DFS, DTFT, and DFT

Herein we describe the relationship between the Discrete Fourier Series (DFS), Discrete Time Fourier Transform (DTFT), and the Discrete Fourier Transform (DFT). Why? The real reason is that the DFT is easily implemented on a computer and is part of every mathematics package, so it would be nice to know how to determine or approximate the DFT and DTFT on a computer.

First, the definitions: $\tilde{x}(n)$ is a periodic function (period N) and $x(n)$ is a non-periodic function.

Discrete Fourier Series (DFS):

$$a_k = \frac{1}{N}\sum_{n=0}^{N-1}\tilde{x}(n)\exp\left(-i\frac{2\pi}{N}kn\right) \text{ (or the sum over any } N \text{ consecutive } x(n)\text{'s)}$$

$$\tilde{x}(n) = \sum_{n=0}^{N-1}a_k\exp\left(i\frac{2\pi}{N}kn\right) \text{ (or the sum over any } N \text{ consecutive } a_k\text{'s)}$$

a_k is discrete and periodic with period N.

Discrete Time Fourier Transform (DTFT):

$$X(\omega) = \sum_{n=-\infty}^{\infty}x(n)\exp(-i\omega n)$$

$$x(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi}X(\omega)\exp(i\omega n)d\omega \text{ (or the integral over any } 2\pi \text{ interval of } X(\omega))$$

$X(\omega)$ is continuous and periodic with period 2π.

Discrete Fourier Transform (DFT) of length N:

$$X_d(k) = \sum_{n=0}^{N-1}x(n)\exp\left(-i\frac{2\pi}{N}kn\right) \text{ where } k \text{ goes from } [0\ldots N\text{-}1]$$

$$x(n) = \frac{1}{N}\sum_{n=0}^{N-1}X(k)\exp\left(i\frac{2\pi}{N}kn\right) \text{ where } n \text{ goes from } [0\ldots N\text{-}1]$$

The DFT and inverse DFT are implemented in Matlab by `fft` and `ifft`, respectively. You'll observe that the formulae are very similar to the DFT formulae and as such the input and output can be seen as periodic if it is convenient to do so. Matlab also includes a functions `fftshift` and `ifftshift` that change the domain of the output or input, respectively, for `fft` and `ifft` from [0…*N*-1] to [-*N*/2…*N*/2-1]. E.g. for even *N*:

`fft(ifftshift(x))` is defined for $x(n)$ for n in [-*N*/2…*N*/2-1], k in [0…*N*-1]
`fftshift(fft(x))` is defined for n in [0…*N*-1], k in [-*N*/2…*N*/2-1]
`fftshift(fft(fftshift(x)))` is defined for n in [-*N*/2…*N*/2-1], k in [-*N*/2…*N*/2-1]
`ifft(ifftshift(X))` is defined for n in [0…*N*-1], k in [-*N*/2…*N*/2-1]

For odd N, `fftshift` changes the output from [0…N-1] to [-(N+1)/2…(N+1)/2] and `ifftshift` changes the output from [-(N+1)/2…(N+1)/2] to [0…N-1]:

`fft(ifftshift(x))` is defined for $x(n)$ for n in [-(N+1)/2…(N+1)/2], k in [0…N-1]
`fftshift(fft(x))` is defined for n in [0…N-1], k in [-(N+1)/2…(N+1)/2]
`fftshift(fft(fftshift(x)))` is defined for n in [-N/2…N/2-1],
k in [-(N+1)/2…(N+1)/2]
`ifft(ifftshift(X))` is defined for n in [0…N-1], k in [-(N+1)/2…(N+1)/2]

`fftshift` and `ifftshift` are exactly the same for even N. With my fading memory, I can't always remember when to use `fftshift` or `ifftshift`, so I often choose to use N even and then it just doesn't matter.

DFS and DFT

First, the relationship between the DFS and DFT is quite clear – we merely apply the DFT to one period [0…N-1] of $\tilde{x}(n)$ and scale the output of the DFT by 1/N to get the DFS coefficients, e.g.

$$a_k = \frac{1}{N} X(k)$$

Similarly, we can determine 1 period [0…N-1] of $\tilde{x}(n)$ from a_k by taking the inverse DFT of Na_k for k in [0…N-1]. One can use [-N/2…N/2-1] for the interval of $\tilde{x}(n)$ or a_k by using `ifftshift`.

