Approximating the Analytic Fourier Transform with the Discrete Fourier Transform

Jeremy Axelrod*
Department of Physics, University of California, Berkeley
(Dated: 26 May 2015)

The Fourier transform is approximated over a finite domain using a Riemann sum. This Riemann sum is then expressed in terms of the discrete Fourier transform, which allows the sum to be computed with a fast Fourier transform algorithm more rapidly than via a direct matrix multiplication. Advantages and limitations of using this method to approximate the Fourier transform are discussed, and prototypical MATLAB codes implementing the method are presented.

I. Introduction

The Fourier transform is a ubiquitous analytical mathematical tool. However, in many problems of interest analytic expressions for transformed functions do not exist, or the function to be transformed is only known at a set of discrete points as is the case for most real-world experimental data. In both of these cases, it is necessary to approximate the Fourier transform on a set of discrete points. This can be done by approximating the integral in the Fourier transform as a Riemann sum. Such a summation implemented as a single matrix multiplication by the vector of points to be transformed results in an undesirable algorithm complexity scaling of $O(N^2)$. Here, an algorithm is presented which allows this Riemann sum to be expressed in terms of the discrete Fourier transform (DFT), which can in turn be computed via a fast Fourier transform (FFT) with complexity scaling of $O(N \log N)$ [1]. The overall complexity of this algorithm also scales as $O(N \log N)$.

II. Method

A. Definitions

Let the Fourier transform, $\tilde{f}(\omega)$, of a function $f(t') : \mathbb{R} \rightarrow \mathbb{C}$ be defined as

$$\tilde{f}(\omega) := \sqrt{\frac{|b|}{(2\pi)^{1-a}}} \int_{-\infty}^{\infty} f(t')e^{ib\omega t'} dt'$$  \hspace{1cm} (1)

where

$$f(t) := \sqrt{\frac{|b|}{(2\pi)^{1+a}}} \int_{-\infty}^{\infty} \tilde{f}(\omega')e^{-ib\omega't} d\omega'$$  \hspace{1cm} (2)

is the inverse Fourier transform of $\tilde{f}$, and $a, b \in \mathbb{R}, b \neq 0$ are arbitrary constants chosen by convention [2]. Let $\vec{x} := [x_1, x_2, \cdots, x_N]$ be a vector of length $N$, and let $j, j', k, k' = 1, 2, 3, ..., N$. The discrete Fourier transform, $\vec{X} := [X_1, X_2, \cdots, X_N]$, of $\vec{x}$ is then defined to be such that

$$X_k := \sum_{j=1}^{N} x_j e^{-\frac{2\pi i}{N}(j-1)(k-1)}$$  \hspace{1cm} (3)

where

$$x_j := \frac{1}{N} \sum_{k'=1}^{N} X_k e^{\frac{2\pi i}{N}(j-1)(k'-1)}$$

is the inverse discrete Fourier transform of $\vec{X}$.

B. Approximating the forward transform

Let the function $f(t')$ be represented by a vector $\vec{x}$ of length $N$, so that $x_{j'} = f(t'_{j'})$ where $t'_{j'} = \tau' (j' - 1) + t'_1$ constitutes a vector of evenly-spaced points beginning at $t'_1$. Then (1) can be approximated via a Riemann sum, $\hat{f}(\omega)$, as

$$\hat{f}(\omega) \approx \hat{F}(\omega) := \sqrt{\frac{|b|}{(2\pi)^{1-a}}} |\tau'| \sum_{j'=1}^{N} x_{j'} e^{ib\omega t'_{j'}}$$  \hspace{1cm} (4)

Letting max $\{t'_{j'}\}$, min $\{t'_{j'}\}$ denote the maximum and minimum values in $\{t'_{j'}\}$, respectively, then

$$\lim_{N \rightarrow \infty} \hat{F}(\omega) = \sqrt{\frac{|b|}{(2\pi)^{1-a}}} \int_{\min \{t'_{j'}\}}^{\max \{t'_{j'}\}} f(t')e^{ib\omega t'} dt'$$

when the limits of integration $\min \{t'_{j'}\}$ and $\max \{t'_{j'}\}$ are held fixed so that $|\tau'|$ decreases as $N$ increases. Therefore, $\hat{F}(\omega)$ can be viewed as a Riemann approximation of the Fourier transform of the function

$$f_0(t') := \begin{cases} f(t') & \forall t' \in [\min \{t'_{j'}\}, \max \{t'_{j'}\}] \\ 0 & \text{elsewhere} \end{cases}$$

