A Note on Upsampling by Integer Factors Using the DFT

Mark Richards
February 24, 2012

Let $x[n]$ be a sequence of length N with discrete-time Fourier transform (DTFT) $X(\omega)$. Let $x_D[n]$ be a decimated sequence such that $x_D[n] = x[D \cdot n]$ for all n with D a positive integer. The nonzero support of $x_D[n]$ will be from $n = 0$ to $N_D - 1$, where N_D satisfies

$$N_D - 1 = \left\lfloor \frac{N-1}{D} \right\rfloor \quad \Rightarrow \quad N_D = \left\lfloor \frac{N-1}{D} \right\rfloor + 1 = \left\lfloor \frac{N+D-1}{D} \right\rfloor \tag{1}$$

Let $X[k]$ be the K-point discrete Fourier transform (DFT) of $x[n]$, $K \geq N$, and $X_D[k]$ be the K_D-point DFT of $x_D[n]$ where $K_D = K/D$. It will always be true that $K_D \geq N_D$ when $K \geq N$.

Our goal is to go the other way, i.e. given x_D and X_D, show how to obtain x by constructing an appropriate X.

Oppenheim and Schafer, 3rd ed, Section 4.6.1 derive the effect on the DTFTs of "decimating" a sequence x by an integer factor D to obtain x_d. The main result is their Eqn. (4.77), which in our notation is[1]

$$X_D(\omega) = \frac{1}{D} \sum_{i=0}^{D-1} X\left(\frac{\omega - 2\pi i}{D} \right) \tag{2}$$

We need a relationship for DFTs, not DTFTs. One way to define the DFT is as a sampled DTFT:

$$X[k] = X(\omega)\big|_{\omega = \frac{2\pi}{K}k}, \quad k = 0, \ldots, K-1 \tag{3}$$

Consider the K_D-point DFT of x_D. Sampling the DTFT of (2) gives us

$$\begin{aligned} X_D[k] &= X_D\left(\frac{2\pi k}{K_D} \right), \quad k = 0, \ldots K_D - 1 \\ &= \frac{1}{D} \sum_{i=0}^{D-1} X\left(\frac{2\pi k}{K_D D} - \frac{2\pi i}{D} \right) = \frac{1}{D} \sum_{i=0}^{D-1} X\left(\frac{2\pi k}{K} - \frac{2\pi i}{D} \right) \end{aligned} \tag{4}$$

Using $K_D = K/D$ gives

$$X_D[k] = \frac{1}{D} \sum_{i=0}^{D-1} X\left(\frac{2\pi}{K}(k - iK_D) \right) = \frac{1}{D} \sum_{i=0}^{D-1} X\left[(k - iK_D) \right], \quad k = 0, \ldots K_D - 1 \tag{5}$$

We now have a relation between the K_D-point DFT of x_d and the K-point DFT of x.

---

[1] You can distinguish DFTs and DTFTs in this note based on whether the argument is in parentheses (DTFT) or square brackets [DFT].

We can accomplish the upsampling, that is, we can get from x_D to x, if we can get from X_D to X. So, given x_D, we take a K_D-point DFT ($K_D \geq N_D$) to get X_D. We then need to form a K-point DFT X_D, $K = D \cdot K_D$, such that Eqn. (5) holds. We can do this by placing the positive and negative frequency halves of X_D at the positive and negative frequency ends of X and filling the middle of X with zeros. If we do this correctly and then do the IDFT of X, we will obtain x of length K such that $x_D[n] = x[D \cdot n]$ and the other samples of x are filled in by sinc interpolation.

To be more precise about "filling in the middle", let's write Eqn. (5) out more explicitly:

$$X_D[k] = \frac{1}{D}\left\{X[k] + X[k - K_D] + X[k - 2K_D] + \ldots + X\left[k - (D-1)K_D\right]\right\}, \quad k = 0, \ldots K_D - 1 \qquad (6)$$

A K-point DFT is periodic in its index k with period K. We can use this to rewrite the right-hand side of Eqn. (6) to avoid negative indices:

$$X_D[k] = \frac{1}{D}\left\{X[k] + X[k + K - K_D] + \ldots + X\left[k + K - (D-1)K_D\right]\right\}, \quad k = 0, \ldots K_D - 1 \qquad (7)$$

Now we can make more specific assignments of values to $X[k]$ that, when aliased according to Eqn. (7), will produce the correct values in $X_D[k]$. The answer depends on whether K_D is even or odd.

