Windowed Fourier analysis and spectrograms

The objective of this lab is to explore the windowed Fourier transform to identify and localize in time which frequencies during a signal and then visualize the time-frequency analysis through spectrograms. This lab is adapted from Nathan Kurz's notes on computational data methods.

Instructions: Submit the requested figures to tleise@amherst.edu.

The motivation for the windowed Fourier transform is that the DFT can't tell us *when* a frequency occurred. For example, in analyzing a music recording, the DFT tells you what notes were played but not in what order. We'd like to know the timing of each note in a song, not just whether that note was played sometime during the song.

One way to localize the frequency information in time is to use a moving window, accomplished by multiplying the signal by a step function or Gaussian to retain just a portion or "window" centered at successive time points.

Exercise 1 (Windows) Create a time array `t` with 2048 components covering a total time of 10 seconds. Then generate the following signal that exhibits different frequencies over time:

```
x=(3*sin(2*t)+0.5*tanh(0.5*(t-3))+ 0.2*exp(-(t-4).^2)+...
1.5*sin(5*t)+4*cos(3*(t-6).^2))/10+(t/20).^3;
```

Compute and plot the absolute value of the DFT, making sure you define the frequencies correctly in Hertz, given that the total duration is 10 seconds.

Now that we have our signal, let's turn to the window function. We'll use a Gaussian here so that the edges of the window taper off rather then ending abruptly: `g=exp(-a*(t-t0).^2)`, where `a` controls the window width and `t0` controls where it is centered. Try plotting `g` for different values of `a` and `t0` to get a sense of how these parameters affect the graph.

Create a figure to submit that has 4 subplots (arranged 2 by 2 works well): the original signal `x`, the absolute value of its DFT, the signal multiplied by a window function (your choice of `a` and `t0`), and the absolute value of the windowed signal's DFT. Clearly label axes, and make sure frequency is plotted with respect to Hertz.

Exercise 2 (Movie of sliding window) We can use a for-loop to visualize the window sliding along the time axis.

```
a=1;
tslide=0:0.1:10;
for j=1:length(tslide)
g=exp(-a*(t-tslide(j)).^2);
xg=g.*x;
xghat=fft(xg);
subplot(3,1,1), plot(t,x,'k',t,g,'r')
axis([0 10 -1 1])
subplot(3,1,2), plot(t,xg,'k')
axis([0 10 -1 1])
```

```
subplot(3,1,3), stem(fftshift(freq),abs(fftshift(xghat))/max(abs(xghat)))
axis([-10 10 0 1])
drawnow
pause(0.2)
end
```

Submit a figure showing the final frame of your sliding window movie. Clearly label axes, and make sure frequency is plotted with respect to Hertz (you need to have defined an appropriate `freq` array).

Exercise 3 (Spectrogram) For the last step, we want to visualize the information from the sliding window movie in a single image, the spectrogram, which color codes the energy (squared DFT coefficient) associated with each frequency at each time point. The spectrogram shows the windowed DFT centered at each time point as a color-coded vertical line so you can see what the strongest frequencies (in red) are over time.

```
for j=1:length(tslide)
g=exp(-a*(t-tslide(j)).^2);
xg=g.*x;
xghat=fft(xg);
xghat_spec(j,:)=abs(fftshift(xghat)).^2;
end

figure
imagesc(tslide,fftshift(freq),xghat_spec')
axis([0 10 -10 10])
colormap(hot)
```

Submit two figures showing the spectrograms for `a=0.2` and with `a=5`. Note the difference in spectrogram resolution between the wide window and the narrow window.