CHAPTER 2. MATLAB FUNDAMENTALS

Exercise 2.1

Show that the Taylor Series expansion of $\cos(x)$ about $x = 0$ is:

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots$$

Solution:

Taylor series expansion of $f(x)$ about $x = 0$ is:

$$f(x) = f(0) + f'(0)x + \frac{f''(0)}{2!}x^2 + \frac{f'''(0)}{3!}x^3 + \frac{f^{IV}}{4!}x^4 + \cdots$$

For $\cos(x)$,

$f(x) = \cos(x),\ f(0) = 1,$

$f'(x) = -\sin(x),\ f'(0) = 0,$

$f''(x) = -\cos(x),\ f''(0) = -1,$

$f'''(x) = +\sin(x),\ f'''(0) = 0,$

$f^{IV}(x) = +\cos(x),\ f^{IV}(0) = 1$

We can see that

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \cdots + \cdots$$

Exercise 2.2

Show that the Taylor Series expansion of $\sin(x)$ about $x = 0$ is:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

Solution:

For $\sin(x)$,

$$f(x)=\sin(x),\ f(0)=0$$

$$f'(x)=\cos(x),\ f'(0)=1$$

$$f''(x)=-\sin(x),\ f''(0)=0$$

$$f'''(x)=-\cos(x),\ f'''(0)=-1$$

$$f^{IV}(x)=\sin(x),\ f^{IV}(0)=0$$

$$f^{V}(x)=\cos(x),\ f^{V}(0)=1$$

We can see that

$$\sin(x)=x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+\frac{x^9}{9!}-\cdots+\cdots$$

Exercise 2.3

Using the Taylor Series expansion of e^x about $x=0$ (see Example 2.1), show that

$e^{jx}=\cos x+j\sin x$ and that $e^{-jx}=\cos x-j\sin x$, where $j=\sqrt{-1}.$

Solution:

For e^{jx},

$$f(x)=e^{jx},\ f(0)=1$$

$$f'(x)=je^{jx},\ f'(0)=j$$

$$f''(x)=-e^{jx},\ f''(0)=-1$$

$$f'''(x)=-je^{jx},\ f'''(0)=-j$$

$$f^{IV}(x)=e^{jx},\ f^{IV}(0)=1$$

$$f^{V}(x) = je^{jx},\ f^{V}(0) = j$$

$$f^{VI}(x) = -e^{jx},\ f^{VI}(0) = -1$$

We can see that

$$e^{jx} = \left(1 - \frac{1}{2!}x^2 + \frac{1}{4!}x^4 - \cdots + \cdots - \right) + j\left(x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots + \cdots\right)$$

Comparing the above expression and the series expansions for $\cos x$ and $\sin x$, we see that

$e^{jx} = \cos x + j\sin x$.

Similarly, for e^{-jx}:

$$f(x) = e^{-jx},\ f(0) = 1$$

$$f'(x) = -je^{-jx},\ f'(0) = -j$$

$$f''(x) = -e^{-jx},\ f''(0) = -1$$

$$f'''(x) = je^{-jx},\ f'''(0) = j$$

$$f^{IV}(x) = e^{-jx},\ f^{IV}(0) = 1$$

$$f^{V}(x) = -je^{-jx},\ f^{V}(0) = -j$$

$$f^{VI}(x) = -e^{-jx},\ f^{V}(0) = -1$$

We can see that

$$e^{-jx} = \left(1 - \frac{1}{2!}x^2 + \frac{1}{4!}x^4 - \cdots + \cdots\right) - j\left(x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots + \cdots\right)$$

Thus, $e^{-jx} = \cos x - j\sin x$.

Project 2.1

Using the Taylor Series expansion of $\cos x$ about $x = 0$ (see Exercise 2.1), develop a computer program in MATLAB that will evaluate $\cos x$ from $-\pi \le x \le \pi$ in steps of 0.1 π. Compute the series by determining each term according to the equation

$$\text{term}(k+1) = \text{term}(k) \times \text{a multiplication factor}$$

where the multiplication factor needs to be determined. Use as many as 50 terms; however, stop adding terms when the last term only affects the 6th decimal place in the answer. Also compute the value of $\cos x$ by using MATLAB's built-in `cos()` function over the same interval and step size. Print a table of your results to a file using the table format below. Use 6 decimal places for $\cos x$.

Table P2.1. Table format for Project 2.1.

x	$\cos x$ by `cos()`	$\cos x$ by series	Terms in the series
-1.0 π			
-0.9 π			
-0.8 π			
⋮			
0.9 π			
1.0 π			

Also create a plot of $\cos x$ for $-\pi \le x \le \pi$.

Solution:

Taylor series expansion of $f(x)$ about $x = 0$ is:

$$f(x) = f(0) + f'(0)\,x + \frac{f''(0)}{2!}x^2 + \frac{f'''(0)}{3!}x^3 + \frac{f^{IV}}{4!}x^4 + \ldots$$

For $\cos x$,

$$f(x) = \cos(x),\ f(0) = 1,$$

$$f'(x) = -\sin(x),\ f'(0) = 0,$$

$$f''(x) = -\cos(x),\ f''(0) = -1,$$

$$f'''(x) = +\sin(x),\ f'''(0) = 0,$$

$$f^{IV}(x) = +\cos(x),\ f^{IV}(0) = 1$$

We can see that

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \cdots + \cdots$$

For $k = 1, 2\ 3, \ldots$

$$\text{term}(k+1) = -\ \text{term}(k) \times \frac{x^2}{2k(2k-1)}$$

The following program evaluates $\cos(x)$ by both an arithmetic statement and by the above series for $-\pi \le x \le \pi$ in steps of $0.1\ \pi$.

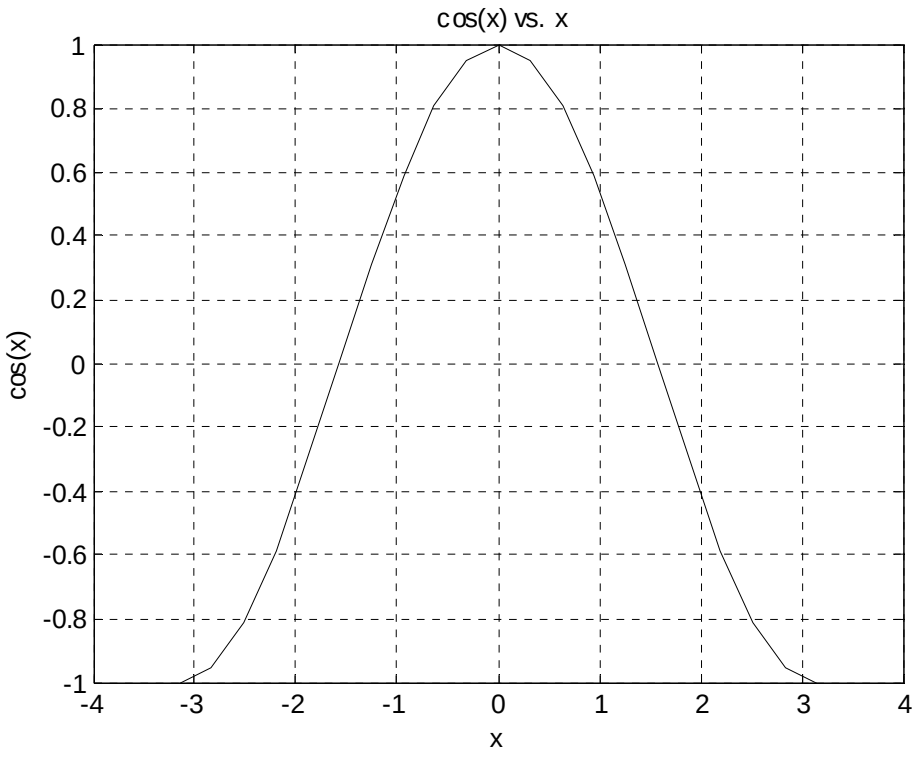
```
% Project_2_1.m
% This program evaluates cos(x) by both arithmetic statement and by
% series for -pi <= x <= pi  in steps of 0.1 pi
clear; clc;
xi=-pi; dx=0.1*pi;
for j=1:21
    x(j)=xi+(j-1)*dx;
    cos_arith(j)= cos(x(j));
    sum=1.0; term=1.0;
```

```
    for k=1:50
        den=2*k*(2*k-1);
        term=-term*x(j)^2/den;
        sum=sum+term;
        test=abs(sum*1.0e-6);
        if abs(term) <= test;
            break;
        end
    end
    cos_ser(j)=sum;
    nterms(j)=k;
end
fo=fopen('output.dat','w');
fprintf(fo,'x       cos(x)             cos (x)     terms in  \n');
fprintf(fo,'      by arith stm        by series   the series  \n');
fprintf(fo,'=====================================================\n');
for j=1:21
    fprintf(fo,'%10.5f   %10.5f    %10.5f       %3i \n',...
                x(j),cos_arith(j),cos_ser(j),nterms(j));
    fprintf(fo,'-----------------------------------------------\n');
end
fclose(fo);
plot(x,cos_arith),xlabel('x'),ylabel('cos(x)'),
title('cos(x) vs. x'),grid;
```

Program output:

```
    x          cos(x)         cos (x)     terms in
             by arith stm     by series   the series
=====================================================
```

```
  -3.14159     -1.00000      -1.00000         9
-------------------------------------------------
  -2.82743     -0.95106      -0.95106         8
-------------------------------------------------
  -2.51327     -0.80902      -0.80902         8
-------------------------------------------------
  -2.19911     -0.58779      -0.58779         8

      ⋮

   1.57080      0.00000       0.00000        14
-------------------------------------------------
   1.88496     -0.30902      -0.30902         7
-------------------------------------------------
   2.19911     -0.58779      -0.58779         8
-------------------------------------------------
   2.51327     -0.80902      -0.80902         8
-------------------------------------------------
   2.82743     -0.95106      -0.95106         8
-------------------------------------------------
   3.14159     -1.00000      -1.00000         9
-------------------------------------------------
```



Project 2.2

Using the Taylor Series expansion of $\sin x$ about $x = 0$ (see Exercise 2.2), develop a computer program in MATLAB that will evaluate the function from $-\pi \le x \le \pi$ in steps of 0.1 π. Compute the series by using MATLAB's `factorial` function and using as many as 50 terms. However, stop adding terms when the last term only affects the 6th decimal place in the answer. Also compute the value of $\sin x$ by using MATLAB's built-in `sin()` function over the same interval and step size. Create a table similar to the table shown in Project 2.1, except replace $\cos x$ with $\sin x$. Also create a plot of $\sin x$ for $-\pi \le x \le \pi$.

Solution:

Taylor series expansion of $f(x)$ about $x = 0$ is:

$$f(x) = f(0) + f'(0)\,x + \frac{f''(0)}{2!}x^2 + \frac{f'''(0)}{3!}x^3 + \frac{f^{IV}}{4!}x^4 + \ldots$$

For $\sin x$,

$$f(x) = \sin(x),\ f(0) = 0$$

$$f'(x) = \cos(x),\ f'(0) = 1$$

$$f''(x) = -\sin(x),\ f''(0) = 0$$

$$f'''(x) = -\cos(x),\ f'''(0) = -1$$

$$f^{IV}(x) = \sin(x),\ f^{IV}(0) = 0$$

$$f^{V}(x) = \cos(x),\ f^{V}(0) = 1$$

We can see that

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \cdots + \cdots$$

For $k = 1, 2, 3, \ldots$.

$$\text{term}(k) = (-1)^{(k-1)} \times \frac{x^{(2k-1)}}{(2k-1)!}$$

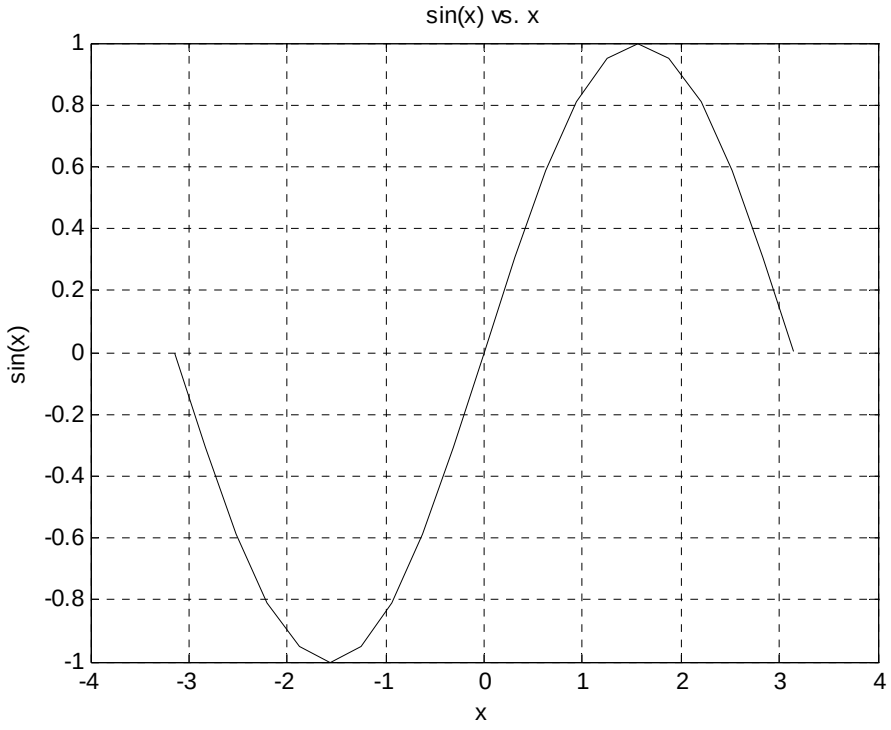
The following program evaluates $\sin(x)$ by both an arithmetic statement and by the above series for $-\pi \le x \le \pi$ in steps of $0.1\ \pi$.

```
% Project_2_2.m
% This program evaluates sin(x) by both arithmetic statment and
% by series.
% The series solution for sin(x) is: x-x^3/3!+x^5/5!-x^7/7! + - ...
% for -pi <  x < pi in steps of 0.1*pi.
clear; clc;
xi=-pi; dx=0.1*pi;
for j=1:21
```

```
    x(j)=xi+(j-1)*dx;
    sin_arith(j)= sin(x(j));
    sum=0;
    for k=1:50
        den=factorial(2*k-1);
        term(k)=(-1)^(k-1)*x(j)^(2*k-1)/den;
        sum=sum+term(k);
        test=abs(sum*1.0e-6);
        if abs(term(k)) <= test;
            break;
        end
    end
    sin_ser(j)=sum;
    nterms(j)= k;
end
fo=fopen('output.txt','w');
fprintf(fo,'x       sin(x)             sin (x)     terms in  \n');
fprintf(fo,'      by arith stm        by series   the series  \n');
fprintf(fo,'=====================================================\n');
for j=1:21
    fprintf(fo,'%10.5f   %10.5f    %10.5f       %3i \n',...
                x(j),sin_arith(j),sin_ser(j),nterms(j));
    fprintf(fo,'-----------------------------------------------\n');
end
fclose(fo);
plot(x,sin_arith);
xlabel('x'),ylabel('sin(x)'),title('sin(x) vs. x'),grid;
```

