MATLAB Lecture 7. Signal Processing in MATLAB

We have seen how to fit data with **polyfit** and how to design shapes with **spline**. In this lecture we cover another way to deal with approximate data, which is well suited for treating signals [1].

7.1 Fourier Transforms

Every signal can be written as a sum of sinusoids with different amplitudes and frequencies. The MATLAB command to compute the Fourier Transform and its inverse are respectively **fft** and **ifft**, for example:

```
>> x = rand(1,10); % suppose 10 samples of a random signal
>> y = fft(x);     % Fourier transform of the signal
>> iy = ifft(y);   % inverse Fourier transform
>> x2 = real(iy);  % chop off tiny imaginary parts
>> norm(x-x2);     % compare original with inverse of transformed
```

The **fft** is the abbreviation of **F**ast **F**ourier **T**ransform. This algorithm implements the discrete Fourier transform to transform data from time into the frequency domain. The study of this algorithm is normally covered in a good linear algebra course. First we give an example of the meaning of the Fourier transform before showing how Fourier transforms can be used to filter noise from signals.

7.2 Waveform and Amplitude Spectrum

Suppose we sample a signal during 4 seconds, at a sampling rate of 0.01:

```
>> dt = 1/100;                        % sampling rate
>> et = 4;                            % end of the interval
>> t = 0:dt:et;                       % sampling range
>> y = 3*sin(4*2*pi*t) + 5*sin(2*2*pi*t); % sample the signal
```

A natural plot is that of amplitude versus time:

```
>> subplot(2,1,1);                    % first of two plots
>> plot(t,y); grid on                 % plot with grid
>> axis([0  et  -8  8]);              % adjust scaling
>> xlabel('Time (s)');                % time expressed in seconds
>> ylabel('Amplitude');               % amplitude as function of time
```

With the Fourier Transform we can visualize what characterizes this signal the most. From the Fourier transform we compute the amplitude spectrum:

```
>> Y = fft(y);                        % compute Fourier transform
>> n = size(y,2)/2;                   % 2nd half are complex conjugates
>> amp_spec = abs(Y)/n;               % absolute value and normalize
```

To visualize the amplitude spectrum, we execute the following commands

```
>> subplot(2,1,2);                    % second of two plots
>> freq = (0:79)/(2*n*dt);            % abscissa viewing window
>> plot(freq,amp_spec(1:80)); grid on % plot amplitude spectrum
>> xlabel('Frequency (Hz)');          % 1 Herz = number of cycles/second
>> ylabel('Amplitude');               % amplitude as function of frequency
```

On the amplitude spectrum we see two peaks: at 2 and 4. The location of the peaks occurs at the two frequencies in the signal. The heights of the peaks (5 and 3) are the amplitudes of the sines in the signal.

7.3 Filtering Noise from Signals

We will see now how to use **fft** and **ifft** to filter out the noise from signals. First we add random noise to our signal and compute the amplitude spectrum.

```
>> noise = randn(1,size(y,2));      % random noise
>> ey = y + noise;                   % samples with noise
>> eY = fft(ey);                     % Fourier transform of noisy signal
>> n = size(ey,2)/2;                 % use size for scaling
>> amp_spec = abs(eY)/n;             % compute amplitude spectrum
```

To interpret these calculations we make a plot of the waveform and amplitude spectrum:

```
>> figure                            % plots in new window
>> subplot(2,1,1);                   % first of two plots
>> plot(t,ey); grid on               % plot noisy signal with grid
>> axis([0  et  -8  8]);             % scale axes for viewing
>> xlabel('Time (s)');               % time expressed in seconds
>> ylabel('Amplitude');              % amplitude as function of time
>> subplot(2,1,2);                   % second of two plots
>> freq = (0:79)/(2*n*dt);           % abscissa viewing window
>> plot(freq,amp_spec(1:80)); grid on % plot amplitude spectrum
>> xlabel('Frequency (Hz)');         % 1 Herz = number of cycles per second
>> ylabel('Amplitude');              % amplitude as function of frequency
```

On the first plot we recognize the shape of the signal. In the plot of the amplitude spectrum, the peaks and their heights are the same as on the plot of the amplitude spectrum of the original signal. The wobbles we see around the peaks show that the amplitude of the noise is less than that of the original signal. We can visualize the output of the Fourier transforms:

```
>> figure                            % new window for plot
>> plot(Y/n,'r+')                    % Fourier transform of original
>> hold on                           % put more on same plot
>> plot(eY/n,'bx')                   % Fourier transform of noisy signal
```

Via the inverse Fourier transform, we filter out the noise. The command **fix** rounds the elements of its argument to the nearest integers towards zero. For this example, we use **fix** to set all elements in eY less than 100 to zero:

```
>> fY = fix(eY/100)*100;             % set numbers < 100 to zero
>> ifY = ifft(fY);                   % inverse Fourier transform of fixed data
>> cy = real(ifY);                   % remove imaginary parts
```

The vector cy contains the corrected samples. So, finally we plot this corrected signal:

```
>> figure                            % new window for plot
>> plot(t,cy); grid on               % plot corrected signal
>> axis([0  et  -8  8]);             % adjust scale for viewing
>> xlabel('Time (s)');               % time expressed in seconds
>> ylabel('Amplitude');              % amplitude as function of time
```

Here we filtered out noise of low amplitude. Note we can also remove noise of high frequency.

7.4 Assignments

1. Consider the following sequence of instructions:

```
>> t = 0:0.1:10;
>> y1 = sin(2*pi*t);
>> y2 = sin(20*pi*t);
>> plot(t,y1);
>> hold on;
>> plot(t,y2);
```

   Why is the output of the second plot like this? Find a better range for `t` to plot `sin(20*pi*t)` right. Can you find a good lower bound for the sampling interval in terms of the frequency?

2. Give the MATLAB commands to plot the amplitude spectrum for the signal

$$f(t) = \sum_{k=10}^{20} (20 - k) \sin(2\pi k t).$$

   In addition, plot the waveform spectrum of this signal.

3. Make a function to plot waveform and amplitude spectrum of a signal. The function has prototype:

```
function specplot ( t, dt, et, y )
%
%  Opens a new figure window with two plots:
%  the waveform and amplitude spectrum of a signal.
%
%  On entry :
%    t          sampling range of the signal;
%    dt         sampling rate;
%    et         end of the range;
%    y          samples of the signal over the range t.
%
```

   So `specplot` computes the amplitude spectrum of the signal. For the abscissa viewing window you may take half of the range of t.

   Test your `specplot` with the signal of the previous assignment.

4. With **fft** we can decompose a signal in low and high frequencies. Take the example signal from page 1. As noise we now add a sine of amplitude 4 and with frequency 50. Plot the waveform and amplitude spectrum of the new signal. Use **fft** and **ifft** to remove this high frequency noise.

References

[1] S.D. Stearns and R.A. David. *Signal Processing Algorithms in MATLAB*. Prentice Hall, 1996.