

# Approximation by Trigonometric Polynomials

**Introduction:** The name of Fourier is associated with the investigation of periodic phenomenon, Fourier series, Fourier Transforms, Discrete Fourier Transforms and Fast Fourier Transforms. These mathematical objects are fundamental to the extraction of periodic signals from noise and form the basis of the fields of Signal and Image Processing. Their applications are widespread from instruments for chemical analysis to Shor's Algorithm for factoring large numbers on the, yet to be built, quantum computer. The study and understanding of these objects is important to modern scientific computing and provides an important subject matter for the review of the Calculus and the basic ideas of linear algebra. The development and investigation of Fourier series and transforms will be done in the spirit of computational science using the MATLAB computational environment to produce illustrative graphics.

**Fourier Series:** Let  $f$  be a given real valued function defined on the interval  $[0, L]$  and suppose that  $f$  has a periodic component. How can both the period and magnitude of this component be determined? To investigate this periodicity, approximate the function by a trigonometric polynomial,  $S(t)$  and look for the frequency having the largest coefficient. That is:

$$f(t) \approx S(t) = c_0 + \sum_{n=1}^M \left[ c_n \cos\left(\frac{2\pi n}{L} t\right) + d_n \sin\left(\frac{2\pi n}{L} t\right) \right]$$

or in complex exponential form:

$$f(t) \approx S(t) = \sum_{n=-M}^{n=M} A_n e^{\frac{i2\pi n}{L} t}$$

The size of  $A_n$  and  $A_{-n}$  or  $c_n$  and  $d_n$  gives the contribution of the frequency  $2\pi n/L$  to the representation of  $f$  by the trigonometric approximation. Note that the trigonometric and exponential forms are equivalent. The following formulas define the transformation between the two (for real valued functions  $f$ ):

$$A_0 = c_0, \quad A_n = \frac{c_n - id_n}{2}, \quad A_{-n} = \frac{c_n + id_n}{2} \quad \text{or} \quad c_n = 2 \operatorname{Real}(A_n), \quad d_n = -2 \operatorname{Im}(A_n)$$

Note that  $A_n$  is the complex conjugate of  $A_{-n}$ . This approximation is determined by the coefficients of the trigonometric polynomial. The least squares condition will be used to select these coefficients. That is, the coefficients are determined by minimizing the integral of the square of the difference between the approximation and the function over the interval  $[0, L]$ . The function to be minimized is given by:

$$E(c_0, c_1, \dots, c_M, d_1, \dots, d_M) = \int_0^L (f(t) - S(t))^2 dt$$

**Approximation Theorem:** The coefficients of the Least Squares Approximation to the function  $f$  defined on the interval  $[0, L]$  by trigonometric polynomials  $S(t)$  are unique and are given by:

$$c_0 = \frac{1}{L} \int_0^L f(t) dt \quad c_n = \frac{2}{L} \int_0^L f(t) \cos\left(\frac{2\pi n}{L} t\right) dt \quad d_n = \frac{2}{L} \int_0^L f(t) \sin\left(\frac{2\pi n}{L} t\right) dt$$

$$\text{or } A_n = \frac{1}{L} \int_0^L f(t) e^{\frac{-i2\pi n}{L} t} dt \quad \text{for } -M \leq n \leq M.$$

**Proof:** Only one case will be considered, that of the formula for  $d_k$ . The other cases follow from a similar calculation. To minimize the function  $E$ , compute the various partial derivatives and set them equal to zero. Solve for the values of the coefficients which satisfies these equations and thus, minimize the function  $E$ .

$$\frac{\partial E}{\partial d_k} = 2 \int_0^L (f(t) - S(t)) \sin\left(\frac{2\pi k}{L} t\right) dt = 0$$

$$\text{or } \int_0^L f(t) \sin\left(\frac{2\pi k}{L} t\right) dt = \int_0^L S(t) \sin\left(\frac{2\pi k}{L} t\right) dt = \frac{L}{2} d_k$$

The last equality comes from using the orthogonality of the sine and cosine functions on the interval  $[0, L]$ . To verify uniqueness, let  $S$  be any  $M$ -term trigonometric polynomial and let  $S_f$  be the  $M$ -term Fourier Approximation. Consider the mean square error for the function  $S$ .

$$E(c_0, c_1, \dots, c_M, d_1, \dots, d_M) = \int_0^L (f(t) - S(t))^2 dt = \int_0^L (f(t) - S_f(t) + S_f(t) - S(t))^2 dt$$

$$= \int_0^L (f(t) - S_f(t))^2 + (S_f(t) - S(t))^2 + 2(f(t) - S_f(t))(S_f(t) - S(t)) dt$$

Assume that  $f$  is represented by its Fourier series, then in the last term the quantity  $(f(t) - S_f(t))$  contains only frequencies higher than  $2\pi M/L$  and the term  $(S_f(t) - S(t))$  contains only frequencies up through  $2\pi M/L$ . Orthogonality of sine and cosine functions on the interval  $[0, L]$  implies that all of these integrals are zero. Hence,

$$E(c_0, c_1, \dots, c_M, d_1, \dots, d_M) = \int_0^L (f(t) - S(t))^2 dt = \int_0^L (f(t) - S_f(t))^2 + (S_f(t) - S(t))^2 dt$$

Thus, the Least Squares Condition is satisfied by  $S(t) = S_f(t)$ .

The Least Squares Approximation to  $f$  by trigonometric polynomials on the interval  $[0, L]$  is just the truncated Fourier Series or the trigonometric polynomial  $S(t)$  defined by the Approximation Theorem. This approximation property is true for each value of  $M$  or for the Fourier Series truncated at the  $M$ th term. Note that  $c_0$  is just the average value of  $f$  on the interval  $[0, L]$ .

**Example #1:** Consider the function given by  $f(t) = 6 \sin\left(\frac{2\pi 5}{L} t\right)$ . Since  $f$  is

already in the form of a trigonometric polynomial, the Approximation Theorem implies that  $c_0 = 0$  and  $c_n = 0$  and  $d_n = 0$  except  $d_5 = 6$ .

**Example #2:** Suppose  $f(t) = 4 \cos\left(\frac{2\pi 5}{L} t + \phi\right)$  Using the trigonometric identity

$$\cos(\alpha + \beta) = \cos(\alpha) \cos(\beta) - \sin(\alpha) \sin(\beta)$$

we obtain:  $f(t) = 4 \cos(\phi) \cos\left(\frac{2\pi 5}{L} t\right) - 4 \sin(\phi) \sin\left(\frac{2\pi 5}{L} t\right)$

Thus, Fourier series terms of the same frequency may be represented in two ways, as a linear combination of cosines and sines or as the cosine of the given frequency with a phase shift  $\phi$ . Like Example #1, using the trigonometric identity,  $f$  is in the form of a trigonometric polynomial. Thus, the coefficients are given by:  $c_0 = 0$ ,  $c_n = 0$  except  $c_5 = 4 \cos(\alpha)$  and  $d_n = 0$  except  $d_5 = -4 \sin(\alpha)$ .

**Example #3:** This example considers the case when the frequency of  $f$  is not a harmonic (integer multiple) of the fundamental frequency of the trigonometric polynomial. Assume that  $f(t) = 3 \cos(15 t)$ . Using the Symbolic Toolbox in MATLAB, the Fourier Coefficients may be calculated.

$$c_0 = \sin(15*L)/(5*L)$$

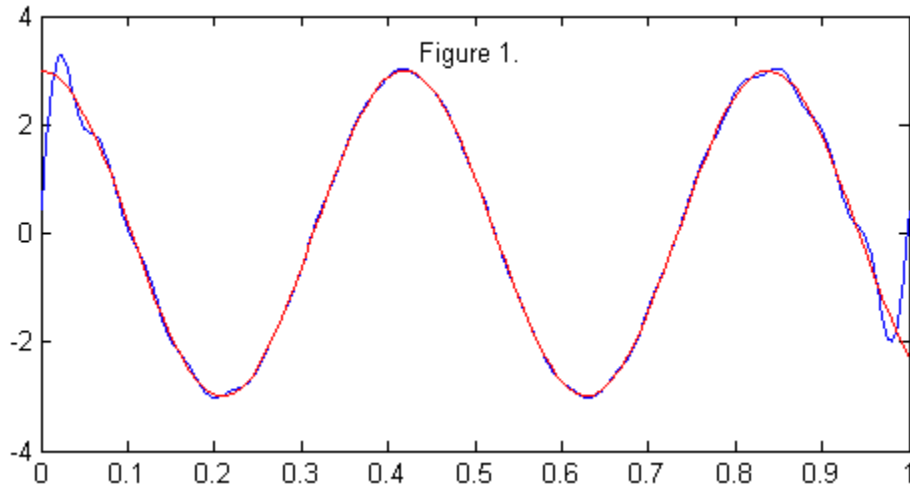
$$c_n = -90*L*\sin(L)*(13440*\cos(L)^6-53248*\cos(L)^{12}+16384*\cos(L)^{14}-2016*\cos(L)^4+112*\cos(L)^2-1-42240*\cos(L)^8+67584*\cos(L)^{10})/(-225*L^2+4*\pi^2*n^2)$$

$$d_n = -12*\pi*n*(-61440*\cos(L)^{13}-15*\cos(L)+28800*\cos(L)^7-6048*\cos(L)^5+560*\cos(L)^3-70400*\cos(L)^9+16384*\cos(L)^{15}+92160*\cos(L)^{11}-1)/(-225*L^2+4*\pi^2*n^2)$$

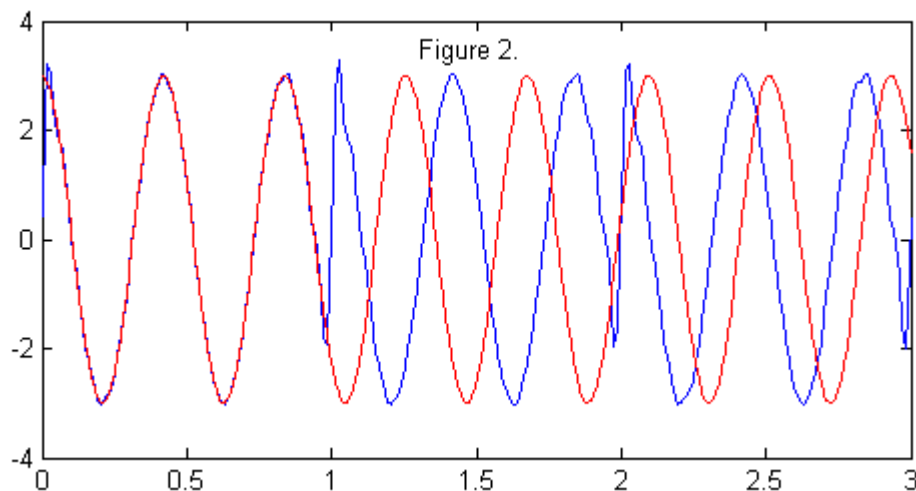
Setting  $L=1$ , the coefficients become:

$$c_0 = 0.1301 \quad c_n = -58.5259/(-255+4*\pi^2*n^2) \quad d_n = (66.3387*n)/(-255+4*\pi^2*n^2)$$

The MATLAB code used in calculating these coefficients may be found in Appendix 1 under Example #3. Also in this Appendix may be found a MATLAB function which plots truncated Fourier Series (FSPlot.m). The following figure shows both  $f$  and the approximation to  $f$  using the first twenty terms of the Fourier Series.

