VISUALIZATION: FREQUENCY DISCRIMINATION
By L.G. Hassebrook
EE630
2011

TURN IN PRINTOUTS OF ALL YOUR MATLAB CODE AND RESULTS. No electronic form please. Replicate all figures using your own MATLAB code. Do not share code but you can talk about it all you want. ANSWER ALL THE QUESTIONS indicated by the ? mark.

PART 1: ZERO PADDING AND WINDOWING OF SINGLE FREQUENCY

Consider a discrete time finite length cosine waveform \( s_1[n] = \cos(2\pi k_c n/N) \) where \( k_c = 40 \) and \( N = 512 \). A plot of this signal is shown in Fig. 1.

![Discrete Time Cosine Wave](image)

**Figure 1: DT Cosine Wave**

The DFT of \( s_1[n] \) is shown in Fig. 2 and note the location of the peaks correspond to \( k_c = 40 \).

![DFT of Cosine Wave](image)

**Figure 2: DFT of waveform in Fig. 1.**

Zero pad \( s_1[n] \) to twice its length such that

\[
s_2[n] = \begin{cases} 
  s_1[n] & n = 0, 1, \ldots, (N - 1) \\
  0 & n = N, N + 1, \ldots, (2N - 1) 
\end{cases}
\]

and is shown in Fig. 3.
As seen in Fig. 4, the zero padding creates a leakage situation.

A cosine edge window is applied to $s_1[n]$ using the MATLAB code

```matlab
Tx=N/4;
wincos=cosedgewindow(Tx,N);
s1W=s1.*wincos;
```

where the cosedgewindow($Tx,N$) code is given in the appendix. After zero padding, the signal is shown in Fig. 5 and its DFT is shown in Fig. 6.
Figure 5: Zero padded cosine with a partial cosine window.

Figure 6: DFT of waveform in Fig. 5.

Note the reduction in leakage when comparing Figs. 4 and 6.

PART 2: DUAL FREQUENCY DISCRIMINATION AND UP-SAMPLING

One of the misperceptions within the DSP community is that by zero padding you will gain improved frequency discrimination between to sine waves that are close in frequency. Unfortunately, this is not true. What is gained in more frequency samples, is lost in leakage characteristics. This part of the visualization will give you the code to try that for yourself. You will find that sometimes, depending on phase and frequency difference, you may or may not be able to discriminate. In this part, we separated the frequencies by a 2 discrete frequency units such that $dk=1$ and $k_{cL} = k_c - dk$ and $k_{cR} = k_c + dk$. Where $k_c$ is the same frequency as in Part 1. The summed cosine waveforms followed by a partial cosine window are shown in Fig. 7
The code we used for this is

```matlab
dk=1;
doff=0;
kxcL=kxc-dk+doff
kxcR=kxc+dk+doff
s1L=cos(2*pi*kxcL*n/N);
s1R=cos(pi/2+2*pi*kxcR*n/N);
sall1=s1L+s1R;
% window
Tx=N/4;
wincos=cosedgewindow(Tx,N);
sall1=sall1.*wincos;
```

Figure 7: Dual frequency with cosine window.

The DFT of the signal in Fig. 7 is shown in Fig. 8 and if observed closely, the peaks are easily separated.

Figure 8: DFT of dual frequency sum.
With zero padding, the discrimination performance does not necessarily improve. Compare Fig. 9 with Fig. 8 and try letting $d_k=0.25$ instead of 1? If you experiment, you will find some values work better than others.

PART 3: UP-SAMPLING AND INTERPOLATION

A handy alternative to zero padding, to reach a specific sequence length is up-sampling combined with interpolation. In this visualization we will show you one way to do this.

Using the code below, we obtain the up-sampled signal in Fig. 10 with $\text{Upsample}=2$:

```matlab
% upsample sall1 to sall3
sall3=zeros(1,N*Upsample);
for n=1:N
    sall3(Upsample*(n-1)+1)=sall1(n);
end;
Tx=N*Upsample/4;
```
wincos = cosedgewindow(Tx, N*Upsample);
sall3 = sall3 .* wincos;

Figure 11: DFT of signal in Fig. 10. Note the replicated spectra.

The DFT of the upsampled signal, shown in Fig. 11, shows the spectrum replication.

Figure 12: Interpolated signal.

Upsampling and interpolation will not attain any better discrimination but it does have some implementation advantages over zeropadding. Let’s consider how to implement the interpolation process. Since we know that every other sample in the sequence is zero, we design an interpolation filter that does not change the non-zero sample values but fills in the zero values with the average of the adjacent non-zero values. Consider the filter h[n] which is two kroneckor delta function located at n-1 and n+1 with a coefficient of ½. As shown in the MATLAB code below, this filter is convolved with the upsampled input to get a result of the interpolated values separated by zeros. To get the final result, the original upsampled
input is added to the interpolated signal, thereby filling in all the zero values with the appropriate values. The resulting signal is shown in Fig. 12.

% interpolate with \( h[n] = \frac{\delta[n-1] + \delta[n+1]}{2} \) to fill in zeros
h = zeros(1, N*Upsample);
h(2) = 1/2;
h(N*Upsample) = 1/2;
H = fft(h);
Sall3 = fft(sall3);
Sall4 = Sall3 .* H; % convolve via the DFT domain
% sum in original upsampled sequence sall3
sall4 = real(ifft(Sall4)) + sall3;

Figure 13: DFT of interpolated dual signal.

The DFT of Fig. 12 signal is shown in Fig. 13. Note the frequency location? Are the peak locations the same as Figs 8 or 9? Explain your answer?

APPENDIX:

% 1-D Symmetric cosine edge function for use with DFTs
% Tx are the widths of the cosine rolloff
% Nx are number of elements of \( w \)
function w = cosedgedwindow(Tx, Nx)
w = ones(1, Nx);
for n = 1:Tx
    nn = Nx - (n-1);
    a = (1 - cos(pi * (n-1)/Tx)) / 2;
    w(n) = a;
    w(nn) = a;
end;