* jaxelrod@berkeley.edu
so that $\tilde{F}(\omega)$ approximates $\hat{f}(\omega)$ well for large $N$, and for $\{t_j, \rangle$ and $f(t')$ such that

$$
\int_{-\infty}^{t_{\min}} f(t') e^{i\omega t'} dt' + \int_{t_{\max}}^{\infty} f(t') e^{i\omega t'} dt' \leq \int_{t_{\min}}^{t_{\max}} f(t') e^{i\omega t'} dt' \forall \omega
$$

Computing the sum in (4) directly for $N$ values of $\omega$ (denoted by $\tilde{\omega} := [\omega_1, \omega_2, \cdots, \omega_N]$) can be written as a matrix multiplication:

$$
\tilde{F}(\tilde{\omega}) = \sqrt{\frac{|b|}{(2\pi)^{1-a} |\tau|^a}} \left[ e^{ib\omega_1 t'_1} e^{ib\omega_2 t'_2} \cdots e^{ib\omega_N t'_N} \right]
$$

where

$$
X_k = \sum_{j=1}^{N} x_j e^{ib\omega_k (t_j - t'_j)} = e^{-ib\omega_k t'_1} \sqrt{\frac{(2\pi)^{1-a}}{|b||\tau|}} \tilde{F}(\omega_k)
$$

so that

$$
\tilde{F}(\omega + \frac{2\pi}{\tau' b} m) = \sqrt{\frac{|b|}{(2\pi)^{1-a} |\tau'|}} \sum_{j=1}^{N} x_j e^{ib\omega t'_j} e^{i\frac{2\pi}{\tau' b} mt'_j} = e^{i\frac{2\pi}{\tau' b} mt'_1} \tilde{F}(\omega)
$$

As such, $\tilde{F}$ has been expressed in terms of the DFT, $X$, for $\tilde{\omega} = \{\omega_k\}$ as defined by (6). However, it is sometimes desirable to retrieve the values of $\tilde{F}$ for $\omega \not\in \{\omega_k\}$, e.g. $\omega < 0$ if $b < 0$. To that end, note that for $m \in \mathbb{Z},$

$$\omega_k := -\frac{2\pi}{\tau' b N} (k - 1)$$

so that $|\tilde{F}|$ is periodic with period $\frac{2\pi}{\tau' b}$, with a phase shift of $\frac{2\pi}{\tau' b} m t'_1$. Since $\tilde{\omega}$ spans an entire period (from 0 to $\frac{2\pi}{\tau' b} N - 1$), $\tilde{F}$ can be determined from $\tilde{F}(\tilde{\omega})$ for any integer multiple of $\frac{2\pi}{\tau' b}$. This is useful in practice, since it is often desirable to retrieve $\tilde{F}$ on the interval $\omega \in [-|\omega_{nyq}|, +|\omega_{nyq}|]$ for $\omega_{nyq} := \frac{\pi}{\tau' b}$ the Nyquist frequency in order to display aliases in a more readily interpretable context—that is, negative frequency components will appear below $\omega = 0$ instead of above $\omega = \omega_{nyq}$. A comparison between the DFT and the Riemann
sum approximation to the Fourier transform is shown in figure 1. Clearly, if the vector being transformed is not entirely real, considering the DFT only below the Nyquist frequency index disregards possible asymmetries between positive and negative frequency components. Also, the vertical scaling of the Riemann sum approximation is independent of \( N \) unlike with the DFT, making direct comparisons of spectral power density between transforms of vectors of different lengths possible.

