Handout 4 – Short-time frequency analysis

Spectrogram in Matlab
Matlab 6.5 and earlier versions contain the function `specgram` to compute the short-time Fourier transform (STFT) or spectrogram of a signal. With the introduction of Matlab 7 and Matlab 2006a, this version has been replaced by a new function, `spectrogram`. The old `specgram` function is still supported, although help for the function is minimal, but it may be removed from future versions of Matlab altogether.

The two functions provide similar functionality. The `specgram` function is simple in use, has a straightforward and very common graphical display of the spectrogram, and has workable default settings. It is sufficient to call the `specgram` function with only the signal as a parameter and get a meaningful result:

```matlab
>> [x,fs] = wavread('ex3_1.wav'); % load a speech signal
>> figure, specgram(x)           % display spectrogram in new window
```

On the other hand, the `spectrogram` function has a little more functionality, as it can also plot and return the power spectral density of a signal, which basically is a spectrogram with magnitude corrected for window power and sampling frequency. But its default graphical output is pretty much useless for most signal processing applications, see for example

```matlab
>> [x,fs] = wavread('ex3_1.wav'); % load a speech signal
>> figure, spectrogram(x)        % display spectrogram in new window
```

And instead of plotting the spectrogram as an image, using `imagesc` like `specgram` does, `spectrogram` plots it as a 3D surface, using the `surf` command. That makes the spectrogram look smoother, which is probably the reason for the change. But it also makes the graphics output of the `spectrogram` function much larger and slower than that of the old `specgram` function.
In this course we will only use the spectrogram function, because it looks like that will be only function supported in future versions of Matlab. The first exercise will introduce the function and the parameters that are necessary to get a meaningful spectrogram plot.

Exercise 4.1 – Speech spectrogram

a. Recreate the spectrogram figure above by using the following Matlab commands:

```matlab
>> [x,fs] = wavread('ex3_1.wav'); % load a speech signal
>> figure, spectrogram(x) % display spectrogram in new window
```

b. As you can see in this figure, normalized frequency is on the horizontal axis, and time is on the vertical axis. It is more common in signal processing to have time on the horizontal axis and frequency on the vertical axis. The spectrogram plot can be changed by adding a 'yaxis' flag to the end of the parameter list, as follows:

```matlab
>> spectrogram(x,'yaxis') % plot frequency on the y-axis
```

c. You may notice that the plot of part b consist of very horizontal lines. This indicates a high resolution in frequency, but a low resolution in time. In other words, spectrogram uses a long window by default. We can specify a window length ourselves as the second parameter:

```matlab
>> spectrogram(x,256,'yaxis') % use a 256-point Hamming window
```

Notice how the 'yaxis' flag remains the last parameter. By specifying a number as the second parameter, we're instructing spectrogram to use a Hamming window of the length. If we'd like another window shape, we can also specify the window as a vector:

```matlab
>> spectrogram(x,rectwin(256),'yaxis') % 256-pt rectangular window
```

d. The plot in part c has decent frequency resolution, but looks a bit ‘grainy’ in time. The spectrogram function uses 50% overlap between windows by default. To increase the resolution in time, we need to specify a higher amount overlap between windows. This is the third parameter of the spectrogram function:

```matlab
>> spectrogram(x,256,224,'yaxis') % 256-pt Hamming, 224 pt overlap
```

This spectrogram has finer resolution and more detail in time.

e. The fourth parameter of the spectrogram function determines the number of points at which the short-time spectra are evaluated. The default setting is to use as many points as the window length. To create a spectrogram that looks smoother in frequency, we can evaluate the spectra at more points than that. This technique is called oversampling in frequency, and is done as follows:

```matlab
>> spectrogram(x,256,224,1024,'yaxis') % evaluate at 1024 points
```

f. Finally, we can add the sampling frequency as a parameter, so that the spectrogram function can change the units of the axis from time index and normalized frequency to time in seconds and frequency in Hz.

```matlab
>> spectrogram(x,256,224,1024,fs,'yaxis') % use sampling frequency
```
Exercise 4.2 – Wideband and narrowband spectrograms
   a. Would you call the spectrogram in 4.1f a wideband spectrogram or a narrowband spectrogram?
   b. Choose different parameters for the spectrogram function to get the other type of spectrogram.

Exercise 4.3 – Improving and comparing spectrograms: color limits
When Matlab displays a spectrogram, it uses a coloring scheme called 'jet'. This coloring scheme makes it easy to identify regions of certain energy levels in the spectrogram: red for high energy, yellow for medium energy, light blue for low energy, and dark blue for very low energy. But often times, the dynamic range of these colors is very large (> 100 dB), and also different for each spectrogram.

   a. Load the two speech signals from the previous exercises into Matlab:

      >> [x1,fs] = wavread('ex1_3.wav'); % load 'zero' speech signal
      >> [x2,fs] = wavread('ex3_1.wav'); % load 's' speech signal

   b. Plot the spectrogram of each signal in its own figure:

      >> figure(1), spectrogram(x1,256,224,1024,fs,'yaxis')
      >> figure(2), spectrogram(x2,256,225,1024,fs,'yaxis')

   c. Put a color bar on each plot using the colorbar function:

      >> figure(1), colorbar
      >> figure(2), colorbar

   d. Notice the large dynamic range in colors (-45 db to -155 dB for figure 1, -40 dB to -155 dB for figure 2).

   e. Download the file climdb.m and climmatch.m from the class website. Put them in the local directory C:\Temp\SPHSC503\, and change Matlab's current directory to that directory. The climdb function limits the dynamic range of the colors to a specified range, see help climdb for details. To restrict the dynamic range of the spectrogram to 80 dB, for example, type

      >> figure(1), climdb(80), colorbar
      >> figure(2), climdb(80), colorbar

      Restricting the dynamic range like this removes some of the noisy details from the spectrogram and highlights the more interesting features.

   f. Still, the color range in the two figures is different (Figure 1 is -43 dB – -123 dB, figure 2 is -39 dB – -119 dB). As a result, a shade of red in figure 1 does not correspond to the same energy level as the same shade of red in figure 2. To make the meaning of the colors the same thing, and to facilitate comparisons between the two speech spectrograms, we must match the color range on the figures. To match the figures, use the climmatch function, see help climmatch for details:
Now, colors in the two spectrograms have the same meaning, and it is fair to compare the two spectrograms.