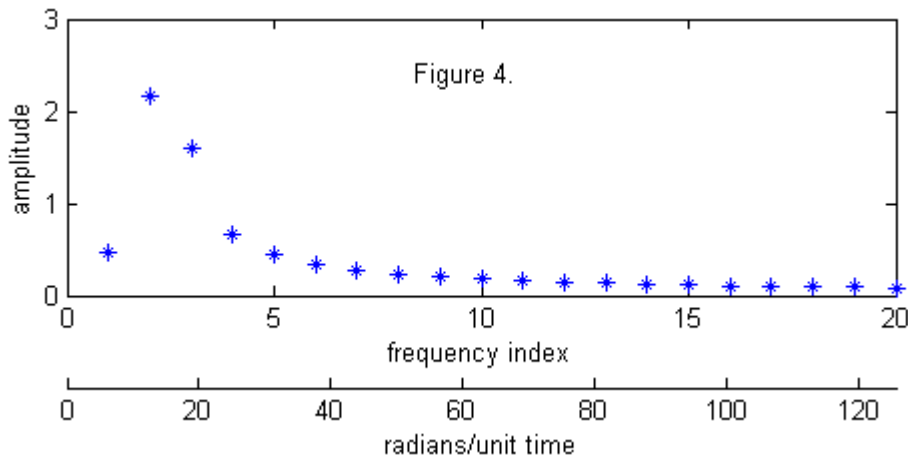
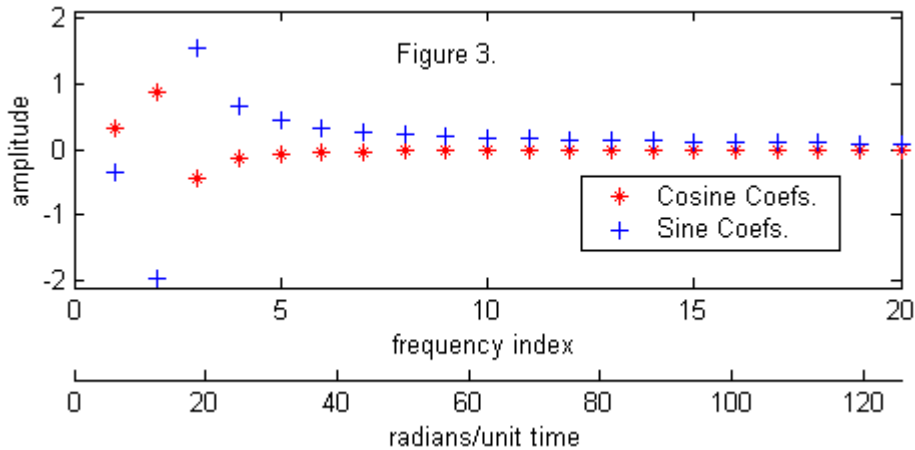


The approximating trigonometric polynomial is defined for all real numbers and not just those in the initial interval  $[0, L]$ . Since the sine and cosine functions in this polynomial are all periodic on  $[0, L]$ , the trigonometric polynomial defines an approximation to the periodic extension of the original function  $f$  beyond the initial interval  $[0, L]$ . Note that the periodic extension of the original function may not coincide with the original function outside of the interval  $[0, L]$ . Figure 2 graphs both  $f$  and the approximation to  $f$  on a larger interval  $[0, 3]$ .



Note that the approximation and the original function  $f$  do not agree outside of the initial interval  $[0, 1]$ . The frequency of the function  $f$  is 15 radians/unit time which lies between two of the harmonics of the fundamental frequency of the trigonometric polynomial  $2\pi \cdot 2 < 15 < 2\pi \cdot 3$ . Figure 3 graphs the values of the Fourier coefficients and Figure 4

graphs the square root of the sum of the squares of  $c_n$  and  $d_n$  for each value of  $n$ . Note that the trigonometric polynomial frequencies adjacent to the frequency of the function  $f$  are most prominent. The frequency is displayed in two ways, in terms of the frequency index, which represents the  $n$ th harmonic of the fundamental frequency  $2\pi/L$ , and in terms of radians/unit time.



From Figure 1, it can be seen that the approximation does not seem as good near the ends of the initial interval. This is because the periodic extension of the function  $f$  is discontinuous at the ends of the interval. Discontinuities cause an oscillation in the approximating series independent of the number of terms used in the approximation. This phenomenon is called the Gibbs Phenomenon, which we will explore in Example #4.

**Example #4:** Consider the unit jump at  $x = a$  given by  $f(t) = H(t-a)$ . Where  $H$  is the Heaviside Function

$$H(t) = \begin{cases} 0 & \text{if } t < 0 \\ 1 & \text{if } t \geq 0 \end{cases}$$

Using the Symbolic Toolbox in MATLAB, we will calculate the Fourier coefficients (See code in Appendix 1).

$$c_0 = 1/L*(L-a) \quad c_n = -2*\sin(a*\pi*n/L)*\cos(a*\pi*n/L)/(pi*n)$$

$$d_n = 2*(-1+\cos(a*\pi*n/L)^2)/(pi*n)$$

Assume that  $L = 1$  and that  $a = 2/3$ . The Fourier coefficients become:

$$c_0 = 0.3333 \quad c_n = -2*\sin(2/3*\pi*n)*\cos(2/3*\pi*n)/(pi*n)$$

$$d_n = 2*(-1+\cos(2/3*\pi*n)^2)/(pi*n)$$

Figure 5 displays both the function and the first 20 terms of it's approximation.

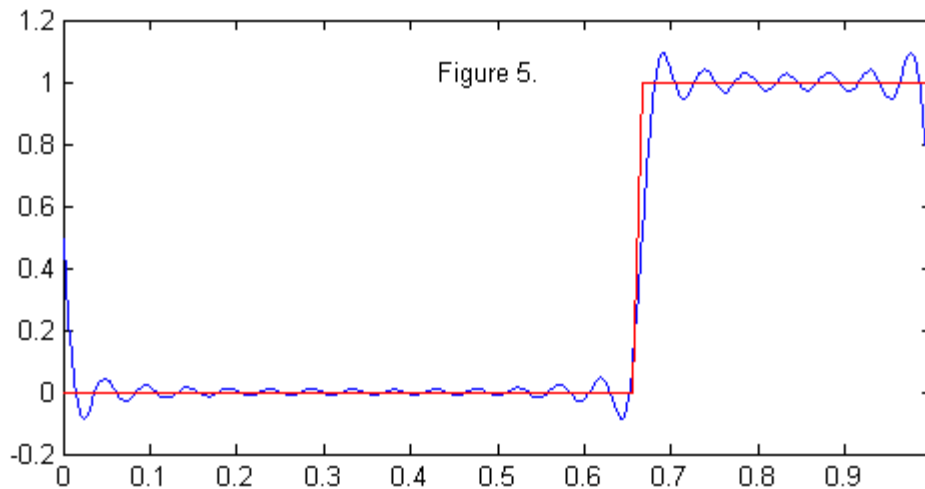
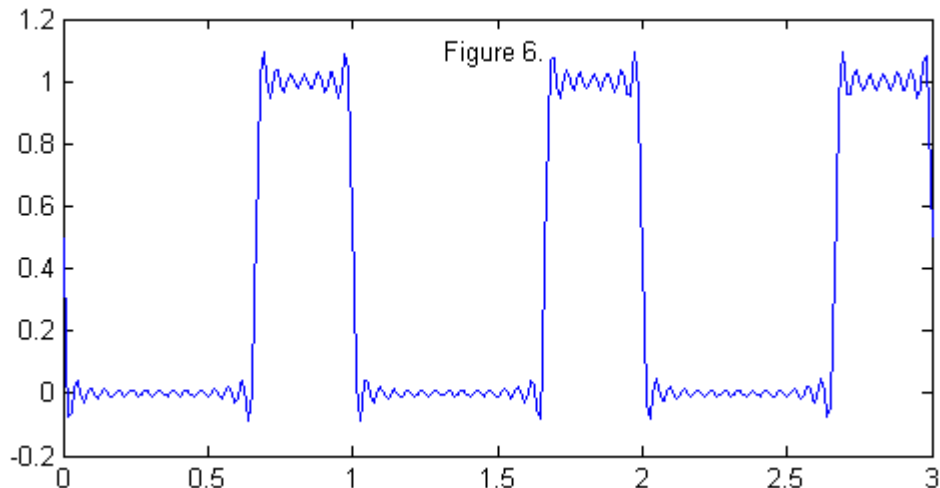
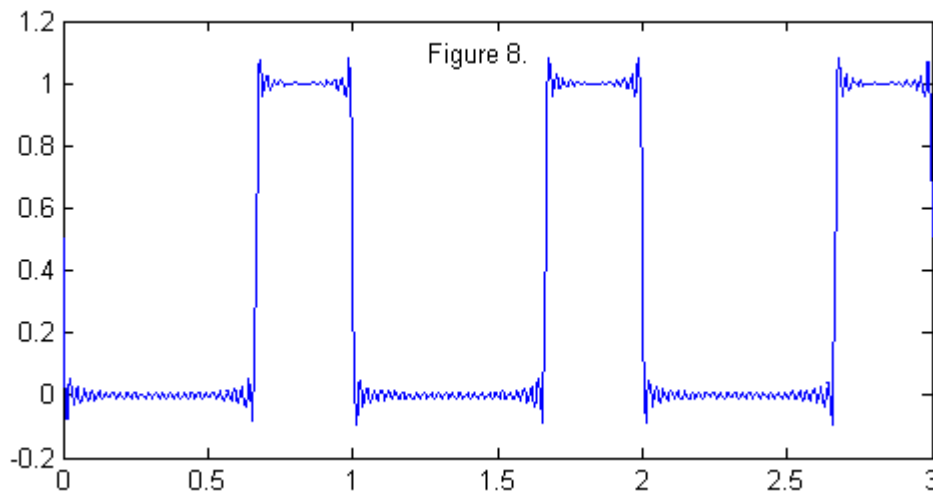
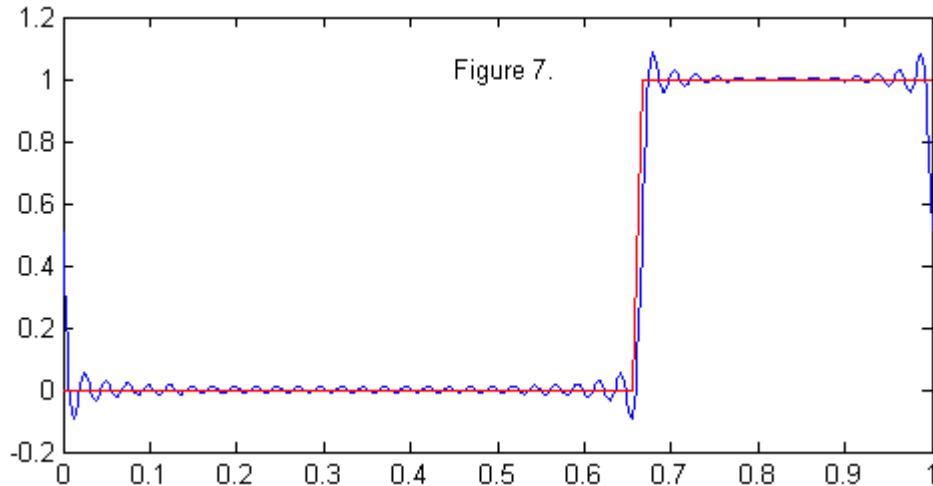