C. Approximating the inverse transform

The inverse transform can be treated in exactly the same way as the forward transform. Letting the function \( \hat{f}(\omega') \) be represented by a vector \( \vec{X} \) of length \( N \) so that \( X_{k'} = \hat{f}(\omega') \) where \( \omega' \neq W'(k'-1) + \omega' \) constitutes a vector of evenly-spaced points beginning at \( \omega' \), then (2) can be approximated via a Riemann sum, \( F(t) \), as

\[
f(t) \approx F(t) := \sqrt{\frac{|b|}{(2\pi)^{1+a}}} \left| W' \right| \sum_{k'=1}^{N} X_{k'} e^{-ib\omega'k't} \tag{7}
\]

Then, after defining \( t_j := -\frac{2\pi}{W'N} (j - 1) \),

\[
F(t_j) = \sqrt{\frac{|b|}{(2\pi)^{1+a}}} N |W'| \ e^{-ib\omega't_j} x_j
\]

and for \( l \in \mathbb{Z} \),

\[
F \left( t + \frac{2\pi}{W'b} \right) = e^{-i\frac{2\pi}{W'b} \omega'_l} F(t)
\]

similarly to before.

D. Invertibility of the approximate transforms

It is of interest to know if the above Riemann sum approximations of the forward/inverse Fourier transforms are inverses of each other, i.e. is \( F \) applied to \( \vec{X} = \vec{F}(\vec{x}) \), where \( \vec{F} \) is applied to vector \( \vec{x} \), equal to \( \vec{x} \)? Letting \( x_{j'} \) be defined on \( t_{j'} = \tau'(j'-1) + t'_j \), and using equations (4) and (7), the inverse transform of the forward transform of \( \vec{x} \) can be written as

\[
F(t) = \sqrt{\frac{|b|}{(2\pi)^{1+a}}} \left| W' \right| \sum_{k'=1}^{N} \left( \sqrt{\frac{|b|}{(2\pi)^{1+a}}} \right| \sum_{j'=1}^{N} x_{j'} e^{ib\omega'k't'} \right) e^{-ib\omega'k't} = \frac{|b|}{2\pi} |W'| |\tau'| \sum_{j'=1}^{N} x_{j'} e^{ib(t_j'-t)} \omega' \left( \sum_{k'=1}^{N} \left( e^{ibW'(t_j'-t)} \right)^{(k'-1)} \right) \tag{8}
\]

assuming that \( \omega'_{j'} = W'(k'-1) + \omega' \) in accordance with equation (6) and the accompanying discussion on discrete shifts. The sum in parentheses in (8) can be evaluated as a geometric sum:

\[
\sum_{k'=1}^{N} \left( e^{ibW'(t_j'-t)} \right)^{k'-1} = \begin{cases} 
1 - e^{iN\omega'(t_j'-t)} & \text{for } e^{ibW'(t_j'-t)} \neq 1 \\
N & \text{for } e^{ibW'(t_j'-t)} = 1
\end{cases}
\]

In order to consider invertibility, it must be that \( t \in \{t_{j'}\} \). If this is the case, then \( \forall j' \in \{1,2,3, ..., N\} \), \( t_{j'} - t = m|\tau'| \) for some \( m \in \{Z \mid |m| \leq N - 1\} \). That is, the difference between \( t \) and \( t_{j'} \) is always an integer multiple of \( |\tau'| \). Thus,

\[
F(t) = \frac{|b|}{2\pi} |\tau'| |W'| N x_{j'} = x_{j'} \quad \text{when } t = t_{j'}
\]

which proves that the Riemann sum approximations are indeed inverses of each other. Since the method for calculating the Riemann sum approximation
FIG. 1. Comparison of the magnitude of the DFT (a) and Riemann sum approximation to the Fourier transform (b) for \( f(t) = \sin(t) + 0.1e^{-2it} \) sampled at 201 evenly-spaced points on the interval \([-100, 100]\) with \( a = 0, b = -1 \). The DFT is more difficult to directly interpret because of aliasing across the Nyquist frequency at \( k = 101 \).

of the inverse Fourier transform using the inverse DFT can only return \( t \) which are integer multiples of \( \frac{2\pi}{NW} \). The transforms \( F(t) \) and \( \tilde{F}(\omega) \) specified above are restricted to being each others’ inverse under the single condition that the input \( \{t'\} \) is of the form \( t'_j = \tau'(j' - 1 + l) \) so that it is possible for the inverse transform to return \( t \in \{t'_j\} \).

III. Performance

A prototypical MATLAB function, \( \text{FT} \), which implements the approximation method described above for the forward Fourier transform is given in Section A1. The Riemann sum approximation of the Fourier transform of \( f(t) = \text{rect}(t - 1) \) calculated using \( \text{FT} \) is shown in figure 2. The analytic Fourier transform of \( f(t) \) for \( a = 0, b = -1 \) is

\[
\hat{f}(\omega) = \frac{1}{\sqrt{2\pi}} e^{-i\omega} \sin\left(\frac{\omega}{2}\right) (9)
\]

and is also plotted in figure 2 for comparison. Referencing the code in \( \text{FT} \), it is evident that adapting the DFT to the Riemann sum approximation only incurs additional computational costs scaling as \( O(N) \) from the element-wise multiplications by phase factors. Since the FFT complexity scales as \( O(N \log N) \), the overall complexity of the adaptation algorithm scales as \( O(N \log N) \). This is demonstrated in figure 3.

IV. Conclusions

It has been shown that the Riemann sum approximation to the Fourier integral over a discrete finite domain can be expressed in terms of the discrete Fourier transform and is therefore calculable using a fast Fourier transform algorithm, which reduces the complexity of the problem from \( O(N^2) \) for a direct matrix multiplication implementation of the sum to \( O(N \log N) \). The Riemann sum approximation is useful when a discrete approximation to the continuous Fourier transform is required, and it may be preferable to the discrete Fourier transform in some cases because it is directly interpretable in the context of the Fourier transform. The Riemann sum approximation of the inverse Fourier transform applied to the Riemann sum approximation of the forward Fourier transform of a vector \( \vec{x} \) returns the same vector \( \vec{x} \)—that is, the approximated transforms are still inverses of each other. However, the method pre-

FIG. 2. The real (solid blue line) and imaginary (solid red line) parts of the analytic Fourier transform of \( f(t) \) given by equation (9) are plotted with the real (blue ·) and imaginary (red ·) parts of the Riemann sum approximation as computed by MATLAB function \( \text{FT} \). \( \text{FT} \) used 201 evenly-spaced samples of \( f(t) \) on the interval \([-10, 10]\) with \( a = 0, b = -1 \). The approximation performs well for \( |\omega| \ll \frac{1}{\tau'} \).
FIG. 3. Algorithm execution times as a function of input vector length for a) the Riemann approximation method as given in the function \( \text{FT} \) (see Section A1) and b) MATLAB’s built-in fast Fourier transform function, \( \text{fft} \). c) The ratio of times in (a) to (b). The ratio approaches a constant value as \( N \) increases, indicating that the complexity of function \( \text{FT} \) scales similarly to \( \text{fft} \)—that is, as \( O(N \log N) \). The transforms were performed on the rectangular function \( f(t) = \text{rect}(t - 1) \), with sample points evenly-spaced on \([-10, 10]\).

Presented here results in the domain of the transformed function being defined only on integer multiples of \( \frac{2\pi \Delta}{\Delta N} \) where \( \Delta \) is the spacing between values in the conjugate domain. Therefore, the transforms can only be implemented in such a way that they are each others’ inverse if the input domain is defined on integer multiples of some number. Future work could try to alleviate this integer multiple constraint, or could try express higher-order integral approximations (like Simpson’s rule) of the Fourier integral in terms of the discrete Fourier transform.

[1] Duhamel, P. and M. Vetterli, “Fast Fourier Transforms: A Tutorial Review and a State of the Art,” Signal Processing, Vol. 19, April 1990, pp. 259-299.
[2] Weisstein, Eric W. “Fourier Transform.” From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/FourierTransform.html
A simple implementation of the forward transform in MATLAB is given below. The function \( FT \) takes input vectors of length \( N \) \( t,f \) and input scalars \( w_1,a,b \) (with \( w_1 \) an integer multiple of \( \frac{2\pi}{\tau b N} \)) and returns output vectors of length \( N \) \( w,ff \) where \( w(1) = w_1 \).