Program output:

```
    x          sin(x)        sin (x)     terms in
               by arith stm    by series   the series
=================================================
  -3.14159     -0.00000      -0.00000        18
-------------------------------------------------
  -2.82743     -0.30902      -0.30902         9
-------------------------------------------------
  -2.51327     -0.58779      -0.58779         9
-------------------------------------------------
  -2.19911     -0.80902      -0.80902         8
-------------------------------------------------
  -1.88496     -0.95106      -0.95106         7
-------------------------------------------------
  -1.57080     -1.00000      -1.00000         7

      ⋮


   1.88496      0.95106       0.95106         7
-------------------------------------------------
   2.19911      0.80902       0.80902         8
-------------------------------------------------
   2.51327      0.58779       0.58779         9
-------------------------------------------------
   2.82743      0.30902       0.30902         9
-------------------------------------------------
   3.14159      0.00000       0.00000        18
-------------------------------------------------
```



Project 2.3

Develop a computer program in MATLAB that will evaluate the following function for $-0.9 \le x \le 0.9$ in steps of 0.1 by:

a) an arithmetic statement.

b) by series allowing for as many as 50 terms. However, stop adding terms when the last term only affects the 6th decimal place in the answer.

The function and its series expansion, valid for $x^2 < 1$, is:

$$f(x) = (1+x^2)^{-1/2} = 1 - \frac{1}{2}x^2 + \frac{1\cdot 3}{2\cdot 4}x^4 - \frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6}x^6 + \frac{1\cdot 3\cdot 5\cdot 7}{2\cdot 4\cdot 6\cdot 8}x^8 - \cdots + \cdots$$

Print out a table (to a file) in the following format; use 6 decimal places for $f(x)$.

Table P2.4. Table format for Project 2.3.

x	$f(x)$ (arithmetic statement)	$f(x)$ (by series)	# of terms used in the series
-0.9			
-0.8			
-0.7			
⋮			
0.7			
0.8			
0.9			

Solution:

For k=1, 2, 3, ..., we can see that

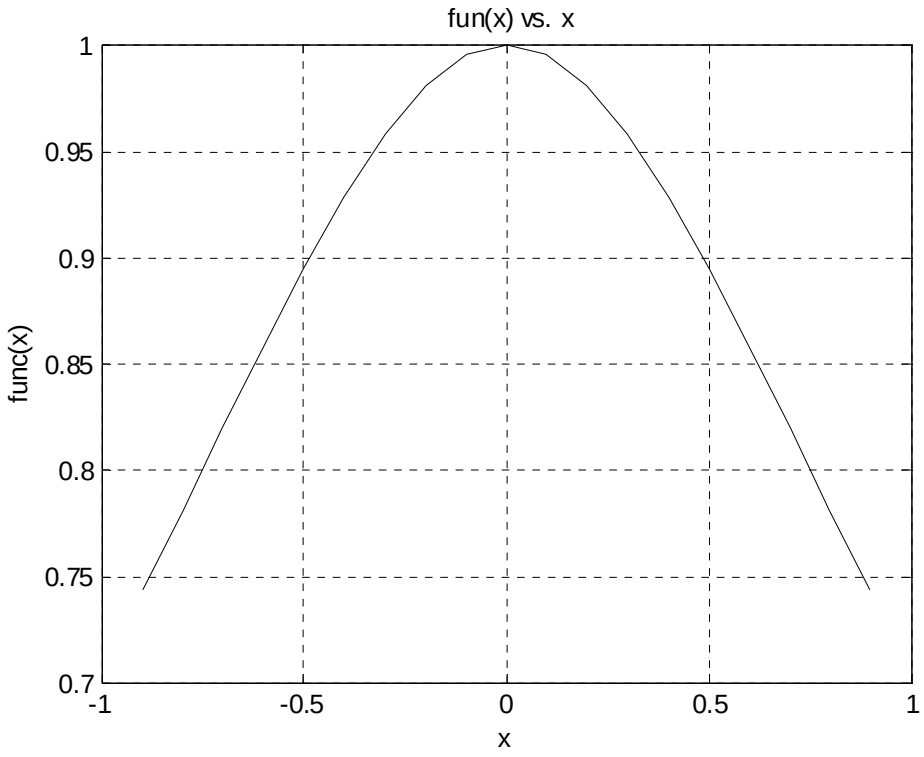
$$\text{term}(k+1) = -\text{term}(k) \times \frac{(2k-1)x^2}{2k}$$

```
% Project_2_3.m
% This program evaluates f(x) by both arithmetic statements and by
% series for -0.9 <= x <= 0.9 in steps of 0.1.
clear; clc;
xi=-0.9; dx=0.1;
for j=1:19
    x(j)=xi+(j-1)*dx;
    fun_arith(j)= (1+x(j)^2)^(-0.5);
    sum=1.0; term=1.0;
    for k=1:50
```

```
        mult=(2*k-1)/(2*k);
        term=-term*mult*x(j)^2;
        sum=sum+term;
        test=abs(sum*1.0e-6);
        if abs(term) <= test;
            break;
        end
    end
    fun_ser(j)=sum;
    nterms(j)=k;
end
fo=fopen('output.dat','w');
fprintf(fo,'x      function(x)       function(x)     terms in  \n');
fprintf(fo,'      by arith stm        by series     the series  \n');
fprintf(fo,'=====================================================\n');
for j=1:19
    fprintf(fo,'%3.1f   %10.5f    %10.5f       %3i \n',...
                x(j),fun_arith(j),fun_ser(j),nterms(j));
    fprintf(fo,'-----------------------------------------------\n');
end
fclose(fo);
plot(x,fun_arith),xlabel('x'),ylabel('func(x)'),title('fun(x) vs. x'),grid;
```

Program output:

```
  x     function(x)    function(x)  terms in
       by arith stm     by series  the series
=================================================
-0.9      0.74329       0.74330        50
-------------------------------------------------
-0.8      0.78087       0.78087        27
-------------------------------------------------
-0.7      0.81923       0.81923        17
-------------------------------------------------
-0.6      0.85749       0.85749        12
-------------------------------------------------
-0.5      0.89443       0.89443         9

  ⋮

0.5      0.89443       0.89443         9
-------------------------------------------------
0.6      0.85749       0.85749        12
-------------------------------------------------
0.7      0.81923       0.81923        17
-------------------------------------------------
0.8      0.78087       0.78087        27
-------------------------------------------------
0.9      0.74329       0.74330        50
-------------------------------------------------
```



Project 2.4

Develop a computer program in MATLAB that will evaluate the function

$$f(x) = (1+x)^{1/3}$$

for $-0.9 \le x \le 0.9$ in steps of 0.1 by:

(a) an arithmetic statement.

(b) by series allowing for as many as 50 terms. However, stop adding terms when the last term only affects the 6th decimal place in the answer.

By using a Taylor series expansion, you can see that the relationship between successive terms is:

$$\text{term}(k+1) \;=\; \text{term}(k) \;\times \left(\frac{1}{3} - (k-1)\right) \times \frac{x}{k}$$

for $k = 1, 2, 3, ...$

Print out a table (to a file) in the format shown in Project 2.3; use 6 significant figures for $f(x)$.

Solution:

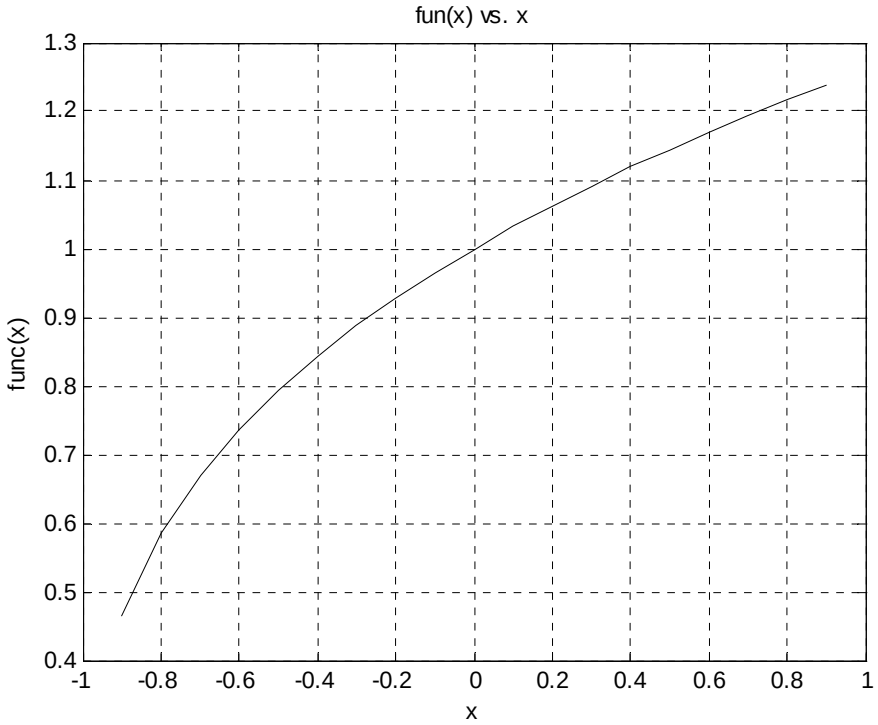
```
% Project_2_4.m
% This program evaluates the following function by both arithmetic
% statement and by series for -0.9 < x < 0.9 in steps of 0.1.
% The function is f(x(=(1+x)^1/3.
clear; clc;
xi=-0.9; dx=0.1;
for j=1:19
    x(j)=xi+(j-1)*dx;
    fun_arith(j)= (1+x(j))^(1.0/3.0);
    sum=1.0; term=1.0;
    for k=1:50
        term=term*(1.0/3.0-(k-1))*x(j)/k;
        sum=sum+term;
        test=abs(sum*1.0e-7);
        if abs(term) <= test;
            break;
        end
    end
    fun_ser(j)=sum;
    nterms(j)=k;
end
fo=fopen('output.txt','w');
fprintf(fo,'x      function(x)       function(x)     terms in  \n');
fprintf(fo,'      by arith stm        by series     the series  \n');
```

```
fprintf(fo,'=================================================\n');
for j=1:19
    fprintf(fo,'%3.1f   %10.5f    %10.5f       %3i \n',...
                x(j),fun_arith(j),fun_ser(j),nterms(j));
    fprintf(fo,'-------------------------------------------------\n');
end
fclose(fo);
plot(x,fun_arith);
xlabel('x'),ylabel('func(x)'),title('fun(x) vs. x'),grid;
```

Program output:

```
x      function(x)    function(x)     terms in
       by arith stm     by series     the series
=================================================
-0.9      0.46416       0.46421          50
-------------------------------------------------
-0.8      0.58480       0.58480          46
-------------------------------------------------
-0.7      0.66943       0.66943          30
-------------------------------------------------
-0.6      0.73681       0.73681          22
-------------------------------------------------
-0.5      0.79370       0.79370          17

    ⋮

0.6      1.16961       1.16961           21
-------------------------------------------------
0.7      1.19348       1.19348           29
```

```
------------------------------------------------
0.8      1.21644       1.21644           43
------------------------------------------------
0.9      1.23856       1.23856           50
------------------------------------------------
```



Project 2.5

The binomial expansion for $(1+x)^n$, where *n* is an integer, is as follows:

$$(1+x)^n = 1 + nx + \frac{n(n-1)x^2}{2!} + \frac{n(n-1)(n-2)x^3}{3!} + \cdots + \frac{n(n-1)(n-2)...(n-r+2)x^{r-1}}{(r-1)!} + \cdots + x^n$$

Construct a MATLAB program that will evaluate $(1+x)^n$ by both the above series and by an arithmetic statement for $n = 10$ and $1.0 \le x \le 10.0$ in steps of 0.5. Print out the results in a table as shown below.