DTFT and DFT

To relate the DFT and DTFT, we will need to truncate the DTFT to a finite range of N samples. In doing so, one would like to insure that the power of the signal in the excluded (truncated) portion is insignificant, e.g. one would not want to approximate the DSFT with the DFT for $x(n) = \sin(4\pi n)$, which will have significant power in any truncated portion. This is equivalent to saying that the DTFT of x cannot have any delta functions.

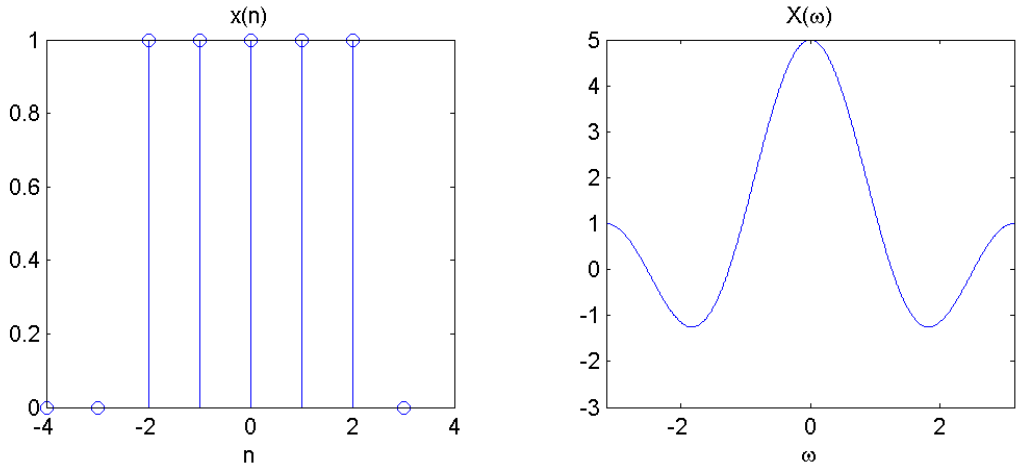
Next, one needs to determine if the range of $x(n)$ should be [0…N-1] or [-N/2…N/2-1]. That will determine if `ifftshift` needs to be used. Now suppose we have a function that is zero for all negative n and we determine a length N over which we'd like to examine. We can now approximate $X(\omega)$ as $X_d(k)$, where $\omega = \frac{2\pi}{N} k$.

Note that the continuous $X(\omega)$ is now discretized or sampled. Locations of samples are either $\omega \in \frac{2\pi}{N}[0...N-1]$ or $\omega \in \frac{2\pi}{N}[N/2...N/2-1]$ depending of whether `fftshift` has been used. This range is approximately [0..2π] or [-π…π], respectively. This introduces a second criterion on selection N. If $X(\omega)$ appears to be sampled too coarsely, then N should be increased, even if it means adding zeros to ends of the definition of x (know as zero padding).

