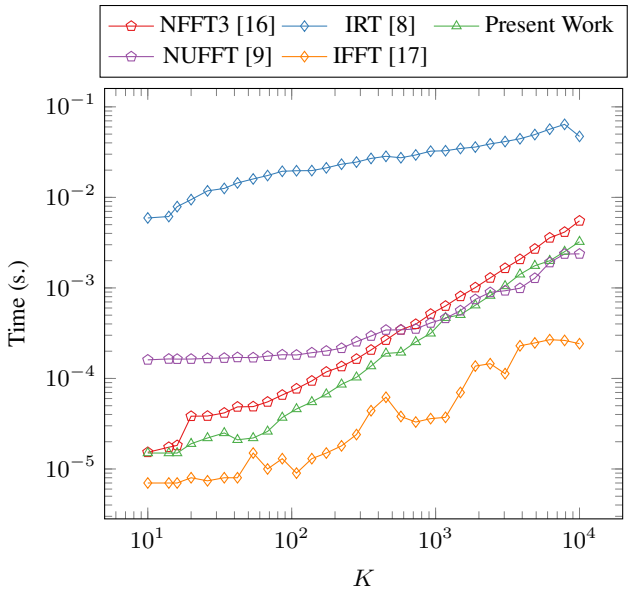

Fig. 1. Characteristic runtimes for varying bandwidths (K), plotted for each NUFFT implementation ($n = 1$, $P = 2$).

For our numerical experiments, we experimentally determined the optimal choice of L. Our NUFFT algorithm follows:

Procedure 3 Bandlimited interpolation (our main algorithm).

Choose $L \geq 2$ optimally for use with the FMM.
for $j = 0, \ldots, J$ **do**
  Compute $\phi_{\text{near}}(\tilde{x}_j)$ using Procedure 2.
end for
Compute c_0 from (15).
for $m = 0, \ldots, P-1$ **do**
  Compute c_m from (16).
end for
for $j = 0, \ldots, J$ **do**
  Compute $\phi_{\text{far}}(\tilde{x}_j)$ from (13).
  Compute $\phi(\tilde{x}_j) = \phi_{\text{near}}(\tilde{x}_j) + \phi_{\text{far}}(\tilde{x}_j)$.
end for

We defer a full analysis of Procedure 3's complexity, but note that the complexity of the original FMM-based NUFFT is not compromised, as the salient difference is the computation of the fitting coefficients.

6. NUMERICAL RESULTS

Our experiments were run on a single 2.2 GHz Intel Core i7 core. We compared our implementation with NFFT3 [16], NUFFT [9, 18], and IRT [8]. Our implementation is single-threaded at present—we compiled the other libraries with parallelism disabled for a fair comparison. Each library is relatively well-optimized. We conducted the following numerical experiments: for varying K and for different test functions [19, 20, 21], we measured the ℓ_∞ error of each method (see Figure 2—the NUFFT library was omitted since it attained machine precision). Next, we timed each method for the same range of K. As a base line, we timed the numpy library's IFFT [17] (see Figure 1). We found that for an optimistic choice of parameters ($n = 1, P = 2$—the minimum possible), our algorithm was reasonably accurate and competitive in terms of speed. We found that

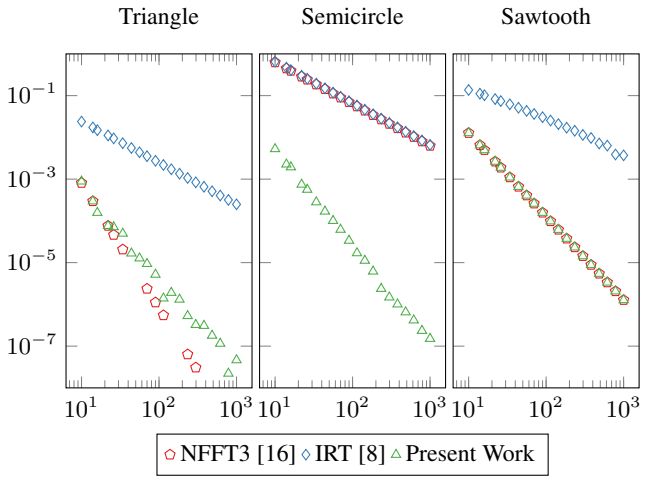

Fig. 2. The ℓ_∞ error for different bandwidths (K) and for each NUFFT implementation ($n = 1, P = 2$). Note: for some problem sizes, NFFT3 attained machine precision.

NFFT3 possesses similar performance and accuracy characteristics, but that these characteristics are more erratic. The NUFFT library is very accurate, but slow for small problem sizes.

As a final test, for different choices of block size, we computed a constant Q transform with 24 filters per octave in order to estimate the CPU load due to Procedure 3. We found that for a range of block sizes and sampling rates which are used commonly, Procedure 3 generally incurred low single-digit CPU load (see Table 2), which is eminently reasonable for soft realtime audio processing.