Table P2.5. Table format for Project 2.5

x	$(1+x)^n$ (arithmetic statement)	$(1+x)^n$ (by series)
1.0		
1.5		
2.0		
2.5		
⋮		
9.5		
10.0		

Solution:

```
% Project_2_5.m
% This program evaluates the expression (x+a)^n by the binomial
% expansion for several values of x.
clear; clc;
fprintf(' The binomial series formula for (1+x)^n is: \n');
fprintf('(1+x)^n=1+nx+n(n-1)*x^2/2!+n(n-1)(n-2)*x^3/3!+');
fprintf('    n(n-1)(n-2)...(n-r+2)x^(r-1)/(r-1)!+..x^n \n');
fprintf('The series terminates with x^n,...
       as long as n is an integer \n');
x=1:0.5:10; n=10;
fout=fopen('bionomial.dat','w');
fprintf('  x            y                y \n');
fprintf('          by expansion       by arith stm \n');
fprintf('-------------------------------------------------------\n');
```

```
for x=1:0.5:10
    y2=(1+x)^n;
    sum=1.0;
    num=1.0;
    den=1;
    for j=1:n
        num=num*(n-j+1);
        den=factorial(j);
        term=num/den*x^j;
        sum=sum+term;
        y1=sum;
    end
    fprintf(' %5.1f   %15.1f   %15.1f \n',x,y1,y2);
end
```

Program output:

```
The binomial series formular for (1+x)^n is:
(1+x)^n=1+nx+n(n-1)*x^2/2!+n(n-1)(n-2)*x^3/3!+..n(n-1)(n-2)..
    (n-r+2)x^(r-1)/(r-1)!+..x^n
The series terminates with x^n, as long as n is an integer.
-------------------------------------------------------
    x                  y                        y
                  by expansion              by arith stm
-------------------------------------------------------
   1.0            1.02400e+003              1.02400e+003
-------------------------------------------------------
   1.5            9.53674e+003              9.53674e+003
-------------------------------------------------------
   2.0            5.90490e+004              5.90490e+004
```

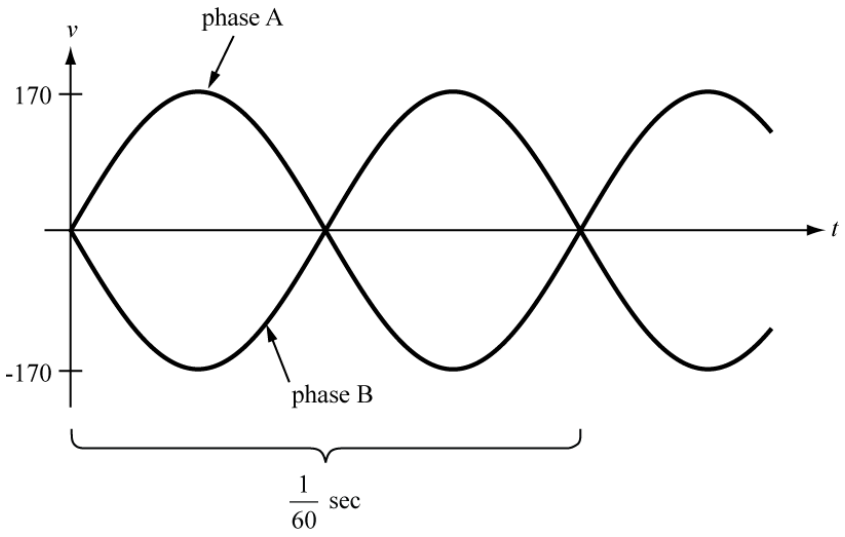
```
-------------------------------------------------------
   2.5            2.75855e+005             2.75855e+005
-------------------------------------------------------
   3.0            1.04858e+006             1.04858e+006
-------------------------------------------------------
   3.5            3.40506e+006             3.40506e+006

    ⋮

   8.0            3.48678e+009             3.48678e+009
-------------------------------------------------------
   8.5            5.98737e+009             5.98737e+009
-------------------------------------------------------
   9.0            1.00000e+010             1.00000e+010
-------------------------------------------------------
   9.5            1.62889e+010             1.62889e+010
-------------------------------------------------------
  10.0            2.59374e+010             2.59374e+010
-------------------------------------------------------
```

Project 2.6

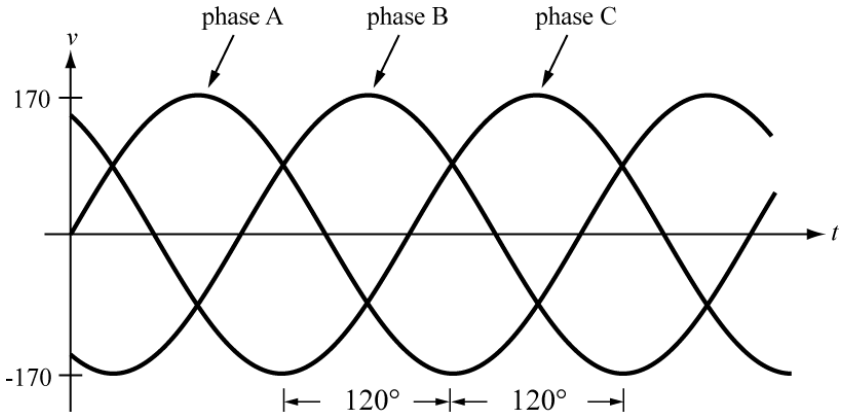
This project will examine a 340 $V_{peak\text{-}to\text{-}peak}$ sinusoidal waveform at 60 Hz which has no DC component (i.e. it is centered around 0 V).

a. Plot this waveform in MATLAB by using the `sin()` function. Plot over the interval $0 \le t \le 100$ millisec in steps of 2 millisec

b. Compute and print to the screen the root-mean-square (RMS) voltage for this waveform by performing the following operations in MATLAB:
    i. squaring it.
    ii. computing the mean value of the squared waveform by averaging it over the 100 millisec time interval.
    iii. taking the square root of the average
c. In North America, residential electrical service is usually delivered as two 340 $V_{peak\text{-}to\text{-}peak}$ sinusoids, each 180° out of phase with each other (Figure P2.6a), referred to as *one-phase* service. For high power appliances (e.g. air conditioners and stoves), the current is drawn from both "legs" of the service.
    i. Plot the *difference* waveform between the two legs of one-phase service.
    ii. Compute and print to the screen the RMS value of the difference waveform (as in part b above).
d. In many commercial and industrial buildings, *three-phase* electrical service is delivered as three 340 $V_{peak\text{-}to\text{-}peak}$ sinusoids, with each leg 120° out of phase with the others (see Figure P2.6b).
    i. Plot the waveform of the voltage difference between any two of the three legs
    ii. Compute and print to the screen the RMS value of this waveform. Most high-voltage appliances are rated to work at either of the RMS voltages computed in part c or part d.



(a)



(b)

Figure P2.6. (a) One-phase power is delivered as two sinusoids which are out of phase by 180°. (b) Three-phase power is delivered as three sinusoids which are out of phase by 120°.

Solution:

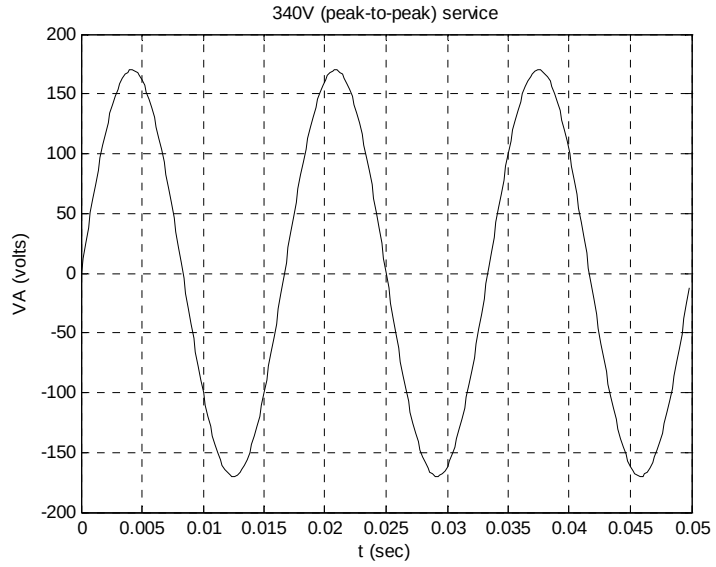
```
% Project_2_6.m
% This program determines the RMS of a sinusoidal waveform of AC voltage.
```

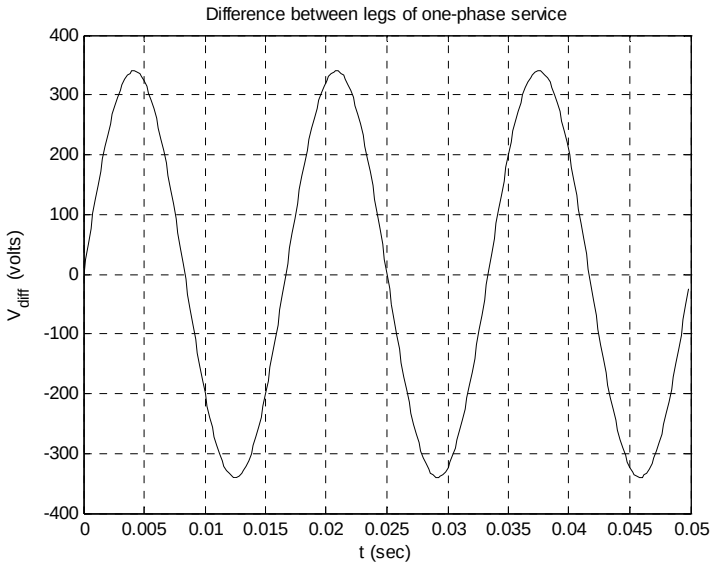
```
% The program also creates a plot of the voltage for 0<=t<=50 millisec
% in steps of 0.2 millisec.
clear; clc;
f = 60;
omega = 2*pi*f;
ampl = 170;
dt = .2e-3;
steps = 50e-3 / dt;
% part A:
for i=1:steps
    t(i) = (i-1)*dt;
    VA(i) = ampl*sin(omega*t(i));
    VA_squared(i) = VA(i)^2;
end
plot(t,VA), xlabel('t (sec)'), ylabel('VA (volts)'), grid;
title('340V (peak-to-peak) service');
figure;
% part B:
VA_avg=mean(VA_squared);
VA_rms=sqrt(VA_avg);
fprintf('RMS of VA = %.3f volts \n',VA_rms);
% part C:
for i=1:steps
    t(i)=(i-1)*dt;
    V_leg1(i)=ampl*sin(omega*t(i));
    V_leg2(i)=-ampl*sin(omega*t(i));
    V_diff(i)=V_leg1(i)-V_leg2(i);
    V_diff_squared(i)=V_diff(i)^2;
end
```

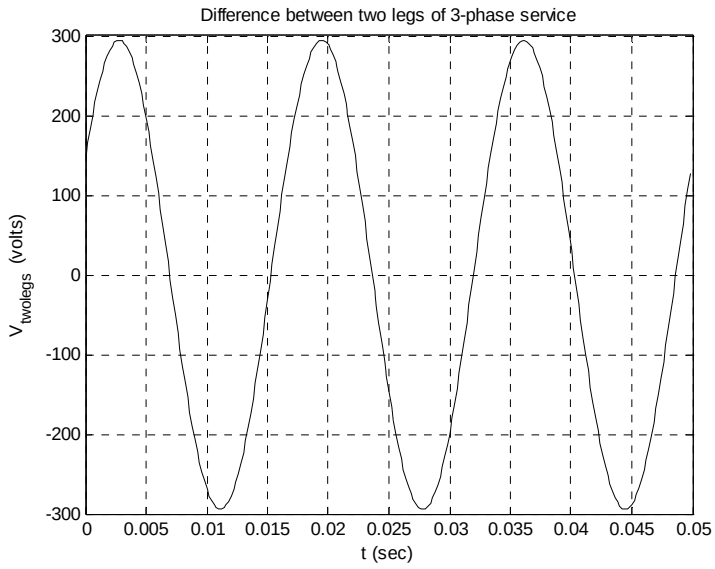
```
plot(t,V_diff);
xlabel('t (sec)'), ylabel('V_{diff} (volts)');
title('Difference between legs of one-phase service'), grid;
figure;
V_diff_avg = mean(V_diff_squared);
V_diff_rms = sqrt(V_diff_avg);
fprintf('RMS of one-phase service (V_leg1-V_leg2) = %.3f volts \n',...
    V_diff_rms);
% part D:
phi=120/180*pi;
for i=1:steps
    t(i)=(i-1)*dt;
    V_leg1(i)=ampl*sin(omega*t(i));
    V_leg2(i)=ampl*sin(omega*t(i)-phi);
    V_leg3(i)=ampl*sin(omega*t(i)+phi);
    V_twolegs(i) = V_leg1(i)-V_leg2(i);
    V_twolegs_squared(i)=V_twolegs(i)^2;
end
plot(t,V_twolegs);
xlabel('t (sec)'), ylabel('V_{twolegs} (volts)');
title('Difference between two legs of 3-phase service'), grid;
V_twolegs_avg = mean(V_twolegs_squared);
V_twolegs_rms = sqrt(V_twolegs_avg);
fprintf('RMS of 3-phase service (V_leg1-V_leg2) = %.3f volts \n',...
    V_twolegs_rms);
```

Program output:

```
RMS of VA = 120.208 volts

RMS of one-phase service (V_leg1-V_leg2) = 240.416 volts

RMS of 3-phase service (V_leg1-V_leg2) = 208.207 volts
```







Project 2.7

Given the following component values in the parallel RLC circuit described in section 2.17

$$R = 100\ \Omega, \quad L = 1\ \text{mH}, \quad C = 1\ \mu\text{F}$$

and the following initial conditions, determine the coefficients A and B for the following 3 cases:

i. $i_L(0) = 0.25\,\text{A}, \quad v_C(0) = 6\,\text{V}$ (Hint: convert $i_L(0)$ to $v'(0)$ with Equation (2.7))

ii. $i_L(0) = -0.25\,\text{A}, \quad v_C(0) = 6\,\text{V}$

iii. $i_L(0) = 0, \quad v_C(0) = 6\,\text{V}$

For each case, determine an expression for A and B in terms of R, L, C, $v_C(0)$ and $i_L(0)$ and write a MATLAB program that will:

a) Determine $v(t)$ for $0 \le t \le 500$ μsec in steps of 1 μsec.

b) Print out a table of v vs. t every 10 μsec.

c) Plot v vs. t for all three cases on the same graph. Label each curve with the value of $i_L(0)$.