Figure 6 displays 3 periods of the first 20 terms of the approximation.



Figures 7 and 8 display the first 40 terms of the approximation. Note that the overshoot at the jump does not decrease with the increase in number of terms, it remains at about 10% of the height of the jump.



The next example will calculate the Fourier coefficients for a trend, that is for the line  $f(t) = a + bt$  on the interval  $[0, L]$ .

**Example #5:** Assume that  $f(t) = a + bt$  on the interval  $[0, L]$ . Using the Symbolic Toolbox in MATLAB (Code found in Appendix 1), the Fourier coefficients are:

$$c_0 = L*b/2+a, \quad c_n = 0, \quad d_n = -b L/(\pi*n)$$

If we make the following choices for the constants:  $L = 1$ ,  $a = 2$ ,  $b = 0.25$  we obtain:

$$c_0 = 2.1250, \quad c_n = 0, \quad d_n = -1/(4\pi n)$$

Figure 9 graphs the approximation to the periodic extension of this trend over three periods.

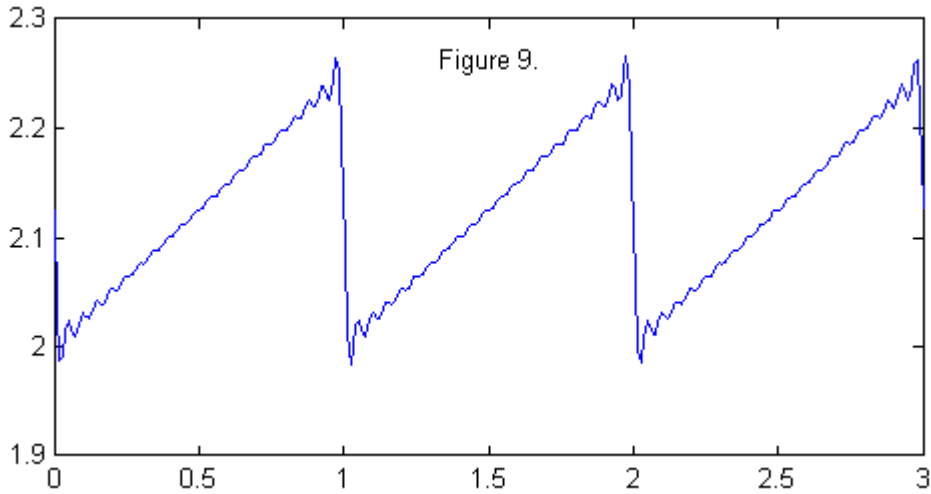
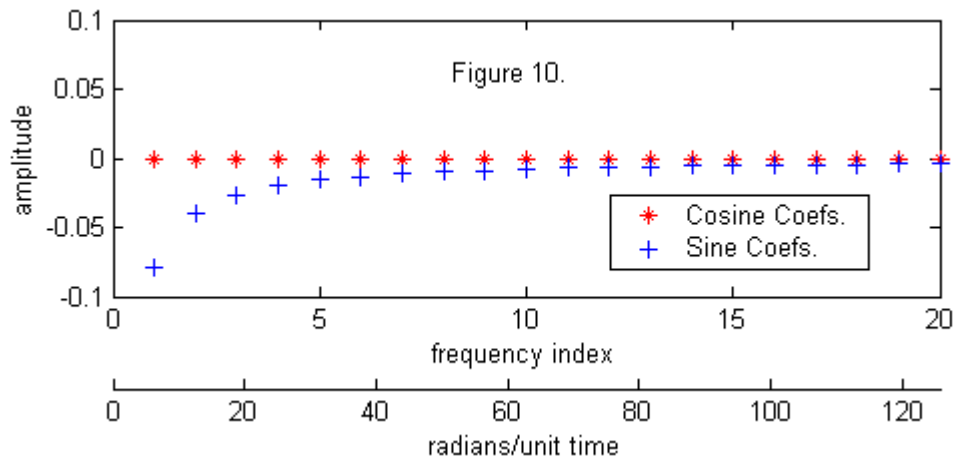


Figure 10 graphs the coefficients of the first twenty terms of the approximation.



If we study Figure 1 from Example #3, we find that the approximating series goes through the average value  $c_0$  at the end points of the interval while the function does not. Hence, the periodic extension of the function has a discontinuity at each end point with the oscillatory behavior of the Gibbs Phenomenon.

**Example #6:** Suppose we multiply the original function in Example #3 by a function, which is zero at the ends of the initial interval. We will scale this function so that its maximum height is 1 at the center of the interval. Consider  $f(t) = 3 \cos(15 t) (16 (t^2)(L - t)^2)/L^4$ . Then redo Example #3.

Figure 11 displays the original function and the modified or windowed function.

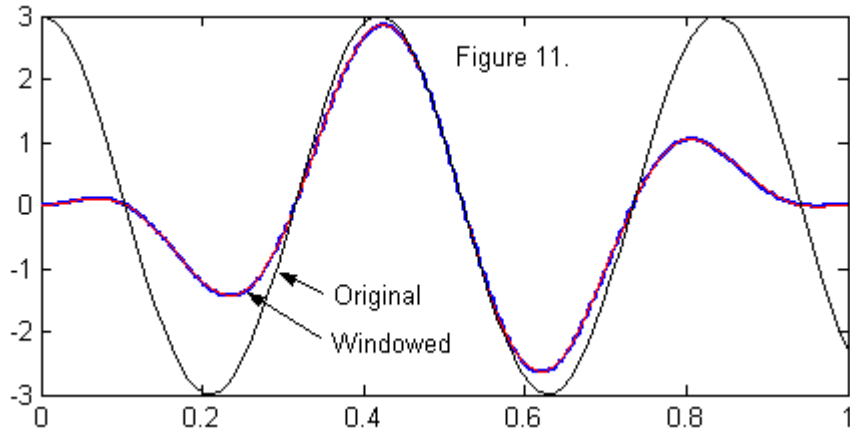


Figure 12 graphs the coefficients.