7. CONCLUSIONS AND FUTURE WORK

In this work, we extend an existing NUFFT algorithm [2]. In the conclusions of the original work, the method is passed over in favor of other algorithms developed by the same authors [1]. Our contribution has been to extend the method to enhance the algorithm's precision for neglible added cost, and to greatly reduce the required number of operations by providing an adaptive FMM which suits the problem data. Our implementation, although only modestly optimized, is competitive with libraries that have existed for years.

Several future lines of research will be pursued. First, the original FMM-based NUFFT [7] is based on a Chebyshev approximation of Φ, with the translation operators compressed and accelerated using singular value decompositions—this approach will be explored. We will also investigate applying parallel and GPU implementations of the FMM [22].

Table 2. Estimated CPU load for computing a 24 filter per octave constant Q transform using a modified Procedure 3.

Block Size	44.1KHz	48KHz	88.2KHz	96KHz
32	5.79%	4.35%	9.1%	10.2%
64	3.31%	3.3%	6.06%	7.05%
128	2.24%	2.85%	7.92%	5.25%
256	2.31%	4.12%	3.34%	3.6%
512	1.31%	1.65%	3.05%	2.87%
1024	1.46%	1.05%	1.94%	2.46%

8. REFERENCES

[1] A. Dutt and V. Rokhlin, "Fast Fourier transforms for nonequispaced data," *SIAM J. Sci. Comput.*, vol. 14, pp. 1368–1393, 1993.

[2] A. Dutt and V. Rokhlin, "Fast Fourier transforms for nonequispaced data, II," *Appl. Comput. Harmon. A.*, pp. 85–100, 1995.

[3] N. A. Gumerov and R. Duraiswami, "A method to compute periodic sums," *J. Comput. Phys.*, vol. 272, pp. 307–326, 2014.

[4] J. C. Brown, "Calculation of a constant Q spectral transform," *J. Acoust. Soc. Am.*, vol. 89, no. 1, pp. 425–434, 1991.

[5] J.C. Brown and M.S. Puckette, "An efficient algorithm for the calculation of a constant q transform," *J. Acoust. Soc. Am.*, vol. 92, 1992.

[6] C. Schörkhuber and A. Klapuri, "Constant-q transform toolbox for music processing," in *Proc. of 7th Sound and Music Computing Conference*, 2010.

[7] A. Dutt, M. Gu, and V. Rokhlin, "Fast algorithms for polynomial interpolation, integration, and differentiation," *SIAM J. Numer. Anal.*, vol. 33, no. 5, pp. 1689–1711, 1996.

[8] J. A. Fessler and B. P. Sutton, "Nonuniform Fast Fourier Transforms Using Min-Max Interpolation," *IEEE Transactions on Signal Processing*, vol. 51, no. 2, pp. 560–574, Feb. 2003.

[9] L. Greengard and J.-Y. Lee, "Accelerating the Nonuniform Fast Fourier Transform," *SIAM Review*, vol. 46, no. 3, pp. 443–454, 2004.

[10] D. Potts, G. Steidl, and M. Tasche, "Fast Fourier transforms for nonequispaced data: A tutorial," in *Modern Sampling Theory: Mathematics and Applications*, chapter 12, pp. 247–270. Springer, 1998.

[11] T. Schanze, "Sinc interplation of discrete periodic signals," *IEEE Transactions on Signal Processing*, vol. 43, no. 6, pp. 1502–1503, 1995.

[12] N. A. Gumerov, R. Duraiswami, and E. A. Borovikov, "Data structures, optimal choice of parameters, and complexity results for generalized multilevel fast multipole methods in d dimensions," Tech. Rep., University of Maryland Institute for Advanced Computer Studes, 2002-2003.

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

[14] M. Abramowitz and I. A. Stegun, Eds., *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, National Bureau of Standards, 1972.

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

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

[17] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy array: a structure for efficient numerical computation," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 22–30, 2011.

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

[19] Eric W. Weisstein, "Fourier series—triangle wave," http://mathworld.wolfram.com/FourierSeriesTriangleWave.html, From MathWorld—A Wolfram Web Resource.

[20] Eric W. Weisstein, "Fourier series—semicircle," http://mathworld.wolfram.com/FourierSeriesSemicircle.html, From MathWorld—A Wolfram Web Resource.

[21] Eric W. Weisstein, "Fourier series—sawtooth wave," http://mathworld.wolfram.com/FourierSeriesSawtoothWave.html, From MathWorld—A Wolfram Web Resource.

[22] N. A. Gumerov and R. Duraiswami, "Fast multipole methods on graphics processors," *J. Comput. Phys.*, vol. 227, no. 18, pp. 8290–8313, 2008.