Solution:

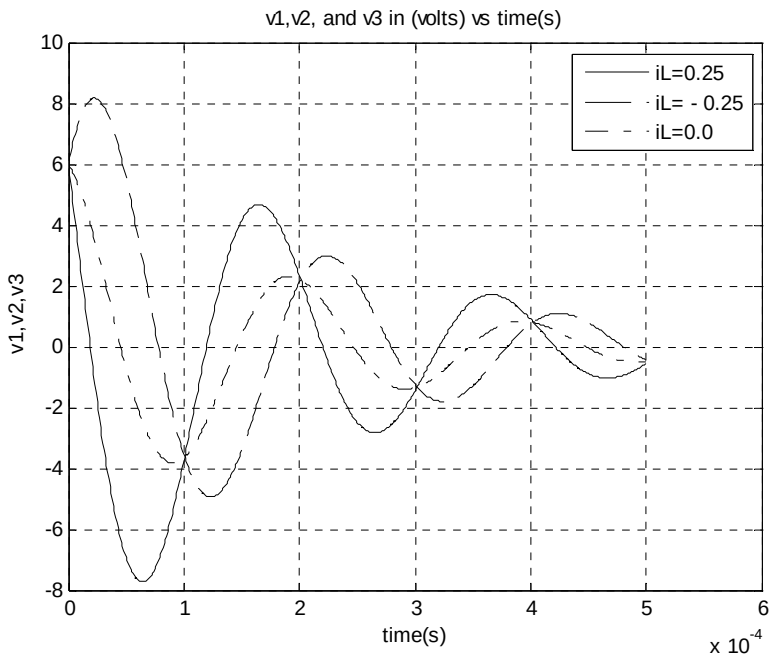
```
% Project_2_7.m
% This script determines the values of the coefficients A and B
% based on the initial conditions.
% R=100 ohms; L=1e-3 henry; C=1e-6 Farads;
clear; clc;
% v(0)=vc(0)=6 V
% (a); iL(0)=0.25 A, (b); iL(0)=-0.25 A, (c); iL(0)=0.0 A,
R=100; L=1e-3; C=1e-6; v0=6;
iL0=[0.25 -0.25 0];
v=zeros(501,1);
% v1 is for case (a), v2 is for case (b), v3 is for case (c)
v1=zeros(501,1);
v2=zeros(501,1);
v3=zeros(501,1);
system_parameter=(2*R*C)^2-(L*C);
fo=fopen('output.txt','w');
fprintf(fo,'system_parameter=%10.2e \n',system_parameter);
for j=1:3
    dvdt0=-1/(R*C)*v0-1/C*iL0(j);
    if system_parameter<0.0
        % system is overdamped
        % v0=A+B or B=v0-A
        % A=1/(2*sqrt(1/(2*R*C)^2-1/(L*C))*
        %  (-v0/(2RC)-iL0/C+v0*sqrt(1/(2*R*C)^2-1/(L*C))
        num=v0*sqrt(1/(2*R*C)^2-1/(L*C))-v0/(2*R*C)-iL0(j)/C;
        den=2*sqrt(1/(2*R*C)^2-1/(L*C));
        A=num/den;
```

```
    B=v0-A;
    D=0;
    fprintf(fo,'System is overdamped,j=%2i A=%8.4f B=%8.4f \n',...
  j,A,B);
 end
 if system_parameter>0.0
    % system is underdamped
    A=v0;
    num = -v0/(R*C)-1/C*iL0(j)+v0/(2*R*C);
    den = sqrt(1/(L*C)-1/(2*R*C)^2);
    B=num/den;
    D=1;
    fprintf(fo,'Sys is underdamped,j=%2i, A=%8.4f B=%8.4f \n', ...
  j,A,B);
end
% Creating table
dt=1.0e-6;
for i=1:501
    tm(i)=(i-1)*dt;
    t=tm(i);
    if D==0; %system is overdamped
        arg1=1/(2*R*C);
        arg2=sqrt(1/(2*R*C)^2-1/(L*C));
        v(i)=exp(-arg1*t)*(A*exp(arg2*t)+B*exp(-arg2*t));
    end
    if D==1; %system is underdamped
        arg1=1/(2*R*C);
        arg2=sqrt(1/(L*C)-1/(2*R*C)^2);
        v(i)=exp(-arg1*t)*(A*cos(arg2*t)+B*sin(arg2*t));
```

```
        end
    end
    if j==1
       v1=v;
    end
    if j==2
        v2=v;
    end
    if j==3
        v3=v;
    end
end
fprintf(fo,'   t(s)x1.0e+4    v1(volts)   v2(volts)  v3(volts)    \n');
fprintf(fo,'-------------------------------------------------------\n');
t2=tm*1.0e+4;
for i=1:10:501
    t(i)=tm(i)*1.0e+6;
    fprintf(fo,'%10.2f        %10.2f    %10.2f    %10.2f \n',...
      t2(i),v1(i),v2(i),v3(i));
end
plot(tm,v1,tm,v2,'--',tm,v3,'-.'), xlabel('time(s)'), ylabel('v1,v2,v3'),
grid,
    title('v1,v2, and v3 in (volts) vs time(s)'),
    legend('iL=0.25','iL= - 0.25','iL=0.0');
fclose(fo);
```

Program output:

```
system_parameter= 3.90e-008
System is underdamped,j= 1, A=  6.0000   B= -8.9672
System is underdamped,j= 2, A=  6.0000   B=  7.0456
System is underdamped,j= 3, A=  6.0000   B= -0.9608

    t(s)x1.0e+4     v1(volts)    v2(volts)   v3(volts)
-------------------------------------------------------
      0.00              6.00         6.00         6.00
      0.10              2.81         7.49         5.15
      0.20             -0.34         8.13         3.90
      0.30             -3.16         7.94         2.39
      0.40             -5.41         7.03         0.81
      0.50             -6.94         5.53        -0.70
      0.60             -7.67         3.66        -2.00
      0.70             -7.60         1.62        -2.99
      0.80             -6.82        -0.38        -3.60
      0.90             -5.48        -2.16        -3.82
        ⋮
      4.20              0.05         1.08         0.57
      4.30             -0.34         1.08         0.37
      4.40             -0.66         0.98         0.16
      4.50             -0.89         0.79        -0.05
      4.60             -1.01         0.55        -0.23
      4.70             -1.03         0.28        -0.37
      4.80             -0.95         0.01        -0.47
      4.90             -0.78        -0.23        -0.51
      5.00             -0.56        -0.44        -0.50
```



Project 2.8

The envelope of the voltage waveform for the parallel RLC circuit described in Section 2.17 is given by:

$$v(t) = \pm\, Ae^{\left(-t/2RC\right)} \tag{P2.8}$$

Given the following component values

$R = 5000\ \Omega$, L=1 μH, C=10 pF

$i_L(0) = 0\,\text{A}$, $v_C(0) = 3.3\,\text{V}$

Determine $v(t)$ for $0 \le t \le 50$ nsec in steps of 1 nsec. Plot v vs. t for both the oscillating function and its envelope on the same graph.

Solution:

```
% Project_2_8.m
% This script determines the values of the arbitrary coef, A and B,
% for project P2.8
```

```
% R=5000 ohms; L=1e-6 henry; C=10e-12 Farads;
% The system is underdamped [1/(2*R*C) < 1/(L*C)];
% v(0)=3.3 V
% iL(0)=0.0 A
clear; clc;
R=5000; L=1e-6; C=1e-12; v0=3.3; iL0=0.0;
v=zeros(501,1); vp=zeros(501,1); vm=zeros(501,1);
fo=fopen('output.txt','w');
dvdt0=-v0/(R*C)-iL0/C;
A=v0;
num = -v0/(R*C)-iL0/C+v0/(2*R*C);
den = sqrt(1/(L*C)-1/(2*R*C)^2);
B=num/den;
fprintf(fo,'systems are underdamped, A=%8.4f   B=%8.4f \n',A,B);
Ap=3.3;
Am=-3.3;
% Creating table
dt=1.0e-10;
for j=1:501
    tm(j)=(j-1)*dt;
    t=tm(j);
    arg1=1/(2*R*C);
    arg2=sqrt(1/(L*C)-1/(2*R*C)^2);
    v(j)=exp(-arg1*t)*(A*cos(arg2*t)+B*sin(arg2*t));
    vp(j)=Ap*exp(-arg1*t);
    vm(j)=Am*exp(-arg1*t);
end
fprintf(fo,'    t(s)x1.0e+9     v(volts)    vp(volts)  vm(volts) \n');
fprintf(fo,'-----------------------------------------------------\n');
```

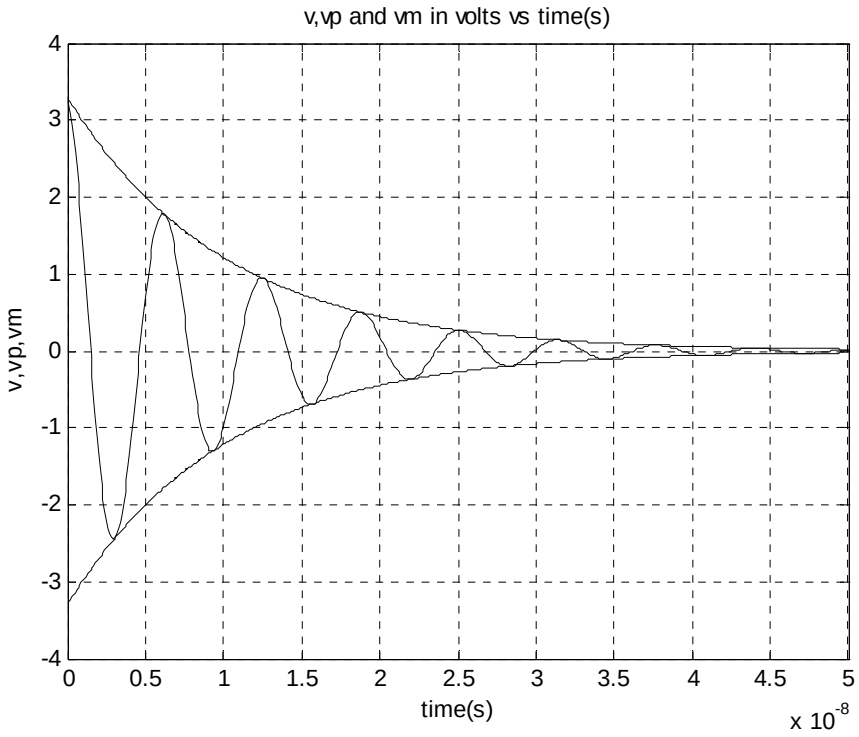
```
t2=tm*1.0e+9;
for j=1:10:501
    fprintf(fo,'%10.2f     %10.2f    %10.2f   %10.2f \n',...
t2(j),v(j),vp(j),vm(j));
end
plot(tm,v,tm,vp,tm,vm),xlabel('time(s)'), ylabel('v,vp,vm'), grid,
    title('v,vp and vm in volts vs time(s)'),
fclose(fo);
```

Program output:

```
systems are underdamped, A=  3.3000   B= -0.3317

    t(s)x1.0e+9     v(volts)    vp(volts)   vm(volts)
-----------------------------------------------------------
      0.00           3.30          3.30        -3.30
      1.00           1.37          2.99        -2.99
      2.00          -1.35          2.70        -2.70
      3.00          -2.45          2.44        -2.44
      4.00          -1.31          2.21        -2.21
      5.00           0.71          2.00        -2.00
      6.00           1.78          1.81        -1.81
      7.00           1.17          1.64        -1.64
      8.00          -0.30          1.48        -1.48
      9.00          -1.26          1.34        -1.34
     10.00          -0.99          1.21        -1.21
     11.00           0.05          1.10        -1.10
     12.00           0.86          0.99        -0.99
```

⋮

39.00	0.02	0.07	-0.07
40.00	-0.04	0.06	-0.06
41.00	-0.05	0.05	-0.05
42.00	-0.02	0.05	-0.05
43.00	0.02	0.04	-0.04
44.00	0.04	0.04	-0.04
45.00	0.02	0.04	-0.04
46.00	-0.01	0.03	-0.03
47.00	-0.03	0.03	-0.03
48.00	-0.02	0.03	-0.03
49.00	0.00	0.02	-0.02
50.00	0.02	0.02	-0.02



Project 2.9

Instead of using *v* as the state variable in the analysis of the parallel RLC circuit , we can use i_L by differentiating both sides of Equation (2.6) and combining with Equation (2.7) to obtain the differential equation

$$\frac{d^2 i_L}{dt^2} + \frac{1}{RC}\frac{di_L}{dt} + \frac{1}{LC} i_L = 0 \qquad \text{(P2.9a)}$$

Note that Equation (P2.9a) has the identical form as Equation (2.12) and thus has solutions of the identical form. For the under damped case,

$$i_L = \exp\left(-\frac{1}{2RC}t\right)\left\{\alpha \cos\left(\sqrt{\frac{1}{LC} - \left(\frac{1}{2RC}\right)^2}\ t\right) + \beta \sin\left(\sqrt{\frac{1}{LC} - \left(\frac{1}{2RC}\right)^2}\ t\right)\right\} \qquad \text{(P2.9b)}$$

where α and β are constants to be determined by initial conditions and have the units of amperes.