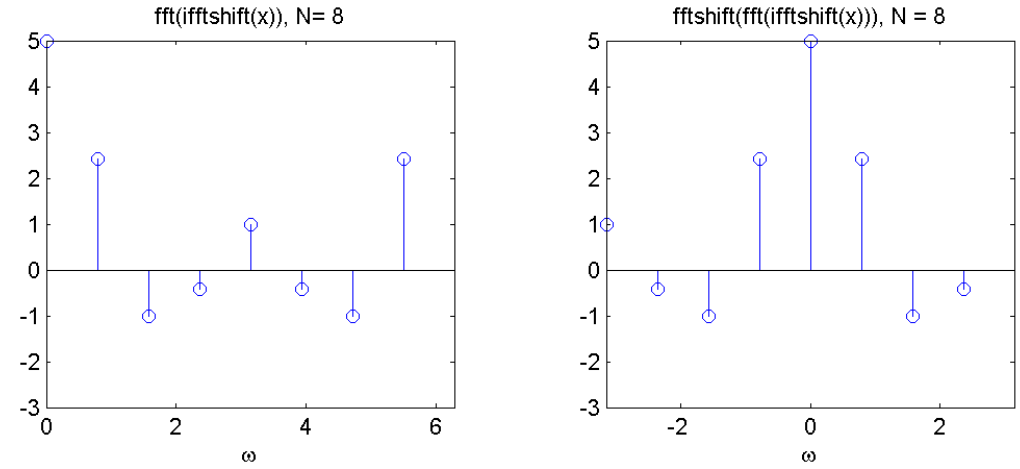
The inverse is relationship is also tricky. First, let's assume that we have defined a continuous $X(\omega)$. Obviously, $X(\omega)$ cannot have any delta functions as these cannot be represented on the computer. This can be sampled at locations $\omega \in \frac{2\pi}{N}[0...N-1]$ or $\omega \in \frac{2\pi}{N}[N/2...N/2-1]$ and one them implements the inverse DFT (`ifft`) to get a signal $x(n)$. The tricky part arises from the observation that the DFT is DFS are closely related. This means that by sampling $X(\omega)$, we have created a periodic $x(n)$. This creates an ambiguity as to whether $x(n)$ is defined for n in $[0\ldots N\text{-}1]$ or $[-N/2\ldots N/2\text{-}1]$. If we know (e.g. based on causality or whatever) that the output will be zero for negative n, then we can use the output of the DFT for n in $[0\ldots N\text{-}1]$. If we don't know, it might be wise to use `fftshift` and define n for $[-N/2\ldots N/2\text{-}1]$. N should be chosen so that $x(n)$ had decayed to near zero at $-N/2$ and $N/2\text{-}1$.

Examples

Consider the following example:

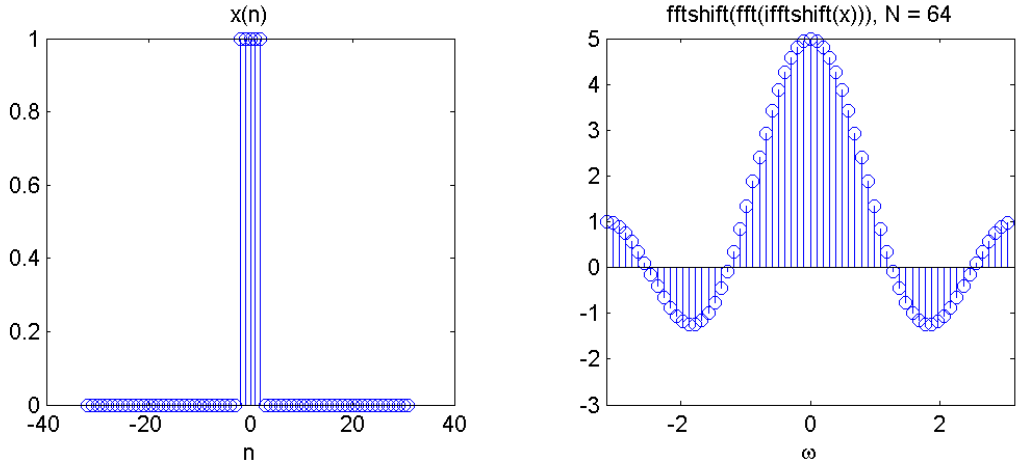


This is the continuous DTFT of a non-periodic signal $x(n)$. If we wish to approximate this by the DFT, we can taking `fft(ifftshift(x))` and the result looks like this:



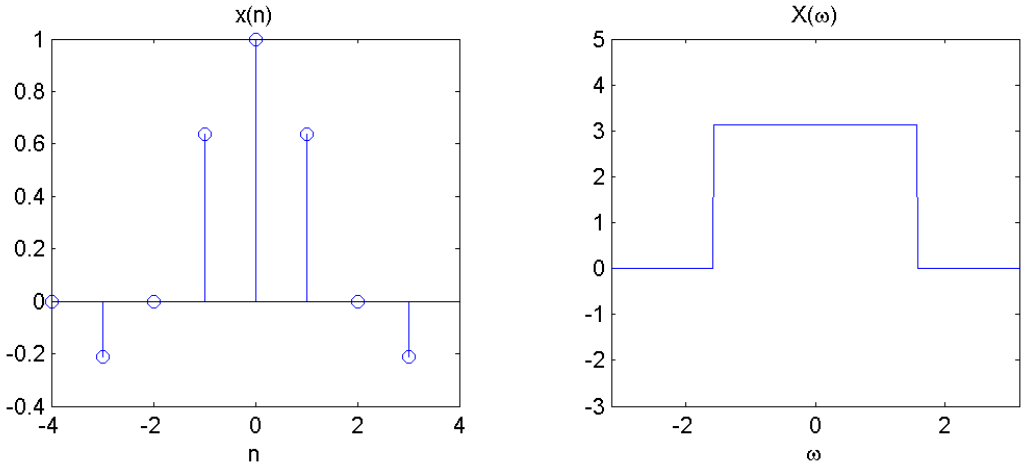
Observe that $X(\omega)$ is now sampled at the locations $\omega \in \frac{2\pi}{N}[0...N-1]$ or $\omega \in \frac{2\pi}{N}[N/2...N/2-1]$ on the left after we have used fftshift. Because $x(n)$ is zero outside of the specified bounds, these samples are exact – there is no error.

