1.2 Review of Fourier Analysis

If we approximate the DTFT

\[ X(f) = \sum_{n=-\infty}^{\infty} x_n e^{-j2\pi fn} \]

by

\[ X(f) \approx \sum_{n=n_1}^{n_2} x_n e^{-j2\pi fn}, \]

then the right-hand side can be plotted in MATLAB as follows. Assuming \( x = [x_{n_1}, \ldots, x_{n_2}] \),

```matlab
f = linspace(-1/2,1/2,201); nvec = [n1:n2]; X = dft(f,x,nvec); plot(f,X)
```

The DFT and the FFT

In Chapter 1, we saw that signal processing for continuous-time, bandlimited waveforms and systems can be accomplished by discrete-time signal processing. However, computers can only evaluate finite sums.

Recall that for an infinite sequence \( x_n \), its DTFT is

\[ X(f) := \sum_{m=-\infty}^{\infty} x_m e^{-j2\pi fm} \approx \sum_{m=M_0}^{M_1} x_m e^{-j2\pi fm} \]

for large \( M_2 \) and large (negative) \( M_1 \). The sum on the right contains \( M_2 - M_1 + 1 \) terms. In order to write the finite sum as a sum starting from zero, we can make the change of variable \( n = m - M_1 \) to get

\[ X(f) \approx \sum_{n=0}^{M_2-M_1} x_{n+M_1} e^{-j2\pi f(n+M_1)}. \]

Although the foregoing approximation involves only a finite sum, a computer cannot evaluate it for all values of \( f \) in one period, say \( |f| \leq 1/2 \). Instead, the computer can only evaluate the sum for finitely many values of \( f \). Let \( N := M_2 - M_1 + 1 \), and let \( f = k/N \) to get

\[
X(k/N) \approx \sum_{n=0}^{N-1} x_{n+M_1} e^{-j2\pi k(n+M_1)/N} \\
= e^{-j2\pi M_1/N} \sum_{n=0}^{N-1} x_n e^{-j2\pi kn/N}. 
\]

Regarding this last sum as a function of \( k \), observe that it has period \( N \); i.e., replacing \( k \) with \( k + N \) does not change the value of the sum. So we only need to evaluate the sum for \( k = 0, \ldots, N-1 \).

3.1. The Discrete Fourier Transform (DFT)

Given a finite sequence \( y_0, \ldots, y_{N-1} \), its **discrete Fourier transform** (DFT) is

\[ Y_k := \sum_{n=0}^{N-1} y_n e^{-j2\pi kn/N}. \]

It is easy to show that \( Y_k \) is a periodic function of \( k \) with period \( N \). We show below in Section 3.1.7 that the sequence \( y_n \) can be recovered from the DFT sequence \( Y_0, \ldots, Y_{N-1} \) by the **inverse DFT** (IDFT)

\[ y_n = \frac{1}{N} \sum_{k=0}^{N-1} Y_k e^{j2\pi kn/N}. \]

Of course the right-hand side is a periodic function of \( n \) with period \( N \). Hence, although \( y_n \) is only defined for \( n = 0, \ldots, N-1 \), we often think of it as being an infinite-duration periodic signal with period \( N \).
3.1 The Discrete Fourier Transform (DFT)

3.1.1. Zero Padding

Given a sequence \( y_0, \ldots, y_{M-1} \), its DTFT is

\[
Y(f) = \sum_{n=0}^{M-1} y_n e^{-j2\pi fn}.
\]

Now consider the zero-padded vector

\[
z = [y_0, \ldots, y_{M-1}, 0, \ldots, 0].
\]

If the length of \( z \) is \( N \), then the DFT of \( z \) is

\[
\sum_{n=0}^{N-1} z_n e^{-j2\pi kn/N} = \sum_{n=0}^{M-1} y_n e^{-j2\pi kn/N} = Y(k/N),
\]

In other words, given \( M \) data samples \( y_0, \ldots, y_{M-1} \), computing a zero-padded DFT gives you samples of \( Y(f) \) that are more closely spaced in frequency.

Now suppose that \( x_n \) is a causal sequence of infinite duration with DTFT

\[
X(f) = \sum_{n=0}^{\infty} x_n e^{-j2\pi fn}.
\]

Put

\[
X_M(f) := \sum_{n=0}^{M-1} x_n e^{-j2\pi fn}.
\]

Then as \( M \to \infty \), \( X_M(f) \to X(f) \). For fixed \( M \), the DFT of \([x_0, \ldots, x_{M-1}, 0, \ldots, 0]\) zero padded to length \( N \) is

\[
X_M(k/N) := \sum_{n=0}^{M-1} x_n e^{-j2\pi kn/N} \quad k = 0, \ldots, N-1.
\]

In other words, for fixed \( M \), if we compute the DFT of \([x_0, \ldots, x_{M-1}, 0, \ldots, 0]\) with more and more zeros padded, we do not get closer to \( X(f) \), we get more closely spaced frequency samples of \( X_M(f) \). This is illustrated in Figure 3.1.

3.1.2. The Fast Fourier Transform (FFT)

If \( y = [y_0, \ldots, y_{N-1}] \), then the DFT of \( y, Y = [Y_0, \ldots, Y_{N-1}] \), can be computed in MATLAB with the command \( Y = \text{fft}(y) \). Here \text{FFT} stands for fast Fourier transform. The FFT is a special algorithm that computes the DFT very quickly.

Since the DFT is periodic,

\[
Y_{-1} = Y_{-1+N} = Y_{N-1} \\
Y_{-2} = Y_{-2+N} = Y_{N-2} \\
\vdots \\
Y_{-N/2} = Y_{-N/2+N} = Y_{N/2}.
\]

If \( N \) is not even, then \( N/2 \) should be replaced the greatest integer that is less than or equal to \( N/2 \), which is denoted by \( \lfloor N/2 \rfloor \) and is given by \text{floor} in MATLAB.
3.1.3. Using the FFT to Compute the DTFT

If we are approximating the sum in (3.1), then we would use $k/N$ to have the horizontal axis run from $-1/2$ to $1/2$. The MATLAB function `dtftfft` given below computes the right-hand side of (3.1) using `fft` and also takes care of the bookkeeping to return to you the corresponding frequencies $f$ in $[-1/2, 1/2]$. The command for this is $[y, f] = \text{dtftfft}(x,M1)$, where the elements of $x$ are $[x_M, ..., x_M]$. If $M1$ is zero, it can be omitted and you can write $[y, f] = \text{dtftfft}(x)$ instead. There is also an optional third argument if you want to force `fft` to zero-pad $x$. Remember, the spacing between frequencies in the DFT and the FFT is $1/\text{length}(x)$. The required command is $[y, f] = \text{dtftfft}(x, M1, N)$ to force `dtftfft` to add enough zeros to $x$ to make its length $N$. If $x$ is longer than $N$, $x$ will be truncated to $N$ elements. Finally, a nonpositive value of $N$ causes `dtftfft` to zero-pad $x$ so its length is a power of 2. This makes `fft` as fast as possible. Here is the function.

```matlab
function [y, f] = dtftfft(x, varargin)
% dtftfft Compute the discrete-time Fourier transform using FFT.
N = length(x);
if nargin == 3
    N = varargin{2};
    if N <= 0 % If N<=0, zero pad x to a power of 2.
        N = 2^ceil(log2(length(x)));
    end
end
y = fft(x, N);
k = [0:N-1];
if nargin >= 2
    M1 = varargin{1};
    if M1 ~= 0
        y = exp(-j * 2 * pi * M1 / N * k) .* y;
    end
end
y = fftshift(y); % For N=2m even, change 0,...,m,...,2m-1 % to -m,...,0,...,m-1. For N=2m+1 odd, change % 0,...,m-1,m,m+1,...,2m to -m,...,-1,0,1,...,m. % Then divide by N to get frequencies in [-1/2,1/2].
f = (k - floor(N/2)) / N;
```

October 1, 2007