For the component values R = 100 Ω, L=1 mH, C=1μF and initial conditions $i_L(0) = 0$ and $v_C(0) = 6\,\text{V}$:

a) Solve for the coefficients α and β.

b) Plot $i_L(t)$ for $0 \le t \le 500$ μsec in steps of 1/2 μsec

c) Plot $i_L(t)$ and $v_C(t)$ (solved in Project 2.7) on the same axes. Note that when v_C is at a maximum, i_L is zero and vice versa. This illustrates how the energy in the circuit oscillates back and forth between the capacitor and inductor.

Solution:

```
% Project_2_9.m
% This script determines the values of the arbitrary coefficients,
```

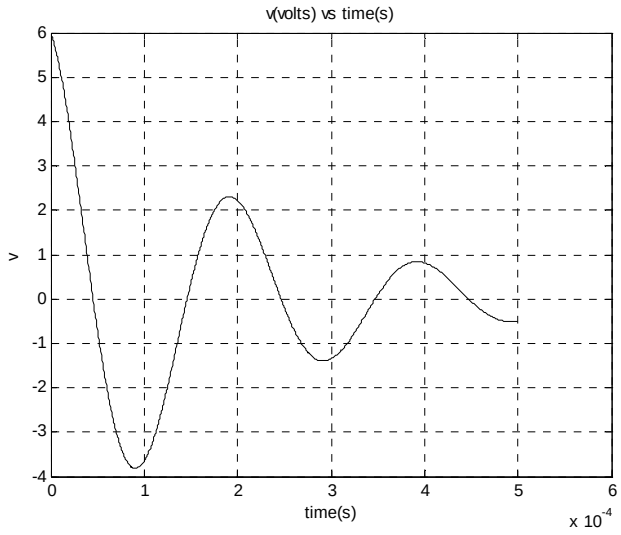
```
% alpha, beta, A and B for project 2.9. It also creates a table of v(t) % and
i(t) and also creates plots of v/6 and i on the same graph.
% R=100 ohms; L=1e-3 henry; C=1e-6 Farads;
% The system is under damped [1/LC > 1/(2*R*C)^2 ];
% v(0)=6 V
% iL(0)=0.0 A
clear; clc;
R=100; L=1e-3; C=1e-6; v0=6; iL0=0.0;
v=zeros(1001,1); i=zeros(1001,1);
fo=fopen('output.txt','w');
fprintf(fo,'Project 2.9 \n');
% i(t)=exp(-t/(2RC))*(alpha*cos(sqrt(1/LC-(1/2RC)^2)t)
% + beta*sin(sqrt(1/LC-(1/2RC)^2)t))
% alpha=iL(0)
% beta= 1/sqrt(1/LC-(1/2RC)^2)*(v(0)/L +iL(0)/2RC)
alpha=iL0;
num=v0/L+iL0/(2*R*C);
den=sqrt(1/(L*C)-1/(2*R*C)^2);
beta=num/den;
A=v0;
num =v0/(2*R*C)+iL0/C;
den = sqrt(1/(L*C)-1/(2*R*C)^2);
B=(-num)/den;
fprintf(fo,'system is underdamped, alpha=%8.4f  beta=%8.4f \n',...
      alpha,beta);
fprintf(fo,'A and B from Project 2.8, A=%8.4f  B=%8.4f \n',A,B);
% Creating table
dt=0.5e-6;
for j=1:1001
```

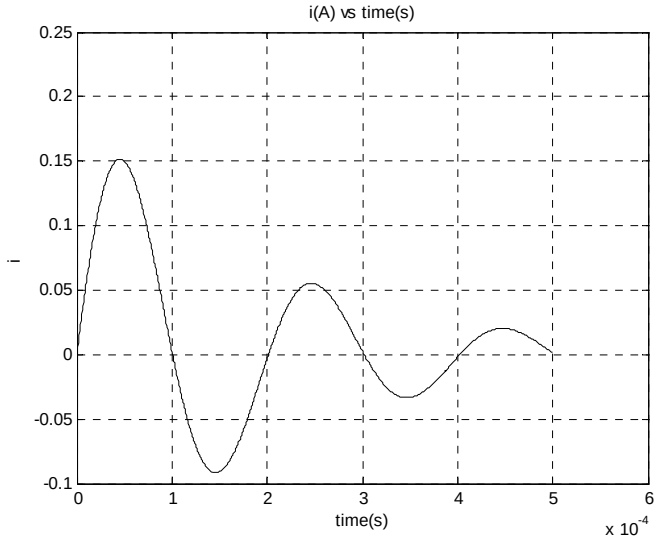
```
    tm(j)=(j-1)*dt;
    t=tm(j);
    arg1=1/(2*R*C);
    arg2=sqrt(1/(L*C)-1/(2*R*C)^2);
    v(j)=exp(-arg1*t)*(A*cos(arg2*t)+B*sin(arg2*t));
    i(j)=exp(-arg1*t)*(alpha*cos(arg2*t)+beta*sin(arg2*t));
end
fprintf(fo,'    t(s)x1.0e+4     v(volts)    i(Amperes)   \n');
fprintf(fo,'-----------------------------------------------------\n');
t2=tm*1.0e+4;
for j=1:20:1001
    fprintf(fo,'%10.2f        %10.2f    %10.2f    \n',t2(j),v(j),i(j));
end
v2=v/6.0;
plot(tm,i), xlabel('time(s)'), ylabel('i'), grid, title('i(A) vs time(s)'),
figure;
plot(tm,v), xlabel('time(s)'), ylabel('v'), grid,
title('v(volts) vs time(s)');
figure;
% To obtain plots of i and v of the same magnitude, plot i and v/6.
plot(tm,i,tm,v2,'--'),xlabel('time(s)'), ylabel('i,v/6'), grid,
    title('i(A) and v/6(volts) vs time(s)'),
    legend('i','v/6');
fclose(fo);
```

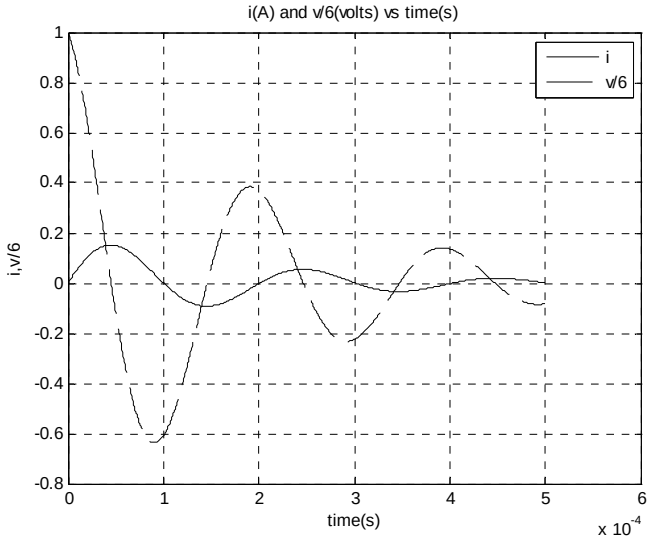
Program output:

```
Project 2.9
system is underdamped, alpha=  0.0000  beta=  0.1922
A=  6.0000  B= -0.9608
```

t(s)x1.0e+4	v(volts)	i(Amperes)
0.00	6.00	0.00
0.10	5.15	0.06
0.20	3.90	0.10
0.30	2.39	0.13
0.40	0.81	0.15
0.50	-0.70	0.15
0.60	-2.00	0.14
0.70	-2.99	0.11
0.80	-3.60	0.08
0.90	-3.82	0.04
1.00	-3.65	0.00
1.10	-3.15	-0.03
1.20	-2.41	-0.06
⋮		
4.00	0.82	-0.00
4.10	0.72	0.01
4.20	0.57	0.01
4.30	0.37	0.02
4.40	0.16	0.02
4.50	-0.05	0.02
4.60	-0.23	0.02
4.70	-0.37	0.02
4.80	-0.47	0.01
4.90	-0.51	0.01
5.00	-0.50	0.00


v(volts) vs time(s)
v
time(s)
x 10^-4


i(A) vs time(s)
i
time(s)
x 10^-4


i(A) and v/6(volts) vs time(s)
i
v/6
i,v/6
time(s)
x 10^-4

Project 2.10

The classical form for a second-order differential equation is

$$\frac{d^2y}{dt^2} + 2\zeta\omega_n \frac{dy}{dt} + \omega_n^2 y = 0 \qquad \text{(P2.10a)}$$

where ζ is the *damping factor* and ω_n is the *natural frequency*. We can match the terms of Equation (P2.9a) with Equation (P2.9b) to find the damping factor and natural frequency for the parallel RLC circuit. Thus,

$$\omega_n = \frac{1}{\sqrt{LC}} \quad \text{and} \quad \zeta = \frac{1}{2RC\omega_n} = \frac{\sqrt{LC}}{2RC} \qquad \text{(P2.10b)}$$

Note that for $\zeta < 1$ the circuit is underdamped, for $\zeta > 1$ the circuit is overdamped, and for $\zeta = 1$, it is critically damped.

For the component values *L*=1 mH, *C*=1 μF and initial conditions $i_L(0) = 0.25$ A and $v_C(0) = 6$ V:

a) Determine the resistor value R_{crit} which makes the circuit critically damped.

b) Plot the inductor current i_L vs. time for the three values of *R:* $R = R_{crit}$, $R = 5R_{crit}$, and $R = \frac{1}{2}R_{crit}$. Assume the interval $0 \le t \le 500$ μsec in steps of 1/2 μsec. Plot all of the waveforms on one graph.

Solution:

(a). Setting $\varsigma = 1$ in Equation (P2.10b) and solving for *R* gives R_{crit}, thus

$$R_{crit} = \frac{\sqrt{LC}}{2C} = \frac{\sqrt{10^{-3} \times 10^{-6}}}{2 \times 10^{-6}} = 15.8114 \text{ ohm}$$

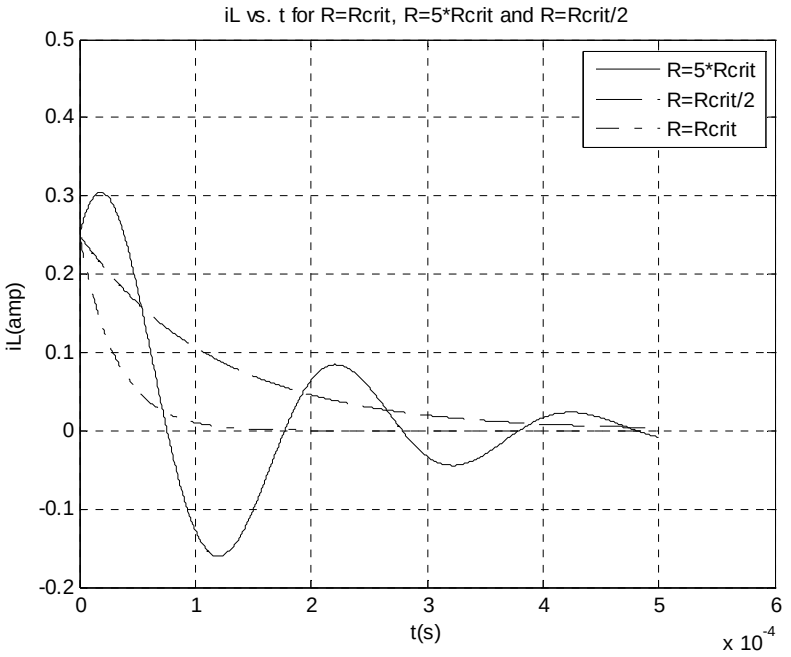
(b). The program follows for part (b).

```
% Project_2_10.m
% This project involves determing iL(t) of a RCL circuit for 3 cases;
% R=5Rcrit, R=Rcrit and R=Rcrit/2. The expressions for iL are in terms
% of there classical form. Component values L=0.001 H and C=1 microF.
% Initial conditions are v0=6 V and iL0=0.25 A.
clear; clc;
L=0.001; C=1.0e-6; iL0=0.25; v0=6;
omegan=1/sqrt(L*C);
Rcrit=1/(2*C*omegan);
fprintf('Project 2.10 \n\n');
fprintf('R-critical=%10.4f ohm \n',Rcrit);
% R=5*Rcrit
% underdamped case.
R=5*Rcrit;
si=1/(2*R*C*omegan);
alpha=iL0;
num=v0/L+si*omegan*iL0;
den=omegan*sqrt(1-si^2);
beta=num/den;
dt=0.5e-6;
arg=omegan*sqrt(1-si^2);
for n=1:1001
    t(n)=(n-1)*dt;
    iL1(n)=exp(-si*omegan*t(n))*(alpha*cos(arg*t(n))+ beta*sin(arg*t(n)));
end
% R=0.5Rcrit
% overdamped case.
R=0.5*Rcrit;
si=1/(2*R*C*omegan);
```

```
num=v0/L+iL0*omegan*(si+sqrt(si^2-1));
den=2*omegan*sqrt(si^2 -1);
A=num/den;
B=iL0-A;
arg=omegan*sqrt(si^2-1);
for n=1:1001
    t(n)=(n-1)*dt;
    iL2(n)=exp(-si*omegan*t(n))*(A*exp(arg*t(n))+ B*exp(arg*t(n)));
end
% R=Rcrit
% critically damped case.
R=Rcrit;
si=1/(2*R*C*omegan);
for n=1:1001
    t(n)=(n-1)*dt;
    iL3(n)=iL0*exp(-si*omegan*t(n));
end
plot(t,iL1,t,iL2,'--',t,iL3,'-.'), xlabel('t'), ylabel('iL'),
title('iL vs. t for R=Rcrit, R=5*Rcrit and R=Rcrit/2'), grid,
legend('R=5*Rcrit','R=Rcrit/2','R=Rcrit');
```

Program output:

```
R-critical= 15.8114 ohm
```



Project 2.11

We wish now to consider the RLC circuit described in Project 2.10 to be subjected to a driving current such that instead of opening the switch at $t = 0$, alternatively we close the switch. In this case, the sum of the currents at the top node is

$$i_R + i_L + i_C - I_o = 0 \qquad \text{(P2.11a)}$$

Note that the I_o term above is negative because I_o is arbitrarily defined in Figure 2.13 as flowing *into* the node instead of *away* from the node (as is true for the other three currents). The governing equation then becomes

$$\frac{d^2 i_L}{dt^2} + \frac{1}{RC}\frac{di_L}{dt} + \frac{1}{LC} i_L = I_o \qquad \text{(P2.11b)}$$

The complete solution of this differential equation is equal to the homogeneous solution (found in Project 2.10) plus a *particular solution*.

