MATLAB 4, Part 1: Circular convolution in Matlab

Architectural acoustics is the discipline of designing rooms (especially auditoriums or concert halls) that have good acoustical characteristics. Basically, this comes down to designing the impulse response of the room. A typical step in the design process is 'auralization', or playing back simulated sounds for the different expected impulse responses (Acentech, a local firm, specializes in this work). That process uses convolution, and because the sound samples may be large, fft-based convolution is used.

In general, more reverberant spaces (longer impulse response) can be good for music, while a less reverberant space (shorter impulse response) is better for speech. The acoustician can control the length of the impulse response by adding absorbing material, for example fabric on the walls. In this case, you'll use a measured impulse response for a very reverberant space (a stone church).

Note, only steps

1) Start by reading in the sound file, 'onscreen.wav', using the Matlab command audioread (or if you have an older version, you can use wavread):

   ```
   [x,Fs]=audioread('onscreen.wav');
   ```

   This returns both the single-channel time series *x* and the sampling frequency *Fs*. Load the data and listen to it (using *soundsc(x,Fs*)). Plot the signal and look at it. You should see (and hear) two bursts of activity ("on" and "screen"), and a third small bump (a beep).

   This sound clip is probably older than most EE125 students, but you can google Jean Luc Piccard if you are wondering what it is….

2) Load the impulse response, stored in the file smallChurch_fs22k .mat (the sampling rate matches here). Then, using linear convolution, determine what the listener will hear if this sound is played back in the room. Plot h(n). Can you see evidence of individual reflections when you look at h(n)?

3) Now, assume you want to implement FFT-based convolution. Calculate how much zero-padding you require to have FFT-based circular convolution equal linear convolution. Describe your calculation in your report.

4) Then, calculate the convolution result with and without zero-padding and compare to linear convolution (note, even without full zero padding, you will still need to pad h(n) so it is of the same length as x(n)). Check the results against linear convolution, and listen to the two outputs. Do you hear any differences? Listen especially for the beeping sounds.

5) Document your calculations of the zero-padding needed. Also produce a figure containing five subplots, which should contain time in seconds on the x axis:
   a. A plot of the original signal
   b. A plot of output produced by linear convolution (signal vs. time)

c. The *difference* between the linear convolution output and your zero-padded FFT-based convolution. If these are of different lengths, your plot should be as long as the shorter of the two. Hint: any differences should be comparable to roundoff error.
d. A plot of the output produced by FFT-based convolution, without zero-padding.
e. The *difference* between linear convolution and your non-padded FFT-based convolution. If these are of different lengths, your plot should be as long as the shorter of the two.

Based on plot d), can you make any statements about where and when time-domain aliasing occurs, if padding is not used?