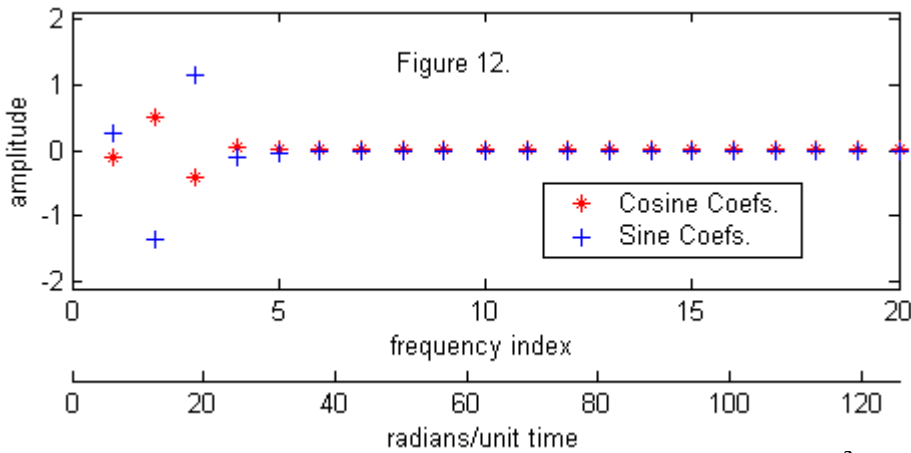
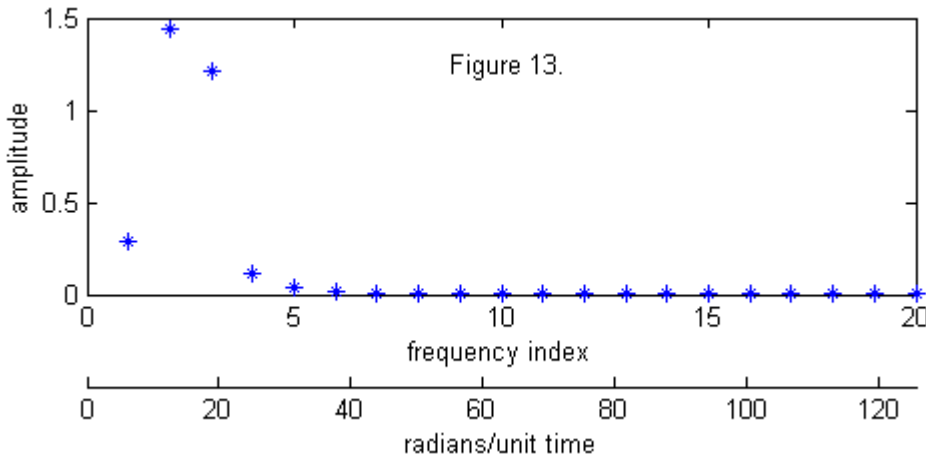


Figure 13 graphs the complex magnitude of the coefficients ( $\sqrt{c_n^2 + d_n^2}$ ).



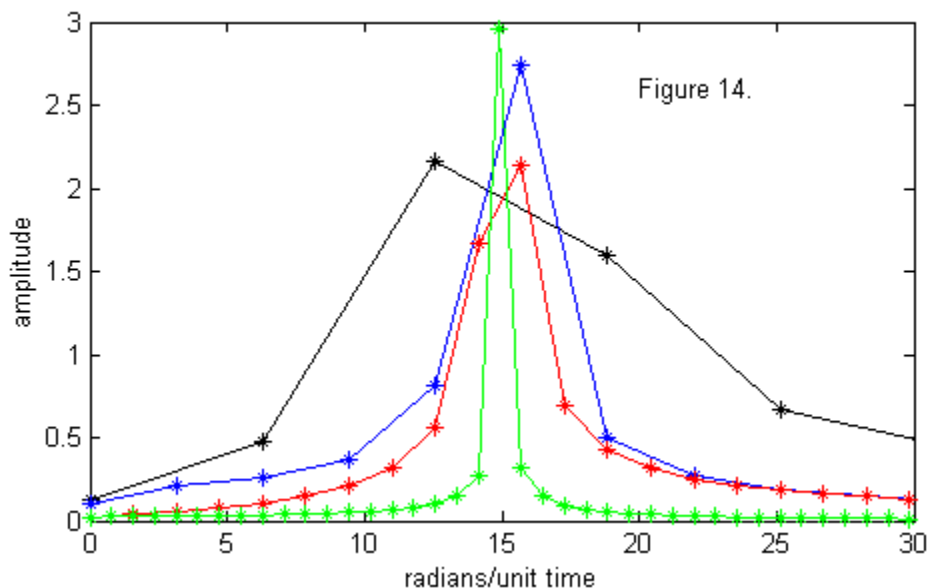
The multiplication by a function, which is zero at the end points of the interval  $[0, L]$ , to reduce the effect of discontinuities at the end points is called windowing. Compare Figures 4 and 13, note that windowing has suppressed the high frequency components due to the discontinuities in the periodic extension of the given function. From the above

graphs (Figures 11, 12 and 13), we notice that the original frequency of 15 radians/unit time which lies between the second and third harmonic of our trigonometric approximation stands out as the only feature.

(The second harmonic  $2\pi \approx 12.57 < 15 < 2\pi \approx 18.85$  the third harmonic).

As the functions become more complicated, numerical integration must be used to evaluate the integrals defining the coefficients. This can be easily done in MATLAB using the built-in integration routine, `Quadl`.

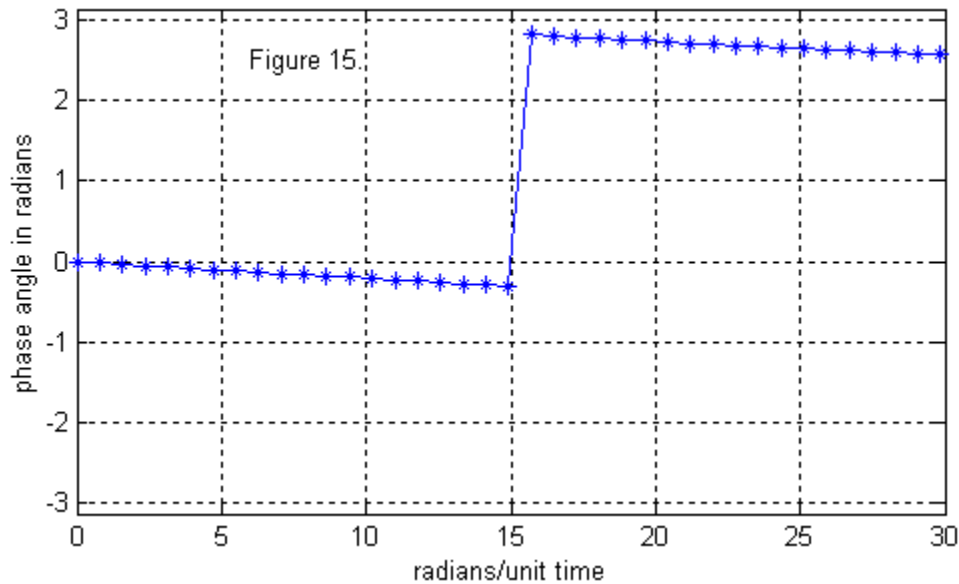
**Example #7:** Continuing with Example #3,  $f(t) = 3 \cos(15t)$ , what happens when we change the length of the initial interval? From the formula for the fundamental frequency,  $2\pi/L$ , as  $L$  increase the fundamental frequency decreases. This decrease provides more detail in the graph of the complex magnitude of the coefficients for a given frequency. Thus, the longer the data set the more resolution is obtained in the frequency spectrum. Consider the cases of  $L = 1, 2, 4$  and  $8$ .



Note that as  $L$  increases the complex magnitude of the coefficients are dominated by the given function's frequency of 15 radians/unit time. Thus, the original function must have the form  $f(t) = \text{amplitude} * \cos(\text{frequency} * t + \phi)$  where  $\phi$  is the phase angle. The phase angle is determined from the individual components of cosine and sine by the formula:

Phase Angle  $\phi = \arctan(\text{sine } n^{\text{th}} \text{ coef.} / \text{cosine } n^{\text{th}} \text{ coef.})$  where  $\arctan$  is the four quadrant version.

Figure 15 plots the phase angle for the case of  $L = 8$ .



From these two plots, the original function  $f(t) = 3 \cos(15t)$  may be recovered.

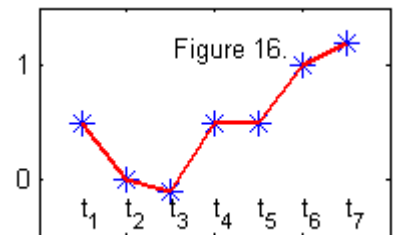
**Fourier Series of a Data Set:** In many cases, the analytic expression for the function  $f$  is not known, only the values of  $f$  at a set of points in the interval  $[0, L]$ . If the Fourier series can be constructed from this data set then the results of Example #7 imply that periodic behavior of this data set can be determined. Suppose this data set is given. To define the Fourier series of this data set, construct the piecewise linear interpolant to this data and from this function compute the approximating Fourier series.

### Piecewise Linear Interpolants

Suppose the following data set on the interval  $[0, L]$  is given.

$0 = t_1$	$t_2$	$t_3$	$t_4$	$t_5$	$t_6$	$L = t_7$
$y_1$	$y_2$	$y_3$	$y_4$	$y_5$	$y_6$	$y_7$

The graph of the piecewise linear interpolant is given in Figure 16.

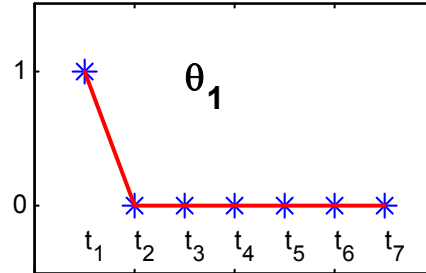


In order to efficiently compute the Fourier series coefficients for this piecewise linear interpolant, write this interpolant as a linear combination of the following basis functions. Each of these basis functions will be the interpolant for a very simple data set based on

the original set of t-values or nodes. The first piecewise linear interpolant basis function  $\theta_1$  and its data set are given by the following:

$0 = t_1$	$t_2$	$t_3$	$t_4$	$t_5$	$t_6$	$L = t_7$
1	0	0	0	0	0	0

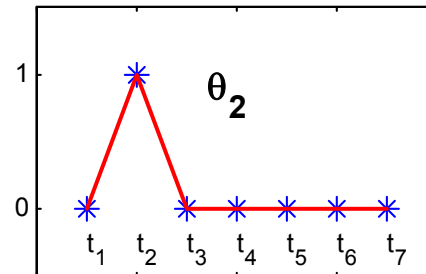
$$\theta_1(t) = \begin{cases} \frac{t - t_2}{t_1 - t_2} & \text{for } 0 \leq t \leq t_2 \\ 0 & \text{otherwise} \end{cases}$$



The second basis function  $\theta_2$  and its data set are given by the following:

$0 = t_1$	$t_2$	$t_3$	$t_4$	$t_5$	$t_6$	$L = t_7$
0	1	0	0	0	0	0

$$\theta_2(t) = \begin{cases} \frac{t - t_1}{t_2 - t_1} & t_1 \leq t \leq t_2 \\ \frac{t - t_3}{t_2 - t_3} & t_2 \leq t \leq t_3 \\ 0 & t_3 \leq t \end{cases}$$



In a similar manner all of the remaining basis functions are defined  $\theta_3$  through  $\theta_7$ . The piecewise linear interpolant of the original data set may now be written in terms of these basis functions.

$$PL_{\text{Data}}(t) = \sum_{i=1}^7 y_i \theta_i(t)$$

Since the computation of the Fourier series coefficients is a linear operation, it is enough to compute the coefficients for the basis functions  $\theta_i$  and multiply by  $y_i$  and sum over all basis functions. Define the Fourier series of a data set on the interval  $[0, L]$  to be the Fourier series of the piecewise linear interpolant.

$$\mathfrak{Z}(\text{Data}) \stackrel{\text{def}}{=} \mathfrak{Z}(\text{PL}_{\text{Data}}(t)) = \sum_{i=1}^7 y_i \mathfrak{Z}(\theta_i(t))$$

Using the MATLAB Symbolic Toolbox, we can easily compute the Fourier coefficients for the basis functions (Using the m-file FSPLCoef.m).