Let us assume that the driving current is a sinusoid of the form $I_o(t) = A_o \sin \omega_o t$. To obtain the particular solution i_P, we seek a solution whose first and second derivatives can be summed to equal the form of the driving current. Thus, assume

$$i_p = A_p \sin \omega_o t + B_p \cos \omega_o t \qquad \text{(P2.11c)}$$

where A_p and B_p are constants. Then,

$$i_p' = \omega_o \left(A_p \cos \omega_o t - B_p \sin \omega_o t \right)$$

and

$$i_p'' = \omega_o^2 \left(- A_p \sin \omega_o t - B_p \cos \omega_o t \right)$$

Substituting these expressions into Equation (P2.11b) gives

$$\left(- A_p \omega_o{}^2 - \frac{1}{RC} \omega_o B_p + \frac{1}{LC} A_p \right) \sin \omega_o t + \left(- B_p \omega_o{}^2 + \frac{1}{RC} \omega_o A_p + \frac{1}{LC} B_p \right) \cos \omega_o t = A_o \sin \omega_o t \qquad \text{(P2.11d)}$$

Collecting coefficients of the sine and cosine terms on the left side of Equation (P2.11d) and equating them to the sine and cosine coefficients on the right side of that equation gives

$$\left(\frac{1}{LC} - \omega_o^2 \right) A_p - \frac{\omega_o}{RC} B_p = A_o$$

$$\frac{\omega_o}{RC} A_p + \left(\frac{1}{LC} - \omega_o^2 \right) B_p = 0$$

Solving the above two equations for A_p and B_P gives

$$A_p = A_o \times \frac{\frac{1}{LC} - \omega_o^2}{\left(\frac{\omega_o}{RC}\right)^2 + \left(\frac{1}{LC} - \omega_o^2\right)^2} \tag{P2.11e}$$

$$B_p = -A_o \times \frac{\frac{\omega_o}{RC}}{\left(\frac{\omega_o}{RC}\right)^2 + \left(\frac{1}{LC} - \omega_o{}^2\right)} \tag{P2.11f}$$

Substituting Equations (P2.11e) and (P2.11f) into Equation (P2.11b) gives

$$i_p = A_o \times \frac{1}{\left(\frac{\omega_o}{RC}\right)^2 + \left(\frac{1}{LC} - \omega_o{}^2\right)^2} \left\{ \left(\frac{1}{LC} - \omega_o{}^2\right) \sin \omega_o t - \frac{\omega_o}{RC} \cos \omega_o t \right\} \tag{P2.11g}$$

We can rewrite Equation (P2.11g) using the trigonometric identity

$$\alpha \sin \omega t + \beta \cos \omega t = \gamma \sin(\omega t - \phi)$$

where

$$\gamma = \sqrt{\alpha^2 + \beta^2}$$

and

$$\phi = \tan^{-1} \frac{\beta}{\alpha}$$

Applying these relations to Equation (P2.11g) gives

$$i_p = A_o \times \frac{1}{\sqrt{\left(\frac{1}{LC} - \omega_o^2\right)^2 + \left(\frac{\omega_o}{RC}\right)^2}} \sin(\omega_o t - \phi) \tag{P2.11h}$$

Using the definitions from Project P2.10 for natural frequency $\omega_n = \sqrt{\frac{1}{LC}}$ and damping factor $\zeta = \frac{1}{2RC\omega_n}$, we can algebraically manipulate, Equation (P2.10h) into

$$i_p = \frac{A_o}{\omega_n} \frac{1}{\sqrt{\left(1-\frac{\omega_o}{\omega_n}\right)^2 + \left(2\zeta\frac{\omega_o}{\omega_n}\right)^2}} \sin(\omega_o t - \phi) \qquad \text{(P2.11i)}$$

The term $\frac{A_o}{\omega_n} \frac{1}{\sqrt{\left(1-\frac{\omega_o}{\omega_n}\right)^2 + \left(2\zeta\frac{\omega_o}{\omega_n}\right)^2}}$ is the amplitude of the oscillation. For a given

$\frac{A_o}{\omega_n}$, the larger the term $\frac{1}{\sqrt{\left(1-\frac{\omega_o}{\omega_n}\right)^2 + \left(2\zeta\frac{\omega_o}{\omega_n}\right)^2}}$, the larger is the amplitude.

Let ampl = $\frac{1}{\sqrt{\left(1-\frac{\omega_o}{\omega_n}\right)^2 + \left(2\zeta\frac{\omega_o}{\omega_n}\right)^2}}$.

a) Construct a MATLAB program to create a plot of ampl vs. $\frac{\omega_o}{\omega_n}$ for values of $\zeta =$ 1.0, 0.5,

0.25, 0.10, 0.05, and $0 < \frac{\omega_o}{\omega_n} < 2$ in steps of 0.01.

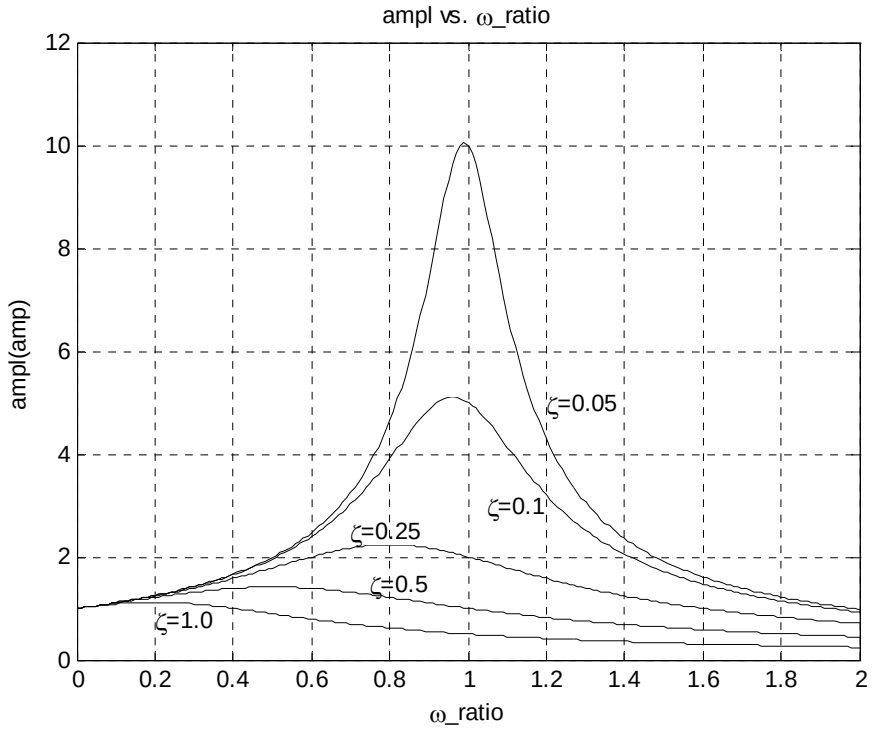
b) What happens as $\omega_o \to \omega_n$?

Solution:

```
% Project_2_11.m
% This program creates plots of ampl vs. wo/wn for several values of c.
% ampl=1/sqrt(arg), where arg=(1-wo/wn)^2+(2*c*wo/wn)^2
clear; clc;
zetam=[1.0 0.5 0.25 0.10 0.05];
dw=0.01;
ampl=zeros(201,5);
% wr=omega ratio
```

```
wr=zeros(201,1);
for i=1:5
    zeta=zetam(i);
    for j=1:201
        wr(j)=(j-1)*dw;
        arg=(1.0-wr(j))^2+(2*zeta*wr(j))^2;
        ampl(j,i)=1/sqrt(arg);
    end
end
plot(wr,ampl(:,1),wr,ampl(:,2), wr,ampl(:,3),wr,ampl(:,4),wr,ampl(:,5)),
grid,
xlabel('\omega\_ratio'), ylabel('ampl(amp)'),
title('ampl vs. \omega\_ratio'),
text(0.2,0.8,'\zeta=1.0'),text(0.75,1.5,'\zeta=0.5'),
text(0.7,2.5,'\zeta=0.25'),
text(1.05,3.0,'\zeta=0.1'),text(1.2,5.0,'\zeta=0.05');
```

Program output:



We see that as ω_o / ω_n approaches 1, the amplitude significantly increases. If there was no damping, the amplitude at $\omega_o / \omega_n = 1$ approaches infinity.

Project 2.12

Modify Example 2.25 as follows:

a) Use a self-written function saved as an *.m* file to do the interpolation. The linear interpolation formula for $y(x) = f(x)$ expression is

$$y = y_1 + \frac{x - x_1}{x_2 - x_1} \times (y_2 - y_1)$$

where points (x_1, x_2) are the nearest surrounding points to *x* and (y_1, y_2) are the *y* values at (x_1, x_2) respectively. Use a global statement to bring T_1, T_2, μ_1, and μ_2 into the self-written function; where T_1, T_2 are the closest temperature table values to the temperature at which μ

is to be determined and μ_1, μ_2 are the μ values at T_1, T_2 respectively.

b) Print out the table data to the screen, then ask the user if he/she wishes to interpolate for μ at a temperature other than those temperatures given in the table. If the answer is yes, ask the user to enter the temperature, do the interpolation and print out the result. Use the switch group to determine whether or not to do the interpolation. If the answer to your query is no, exit the program.

c) If the answer to the query in part (b) is yes, then ask the user if he/she wishes to enter another temperature, if yes, ask the user to enter the temperature, do the interpolation and printout the result. The process is to continue until the answer to the query is no. The user is to enter the following five temperatures in (K), one at a time (125, 180, 218, 334, 427).

Solution:

```
% Project_2_12.m: Interactive program to do interpolation of
%     electron mobility.
clear; clc;
global T1 T2 Mu1 Mu2;
T_data = [ 100 150 200 250 300 350 400 450 ];
Mu_data = [ 19650 7427 3723 2180 1407 972.0 705.5 531.8 ];
fprintf('I know the following values for mobility vs. temperature:\n');
fprintf('  T (K)     Mu (cm^2/V-s)\n');
fprintf('-------------------------\n');
for i=1:length(T_data)
    fprintf('%.1f      %10.1f\n',T_data(i),Mu_data(i));
end
response = 'y';
while response == 'y'
```

```
fprintf('Do you wish to interpolate for Mu at a temperature other\n');
response = input('than those listed above? (y/n): ', 's');
switch(response)
    case 'y'
        Mu=-1;
        T = input('Enter T at which Mu is to be determined: ');
        % check for valid input.  Just hitting return will cause
        % input() to return an empty vector.
        if length(T)==0
            continue;
        end
        % search for the two closest values to T that we know about:
        for i=1:length(T_data)-1
            if T > T_data(i) && T < T_data(i+1)
                % assign global variables and run our
                % interpolation function:
                T1=T_data(i);
                T2=T_data(i+1);
                Mu1=Mu_data(i);
                Mu2=Mu_data(i+1);
                Mu = interp_mobility(T);
                break;
            end
        end
        % If we didn't find two closest values, then give error:
        if Mu==-1
            fprintf('Temperature must be between %.1f and %.1f\n',...
                T_data(1),T_data(end));
        else
```

```
                fprintf('At %.1f degrees, Mu = %.1f cm^2/V-s\n',T,Mu);
            end
        case 'n'
            fprintf('Thanks for visiting.\n');
            break
        otherwise
            fprintf('Please answer ''y'' or ''n''\n');
            response = 'y';
    end
end
--------------------------------------------------------------------------
function [ Mu ] = interp_mobility( T )
global T1 T2 Mu1 Mu2;
Mu = Mu1 + ((T-T1)/(T2-T1)) * (Mu2-Mu1);
end
```

Program output:

```
I know the following values for mobility vs. temperature:
  T (K)     Mu (cm^2/V-s)
-------------------------
100.0         19650.0
150.0          7427.0
200.0          3723.0
250.0          2180.0
300.0          1407.0
350.0           972.0
400.0           705.5
450.0           531.8
Do you wish to interpolate for Mu at a temperature other
```

```
than those listed above? (y/n): y
Enter T at which Mu is to be determined: 125
At 125.0 degrees, Mu = 13538.5 cm^2/V-s
Do you wish to interpolate for Mu at a temperature other
than those listed above? (y/n): y
Enter T at which Mu is to be determined: 180
At 180.0 degrees, Mu = 5204.6 cm^2/V-s
Do you wish to interpolate for Mu at a temperature other
than those listed above? (y/n): y
Enter T at which Mu is to be determined: 218
At 218.0 degrees, Mu = 3167.5 cm^2/V-s
Do you wish to interpolate for Mu at a temperature other
than those listed above? (y/n): y
Enter T at which Mu is to be determined: 334
At 334.0 degrees, Mu = 1111.2 cm^2/V-s
Do you wish to interpolate for Mu at a temperature other
than those listed above? (y/n): y
Enter T at which Mu is to be determined: 427
At 427.0 degrees, Mu = 611.7 cm^2/V-s
Do you wish to interpolate for Mu at a temperature other
than those listed above? (y/n): x
Please answer 'y' or 'n'
Do you wish to interpolate for Mu at a temperature other
than those listed above? (y/n): y
Enter T at which Mu is to be determined: 20
Temperature must be between 100.0 and 450.0
Do you wish to interpolate for Mu at a temperature other
than those listed above? (y/n): n
Thanks for visiting.
```

Project 2.13

This project is a modification of Example 2.25 in which the "anonymous function" that does the interpolation is to consist of arguments of x, x_1, x_2, y_1, y_2 .and which uses the interpolation formula

$$y = y_1 + \frac{x - x_1}{x_2 - x_1} \times (y_2 - y_1)$$

where points (x_1, x_2) are the nearest surrounding points to *x* and (y_1, y_2) are the *y* values at (x_1, x_2) respectively. In Example 2.6 it was shown that the input variables to a function need not have the same names as those used in the calling program. For one-to-one correspondence, the argument positions in the calling program must be the same as the argument positions in the function. Table values for electron mobility μ as a function of temperature *T* are shown below:

T = [100 150 200 250 300 350 400 450]

μ = [19650 7427 3723 2180 1407 972.0 705.5 531.8]

The units for *T* are degrees (K) and the units for μ are cm^2/V-s.

a) Create a MATLAB program that contains the "anonymous function" as described above and which interpolates for the electron mobility μ at the following temperatures *TT:*

   TT = [110 165 215 365 440]

b) Print out a table of values of μ at temperatures *TT*.