If K_D is odd: This means that there is no sample of X_D that corresponds to half the Nyquist sampling frequency; there is a DC sample, and then all of the others can be paired in equal-magnitude positive and negative frequencies. In this case, make the following assignment:

$$\begin{aligned} &X[k] = D \cdot X_D[k], && k = 0, 1, \ldots, (K_D - 1)/2 \\ &X[k + K - K_D] = D \cdot X_D[k], && k = (K_D + 1)/2, \ldots, K_D - 1 \\ &X[k] = 0, && \text{otherwise} \end{aligned} \qquad (8)$$

If K_D is even: This means that there is a sample of X_D that corresponds to half the Nyquist sampling frequency and it has to be "split" in some fashion. In this case, make the following assignment:

$$\begin{aligned} &X[k] = D \cdot X_D[k], && k = 0, 1, \ldots, K_D/2 - 1 \\ &X[K_D/2] = D \cdot X_D[K_D/2]/2, && \\ &X[K - K_D/2] = D \cdot X_D[K_D/2]/2, && \\ &X[k + K - K_D] = D \cdot X_D[k], && k = K_D/2 + 1, \ldots, K_D - 1 \\ &X[k] = 0, && \text{otherwise} \end{aligned} \qquad (9)$$

The particular assignment given is not unique. As one example, in the even-K_D case we could choose not to "split" $X_D[K_D/2]$ but instead to assign it to $X[K_D/2]$ in its entirety, and set $X[K - K_D/2]$ to zero. When X is decimated to get X_D the result will be the same as with the assignment in Eqn. (9). However, Eqn. (9) has the following properties that are consistent with a typical bandlimited signal processing viewpoint:

1. The constructed spectrum X is bandlimited with compact support, i.e. all the energy is concentrated around zero frequency and the center of the spectrum is empty; and
2. If x_D is real-valued so that X_D exhibits the appropriate DFT conjugate symmetry, namely $X_D[k] = X_D^*[k]$ then X will maintain that symmetry, $X[k] = X^*[k]$, and therefore x will also be real-valued.

The following pages include a short MATLAB script that can be used to demonstrate these relations.

It is important to keep in mind what this method claims and what it doesn't claim. This algorithm is designed to start with a sequence x_D and then create a sequence x upsampled by an integer factor D. The sequence x is designed such that, when you decimate it by the same factor D, you get the original spectrum X_D back again, which means you get the original sequence x_D back again. However, this is only guaranteed on the original sequence's region of support. If the original DFT is bigger than the original data, $K_D > N_D$, then after you upsample the original DFT and invert the expanded DFT, you'll have a sequence that is K points long in time instead of N. While it will match the original x_1 for $N = 0{:}N{-}1$, the behavior for $n = N{:}K{-}1$ has not been constrained. In practice, this region will not be zero, but instead will exhibit the asinc interpolation "tails" from the ending portion of the original data sequence.

This behavior can be seen by letting $K_D > N_D$ and turning off the window in the attached MATLAB demonstration code by commenting it out (line 19). In the plot of the IFFT of the larger spectrum, i.e. the final upsampled data, we get some funky behavior away from the original regions of support. This upsampled domain is an asinc interpolation of the original data; the big discontinuity in my data at $n = 0$ causes that ringing at the other end. If you redefine the test sequence to be smoother at the beginning (e.g., uncomment the line with the Tukey window in the code), it looks much better.

```
% dft_upsample.m
%
% toy example of using DFTs to upsample data, with careful (I hope)
% attention to the end points of everything
clear all
close all

ND = input('Enter sequence length ND: ');
KD = input(['Enter DFT size KD (KD >= ',num2str(ND),'): ']);
D = input('Enter upsampling factor (integer): ');

% create some data xd and its DFT Xd. I'll do a decaying, damped sinusoid.
xd = zeros(ND,1);
nd = (0:ND-1)';
f0 = 0.1; % oscillation frequency in cycles per sample
alpha = 0.04; % decay rate for amplitude
xd = exp((1i*2*pi*f0 - alpha)*nd);
% optional window to guarantee no discontinuity at the edges
xd = xd.*tukeywin(length(xd),0.1);
XD = fft(xd,KD);
kd = (0:KD-1)';

subplot(4,2,1)
plot(nd,real(xd)); grid; axis('tight')
subplot(4,2,2)
plot(kd,abs(XD)); grid; axis('tight')
shg

% Now let's build the bigger FFT according to my tech note

K = D*KD;
X = zeros(K,1);
k = (0:K-1)';

if mod(KD,2)==0
 % KD is even. Add one to all indices in tech note for MATLAB indexing
 X(1:(KD/2)) = XD(1:KD/2);
 X(KD/2+1) = XD(KD/2+1)/2;
 X(K-KD/2+1) = XD(KD/2+1)/2;
 X((KD/2+2:KD)+K-KD) = XD(KD/2+2:KD);
 X = D*X;
else
 % KD is odd
 X(1:(KD-1)/2+1) = XD(1:(KD-1)/2+1);
 X(((KD+1)/2+1:KD)+K-KD) = XD((KD+1)/2+1:KD);
 X = D*X;
end

% OK, let's see what we've got.
x = ifft(X);

subplot(4,2,3)
plot(k,real(x)); grid; axis('tight')
subplot(4,2,4)
plot(k,abs(X)); grid; axis('tight')
```

```
% Now to close the loop, let's explicitly decimate x to see if it matches
% the original xd

y = x(1:D:end);
ny = (0:length(y)-1)';
Y = fft(y,KD);

subplot(4,2,5)
plot(nd,real(y(1:ND))); grid; axis('tight')
subplot(4,2,6)
plot(kd,abs(Y)); grid; axis('tight')

subplot(4,2,7)
plot(ny,real(y)); grid; axis('tight')


norm(Y-XD)/norm(XD)
norm(y(1:ND)-xd)/norm(xd)
```