\[
function \ [w,ff] = FT(t,f,w_1,a,b)
\]
\[
N = numel(t); \quad \% \text{size of input vectors}
\]
\[
\text{tau} = ((\text{max}(t)\text{-}\text{min}(t))./(N-1)); \quad \% \text{input sampling period}
\]
\[
W = -2*\pi/(\text{tau}\times b\times N); \quad \% \text{define frequency spacing}
\]
\[
w_k = W\times(0:N-1); \quad \% \text{define frequency vector beginning at zero}
\]
\[
X = \sqrt{\text{abs}(b)}/((2\times \pi).^{1-a})\times \text{tau}\times \exp(1i\times b\times w_k\times t(1))\times \text{fft}(f);
\]
\[
\% \text{unshifted Riemann sum approximation using built-in "fft" function}
\]
\[
w = w_k+w_1; \quad \% \text{shift frequency vector so that } w(1) = w_1
\]
\[
n = \text{round}(w_1/W); \quad \% \text{number of indices that } w \text{ is shifted from } w_k \text{ by}
\]
\[
m = \text{zeros(size}(w_k)); \quad \% \text{initialize vector of periodic shift indices}
\]
\[
\text{if } n > 0
\]
\[
\quad \% \text{determine } m \text{ for each entry in } w:
\quad \text{idx} = \text{mod}(n,N);
\quad m(N-\text{idx}:N) = \text{floor}(n/(N+1)) + 1;
\quad m(1:N-\text{idx}-1) = \text{floor}(n/(N+1));
\quad \text{inds} = \text{circshift}(1:N,-\text{idx},2); \quad \% \text{circularly shift indices of } X \text{ to}
\quad \% \text{corresponding new } w \text{ positions}
\]
\[
\text{elseif } n < 0
\]
\[
\quad \% \text{determine } m \text{ for each entry in } w:
\quad \text{idx} = \text{mod}(-n,N);
\quad m(1:\text{idx}) = -((\text{floor}(-n/(N+1)) + 1));
\quad m(\text{idx}+1:N) = -\text{floor}(-n/(N+1));
\quad \text{inds} = \text{circshift}(1:N,\text{idx},2); \quad \% \text{circularly shift indices of } X \text{ to}
\quad \% \text{corresponding new } w \text{ positions}
\]
\[
\text{end}
\]
\[
\text{phase} = (\exp(1i\times(2\times \pi/\text{tau}\times t(1))))\times m; \quad \% \text{generate phases to shift by}
\]
\[
\text{ff} = \text{phase}\times X(\text{inds}); \quad \% \text{phase shift elements of circularly-shifted } X, \text{ output}
\]
\[
\text{end}
\]
The corresponding inverse transform function IFT is also given for convenience:

```matlab
function [t,ξ] = IFT(w,ff,τ₁,a,b)

N = numel(w); % size of input vectors
W = ((max(w)-min(w))./(N-1)); % input sampling period
τ = -2*pi/(W*b*N); % define time spacing
τ = τ.*(0:N-1); % define time vector beginning at zero
x = sqrt(abs(b)./(2*pi).ˆ(1+a)).*W.*exp(-1i*b.*τ).*N.*ifft(ff);
% unshifted Riemann sum approximation using built-in "ifft" function
τ = τ+τ₁; % shift time vector so that τ(1) = τ₁
n = round(τ₁/τ); % number of indices that τ is shifted from τ by
m = zeros(size(τ)); % initialize vector of periodic shift indices
if n > 0
    % determine m for each entry in w:
    idx = mod(n,N);
    m(N-idx:N) = floor(n./(N+1)) + 1;
    m(1:N-idx-1) = floor(n./(N+1));
    inds = circshift(1:N,-idx,2); % circularly shift indices of X to
    % corresponding new w positions
else n < 0
    % determine m for each entry in w:
    idx = mod(-n,N);
    m(1:idx) = - floor(-n./(N+1)) + 1;
    m(idx+1:N) = - floor(-n./(N+1));
    inds = circshift(1:N,idx,2); % circularly shift indices of X to
    % corresponding new w positions
end
phase = (exp(-1i*(2*pi/W)*w(1))).ˆm; % generate phases to shift by
f = phase.*x(inds); % phase shift elements of circularly-shifted x, output
% answer
end
```