Solution:

```
% Project_2_13.m: use an anonymous function to interpolate mobility
clear; clc;
T_data = [ 100 150 200 250 300 350 400 450 ];
Mu_data = [ 19650 7427 3723 2180 1407 972.0 705.5 531.8 ];
```

```
% Interpolation function:
interp_func = @(x,x1,x2,y1,y2) y1+(x-x1)/(x2-x1)*(y2-y1);
fprintf('This program interpolates for the electron mobility (Mu) at \n');
fprintf('a specified temperature T. \n');
fprintf('The allowable temperature range is 100-450K. \n\n');
TT = [110 165 215 365 440 ];
fprintf('  T (K)   Mu (cm^2/V-s)  \n');
fprintf('-----------------------------------\n');
for i=1:length(TT)
    % search for the two closest values to T that we know about:
    for j=1:length(T_data)-1
        if TT(i) > T_data(j) && TT(i) < T_data(j+1)
            Mu = interp_func(TT(i),T_data(j),T_data(j+1),...
                Mu_data(j),Mu_data(j+1));
            fprintf(' %.1f    %10.1f\n',TT(i),Mu);
            break;
        end
    end
end
```

Program output:

```
This program interpolates for the electron mobility (Mu) at
a specified temperature T.
The allowable temperature range is 100-450K.
  T (K)   Mu (cm^2/V-s)
-----------------------------------
 110.0       17205.4
 165.0        6315.8
 215.0        3260.1
```

```
365.0          892.0
440.0          566.5
```

Project 2.14

In discrete-time signal processing, signals are represented as a sequence of discrete numerical values, as opposed to the "smooth" waveforms in continuous-time signal processing.

Figure P2.14a shows a continuous-time waveform (a 1 kHz sine wave) and its discrete-time counterpart which is generated by sampling the continuous-time signal at a rate of 8000 sample/sec (every 125 μsec). This sampling rate is commonly used for voice-grade telephony equipment and the resulting discrete-time sequence might subsequently be transmitted by methods such as PAM (pulse amplitude modulation) or PCM (pulse code modulation).

Now suppose we want to use this sampled waveform in a movie soundtrack to be recorded on DVD. The digital audio on DVDs typically uses a sampling rate of 48000 sample/sec (every 20.8333 μsec), and thus we will need to *upsample* our 8000 sample/sec waveform in order to use it on DVD.

In upsampling, we augment a given discrete waveform by adding samples to the sequence in order to achieve the desired higher sampling rate. In this case, upsampling from 8000 to 48000 sample/sec will require adding 5 new samples for every one in the original waveform. There are many methods for doing this, and one simple approach is to compute values for the added samples by interpolating between adjacent values of the original waveform (Figure P2.14b).

Develop a MATLAB program that will:

a) Compute the signal values of the original 8000 sample/sec waveform over the period of $0 < t \leq 2$ millisec, for a total of 17 samples. Assume signal peak values of ± 1 for the sine

wave.

b) Create a table consisting of sample number, sample time and sample signal value.

c) Use MATLAB's `interp1` function to calculate the upsampled waveform, i.e., at a sample rate of 48000 samples/s. To avoid round off error, use a $\Delta t = 20.833333$ μsec. You should end up with a sample size of 97 values.

d) Plot the original and the upsampled waveforms on the same graph. For the original sample waveform, use circles to display the data points. For the upsampled waveform, use x's to display the data points.

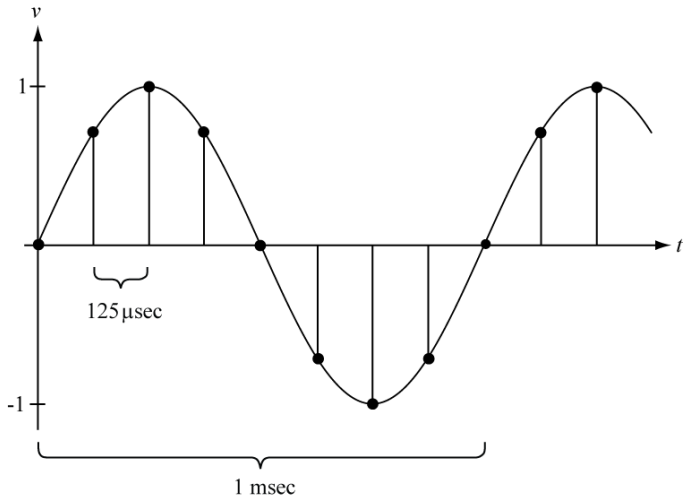


Figure P2.14a. A 1 kHz sine wave is sampled at 8000 samples/sec.

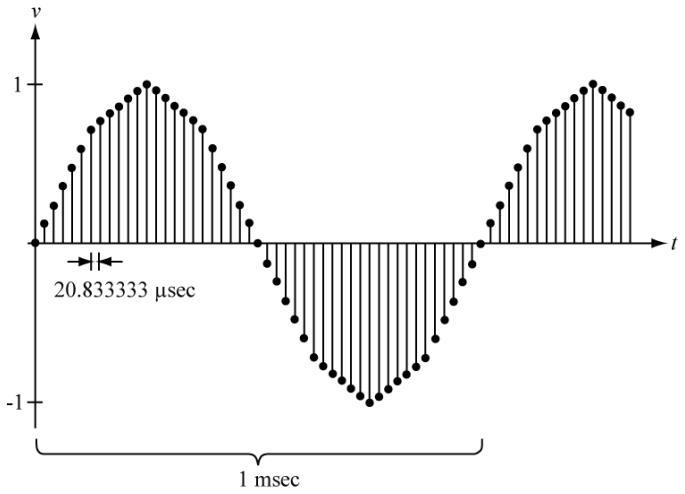


Figure P2.14b. If we want to upsample the sine wave to 48000 samples/sec, we need to interpolate five additional samples between points of the original waveform.

Solution:

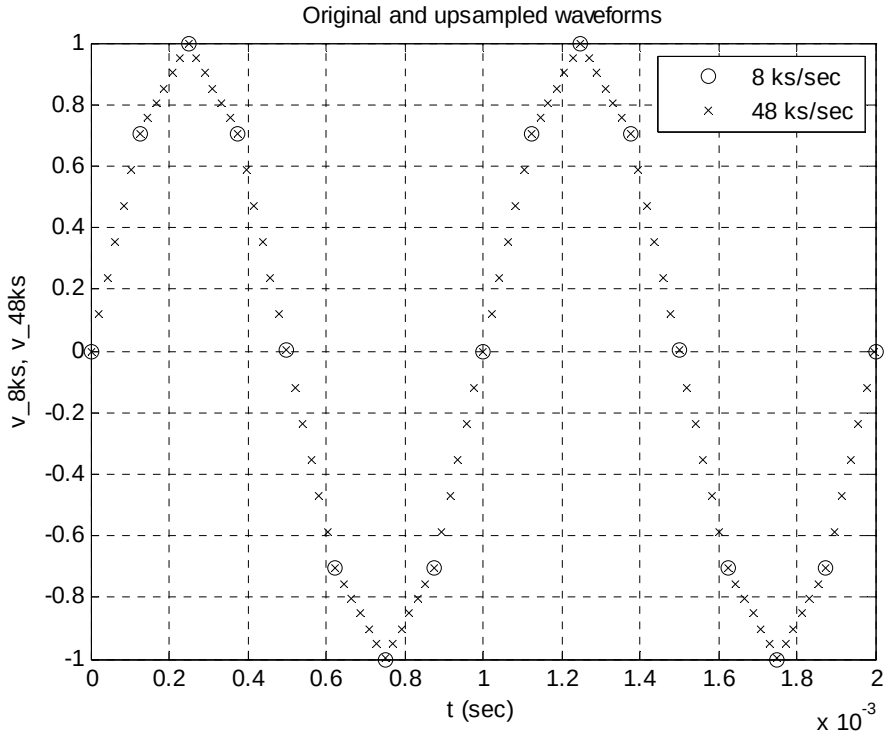
```
% Project_2_14.m
% This project involves discrete-time signal processing of a 1kHz
% sine wave. The object is to increase the number of discrete data
% points by interpolating between 17 sample data points.
clear; clc;
f=1000; % 1 kilohertz
dt_8ks = (1/f)/8;  % 125 usec
% Part a: compute 17 samples of 1kHz sine wave at 8 ksample/sec
for i=1:17
    t_8ks(i)=(i-1)*dt_8ks;
    v_8ks(i)=sin(2*pi*f*t_8ks(i));
end
% Part b: create table of 17 sample data points
fo=fopen('output.dat','w');
fprintf(fo,'sample #    t_8ks         v_8ks  \n');
fprintf(fo,'-----------------------------------------\n');
```

```
for i=1:17
    fprintf(fo,'%5i    %10.6f   %10.6f \n',i,t_8ks(i),v_8ks(i));
end
% Part c: upsample to 48kHz via interpolation
dt2=20.8333333e-6;
for i=1:97
    t_48ks(i)=(i-1)*dt2;
end
v_48ks=interp1(t_8ks,v_8ks,t_48ks);
% Part d: plot
plot(t_8ks,v_8ks,'o',t_48ks,v_48ks,'x');
xlabel('t (sec)'), ylabel('v\_8ks, v\_48ks'), grid;
title('Original and upsampled waveforms');
legend('8 ks/sec','48 ks/sec');
```

Program output:

```
sample #    t_8ks         v_8ks
-----------------------------------------
    1      0.000000     0.000000
    2      0.000125     0.707107
    3      0.000250     1.000000
    4      0.000375     0.707107
    5      0.000500     0.000000
    6      0.000625    -0.707107
    7      0.000750    -1.000000
    8      0.000875    -0.707107
    9      0.001000    -0.000000
   10      0.001125     0.707107
   11      0.001250     1.000000
```

```
12      0.001375     0.707107
13      0.001500     0.000000
14      0.001625    -0.707107
15      0.001750    -1.000000
16      0.001875    -0.707107
17      0.002000    -0.000000
```



Project 2.15

This project involves determining the quality of the upsampled waveform of Project 2.14. This is accomplished by comparing the upsampled waveform with an ideal 1 kHz sine wave. A common performance metric for audio systems is the *total harmonic distortion* (THD), which is a measure of the noise at harmonic frequencies other than the frequency of interest (1 kHz in our case).

Because our signal is periodic and of known frequency, we can assume that *all* the imperfections will contribute to the THD. Develop a MATLAB program that will:

a) Compute the values of an ideal 48000 sample/sec waveform V_{ideal} over the period of $0 < t \leq 2$ millisec, for a total of 97 samples. Assume waveform peak values of ± 1 for the sine wave.

b) Subtract this ideal waveform from our upsampled 48000 sample/sec waveform to calculate the waveform *error*, V_{error}, for the interpolation method that was used in Project 2.14.

c) Calculate the *total harmonic distortion* (THD) of the upsampled waveform as follows:

$$\text{THD} = \sqrt{\frac{\left(V_{error,RMS}\right)^2}{\left(V_{ideal,RMS}\right)^2}} \qquad \text{(P2.15)}$$

Use the method described in Project 2.6 to calculate $\left(V_{error,RMS}\right)$ and $\left(V_{ideal,RMS}\right)$.

d) Print out $V_{error,RMS}$, $V_{ideal,RMS}$ and THD.