If we want to know $X(\omega)$ more finely, we increase N, yielding

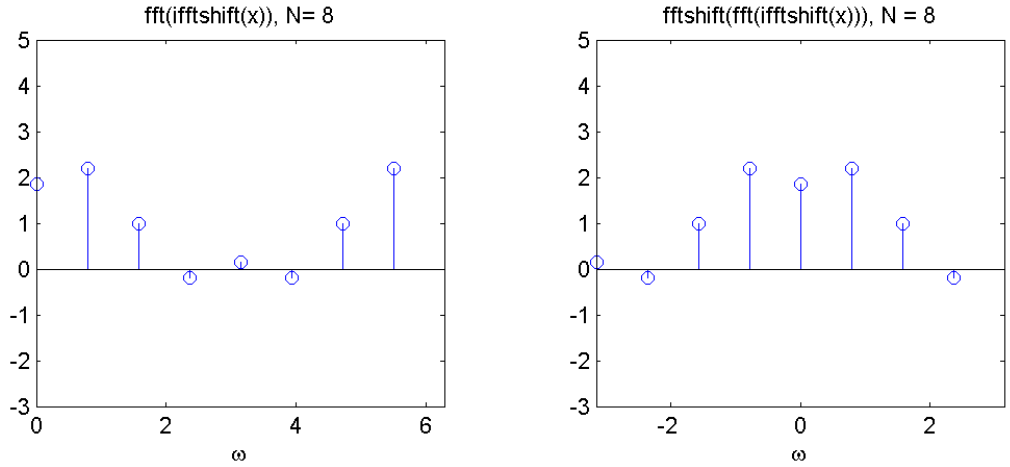


Example 2

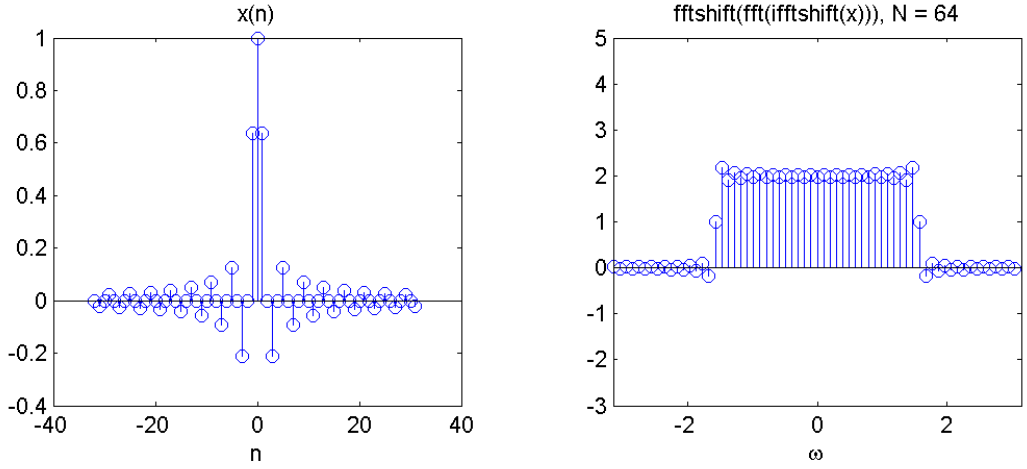
In this example, we use a signal that is not limited in time, in our case, we will use $x(n) = \text{sinc}(n/2)$. Here we show a short range of $x(n)$ as well as the true $X(\omega)$:



Now we take `fft(ifftshift(x))` and the result looks like this:

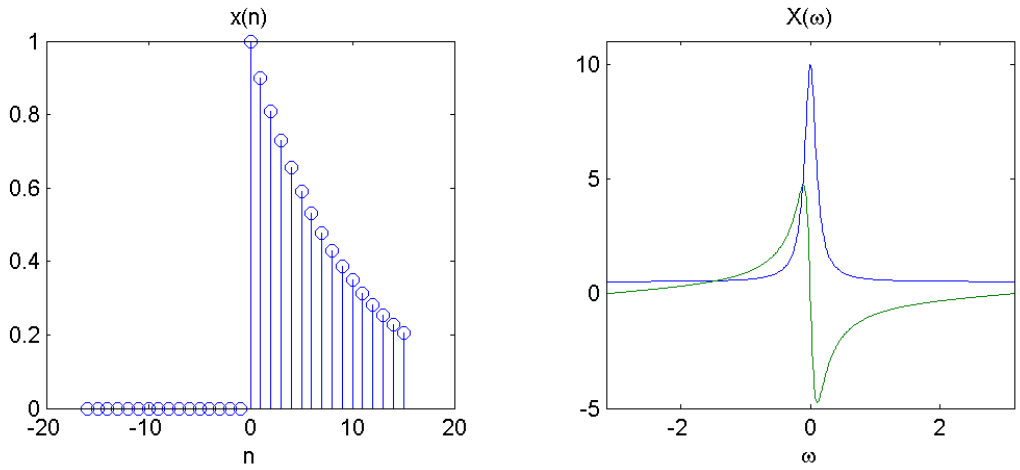


Clearly, $X(\omega)$ is not only sampled rather coarsely, but also has a substantial amount of error. This error results from truncation of $x(n)$ in our DFT approximation of the DTFT. We can reduce error and improve sampling by increasing N as shown here:

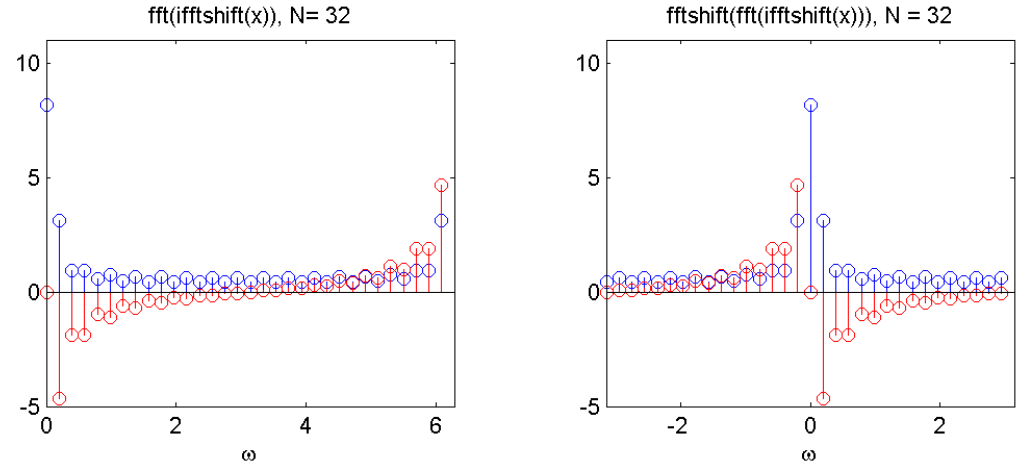


Example 3

In this example, we use a causal system response: $x(n) = (0.9)^n u(n)$. Here we show a short range of $x(n)$ as well as the true $X(\omega)$, which has a peak signal of 10 (real part – blue, imaginary – green):



If we us a range of input signals from -*N*/2 and *N*/2-1 for *N* = 32, we get the following approximation to the DTFT (real – blue, imaginary – red):



Observe that there is quite a bit of error due to the truncation of the time series. We could increase *N*, or in this case, we could defined *n* to go from 0 to *N*-1. If we do the latter, we then do not need the initial ifftshift operation. Here the error is much reduced: