MATH.2720 Introduction to Programming with MATLAB
Fourier Series Basics

A function f that is periodic on the interval $[-\pi, \pi]$ can be represented by the *Fourier series*

$$f(t) = \frac{a_0}{2} + \sum_{j=1}^{\infty} [a_j \cos(jt) + b_j \sin(jt)]$$

In practice, the infinite series is approximated by a partial sum:

$$f(t) \approx \frac{a_0}{2} + \sum_{j=1}^{m} [a_j \cos(jt) + b_j \sin(jt)]$$

To calculate the $n = 2m + 1$ coefficients $a_0, a_1, \ldots, a_m$ and $b_1, b_2, \ldots, b_m$ we need n values of f.

These are usually taken to be values of f at equally spaced t values such as

$t_1 = 0, t_2 = \dfrac{2\pi}{n}, t_3 = \dfrac{4\pi}{n}, \ldots, t_n = \dfrac{2(n-1)\pi}{n}$.

Let $x_k = f(t_k),\ k = 1, 2, 3, \ldots, n$ and let x denote the array $x = [x_1, x_2, x_3, \ldots, x_n]$.

The MATLAB command `z = fft(x)` will generate an array `z` containing n complex numbers. The arrays of coefficients a_j and b_j can be recovered from the array `z` as follows:

```
a = real(2*z(1:m+1)/length(z))

b = -imag(2*z(2:m+1)/length(z))
```

Here are three examples. You can download the script files containing these commands from the course web page.

Example 1.

```
n = input('Enter number of data points: ');
t = (2*pi/n)*(0:(n-1));
x = 1 + cos(t) + 2*sin(2*t); %Create a simple signal.
z = fft(x); %fft is the fast Fourier transform algorithm
%Note that if n is odd z(n) is the complex conjugate of z(2),
%z(n-1) is the complex conjugate of z(3), etc.
m = (n-1)/2;
a = real(2*z(1:m+1)/n); %Recover frequency content of signal.
b = -imag(2*z(2:m+1)/n);
```

Example 2.

```
n = input('Enter number of data points: ');
t = (2*pi/n)*(0:(n-1));
x = cos(t) + 2*sin(2*t) - sin(3*t); %Create a simple signal
x = x + 0.1*(-1+2*rand(1,length(x))); %Add random noise to the signal.
z = fft(x);
m = (n-1)/2;
a = real(z(1:m+1)); %Recover frequency content of signal.
b = -imag(z(2:m+1));
```

Example 3.

```
load handel; %Load an audio signal built into MATLAB
%Array y contains the data. Fs equals the number of samples per second,
%usually 8192.
n = length(y);
plot((1:n)/Fs,y)
xlabel('Time (s)')
ylabel('Amplitude')
sound(y) %Play the audio signal
z = fft(y);
m=(n-1)/2;
z_half = z(1:m+1);
figure
plot(Fs*(0:m)/n,abs(z_half))
xlabel('Frequency (Hz)')
ylabel('Amplitude')
f_cutoff = 2500; %Hz
z_half(round(n*f_cutoff/Fs):end) = 0; %This zeros out the coefficients of terms corresponding
%to frequencies of f_cutoff Hz or more.
figure
plot(Fs*(0:m)/n,abs(z_half))
xlabel('Frequency (Hz)')
ylabel('Amplitude')
pause
z2 = [z_half; conj(z_half(end:-1:2))]; %Reconstruct the full fft
y2=ifft(z2); %ifft is the inverse fft algorithm
sound(y2) %Play the filtered audio signal
```

Practice Problem

Issue the command `load train`

Play this audio signal. Filter our frequencies above 2000 Hz and play the filtered signal.