For  $\theta_1$  defined on the nodes (0,c) we obtain:  $c0_{\text{first}} = c/(2*L)$

$$cn_{\text{first}} = -2*(\cos((2*\pi*n/L)*c)-1)/(L*c*(2*\pi*n/L)^2)$$

$$dn_{\text{first}} = 2*(-\sin((2*\pi*n/L)*c)+(2*\pi*n/L)*c)/(L*c*(2*\pi*n/L)^2)$$

The computation of the coefficients for the first basis function is given by the following MATLAB function subprogram.

```
function [c0,cn,dn]=FSPLBasisF(N,L,c)
%Calculates the Fourier Coefficients of the first PL Basis Function
%where N is the number of terms in the series
c0=c/(2*L);
n=1:N;
w=2*pi*n/L; Const=(2/(L*c))./(w.*w);
cn=-Const.*(cos(w*c)-1);
dn=Const.*(-sin(w*c)+w*c);
```

For  $\theta_i$  defined on the nodes (a,b,c) we obtain:  $c0 = (c-a)/(2*L)$

$$cn = 2*(\cos((2*\pi*n/L)*b)*(c-a)+\cos((2*\pi*n/L)*a)*(b-c) +\cos((2*\pi*n/L)*c)*(a-b)) / (L*(-b+a)*(-c+b)*(2*\pi*n/L)^2)$$

$$dn = 2*(\sin((2*\pi*n/L)*b)*(c-a)+\sin((2*\pi*n/L)*a)*(b-c) +\sin((2*\pi*n/L)*c)*(a-b)) / (L*(-b+a)*(-c+b)*(2*\pi*n/L)^2)$$

The computation of the coefficients for the general basis function is given by the following MATLAB function subprogram.

```
function [c0,cn,dn]=FSPLBasis(N,L,a,b,c)
%Calculates the Fourier Coefficients for the PL Basis Functions
%where N is the number of terms in the series
c0=(c-a)/(2*L);
n=1:N;
w=2*pi*n/L; wsq=w.*w; Const=2/(L*(c-b)*(b-a))./wsq;
cn=Const.*(cos(w*b)*(c-a)+cos(w*a)*(b-c)+cos(w*c)*(a-b));
dn=Const.*(sin(w*b)*(c-a)+sin(w*a)*(b-c)+sin(w*c)*(a-b));
```

For  $\theta_{\text{last}}$  defined on the nodes (a,L) we obtain:  $c0_{\text{last}} = (L-a)/(2*L)$

$$cn_{\text{last}} = 1/2*L*(-1+\cos(2*\pi*n/L*a))/((-L+a)*\pi^2*n^2)$$

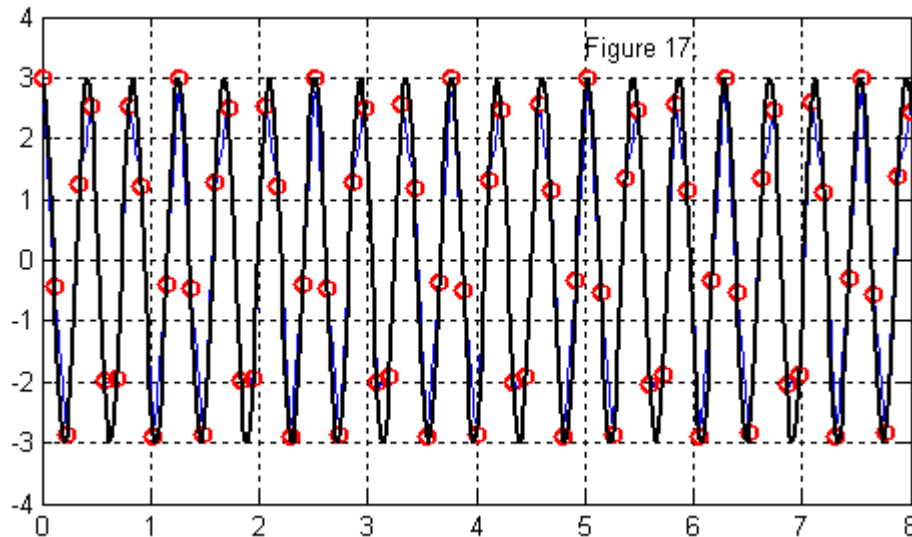
$$dn_{last} = -1/2*(-2*\pi*n*(L-a)-\sin(2*\pi*n/L*a)*L)/((-L+a)*\pi^2*n^2)$$

The computation of the coefficients for the last basis function is given by the following MATLAB function subprogram.

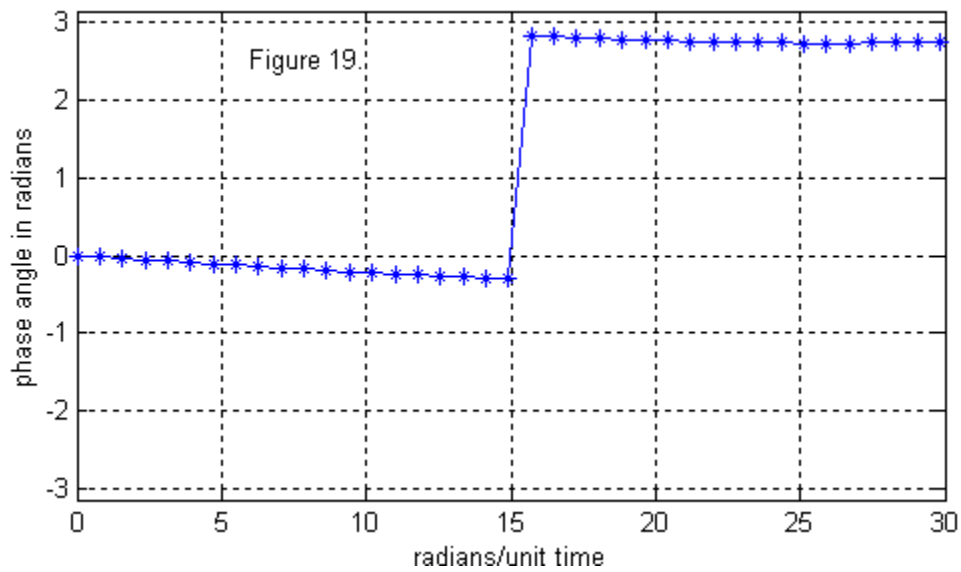
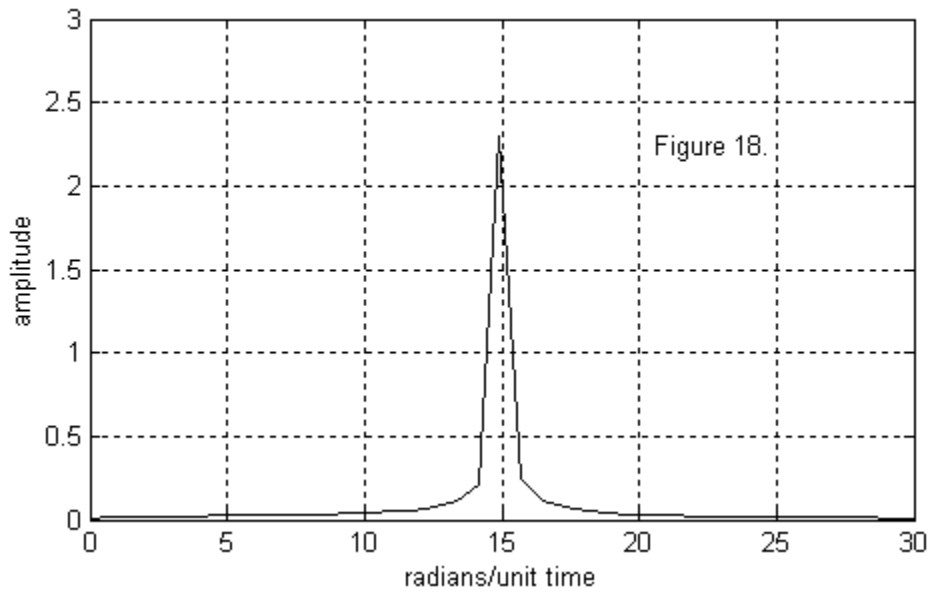
```
function [c0,cn,dn]=FSPLBasisL(N,L,a)
%Calculates the Fourier Coefficients of the last PL Basis Function
%where N is the number of terms in the series
c0=(L-a)/(2*L);
n=1:N;
w=2*pi*n/L; Const=(2/(L*(L-a)))./(w.*w);
cn=Const.*(-cos(w*a)+1);
dn=Const.*(-sin(w*a)-(L-a)*w);
```

The Fourier series of a data set, can now be calculated using the above Fourier series coefficient functions for the piecewise linear interpolation basis functions and our definition. This methodology is exhibited in the next example.

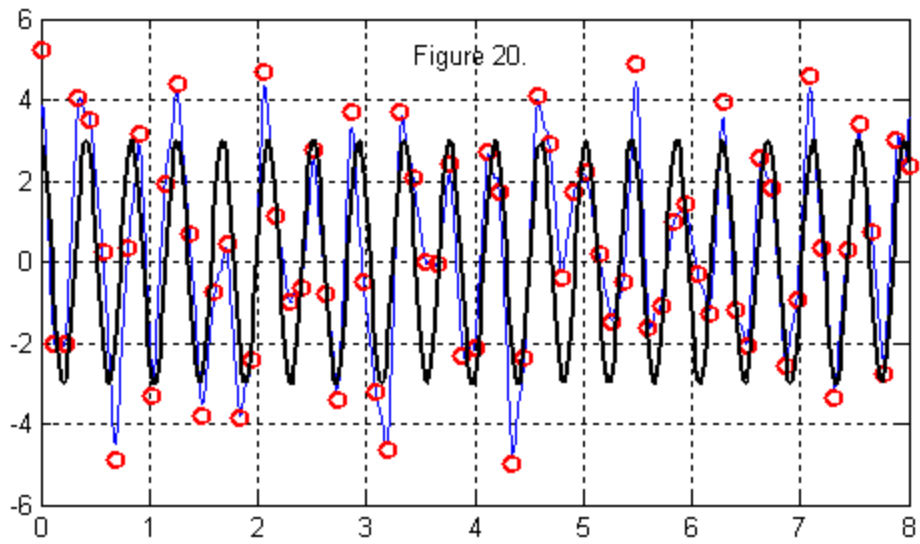
**Example #8:** Continuing with the example given by  $f(t) = 3 \cos(15t)$ , let  $L = 8$  and choose 71 equally spaced points as samples of this function.



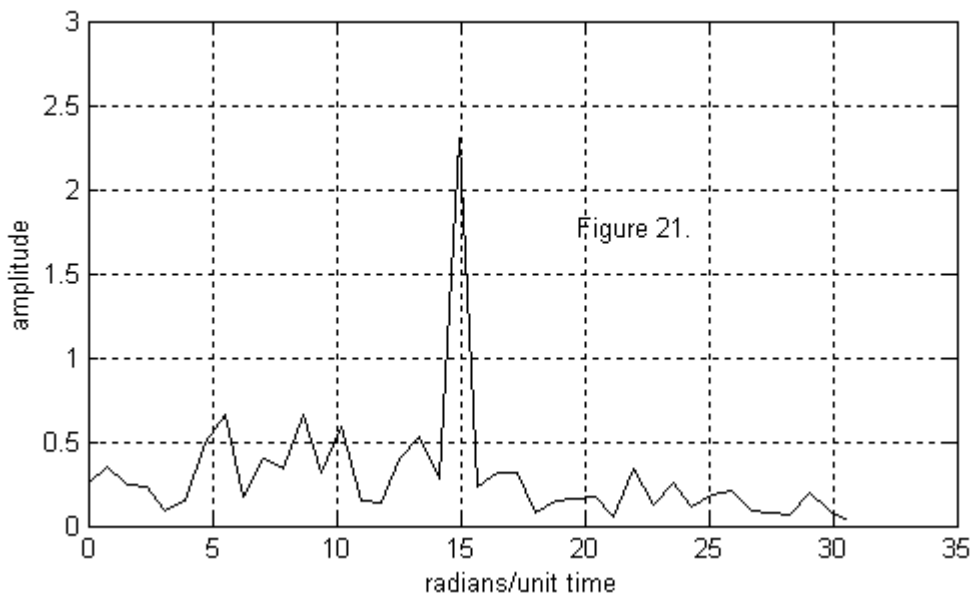
The magnitude and phase of the piecewise linear interpolant of this function are given by:



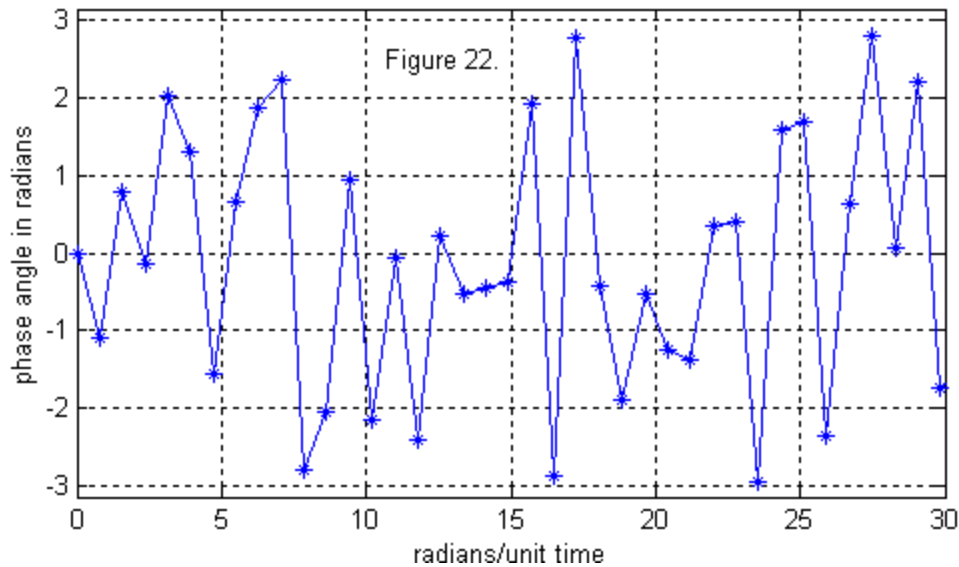
From these graphs, the original function may be reconstructed. Suppose we now add noise to each of the sample points. Assume the noise is taken from a uniform distribution of amplitude 3 about zero. The following graphs give the results. The thick line in Figure 20 is the original function, the thin line is the PL interpolant and the circles are the sampled points.



The results are:



Note that the noise has a very large effect on the phase graph (Figure 22.) except at the main peak of 15 radians/unit time.



Using this technique, Fourier coefficients may be computed for a PL interpolant for any data set having unique independent variable values. Example #8 provides evidence that periodic behavior can be recovered from noisy data. While this process is effective, it is computationally intensive.

### Discrete Fourier Transform:

The Discrete Fourier Transform, or DFT, produces a complex vector  $X$  of length  $N$  from an input vector  $x$  of length  $N$  by the following formula:

$$X(k) = \sum_{n=1}^N x(n) \exp\left(\frac{-2\pi i}{N}(k-1)(n-1)\right) \text{ for } 1 \leq k \leq N$$

The computation of the DFT is usually accomplished using the Fast Fourier Transform Algorithm or FFT. In MATLAB, the DFT is computed using the built-in function `fft`

The DFT operation has an inverse, or IDFT, given by the following formula:

$$x(n) = \frac{1}{N} \sum_{k=1}^N X(k) \exp\left(\frac{2\pi i}{N}(k-1)(n-1)\right) \text{ for } 1 \leq n \leq N$$

**Example #9:** Let the input vector  $x = [1, -2, 3, 6, 0, -1]$ . Thus,  $N = 6$ . The following MATLAB program calculates the DFT and IDFT.

```
%Example #9
clear
x=[1 -2 3 6 0 -1];
X=fft(x);
```

```
xx=ifft(X);
R=[(1:length(x))',x',X',xx']
```

The results are:

k	vector x	X(k) = DFT(x)	IDFT(DFT(x))
1.0000	1.0000	7.0000	1.0000
2.0000	-2.0000	-8.0000 + 1.7321i	-2.0000
3.0000	3.0000	7.0000 - 3.4641i	3.0000
4.0000	6.0000	1.0000	6.0000
5.0000	0	7.0000 + 3.4641i	0
6.0000	-1.0000	-8.0000 - 1.7321i	-1.0000

For this particular vector x, note that IDFT(DFT) is just the identity and that

$$X(k) = \text{Conjugate}(X(N - k + 2)) \text{ for } 2 \leq k \leq N$$

Both of these observations can be proved for an arbitrary input vector.

### Relationship of the DFT to the Interpolation of a Data Set:

Let  $x = [x_1, x_2, \dots, x_N]$  be a given vector of real values. Using these values form the following equally spaced data set on the interval  $[0, L]$ :

DATA SET

$t_1 = 0$	$t_2 = \Delta t$	.....	$t_N = (N - 1) \Delta t$	$t_{N+1} = N \Delta t = L$
$x_1$	$x_2$	.....	$x_N$	$x_1$

**Interpolation Theorem:** The above data set is interpolated by the following trigonometric polynomial.

$$x(t) = a_0 + \sum_{k=1}^{k \leq (N+1)/2} \left[ a_k \cos\left(\frac{2\pi k}{L} t\right) + b_k \sin\left(\frac{2\pi k}{L} t\right) \right] \text{ where } a_0 = X(1)/N,$$

$$a_k = \frac{2 \text{Real}(X(k+1))}{N}, \quad b_k = \frac{-2 \text{Im}(X(k+1))}{N}, \text{ if } N \text{ is even } a_{N/2} = \frac{X(N/2+1)}{N} \text{ and } X = \text{DFT}(x).$$

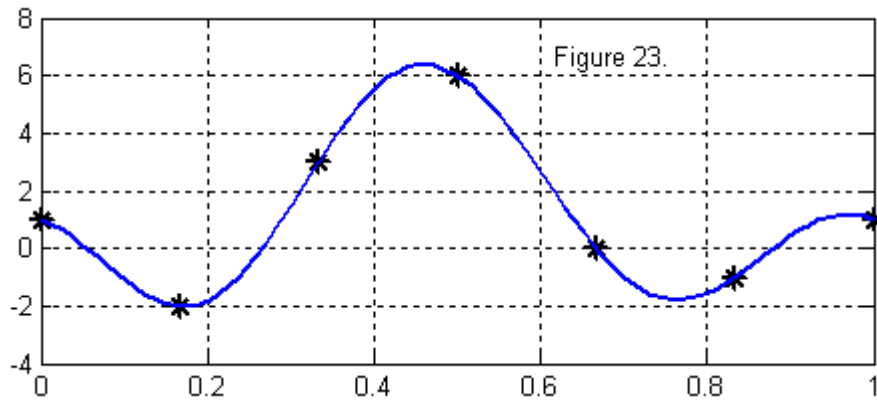
(Note that if N is even  $X(N/2+1)$  is real, see Example #9)

The proof of this theorem begins with the definition of the IDFT, using the properties observed in Example #9, the theorem follows by a careful manipulation and recasting of the terms. Note that we may choose L arbitrarily.

**Example #10:** Consider the vector of values given in Example #9. Let  $L = 1$ , then the Data Set is given by:

0	1/6	1/3	1/2	2/3	5/6	1
1	-2	3	6	0	-1	1

The following MATLAB code graphs both this data set and the Trigonometric Polynomial defined in the Theorem.



The coefficients are:

$c_0 = 1.1667$     $c_n = -2.6667$     $2.3333$     $0.1667$     $d_n = 0.5774$     $-1.1547$     $0$

Note that the Trigonometric Polynomial defined by these coefficients interpolates the Data Set.

If a data set has evenly spaced points, then this theorem provides an alternative to using PL approximations in an investigation of periodic behavior. Since we may use the FFT algorithm to compute the DFT, this method is much more computationally efficient. However, the tradeoff is the requirement that the data points be evenly spaced.

# Appendix 1

This appendix contains the MATLAB code used to generate the graphs and calculations. This code is available on the course web site.

## Index

Example #3	Page 20
Example #4	Page 21
Example #5	Page 22
Example #6	Page 24
Example #7	Page 25
Example #8	Page 26
Example #9	Page 17
Example #10	Page 27
FSPLCoef	Page 27
FSPlot	Page 28

```
%calculation of Fourier Coef. for Example #3
clear
syms L x n
%Compute c0
Int1=(1/L)*3*cos(15*x);
c0=int(Int1,x,0,L)
%Compute cn
Int2=(2/L)*3*cos(15*x)*cos(2*pi*n/L*x);
cn=int(Int2,x,0,L);
cn=simplify(cn);
%Compute dn
Int3=(2/L)*3*cos(15*x)*sin(2*pi*n/L*x);
dn=int(Int3,x,0,L);
dn=simplify(dn);
%Further Simplify using sin(pi*n)=0 and cos(pi*n)=((-1)^n)
C=char(cn);
C=strrep(C,'sin(pi*n)','0');
C=strrep(C,'cos(pi*n)','((-1)^n)');
cn=sym(C);
cn=simplify(cn);
C=char(cn);
C=strrep(C,'(-1)^(2*n)','1');
C=strrep(C,'(-1)^(1+2*n)','(-1)');
cn=sym(C);
cn=simplify(cn)
D=char(dn);
D=strrep(D,'sin(pi*n)','0');
D=strrep(D,'cos(pi*n)','((-1)^n)');
dn=sym(D);
dn=simplify(dn);
D=char(dn);
D=strrep(D,'(-1)^(2*n)','1');
D=strrep(D,'(-1)^(1+2*n)','(-1)');
dn=sym(D);
```

```

dn=simplify(dn)
%Create inline functions for the coefficients
cnf=inline(char(vectorize(cn)), 'n', 'L');
dnf=inline(char(vectorize(dn)), 'n', 'L');
%Compute the first 20 coefficients and set L=1
c0=subs(c0,L,1)
cn=subs(cn,L,1)
dn=subs(dn,L,1)
M=20,L=1
n=1:M;
an(n)=cnf(n,L);
bn(n)=dnf(n,L);
%Plot the series and the function on [0, 1]
figure(1),clf reset
FSPlot(L,c0,an,bn,0,1)
hold on
x=linspace(0,1);
plot(x,3*cos(15*x),'r')
%Plot the series and the function on [0, 3]
figure(2),clf reset
FSPlot(L,c0,an,bn,0,3)
hold on
x=linspace(0,3,300);
plot(x,3*cos(15*x),'r')
%Plot the coefficients
figure(3), clf reset
axes('pos',[.1 .35 .8 .6])
plot(n,an,'r*',n,bn,'b+')
axis([0 M -2.1 2.1])
legend('Cosine Coefs.','Sine Coefs.')
xlabel('frequency index'), ylabel('amplitude')
p=get(gca,'position')
axes('pos',[p(1) .15 p(3) .01],'xlim',[0 M]*2*pi/L)
xlabel('radians/unit time')
%Plot the square root of the sum of the squares of cn and dn
figure(4), clf reset
axes('pos',[.1 .35 .8 .6])
plot(n,sqrt(an.^2+bn.^2),'b*')
xlabel('frequency index') , ylabel('amplitude')
p=get(gca,'position')
axes('pos',[p(1) .15 p(3) .01],'xlim',[0 M]*2*pi/L)
xlabel('radians/unit time')

%calculation of Fourier Coef. for Example #4
clear
syms L x n a
%Compute c0
Int1=(1/L);
c0=int(Int1,x,a,L)
c0=subs(c0,L,1);
c0=subs(c0,a,2/3)
%Compute cn
Int2=(2/L)*cos(2*pi*n/L*x);
cn=int(Int2,x,a,L)
cn=simplify(cn);
%Compute dn
Int3=(2/L)*sin(2*pi*n/L*x);

```

```

dn=int(Int3,x,a,L)
dn=simplify(dn);
%Further Simplify using sin(pi*n)=0 and cos(pi*n)=((-1)^n)
C=char(cn);
C=strrep(C,'sin(pi*n)','0');
C=strrep(C,'cos(pi*n)','((-1)^n)');
cn=sym(C);
cn=simplify(cn);
C=char(cn);
C=strrep(C,'(-1)^(2*n)','1');
C=strrep(C,'(-1)^(1+2*n)','(-1)');
cn=sym(C);
cn=simplify(cn)
D=char(dn);
D=strrep(D,'sin(pi*n)','0');
D=strrep(D,'cos(pi*n)','((-1)^n)');
dn=sym(D);
dn=simplify(dn);
D=char(dn);
D=strrep(D,'(-1)^(2*n)','1');
D=strrep(D,'(-1)^(1+2*n)','(-1)');
dn=sym(D);
dn=simplify(dn)
%Substitute in the values for L and a
cn=subs(cn,L,1);
cn=subs(cn,a,2/3)
dn=subs(dn,L,1);
dn=subs(dn,a,2/3)
%Create inline functions for the coefficients
cnf=inline(char(vectorize(cn)))
dnf=inline(char(vectorize(dn)))
%Compute the first 20 coefficients
M=20, a=2/3
n=1:M;
an(n)=cnf(n);
bn(n)=dnf(n);
%Plot the series and the function on [0, 1]
figure(1),clf reset
FSPlot(1,c0,an,bn,0,1)
hold on
x=linspace(0,1);
y=zeros(1,length(x)); index=find(x>=a);y(index)=1;
plot(x,y,'r')
%Plot the series and the function on [0, 3]
figure(2),clf reset
FSPlot(1,c0,an,bn,0,3)
%Plot the coefficients
figure(3), clf reset
plot(n,an,'r*',n,bn,'b+')
legend('Cosine Coefs.','Sine Coefs.')
%Plot the square root of the sum of the squares of cn and dn
figure(4), clf reset
plot(n,sqrt(an.^2+bn.^2),'b*')

%calculation of Fourier Coef. for Example #5
clear
syms L x n a b

```

```

%Compute c0
Int1=(1/L*(a+b*x));
c0=int(Int1,x,0,L)
c0=subs(c0,L,1);
c0=subs(c0,a,2);
c0=subs(c0,b,1/4)
%Compute cn
Int2=(2/L)*(a+b*x)*cos(2*pi*n/L*x);
cn=int(Int2,x,0,L)
cn=simplify(cn);
%Compute dn
Int3=(2/L)*(a+b*x)*sin(2*pi*n/L*x);
dn=int(Int3,x,0,L)
dn=simplify(dn);
%Further Simplify using sin(pi*n)=0 and cos(pi*n)=((-1)^n)
C=char(cn);
C=strrep(C,'sin(pi*n)','0');
C=strrep(C,'cos(pi*n)','((-1)^n)');
cn=sym(C);
cn=simplify(cn);
C=char(cn);
C=strrep(C,'(-1)^(2*n)','1');
C=strrep(C,'(-1)^(1+2*n)','(-1)');
cn=sym(C);
cn=simplify(cn)
D=char(dn);
D=strrep(D,'sin(pi*n)','0');
D=strrep(D,'cos(pi*n)','((-1)^n)');
dn=sym(D);
dn=simplify(dn);
D=char(dn);
D=strrep(D,'(-1)^(2*n)','1');
D=strrep(D,'(-1)^(1+2*n)','(-1)');
dn=sym(D);
dn=simplify(dn)
%Substitute in the values for L=1, a=2, b=0.25
cn=subs(cn,L,1);
cn=subs(cn,a,2);
cn=subs(cn,b,1/4)
dn=subs(dn,L,1);
dn=subs(dn,a,2);
dn=subs(dn,b,1/4)
%Create inline functions for the coefficients
dnf=inline(char(vectorize(dn)))
%Compute the first 20 coefficients
M=20, A=2, B=1/4, L=1
n=1:M;
an(n)=zeros(1,M);
bn(n)=dnf(n);
%Plot the series and the function on [0, 1]
figure(1),clf reset
FSPlot(1,c0,an,bn,0,1)
hold on
x=linspace(0,1);
plot(x,A+B*x,'r')
%Plot the series and the function on [0, 3]
figure(2),clf reset

```

```

FSPlot(1,c0,an,bn,0,3)
%Plot the coefficients
figure(3), clf reset
axes('pos',[.1 .35 .8 .6])
plot(n,an,'r*',n,bn,'b+')
axis([0 M -0.1 0.1])
legend('Cosine Coefs.','Sine Coefs.')
xlabel('frequency index') , ylabel('amplitude')
p=get(gca,'position')
axes('pos',[p(1) .15 p(3) .01],'xlim',[0 M]*2*pi/L)
xlabel('radians/unit time')
%Plot the square root of the sum of the squares of cn and dn
figure(4), clf reset
axes('pos',[.1 .35 .8 .6])
plot(n,sqrt(an.^2+bn.^2),'b*')
xlabel('frequency index') , ylabel('amplitude')
p=get(gca,'position')
axes('pos',[p(1) .15 p(3) .01],'xlim',[0 M]*2*pi/L)
xlabel('radians/unit time')

%calculation of Fourier Coef. for Example #6
clear
syms L x n
%Compute c0
Int1=(1/L)*(3*cos(15*x)*16*x^2*(x-L)^2);
c0=int(Int1,x,0,L);
c0=subs(c0,L,1)
%Compute cn
Int2=(2/L)*(3*cos(15*x)*16*x^2*(x-L)^2)*cos(2*pi*n/L*x);
cn=int(Int2,x,0,L);
cn=subs(cn,L,1);
cn=simplify(cn);
%Compute dn
Int3=(2/L)*(3*cos(15*x)*16*x^2*(x-L)^2)*sin(2*pi*n/L*x);
dn=int(Int3,x,0,L);
dn=subs(dn,L,1);
dn=simplify(dn);
%Further Simplify using sin(pi*n)=0 and cos(pi*n)=((-1)^n)
C=char(cn);
C=strrep(C,'sin(pi*n)','0');
C=strrep(C,'cos(pi*n)','((-1)^n)');
cn=sym(C);
cn=simplify(cn);
C=char(cn);
C=strrep(C,'(-1)^(2*n)','1');
C=strrep(C,'(-1)^(1+2*n)','(-1)');
cn=sym(C);
cn=simplify(cn)
D=char(dn);
D=strrep(D,'sin(pi*n)','0');
D=strrep(D,'cos(pi*n)','((-1)^n)');
dn=sym(D);
dn=simplify(dn);
D=char(dn);
D=strrep(D,'(-1)^(2*n)','1');
D=strrep(D,'(-1)^(1+2*n)','(-1)');
dn=sym(D);

```

```

dn=simplify(dn)
%Create inline functions for the coefficients
cnf=inline(char(vectorize(cn)))
dnf=inline(char(vectorize(dn)))
%Compute the first 20 coefficients
M=20,L=1
n=1:M;
an(n)=cnf(n);
bn(n)=dnf(n);
%Plot the series and the function on [0, 1]
figure(1),clf reset
FSPlot(1,c0,an,bn,0,1)
hold on
x=linspace(0,1);
plot(x,(3*cos(15*x))*16.*x.^2.*(x-L).^2),'r')
plot(x,(3*cos(15*x)),'k')
%Plot the coefficients
figure(3), clf reset
axes('pos',[.1 .35 .8 .6])
plot(n,an,'r*',n,bn,'b+')
axis([0 M -2.1 2.1])
legend('Cosine Coefs.','Sine Coefs.')
xlabel('frequency index') , ylabel('amplitude')
p=get(gca,'position')
axes('pos',[p(1) .15 p(3) .01],'xlim',[0 M]*2*pi/L)
xlabel('radians/unit time')
%Plot the square root of the sum of the squares of cn and dn
figure(4), clf reset
axes('pos',[.1 .35 .8 .6])
plot(n,sqrt(an.^2+bn.^2),'b*')
xlabel('frequency index') , ylabel('amplitude')
p=get(gca,'position')
axes('pos',[p(1) .15 p(3) .01],'xlim',[0 M]*2*pi/L)
xlabel('radians/unit time')

%Computations for Example #7
%Numerical Computation of the first M Fourier Coeffs
%of the function f on the interval [0, L]
clear,
figure(1), clf reset,
ind=0; col=['k','b','r','g'];sy='*';
for L=[1,2,4,8]
M=ceil(30*L/(2*pi)); ind=ind+1;
f='3*cos(15*t)'
IntC0=inline(f,'t')
IC=[f,'.*cos(2*pi*n*t/L)']
IntC=inline(IC,'t','n','L')
ID=[f,'.*sin(2*pi*n*t/L)']
IntD=inline(ID,'t','n','L')
for i=1:M
    c(i)=2/L*quadl(IntC,0,L,1.e-6,[],i,L);
    d(i)=2/L*quadl(IntD,0,L,1.e-6,[],i,L);
end
c0=1/L*quad(IntC0,0,L);
%Plot the square root of the sum of the squares of cn and dn
m=0:M; rr=2*pi/L*m;
pp=[abs(c0),sqrt(c.^2+d.^2)];

```

```

str=[sy,col(ind)];
plot(rr,pp,str,rr,pp,col(ind))
hold on
axis([0,30,0,3])
end
xlabel('radians/unit time')
ylabel('amplitude')
text(20,2.6,'Figure 14.')
figure(2), clf reset
ph=[0,atan2(d,c)];
plot(rr,ph,'*b',rr,ph,'b')
xlabel('radians/unit time')
ylabel('phase angle in radians')
axis([0,30,-pi,pi])
grid on
text(6,2.5,'Figure 15.')

%Computation for Example #8
%Computes an approximate Fourier Series for the
%PL approximation to the data on the interval [0,L]
clear
N=100;L=8; %N=number of terms in the series
f=inline('3*cos(15*x)'), %Test function
n=71; x=linspace(0,L,n);%n=number of samples
y=(f(x)+6*(rand(1,n)-.5));
[c0,cn,dn]=FSPLBasisF(N,L,x(2));
c0=y(1)*c0;cn=y(1)*cn;dn=y(1)*dn;
for i=2:n-1
    [cc0,ccn,dcn]=FSPLBasis(N,L,x(i-1),x(i),x(i+1));
    c0=y(i)*cc0+c0;cn=cn+y(i)*ccn;dn=dn+y(i)*dcn;
end
[cc0,ccn,dcn]=FSPLBasisL(N,L,x(n-1));
c0=c0+y(n)*cc0;cn=cn+y(n)*ccn;dn=dn+y(n)*dcn;
figure(1), clf reset
FSPlot(L,c0,cn,dn,0,L)
grid on
hold on
plot(x,y,'ro','markersize',7,'linewidth',2)
xx=linspace(0,L,500);
plot(xx,f(xx),'k','linewidth',2)
axis([0,L,-6,6])
text(5,3.5,'Figure 20.')
c0
M=ceil(30*L/(2*pi));
nn=0:M;
figure(2), clf reset
mag=abs(cn+dn*sqrt(-1));
plot(2*pi/L*nn,[abs(c0),mag(1:M)],'k')
grid on
axis([0,35,0,3])
xlabel('radians/unit time')
ylabel('amplitude')
text(20,1.5,'Figure 21.')
figure(3), clf reset
ph=[0,atan2(dn,cn)];
rr=2*pi/L*nn;

```

```

plot(rr,ph(1:M+1),'*b',rr,ph(1:M+1),'b')
xlabel('radians/unit time')
ylabel('phase angle in radians')
axis([0,30,-pi,pi])
grid on
text(6,2.5,'Figure 22.')
z=fft(y);
figure(4), clf reset
plot(2*pi/L*(0:n-1),[abs(z(1))/n,2*abs(z(2:n))/n],'k')
grid on
axis([0,35,0,3])

%Example #10
clear, clf reset
x=[1 -2 3 6 0, -1]; L=1;
N=length(x);
Data=[(0:N)*L/N;x, x(1)]
plot(Data(1,:),Data(2:,:), 'k*', 'markersize',10, 'linewidth',2)
hold on
X=fft(x), M=floor((N+1)/2)
c0=X(1)/N, cn=2*real(X(2:M))/N, dn=-2*imag(X(2:M))/N
if M==N/2
    cn(M)=X(M+1)/N, dn(M)=0
end
FSplot(L,c0,cn,dn,0,L)
grid on

%calculation of Fourier Coef. for PL Basis Functions
syms a b c L x n w
%Calculation of cn and dn for a basis function Theta defined on a,b,c
Int1=(2/L)*(x-a)/(b-a)*cos(w*x);
Int2=(2/L)*(c-x)/(c-b)*cos(w*x);
cn=int(Int1,x,a,b)+int(Int2,x,b,c);
cn=factor(cn);
cn=subs(cn,'w','2*pi*n/L')
Int3=(2/L)*(x-a)/(b-a)*sin(w*x);
Int4=(2/L)*(c-x)/(c-b)*sin(w*x);
dn=int(Int3,x,a,b)+int(Int4,x,b,c);
dn=factor(dn);
dn=subs(dn,'w','2*pi*n/L')
%Calculation of the special case of the first basis function
cnfirst=int(Int2,x,0,c);
%cnfirst=simplify(cnfirst);
cnfirst=subs(cnfirst,'b',0);
cnfirst=subs(cnfirst,'w','2*pi*n/L')
%Calculation of the special case of the last basis function
dnfirst=int(Int4,x,0,c);
%dnfirst=simplify(dnfirst);
dnfirst=subs(dnfirst,'b',0);
dnfirst=subs(dnfirst,'w','2*pi*n/L')
cnlast=int(Int1,x,a,L);
%cnlast=simplify(cnlast);
cnlast=subs(cnlast,'w','2*pi*n/L');
cnlast=subs(cnlast,'b',sym('L'));
cnlast=simplify(cnlast);
dnlast=int(Int3,x,a,L);
%dnlast=simplify(dnlast);

```

```

dnlast=subs(dnlast,'w','2*pi*n/L');
dnlast=subs(dnlast,'b',sym('L'));
dnlast=simplify(dnlast);
%Further Simplify using sin(pi*n)=0 and cos(pi*n)=((-1)^n)
C=char(cnlast);
C=strrep(C,'sin(2*pi*n)','0');
C=strrep(C,'cos(2*pi*n)','1');
cnlast=sym(C);
cnlast=simplify(cnlast)
D=char(dnlast);
D=strrep(D,'sin(2*pi*n)','0');
D=strrep(D,'cos(2*pi*n)','1');
dnlast=sym(D);
dnlast=simplify(dnlast)

```

```

function FSPlot(L,c0,cn,dn,a,b)
%Plots the truncated Fourier Series having coefficients
%c0, cn and dn defined on the interval [0, L] and
%plotted on the interval [a, b]
%c0, cn, dn must be row vectors containing the coefficients
M=length(cn);
x=linspace(a,b,15*M);
n=1:M;
y=c0+cn*cos(2*pi*n'*x/L)+dn*sin(2*pi*n'*x/L);
plot(x,y)

```