SIGNALS AND SYSTEMS LABORATORY 5:
Periodic Signals and Fourier Series

INTRODUCTION

The time base signal in an oscilloscope is a sawtooth wave. Function generators produce sine waves, square waves, and triangular waves. Oscillators in radio transmitters and receivers produce high frequency sinusoids. All of these are examples of periodic signals. By definition, a signal $x(t)$ is *T-periodic* if

(1) $x(t+T) = x(t)$ for all t.

If the signal is known over one period $0 \le t < T$, and it is periodic, then it is known for all values of t. Such signals have a *Fourier Series* representation which is a sum of sinusoids, where the frequencies of the sinusoidal terms are integer multiples of the fundamental frequency $f_0 = 1/T$. Joseph Fourier asserted in 1804 that *any* function has such a representation, at least over an interval of length T. The notion of synthesizing a signal with arbitrary wave-shape out of sinusoids is fundamental in many applications in mathematics, science, and engineering. In this exercise we will gain some experience in visualizing this synthesis in both time and frequency.

For real valued $x(t)$ the Fourier series representation can be written two ways: as an exponential series, or as a sine-cosine series:

(2) $$x(t) = \sum_{n=-\infty}^{\infty} X[n]e^{jn\omega_0 t} = a[0] + \sum_{n=1}^{\infty}\left[a[n]\cos(n\omega_0 t) + b[n]\sin(n\omega_0 t)\right]\ ;\ \omega_0 = 2\pi f_0 = \frac{2\pi}{T}$$

In these formulas, the fundamental frequency is $f_0 = 1/T$ and the radian version of the fundamental frequency is $\omega_0 = 2\pi f_0$ The exponential series is parameterized by the complex Fourier coefficient sequence $\{X[n] : -\infty < n < \infty\}$, and the sine-cosine series is parameterized by the real sequence $\{(a[n], b[n]) : 0 \le n < \infty\}$. Note that $X[n] = \frac{a[n] - jb[n]}{2}$. These Fourier coefficients are computed from $x(t)$ as follows:

(3) $$X[n] = \frac{1}{T}\int_0^T x(t)e^{-jn\omega_0 t}\,dt\,.$$

For real signals, the coefficients of the sine-cosine series are computed as

(4) $$a[n] = 2\,\mathrm{Re}(X[n]) = \frac{2}{T}\int_0^T x(t)\cos(n\omega_0 t)\,dt\,, \text{ and}$$

$$b[n] = -2\,\mathrm{Im}(X[n]) = \frac{2}{T}\int_0^T x(t)\sin(n\omega_0 t)\,dt$$

Moreover, for real signals the complex Fourier coefficients obey the symmetry conditions

(5) $X[0]$: real valued
$X[-n] = \overline{X[n]}$, for $n > 0$.
$a[0]$: real valued and $b[0] = 0$
$a[-n] = a[n]$ and $b[-n] = -b[n]$, for $n > 0$

When $X[n]$ is written in its polar form $X[n] = A[n]e^{j\theta[n]}$, then these symmetry conditions are

(6) $X[0] = A[0]$, real
$A[-n] = A[n]$ and $\theta[-n] = -\theta[n]$, for $n > 0$

We often say the complex Fourier series has *even magnitude* $A[-n] = A[n]$ and *odd phase* $\theta[-n] = -\theta[n]$.

SIGNAL REPRESENTATION IN MATLAB

We have a choice of ways to numerically describe such signals. Since the signal is periodic, we need to specify only one period. Our representation will involve L samples, taken over the period T. In our homebrew function 'plotFS.m' (discussed later), the data structure is (x,T), where x is a row vector of samples $x = [x(0), x(t_s), \ldots, x((L-1)*t_s]$, which describes the signal over one period T

What can we infer from this data structure? To begin, the number of samples is L=length(x). Thus the sampling interval is $t_s = T/L$ and the sampling rate is $f_s = L/T$. The fundamental frequency is $f_0 = 1/T$ and the maximum frequency that can be resolved is $f_s/2 = Mf_0 = M/T$, where M is $M = f_s/2f_0 = L/2$. If you want to resolve a large bandwidth, then you must have many samples L on the period T, which is to say the sampling frequency $f_s = L/T$ must be high. To make a long story short, the Fourier series tool, 'plotFS.m' requires only L, which it steals from x, and f0, which it steals from T, to compute the Fourier series coefficients $X[n]$ and place them at the harmonics $f[n] = nf_0$, for $n = 0, \pm 1, \pm 2, \ldots, \pm M$, where $M \le L/2$.

But this leaves some ambiguity. What does the signal do between samples? In order to resolve this ambiguity we must make some assumptions about how to interpolate between samples, which for large L makes little difference, but for small L makes a lot of difference. We might assume, for example that $x(t)$ is linear between samples so that we just connect straight lines between the sample values. This is called *linear interpolation* and is computationally convenient but is not good practice for real signal processing applications because real signals don't behave in that way. We will adopt a *band limited interpolation* assumption that is a bit harder to work with but conforms more closely to laboratory practice.

Nyquist Sampling

We make the assumptions that the highest frequency in the signal $x(t)$ is the half sampling frequency

(7) $W \le f_s/2 = 1/2t_s = L/2T$ Hz,

or equivalently, the (Nyquist) sampling frequency is twice the highest frequency: $f_s \ge 2W$. The other way to read this equation is to say that the sampling frequency $f_s = L/T$ must be greater than $2W$ or that the number of samples must exceed twice the time-bandwidth product: $L \ge 2TW$.

This assumption is standard whenever sampling is done, and for good reason. (We will develop the theory of sampling in later experiments.) Because of the band-limitation assumption, there will only be one consistent way to interpolate between samples. All the Fourier coefficients corresponding to frequencies above the half sampling frequency are zero. Consequently the infinite sum in equation (2) is now finite:

(8) $$x(t) = \sum_{-M}^{M} X[n]e^{jn\omega_0 t}.$$

Therefore $2M \leq L$. Thus the number of terms in the sum (and consequently the highest frequency in $x(t)$) is limited by the number of samples. For large bandwidth signals like square waves, or anything with jumps, we will need to specify many samples to get enough bandwidth for the accuracy we want. Because of the band-limitation hypothesis, the Fourier integral in equation (3) can be done *exactly* using the Discrete Fourier Transform. This is the strategy used in the following homebrew function 'Fseries.m' found on the web page under 'Functions for Lab 5'. Type:

```
» help Fseries

 X=Fseries(x)
 Returns complex Fourier Series coeffs
    X=[X(0), X(1), ... , X(M)] of x(t).

 Assumptions: x(t) is real, T-periodic and
 x=[x(0),x(ts),...,x((L-1)*ts)], where ts=T/L.
 Also, x(t) is bandlimited to  fs/2 Hz,
 and therefore M <= L/2.
```

You should study the code inside the m-file 'Fseries.m'. The Fourier series integral is computed using the fast Fourier transform, or FFT. If you were to look up the equations for the discrete Fourier transform and its inverse and make the band-limitation assumption, then you could show that the integral can be done in this way. For the moment, we will simply accept it. Note that we do not need to specify the period T to compute the Fourier coefficients using 'Fseries.m'. Try this example:

```
»x=cos((2*pi/5)*[0:4]);
»Fseries(x)

  -0.0000           0.5000 - 0.0000i   0.0000 - 0.0000i
```

We can conclude that $X[1] = X[-1] = 1/2$, and that all other Fourier coefficients are zero. From equation (2) this generates a pure cosine.

'plotFS.m': A TOOL TO DISPLAY PERIODIC SIGNALS

The function 'plotFS.m', on the web page under 'Functions for Lab 5', displays the signal in a plot containing three panels. These will show two periods of the signal in time, and the magnitude and phase of the Fourier coefficients. To use this tool, you need to specify the signal using the row vector x, the period T, and the number of harmonics M that you want plotted. Use 'help' and 'type' to learn more about 'plotFS.m'

The function 'plotFS.m' will plot two periods of the signal, on the range -T to T. When you examine the code for 'plotFS.m', you will see the use of the 'stem' function, in lieu of the 'plot' function for the frequency domain panels. This makes a specialized plot suitable for sequences, wherein each data value is plotted as a vertical line with a small circle around the actual value. We use this to emphasize that the Fourier coefficients form a discrete set. To plot them in the usual way would convey the wrong idea, namely that the spectrum is continuous.

AN EXAMPLE USING ACTUAL ELECTROCARDIAGRAM OR "ECG" DATA

To compare the Fourier series plot (which is discrete) with a plot from a spectrum analyzer (which is continuous), consider actual samples from an electrocardiogram signal. Such signals are not really periodic, but are nearly so, with one period per heartbeat. The file 'ecg.mat' contains 25 seconds worth of an electrocardiogram, sampled at the rate $f_s = 200$ samples per second. The duration of one heartbeat is about $T = 0.78$ seconds and the number of samples in a single period is about $L = T \cdot f_s = 0.78(200) = 156$. MATLAB data or *workspace* files can be recognized by the ".mat" extension. Load one periods worth of data in ‘ecg.mat’ from the web page under ‘Workspace files for Lab 5’, and display the Fourier series by typing

```
»load ecg; start=1;ecgp=ecg([start:start+155]);T=length(ecgp)/200;plotFS(ecgp,T,30)
```

This result should be similar to the display shown in Figure One of the signal in time and frequency. Pay no attention to the vertical scale for the time plot. It is simply the A/D converter value as a signed integer. The raw signal is very low voltage.

Notice that there are nontrivial harmonics of high order in this signal. This is characteristic of signals that are close to a train of impulses (which is what the ECG looks like in the time domain). The variable ecgp, above, was extracted from an experimentally measured electrocardiogram signal. Experiment with various choices of the variable start.

Assignment:

1.) Approximately what value of start produced the graph seen below?

The file ‘ecg.mat’ contains 25 seconds worth of this signal, including 32 pulses that are similar but not identical. Thus the signal is not really periodic. If we were to drive a spectrum analyzer with this signal, would the result look anything like the graph of Fourier series coefficient magnitude spectrum in Figure One?

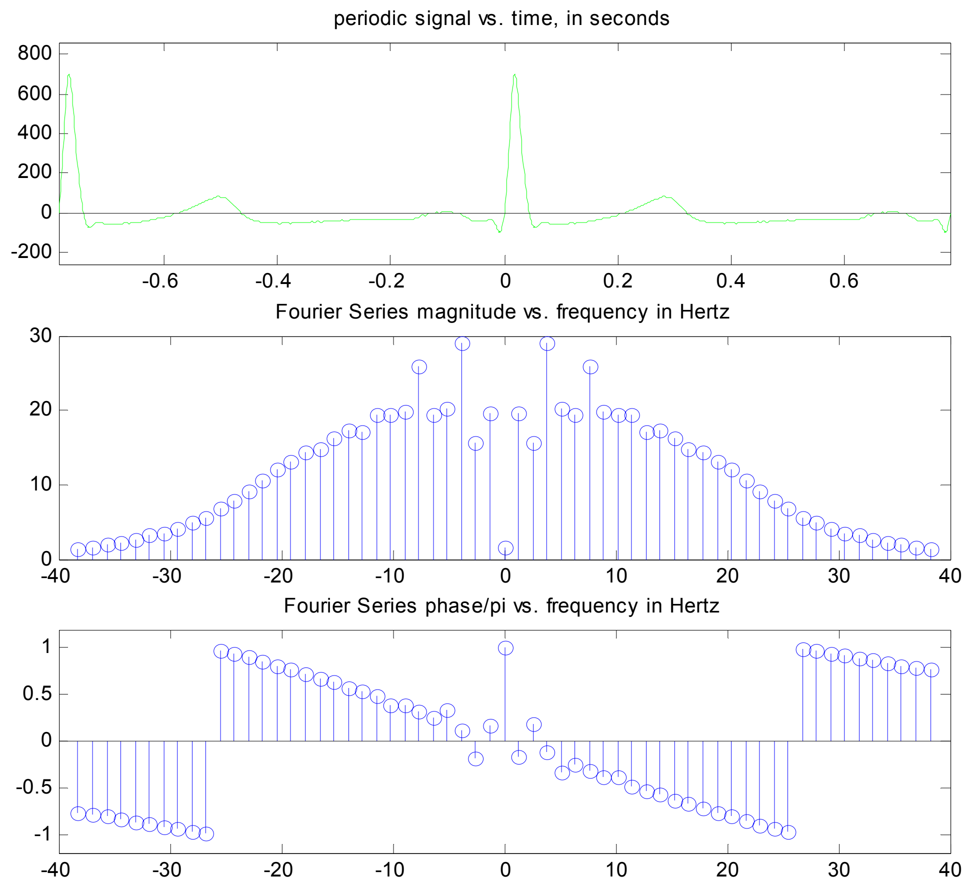


Figure One Fourier series for one period of an electrocardiogram signal.

Type

```
»load ecg;fs=200;plotsig(ecg,fs)  % 'plotsig' function on the web page for Lab 5
```

The result is Figure Two. The spectrum in this plot is what a spectrum analyzer would produce, from roughly 32 heartbeats. These heartbeat ECGS are not identical and are not equally spaced, although you need to look closely to see the differences. Nevertheless, the spectrum agrees strongly with the Fourier series analysis for one heartbeat. Compare the second panel of Figure One, which is discrete, with the second panel of Figure Two. Even though the spectrum is continuous, it is very peaky, and the locations and magnitudes of the peaks correspond well to the Fourier series values at the multiples of the fundamental frequency. The best match is at the lower frequencies.

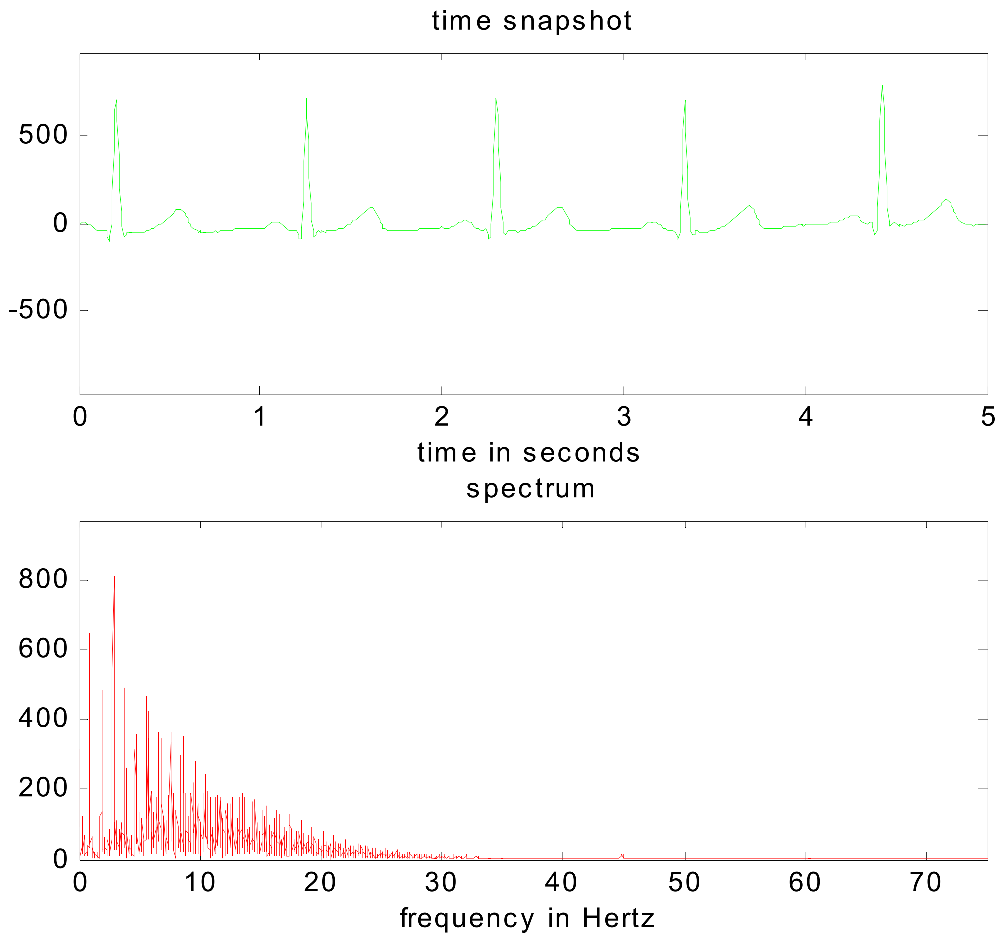


Figure Two Spectrum of a signal containing many beats of an electrocardiogram.

PARTIAL SUMS OF FOURIER SERIES, AND LOWPASS FILTERS

If a periodic signal $x(t)$ is bandlimited to $M \cdot f_0$ Hertz, where $f_0 = 1/T$ is the fundamental frequency, then the sum in equation (1) is finite. This situation can arise by passing a periodic input through a lowpass filter with cutoff frequency between $M \cdot f_0$ Hertz and $(M+1) \cdot f_0$ Hertz. The filter will zero out the harmonics above the cutoff frequency. A partial Fourier series is a sum over a finite number of terms in the infinite sum of equation (2). Figures like those in textbooks that show how the sequence of partial sums converges can be though of as depicting what happens when one passes a periodic signal through a lowpass filter. As the cutoff frequency is reduced, harmonics are suppressed. Suppose $x(t)$ contains lots and lots of Fourier series coefficients. If we extract the first M harmonics (actually $M + 1$ harmonics if you include the DC term), and zero out the rest, then we will get a partial sum like equation (8). The homebrew function 'partial.m', on the web page under 'Functions for Lab 5', takes the time domain representation (x,T), computes the full range of Fourier series coefficients, and then plots M partial Fourier sums. Use 'help' and 'type' to learn more about 'partial.m'. Try this example of a non-symmetric square wave:

```
»x=[ones(1,20),-ones(1,40)];
»T=1;
»partial(x,T,12)
```

The result is Figure Three. We asked for 12 plots (actually 13 plots with when you include the DC term). Why does it appear that we have only 9?

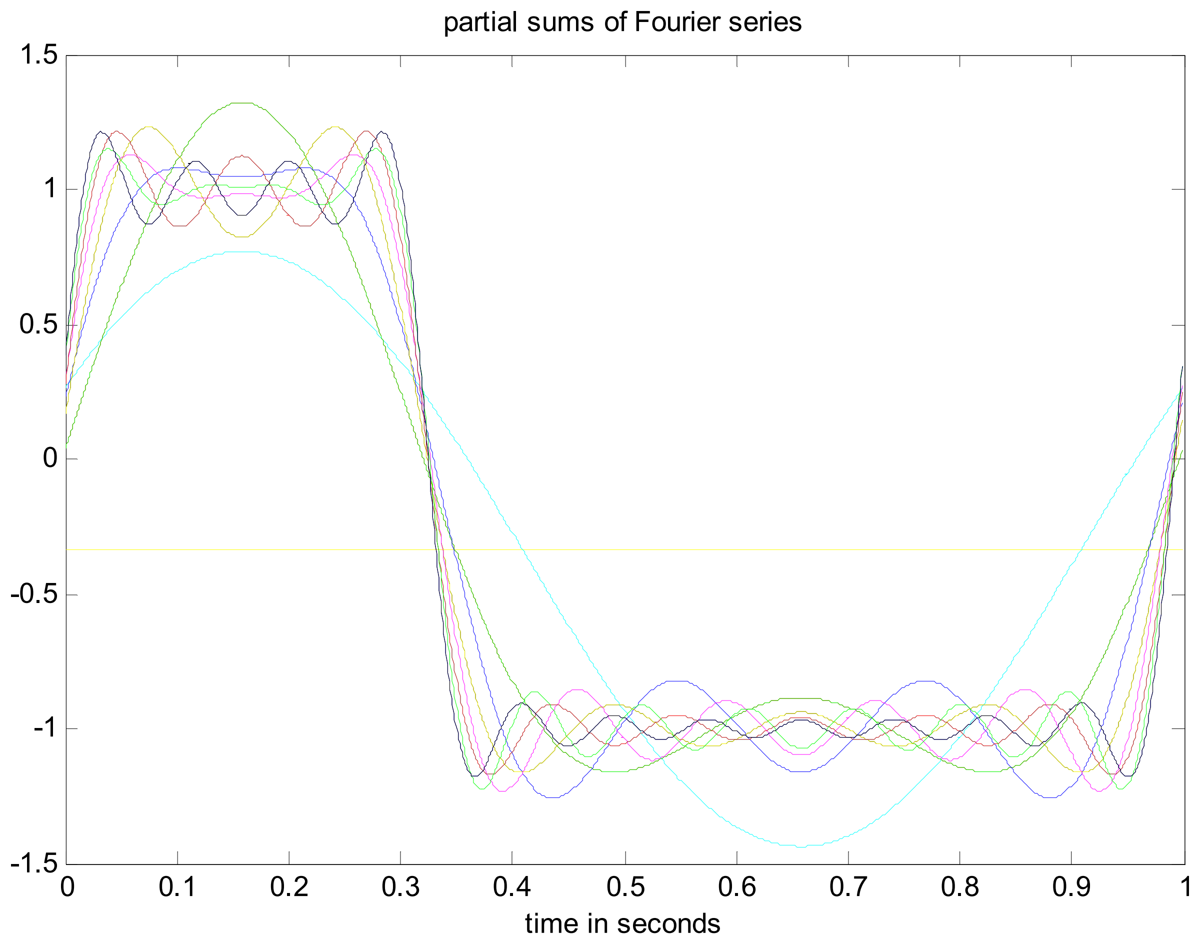


Figure Three Fourier series partial sums, out to 12 terms, for a nonsymmetric square wave.

SOUND

It is instructive to generate a piece of a periodic signal and then use our 'plotsig.m' function, which produces sound when a hardware sound port is available. The following short code body does the job.

```
% a short code body for generating a periodic signal,
% displaying it, and listening to it.
% You must specify the data vector x, and the period T.
        u=zeros(1,7000);
        k=[1:length(x):7000];
        u(k)=ones(size(k));
        plotsig(conv(x,u),length(x)/T)
```

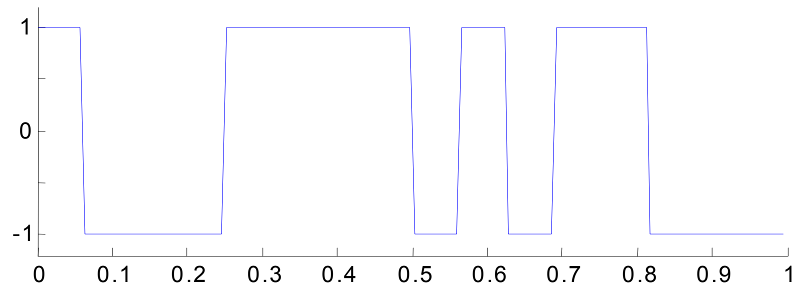
> ***Assignment:***
>
> 2. Explain clearly what the code body shown above is doing.

SIGNALS WHICH ARE IDENTICAL EXCEPT FOR PHASE

It is claimed that the ear can hear only magnitude information. The relative phases of different frequencies are not heard. However, the phase information can make a drastic difference in the appearance of the signal in the time domain. We demonstrate by example. First we construct a *random telegraph* signal $x(t)$, which has levels 1 and -1 depending on the state of a small pn generator with period 15. (Normally the period is much higher, but we want to see an entire period.) The signal $x(t)$ is constructed as follows.

```
»pn=[1 0 0 0 1 1 1 1 0 1 0 1 1 0 0 ];
»L=length(pn);
»x=zeros(1,10*L);
»k=[1:10:length(x)];
»x(k)=pn;
»x=conv(x,ones(1,10));
»x=2*x-1;
»t=(1/length(x))*[0:length(x)-1];
»hold on
»axis([0 1 -1.2 1.2])
»plot(t,x)  % plot signal as one second worth of information
```

Here is the portrait of the random telegraph signal $x(t)$.



> ***Assignment:***
>
> 3. Explain clearly what the code body shown above is doing.

Now inspect the signal *x*(*t*) with 'plotFS.m' tool. Type:

```
»plotFS(x,1,16)
```

You should see the graph below.

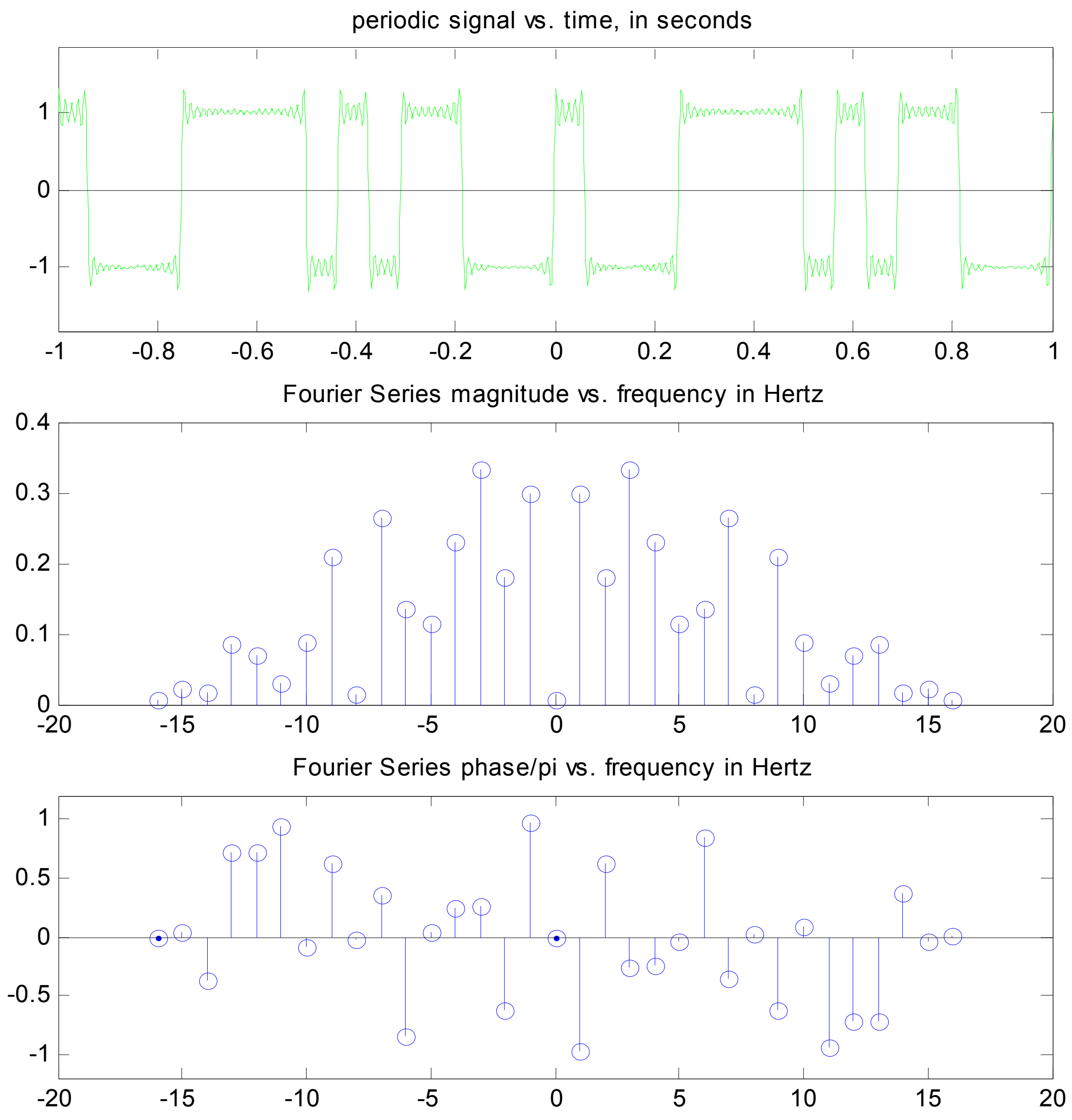


Figure Four A periodic signal which looks like a random data signal over one period.

Now construct a periodic signal $y(t)$ whose Fourier series coefficients are $Y[n] = |X[n]|$. All the phases are now zero, which will make $y(0)$ large. (See the homebrew function 'zerophas.m' on the web page under 'Functions for Lab 5'. Look at a listing of it.)

```
»y=zerophas(x);plotFS(y,1,16)
```

Now the signals $x(t)$ and $y(t)$ have the same magnitude spectrum. If you listened to them, they should sound the same. (You may want to try this.) But they look very different in the time domain. Here is a portrait of $y(t)$.

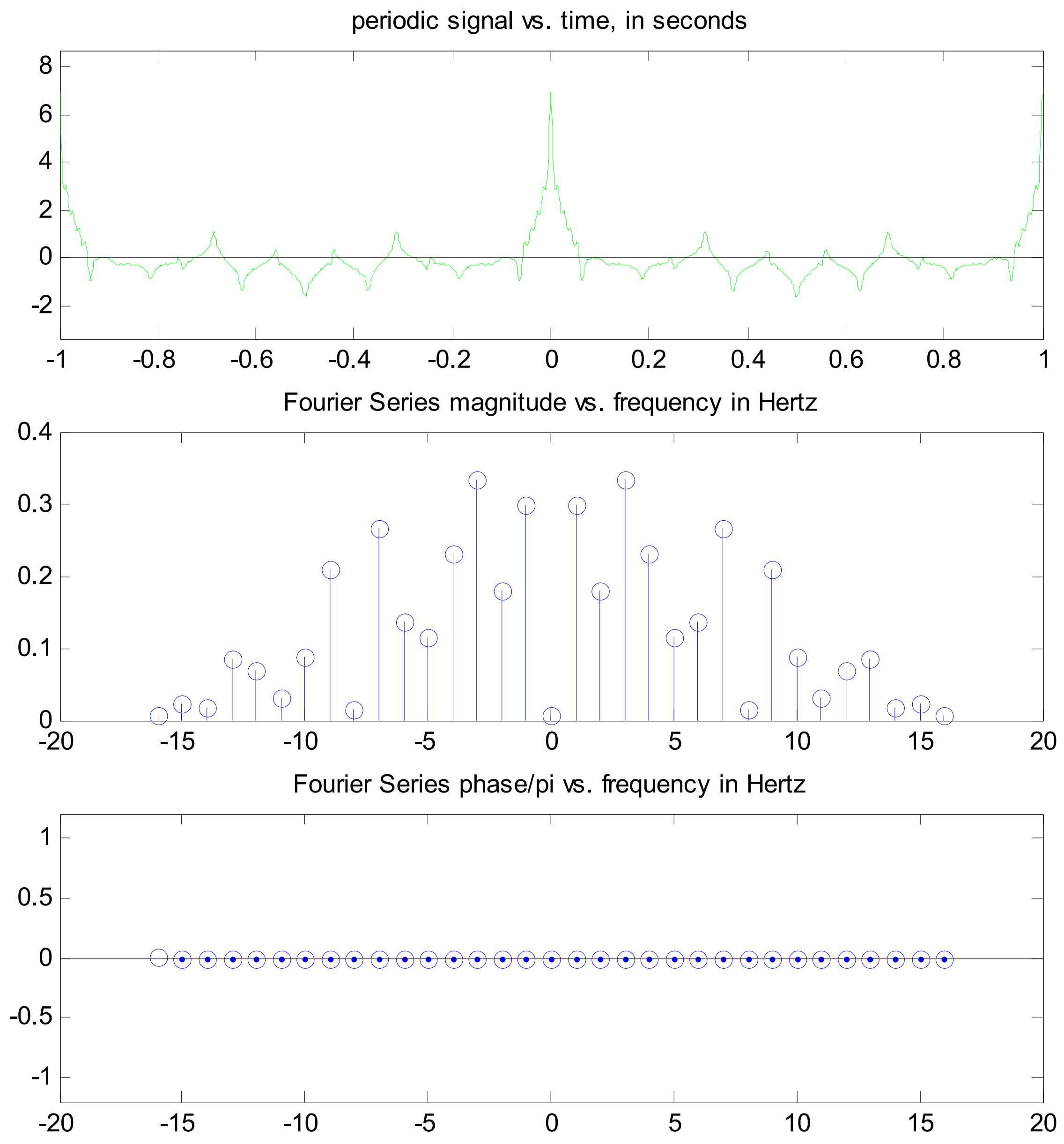


Figure Five A periodic signal that has the same Fourier series magnitude as the signal in Figure Four, but has zero phase. Note the pronounced difference in shape and amplitude in the time domain.

Assignment:

4. *Exercise the tool* 'plotFS.m'

   For each of the following signals: (i) construct the data vector x, (ii) print the graph produced by 'plotFS.m', and (iii) check the results by finding formulas for the Fourier Series coefficients $X[n]$. In every case, choose M =10 harmonics. Be sure to include in your report the method used to compute the data vector x.

   a.) $x(t)$ is a 10 Hz square wave with 66.7% duty cycle. Construct x via

   ```
   »L=500;u=ones(1,L);x=[u,0*u,u];
   ```

   b.) Same as part (a) except set L = 10. Explain why the graph of $x(t)$ is different.
   (The values of $X[n]$ will be the same. You need not repeat the analysis.)

   c.) $x(t)$ is a sinusoid with frequency 100 Hz. Construct a data vector x having 200 samples.

   d.) $x(t)$ is a 1 Hz sawtooth, and satisfies $x(t)=2t-1$ on the period $0 < t < 1$.
   The data vector x should have 500 samples.

5. Construct one period of the sawtooth wave used in problem 4, part (d). Use the function 'partial.m' to display 12 partial sums for this signal. Print out the results.

6. What is the heartbeat for the patient whose ECG is illustrated in Figure One?

7. *Examine symmetry*

   Here are definitions for three kinds:

   (i) even symmetry: $x(-t) = x(t)$ for all t. A cosine has even symmetry.
   The data vector for an even symmetric signal will have the MATLAB form
   [a0, y, zeros(1,m), fliplr(y)] for some real number a0, and row vector y.

   (ii) odd symmetry: $x(-t) = -x(t)$ for all t. A sine has odd symmetry.
   The data vector for an odd symmetric signal will have the MATLAB form
   [0, y, zeros(1,m), -fliplr(y)] for some row vector y.

   (iii) half wave odd symmetry: $x(-t) = -x(t + T/2)$
   The data vector for a half wave odd symmetric signal will have the MATLAB form
   [y, -y] for some row vector y.

   The function 'fliplr' is a MATLAB standard one. Use the 'help' facility to see what it does. For each of the three cases of symmetry, construct a simple signal, and display it using 'plotFS.m'. Print the plot and indicate by hand annotation what special conditions the Fourier coefficients satisfy.

   Suggestion: let y=randn(1,6).

Assignment:

8. *Design Problem*

We want a real 1 Hz periodic signal $x(t)$ for which the magnitudes of $X[0]$, $X[1]$, $X[2]$, and $X[3]$, are all one, but the higher harmonics all vanish. In other words, the highest frequency in $x(t)$ is 3 Hz. Now there are many such signals, since the phases have not been specified. We want a low amplitude signal (to prevent overdriving some physical device) with a specified frequency content. Your problem is to choose the phases so that the amplitude of $x(t)$ is minimized. To construct $x(t)$ you might use the following script:

```
»phi=[0, .3, .4]; % change these values each time. the first can be zero.
»x=128*real(ifft([1,exp(j*phi),zeros(1,121),exp(-j*fliplr(phi))]));
»amplitude=max(abs(x))
amplitude =
   6.9766

»plotFS(x,1,10)
```

It would be wise to change only one phase at a time.