1 Outline

Agenda: Non-Uniform Fast Fourier Transforms

1. Type I
2. Type II

Last Time: We concluded our discussion of computerized tomography by examining the computational issues of ill-posedness. Radon inversion was identified to be an ill-posed problem, resulting from the unboundedness of the inverse transform. This, we mentioned, is equivalent to the inverse because discontinuous, meaning two disparate images can have Radon transforms that are very close. Consequently, numerical uncertainty resulting from noise and discretization create ambiguity as to which signal produces observed data. To resolve this ambiguity, we introduced the method of regularization, which identifies an inverse solution that closely matches the observed data while also exhibiting function characteristics known a priori.

2 Introduction to non-uniform Fourier transforms

In our study of X-ray tomography and the Radon transform, we identified a key role played by the Fourier transform. We shall soon see that Fourier transforms are also central in another mode of medical imaging: magnetic resonance (MR). And as in many applications, the relevant Fourier data is non-uniform, in time and/or in frequency. We introduced this notion in Lecture 8, and today we describe some of the algorithms available to compute non-uniform discrete Fourier transforms.

Recall that the general form of a discrete Fourier transform is

\[ y[k] = \sum_{n=0}^{N-1} x(t_n)e^{-i\omega_k t_n}, \]  \hspace{1cm} (1)

for \( k = 0, \ldots, M - 1 \), where one has samples \( \{x(t_n)\} \) on the grid \( \{t_n\}_{n=0}^{N-1} \), possibly non-equispaced, and one wants to compute the Fourier transform at frequencies \( \{\omega_k\}_{n=0}^{N-1} \), also possibly non-equispaced. From (1), we could compute \( \mathbf{y} \) by directly evaluating \( \mathbf{y} = \mathbf{A} \mathbf{x} \), where \( \mathbf{A} \) is the \( M \times N \) matrix with entries

\[ A_{kn} = e^{-i\omega_k t_n}, \]
although doing so raises the same problem as did the uniform DFT: high computational cost. When \( N \) and \( M \) become large as the signals considered are two- or three-dimensional, it is difficult to even store in memory the matrix \( A \). The use of alternative computational schemes, known as Non-Uniform FFTs (NUFFTs), is thus essential.

Consider the case when at least one of the spatial and frequency domains is sampled uniformly. Since the inverse Fourier transform is just (a multiple of) the adjoint of the Fourier transform, the two possible scenarios are equivalent. So without loss of generality, assume frequencies are taken uniformly. That is, \( \omega_k = k \) for \( k = 0, \ldots, M - 1 \). In this case, (1) becomes

\[
y[k] = \sum_{n=0}^{N-1} x[n] e^{-ikt_n}.
\]

There are now two tasks to consider. First, given \( x \) sampled on a non-equispaced grid \( \{t_n\}_{n=0}^{N-1} \), how can \( y \) be computed efficiently? And conversely, how can \( x \) be recovered, on that non-equispaced grid, from the uniformly sampled \( y \)? An algorithm that performs the first task is called a Type I NUFFT, while one that performs the second is Type II. These operations have the form

\[
y = Ax \iff y[k] = \sum_{n=0}^{N-1} x[n] e^{-ikt_n}, \tag{2}
\]

and

\[
x = A^* y \iff x[n] = \sum_{k=0}^{M-1} y[k] e^{ikt_n}. \tag{3}
\]

How can we compute these expressions tractably? We cannot use Cooley & Tukey’s algorithm (see Lecture 7), as their approach breaks down when we sample or evaluate the transforms on a non-equispaced grid. But is there still a \( O(N \log N) \) algorithm to implement (2) and (3)? In what follows, we will do only slightly worse, while keeping approximation errors low.

Two algorithms are presented, the first for a Type I computation and the second for a Type II. Nevertheless, the difference between the two types is an adjoint, and so each algorithm can be adapted to the other type. Abstractly, the algorithms perform linear operations, meaning the steps can be reversed and individually “adjointed” to produce an algorithm of the opposite type:

\[
\text{Type I} : \quad L_1 \circ L_2 \circ \cdots \circ L_p \iff \text{Type 2} : \quad L_p^* \circ L_{p-1}^* \circ \cdots \circ L_1^*
\]

3 Dutt & Rohklin’s algorithm

Here we present Dutt & Rohklin’s [1] construction of a Type I NUFFT. To motivate their approach, note that we can write

\[
y[k] = \sum_{n=0}^{N-1} x[n] e^{-ikt_n} = \int \left( \sum_{n=0}^{N-1} x[n] \delta(t-t_n) \right) e^{-ikt} dt = \hat{X}(k),
\]

where

\[
X(t) = \sum_{n=0}^{N-1} x[n] \delta(t-t_n).
\]
By periodicity of the complex exponentials, we may assume without loss that \( \{t_n\}_{n=0}^{N-1} \subseteq [-\pi, \pi] \). We are trying to compute the values of the Fourier transform of \( X \) at \( \omega = 0, \ldots, M-1 \), given coefficients \( \{x[n]\}_{n=0}^{N-1} \) and non-uniform times \( \{t_n\}_{n=0}^{N-1} \).

If \( X \) were continuous and bandlimited, one could simply resample \( X \) on a sufficiently fine, equispaced grid, and then perform a standard FFT. This would yield an approximation of \( \hat{X} \) evaluated at \( k = 0, \ldots, M-1 \). Unfortunately, \( X \) is neither continuous nor bandlimited, but we can convolve it with a function \( \varphi \) to make it continuous and nearly bandlimited.

Let \( \varphi \) be a function \( L_1 \), such that \( |\varphi(t)| \) is negligible outside the interval \( [-t_0, t_0] \) for some \( t_0 > 0 \), and \( |\hat{\varphi}(\omega)| \) is negligible outside the interval \( [-\omega_0, \omega_0] \) for some \( \omega_0 > 0 \). Define \( H = X \ast \varphi \), which we can evaluate exactly for any \( t \), as we have the explicit expression

\[
H(t) = (X \ast \varphi)(t) = \sum_{n=0}^{N-1} x[n] \varphi(t - t_n).
\]

Since \( \varphi(t) \) decays rapidly, \( H \) is essentially time-limited. Using the convolution theorem, its Fourier transform is given by

\[
\hat{H}(\omega) = \hat{X}(\omega) \hat{\varphi}(\omega),
\]

and consequently \( H \) is also essentially bandlimited.

Under this setting the discretization

\[
H_d(t) = \sum_{\ell \in \mathbb{Z}} h_d[\ell] \delta(t - \ell \Delta t) \quad \text{with} \quad h_d[\ell] = H(\ell \Delta t) \quad \text{for} \quad \ell \in \mathbb{Z},
\]

with uniform spacing \( \Delta t > 0 \), has a finite number of significant terms, and the aliasing formula shows that

\[
\hat{H}_d(\omega) = \frac{1}{\Delta t} \sum_{k \in \mathbb{Z}} \hat{X} \left( \omega - \frac{2\pi k}{\Delta t} \right) \hat{\varphi} \left( \omega - \frac{2\pi k}{\Delta t} \right).
\]

From this, we obtain the approximation for \( \Delta t \) sufficiently small (i.e. \( \frac{\pi}{\Delta t} \geq \omega_0 \)),

\[
\Delta t \mathbb{I}\{|\omega| \leq \omega_0\} \hat{H}_d(\omega) \approx \hat{X}(\omega) \hat{\varphi}(\omega),
\]

or more generally

\[
\Delta t \mathbb{I}\{\omega \in [(2k - 1)\omega_0, (2k + 1)\omega_0]\} \hat{H}_d(\omega) \approx \hat{X}(\omega) \hat{\varphi}(\omega), \tag{4}
\]

for any \( k \in \mathbb{Z} \). The simple deconvolution step of dividing by \( \hat{\varphi}(\omega) \) completes the transform. This will be the approach we follow to compute the NUFFT.

Let \( \delta \geq 0 \) be such that \( H \) is essentially time-limited on \( [-1 + \delta \pi, 1 + \delta \pi] \). We typically have \( \pi \delta \leq t_0 \) and in many cases we can even choose \( \delta = 0 \). Let \( N_c \) be an even integer so that \( N_c \Delta t = 2\pi(1+\delta) \) (i.e. assume \( \Delta t = \frac{2\pi(1+\delta)}{N_c} \) for some large \( N_c \in 2\mathbb{Z} \)) and large enough that we can take

\[1\]For instance, a Gaussian
\( \omega_0 = \frac{\pi}{\Delta t} \). Then \( h_d[\ell] \approx 0 \) for \( |\ell| > N_c/2 \), and so we truncate \( h_d \) to a vector of length \( N_c \). Then

\[
\hat{h}_d[k] = \sum_{\ell=-N_c/2}^{N_c/2-1} h_d[\ell] e^{-\frac{i2\pi k \ell}{N_c}} \\
\approx \sum_{\ell \in \mathbb{Z}} H(\ell \Delta t) e^{-i\left(\frac{2\pi}{N_c \Delta t} k\right)(\ell \Delta t)} \\
= \hat{H}_d \left( \frac{2\pi}{N_c \Delta t} k \right) \\
\approx \frac{1}{\Delta t} \hat{X} \left( \frac{k}{1+\delta} \right) \hat{\varphi} \left( \frac{k}{1+\delta} \right). \tag{5}
\]

Consequently, for \( N_c \) sufficiently large, \( \hat{h}_d[k] \) corresponds to sampling \( \hat{X} \hat{\varphi} \) at points \( k/(1+\delta) \) for \( k = 0, \ldots, N_c - 1 \). So we make the approximation

\[
\hat{X} \left( \frac{k}{1+\delta} \right) \approx \Delta t \frac{\hat{h}_d[k]}{\hat{\varphi}(k/(1+\delta))}.
\]

When \( \delta \) can be made close to or equal to 0, the above yields exactly the values we are attempting to estimate. In practice one can choose \( \varphi \) in such a way that it is time-limited and essentially bandlimited. For instance (Beylkin [2], 1995),

\[
\hat{\varphi}(\omega) = \text{sinc}(\omega)^q,
\]

where \( q = 4 \) is empirically a good choice.

As desired, this algorithm is computationally tractable. The two main steps are: (i) evaluation of \( H \) on a grid of \( N_c \) points, and; (ii) computing a DFT. To compute the values of \( H \), since \( \varphi \) is essentially time-limited, we do not need to perform \( N N_c \) evaluations, but rather \( q N \), where \( q > 0 \) is the number of points in \([-t_0, t_0]\). Consequently, the computational cost is \( O(q N) \) if function evaluations are \( O(1) \). The computation of the DFT in [5] of a signal of size \( N_c \) is \( O(N_c \log N_c) \). Therefore, the cost of this is \( O(q N + N_c \log N_c) \). The ratio \( \rho = N_c/N \) is called the **oversampling factor**, and we can rewrite the runtime as \( O(q N + \rho N \log N) \). This is a dramatic improvement over the \( O(MN) \) runtime needed to compute \( y = Ax \) directly. Note that the values \( q \) and \( \rho \) here depend on the particular choice of \( \varphi \). In particular, \( q \) depends on \( t_0 \) while \( \rho \) depends on \( \omega_0 \).

It is fair to say that the error analysis of this approach would be quite complex. The runtime calculation was not so simple either. The next section gives a simpler approach to NUFFTs, even if mildly slower in achieving the same level of accuracy.

### 4 An alternative approach

Here we follow the approach of Candès et al [3], shown for a Type II NUFFT. In this case, we have the non-equispaced grid \( \{ t_n \}_{n=0}^{N-1} \), on which we would like to compute

\[
x[n] = \sum_{k=0}^{M-1} y[k] e^{i t_n k} = Y(t_n)
\]
from equispaced samples \( \{y[k]\}_{k=0}^{M-1} \), where

\[
Y(t) = \sum_{k=0}^{M-1} y[k]e^{ikt} = e^{i\frac{M}{2}t}\sum_{k=-\left\lfloor \frac{M}{2} \right\rfloor}^{\left\lfloor \frac{M}{2} \right\rfloor-1} y[k]e^{ikt}
\] (6)

is a trigonometric polynomial. Since \( Y \) is \( 2\pi \)-periodic, we will assume without loss that \( \{t_n\}_{n=0}^{N-1} \subset \left[ -\pi, \pi \right] \). The algorithm will employ two simple ideas: zero-padding and Taylor approximation.

We start by lengthening the observed data \( y \) by a factor of \( D \) (often \( D = 8 \) or 16 is sufficient), namely by appending \((D-1)M\) zeros:

\[
y_{DM}[k] := \begin{cases} y[k] & k = 0, 1, \ldots, M-1 \\ 0 & k = M, M+1, \ldots, DM-1. \end{cases}
\] (7)

The allows us to evaluate \( Y \) and its derivatives on an equispaced but dense grid, consisting of \( DM \) points in \([ -\pi, \pi ]\). We have

\[
Y^{(\ell)}\left( -\pi + \frac{2\pi j}{DM} \right) = \sum_{k=0}^{DM-1} (ik)^\ell y[k]e^{i(\pi + \frac{2\pi j}{DM})}
\]
\[
= \sum_{k=0}^{DM-1} (ik)^\ell y_{DM}[k]e^{i(\pi + \frac{2\pi j}{DM})}, \quad j = 0, 1, \ldots, DM-1,
\] (8)

where the second equality is trivial from (7). Now that \( Y \) and its derivatives have been evaluated on a dense grid, there is for any \( t \in [ -\pi, \pi ] \) some \( t_0 \) on the grid that is close to \( t \). More precisely, there is \( t_0 = -\pi + \frac{2\pi j}{DM} \) satisfying \(|t-t_0| \leq \frac{\pi}{DM} \).

Now, we have the Taylor expansion

\[
Y(t) = Y(t_0) + Y'(t_0)(t-t_0) + \cdots + \frac{Y^{(L-1)}(t_0)}{(L-1)!}(t-t_0)^{L-1} + R_L(t),
\]

\(^2\)Unless \( t \in [ \pi - \frac{2\pi}{DM}, \pi ] \), in which case \( t_0 = \pi - \frac{2\pi}{DM} \) gives \(|t-t_0| \leq \frac{2\pi}{DM} \).
where the residual satisfies
\[ R_L(t) = \frac{Y^{(L)}(\xi_t)}{L!}(t - t_0)^L \] (9)
for some \( \xi_t \in (t_0 - t, t_0 + t) \). We use this Taylor polynomial (up to any desired order \( L \)) to approximate \( x[n] = Y(t_n) \):

\[ x[n] = Y(t_n) \approx Y(t_0) + Y'(t_0)(t_n - t_0) + \ldots + \frac{Y^{(L-1)}(t_0)}{(L-1)!}(t_n - t_0)^{L-1}. \] (10)

The equality (9) makes error analysis straightforward. Indeed,

\[ |Y(t_n) - Y(t_0)| = |R_L(t_n)| \leq \frac{\|Y^{(L)}\|_\infty}{L!}|t_n - t_0|^L \leq \frac{\|Y^{(L)}\|_\infty}{L!}(\frac{\pi}{DM})^L. \]

To bound the \( L \)th derivative, we use Bernstein’s inequality:

**Proposition 1** (Bernstein). Let \( P \) be a \( 2\pi \)-periodic trigonometric polynomial of degree \( n \) with derivative \( P' \). Then

\[ \|P'\|_\infty \leq n\|P\|_\infty. \]

The representation (6) shows that \( Y \) is a modulation of a polynomial of degree at most \( \frac{M}{2} \), and so iterating Bernstein’s inequality gives

\[ \|Y^{(L)}\|_\infty \leq \left(\frac{M}{2}\right)^L\|Y\|_\infty. \]

We conclude

\[ |Y(t_n) - Y(t_0)| \leq \frac{\|Y\|_\infty}{L!}\left(\frac{\pi}{2D}\right)^L, \]

which decays rapidly in \( D \) and very rapidly in \( L \). For instance, when \( \|Y\|_\infty \) is on the order of \( 10^0 \), choosing \( D = 16 \) and \( L = 6 \) yields 9 digits of accuracy.

How fast is the algorithm? The computation (8) is a FFT of length \( DM \), thus requiring \( O(DM \log(DM)) \) operations. And this must be carried out for \( L \) derivatives, giving a total runtime of \( O(LDM \log(DM)) \). In relative runtime, the cost of evaluating the Taylor polynomials (10) is negligible.

### 4.1 As a Type I NUFFT

For the sake of completeness, we now describe how the Type II algorithm just described can be performed “backwards” as Type I. Here we consider the samples \( \{x(t_n)\} \) as given on a non-equispaced grid \( \{t_n\}_{n=0}^{N-1} \). Suppose we need to compute

\[ y(k) = \sum_{n=0}^{N-1} x(t_n)e^{ikt_n}, \quad k \in \mathbb{Z}. \]

Begin with a Taylor approximation

\[ e^{ikt_n} = e^{ik\tau}e^{ik(t_n-\tau)} \approx e^{ik\tau} \left[ 1 + \frac{ik}{1!}(t_n - \tau) + \ldots + \frac{(ik)^{L-1}}{(L-1)!}(t_n - \tau)^{L-1} \right]. \]

6
This representation suggests $\tau$ should be taken on a sufficiently dense, equispaced grid. In analogy with the degree condition from above, we assume

$$|k| \leq \frac{N}{2},$$

and we will sample the $t$-domain sufficiently regularly that we may take

$$|t_n - \tau| \leq \frac{\pi}{DN}.$$ 

These inequalities justify the same type of error analysis we performed above.

For each $\tau$ on the fine lattice, the quantity

$$\tilde{x}^{(\ell)}(\tau) = \sum_{n : t_n \in N(\tau)} (t_n - \tau)\ell x(t_n)$$

(11)

can be computed with negligible cost. Here

$$N(\tau) = \{ t_n : \tau \text{ is the nearest neighbor to } t_n \text{ on the fine grid} \},$$

and so $\{t_n\}_{n=0}^{N-1}$ is the disjoint union of the various $N(\tau)$. We can now write

$$y(k) = \sum_{n=0}^{N-1} x(t_n)e^{ikt_n}, \quad k \in \mathbb{Z}$$

$$= \sum_{\tau} \sum_{n : t_n \in N(\tau)} x(t_n)e^{ikt_n}$$

$$\approx \sum_{\tau} e^{ikt_\tau} \sum_{n : N(\tau)} x(t_n) \sum_{\ell=0}^{L-1} \frac{(ik)^\ell}{\ell!} (t_n - \tau)^\ell$$

$$= \sum_{\ell=0}^{L-1} \frac{(ik)^\ell}{\ell!} \sum_{\tau} \tilde{x}^{(\ell)}(\tau)e^{ikt_\tau}.$$

For each $\ell$, we recognize

$$\tilde{y}^{(\ell)}(k) = \sum_{\tau} \tilde{x}^{(\ell)}(\tau)e^{ikt_\tau}$$

(12)

as the uniform inverse discrete Fourier transform of $\tilde{x}^{(\ell)}$. We can thus use an IFFT to efficiently compute $\tilde{y}^{(\ell)}$. Our NUFFT is then completed by summing

$$y(k) \approx \sum_{\ell=0}^{L-1} \frac{(ik)^\ell}{\ell!} \tilde{y}^{(\ell)}(k).$$

Notice that the Type II algorithm ended with a Taylor approximation, whereas this Type I algorithm begins with one. That is, the Type II algorithm works by summing transforms—using (8) in (10)—while the Type I algorithm is transforming sums: using (11) in (12). Their computational complexity is, of course, the same.
5 Non-uniform in both domains

We have now seen methods of performing discrete Fourier transforms on equispaced data to non-equispaced data, and on non-equispaced data to equispaced data. The Type III problem of “non-equispaced to non-equispaced” remains, however. Given a Type I algorithm, though, it is easy to develop a solution using the ideas of Section 4. In particular, non-uniform data can be transformed onto a dense grid of equispaced points, and then Taylor approximation shows we can have good estimates at nearby, non-equispaced points. In this way, non-uniform data in one domain can be accurately transformed to non-uniform data in the other, with essentially the same runtime as what we have seen above.

References