Solution:

```
% Project_2_15.m
% This program determines the total harmonic distortion (THD) of an
% upsampled of 48000 sample/sec waveform. The upsampled waveform is
% compared to an ideal 1 kHz sine wave. The THD is determined over a
% 2 millisec period.
clear; clc;
f=1000; % 1 kilohertz
w=2*pi*f;
fprintf('This project calculates the total harmonic distortion \n');
fprintf('of an upsampled waveform as compared to an ideal 1 kHz \n');
fprintf('sine wave \n\n');
```

```
% Part A: calculate ideal waveform at 48 ks/sec
dt_48ks = 20.8333333e-6;
for i=1:97
    t_48ks(i)=(i-1)*dt_48ks;
    v_ideal(i)=sin(w*t_48ks(i));
end
% Part B: calculate upsampled waveform and subtract from ideal
% First, calculate 8 ks/sec waveform
dt_8ks = 125e-6;  % 125 usec
for i=1:17
    t_8ks(i)=(i-1)*dt_8ks;
    v_8ks(i)=sin(2*pi*f*t_8ks(i));
end
% Next, calculate upsampled waveform via interpolation of 8 ks waveform
v_48ks=interp1(t_8ks,v_8ks,t_48ks);
for i=1:97
    v_error(i) = v_48ks(i)-v_ideal(i);
end
% Part C: calculate square at each point for ideal and upsampled waveforms
for i=1:97
    v_error_sq(i) = v_error(i)^2;
    v_ideal_sq(i) = v_ideal(i)^2;
end
v_error_rms = sqrt( mean(v_error_sq) );
v_ideal_rms = sqrt( mean(v_ideal_sq) );
THD = v_error_rms / v_ideal_rms;
% Part D: print results
fprintf('V_error_RMS = %f\n',v_error_rms);
fprintf('V_ideal_RMS = %f\n',v_ideal_rms);
```

```
fprintf('THD = %f\n',THD);
```

Program output:

```
This project calculates the total harmonic distortion
of an upsampled waveform as compared to an ideal 1 kHz
sine wave
V_error_RMS = 0.039067
V_ideal_RMS = 0.703452
THD = 0.055536
```

Project 2.16

Mathematician Joseph Fourier is credited with the theorem that any periodic waveform may be expressed as a summation of pure sines and cosines. For example, the square wave of Figure P2.16a can be written as a sum of sines:

$$v(t) = \frac{4}{\pi}\sin\frac{2\pi t}{T} + \frac{4}{3\pi}\sin\frac{6\pi t}{T} + \frac{4}{5\pi}\sin\frac{10\pi t}{T} + \cdots$$

$$= \sum_{\substack{k=1 \\ k \text{ odd}}}^{\infty} \frac{4}{\pi k}\sin\frac{2\pi k t}{T}$$

Figure P2.16b shows the first three terms of the series and their summation.

a) Write a MATLAB script that utilizes the self-written function `sqwave(n,T,i)` which takes the following arguments:

   `n` the number of terms of the Fourier series.

   `T` the period of the square wave in seconds.

   `i` the number samples per period.

   The function should return two arrays, `t` and `V`, each containing `i` elements, where

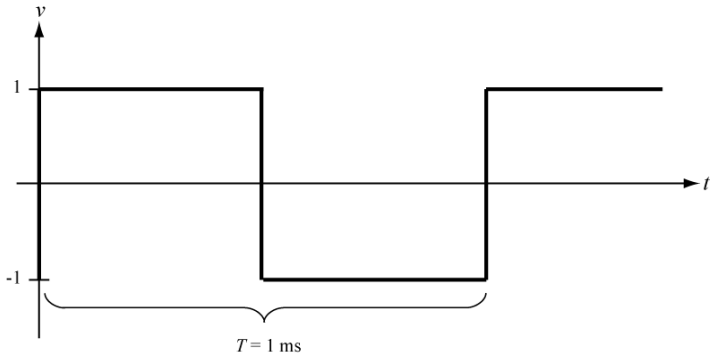
   `t =` an array of `i` time points.

`V =` an array of `i` computed values of the `n`th-degree approximated square wave

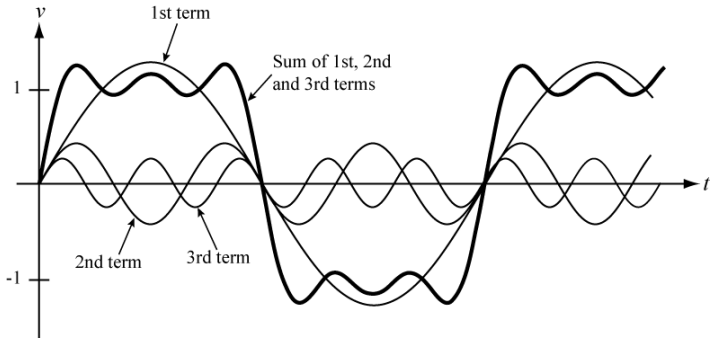
b) Run your `sqwave(n,T,i)` function and plot the results for the following arguments:

`T` = 1 millisec, `i` = 1001, `n` = 1, 3, 10, 100

c) For n = 100, print out a table of `(t,V)` at every 20 time points.



(a)



(b)

Figure P2.16. (a) Square wave. (b) Three term in a Fourier series of a square wave.

Solution:

```
% Project_2_16.m
% This program calculates the Fourier series of a square wave.
clear; clc;
% number of terms to calculate in Fourier series
fourier_order = [1 3 10 100];
```

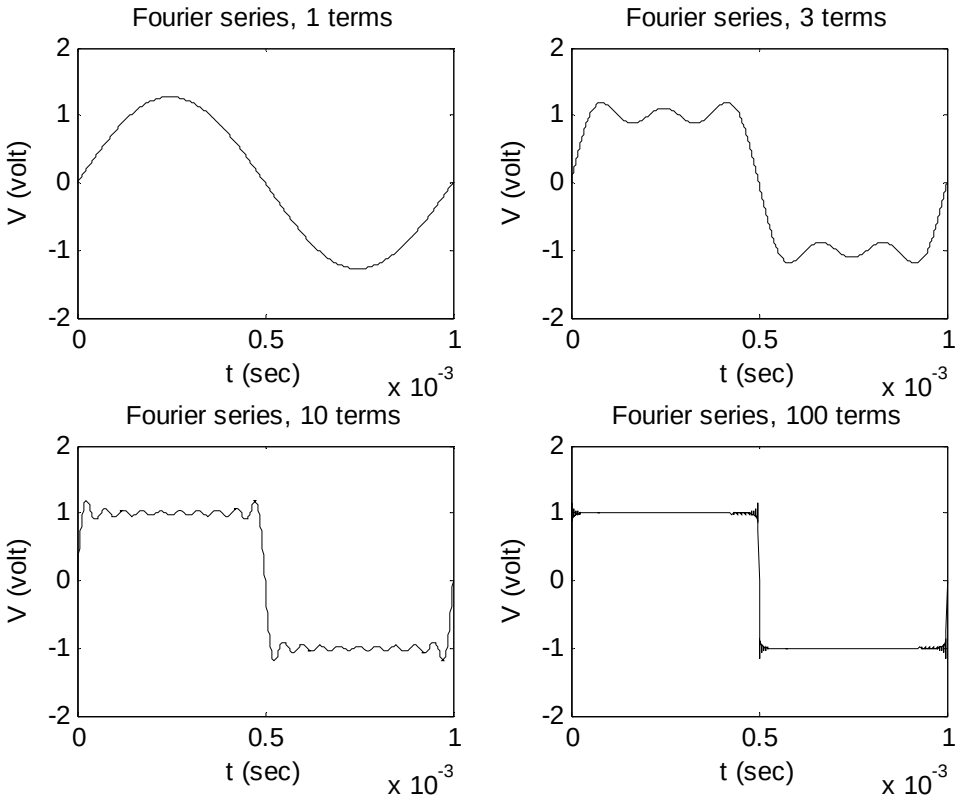
```
T=0.001;
i=1001;
for m=1:length(fourier_order)
    n=fourier_order(m);
    [t V]= sqwave(n,T,i);
    subplot(2,2,m);
    plot(t,V),xlabel('t (sec)'),ylabel('V (volt)');
    title(sprintf('Fourier series, %d terms',fourier_order(m)));
    if m==4
        fo=fopen('output.txt','w');
        fprintf(fo,'For 100-term Fourier series: \n');
        fprintf(fo,'      t (s)      V (volt)  \n');
        fprintf(fo,'------------------------- \n');
        for j=1:4:101
            fprintf(fo,'%10.6f  %10.6f  \n',t(j),V(j));
        end
        fclose(fo);
    end
end
--------------------------------------------------------------------------
% sqwave.m: compute Fourier series of square wave for n terms and i samples
%     for time period T.
function [t,V] = sqwave(n ,T, i)
dt=T/(i-1);
for j=1:i
    t(j) = (j-1)*dt;
    for term_no=1:n
        k=2*term_no-1;
        fourier_terms(term_no)=4/(pi*k) * sin(2*pi*k*t(j)/T);
```

```
    end
    V(j)=sum(fourier_terms);
end
```

Program output:

```
For 100-term Fourier series:
    t (s)      V (volt)
-------------------------
  0.000000    0.000000
  0.000004    0.983490
  0.000008    1.053849
  0.000012    1.032297
  0.000016    0.988772
  0.000020    0.974682
  0.000024    0.994125
  0.000028    1.014989
  0.000032    1.012650
  0.000036    0.995326
  0.000040    0.987211
  0.000044    0.996594
  0.000048    1.008766
  0.000052    1.007934
  0.000056    0.997028
  0.000060    0.991356
  0.000064    0.997578
  0.000068    1.006262
  0.000072    1.005846
  0.000076    0.997797
  0.000080    0.993394
```

```
0.000084    0.998099
0.000088    1.004932
0.000092    1.004686
0.000096    0.998228
0.000100    0.994585
```



Project 2.17

Bob's Hardware Store wishes to create an online program to sell inventory items in its store. You are to create an interactive MATLAB program for this purpose. The program is to contain the following items:

Data file:

a) A data file that contains a description of the inventory items for sale, their cost and the quantity available for sale. The list should contain at least 10 items. This data file will be read

into the main script. Whenever an item is purchased, the inventory available for sale should be updated in the data file.

Main script:

b) The script is to print to the screen the items for sale, their cost, and the quantity available for purchase.

c) The script is to ask the user if he/she wishes to make a purchase. If yes, the script is to ask the user the line number of the item he/she wishes to purchase and the quantity.

d) If the user does not wish to make a purchase, exit the program. If the user does not respond correctly to the query, print an error message to the screen and inform the user to restart the program. Then exit the program.

e) If the user does make a purchase, the program is to continue asking the user if he/she wishes to make another purchase. If yes, print to the screen the list of items, their cost and the updated items available for purchase. Then, the script is to ask the user the line number of the item he/she wishes to purchase and the quantity. If the answer is no, exit the program.

Billing function:

When the user is finished making all his/her purchases, the script is to call a function that will print out a bill for the items purchased. The bill should contain the name and address of the store, the user's first and last names, the user's address, bill headings, a list of all the items purchased, their unit prices, the total price for each item purchased, and finally the total price of all the items purchased.

Solution:

```
% Project_2_17.m
% retail
% This program was created for an on line retail store that sells
% hardware. The program is interactive. It displays various items for
% sale and asks the user if he/she wishes to make a purchase. If the
% user wishes to make a purchase a bill of sale is produced.
clear; clc;
global des;
fid=fopen('inv1.txt');
C = textscan(fid,'%d %14c %f  %d',5);
% Contents of cell block C contains 1 row and 4 columns
LN=C{1};
des=C{2};
cost=C{3};
quant=C{4};
fclose(fid);
fprintf('Catalog     Item description    cost($)  quantity \n');
fprintf('number                                   available \n');
fprintf('-------------------------------------------------\n');
for i=1:5
    fprintf('%5i   %19s   %6.2f   %5i \n\n',...
      LN(i),des(i,1:14),cost(i),quant(i));
end
nr=0;
fprintf('The items listed above are for sale \n');
fprintf('Do you wish to make a purchase? \n');
ans=input('Enter Y for yes and N for no \n','s');
```

```
switch(ans)
    case 'N'
        quit;
    case 'Y'
        fname=input('enter your first name \n','s');
        Lname=input('enter your last name \n','s');
        nr=nr+1;
        LN1(nr)=input('Enter the catalog number of the item that ...
            you wish to purchase \n');
        LN2(nr)=LN1(nr)-1000;
        quan(nr)=input('Enter the amount of this item to be ...
            purchased \n');
        quant(LN2(nr))=quant(LN2(nr))-quan(nr);
end
while(ans=='Y')
    fprintf('Do you wish to make another purchase? \n');
    ans=input('Enter Y for yes and N for no \n','s');
    switch(ans)
        case 'N'
            break;
        case 'Y'
        clc;
        fprintf('Catalog   Item description  cost($)  quantity \n');
        fprintf('number                                 available \n');
        fprintf('-----------------------------------------------\n');
        for i=1:5
            fprintf('%5i   %19s   %6.2f   %5i \n\n',...
      LN(i),des(i,1:14),cost(i),quant(i));
        end
```

```
        nr=nr+1;
        LN1(nr)=input('Enter the line number of the item that you ...
             wish to purchase \n');
        LN2(nr)=LN1(nr)-1000;
        quan(nr)=input('Enter the amount of this item to be ...
            purchased \n');
        quant(LN2(nr))=quant(LN2(nr))- quan(nr);
    end
end
clc;
billf(fname,Lname,nr,LN2,cost,quan);
fo=fopen('inv1.txt','w');
for i=1:5
    fprintf(fo,'%5i   %14s   %6.2f  %5i  \n\n',...
LN(i),des(i,1:14),cost(i),quant(i));
end
------------------------------------------------------------------------
% This function works with script 'retail'
function []=billf(fname,Lname,nr,LN2,cost,quan)
global des;
clc;
fprintf('               Bob Retail store  \n');
fprintf('               779 Glades Road \n');
fprintf('               Boca Raton, FL 33431 \n\n');
fprintf('-----------------------------------------------------\n\n');
fprintf('Customer Name: %s  %s  \n',fname, Lname);
fprintf('-------------------------------------------\n\n');
fprintf('Catalog # item description unit cost quantity sub-total  \n');
fprintf('-----------------------------------------------------\n');
```

```
for i=1:nr
    LN=LN2(i);
    cat_num=LN+1000;
    sub_total(i)=quan(i)*cost(LN);
    fprintf('%5i  %20s     %6.2f   %10i  %10.2f \n',...
        cat_num,des(LN,1:14),cost(LN),quan(i),sub_total(i));
end
fprintf('-----------------------------------------------------------\n')
total=sum(sub_total);
fprintf('                          Total cost:        $%7.2f \n',total);
```

Program output:

```
Catalog   Item description    cost($)  quantity
number                                 available
-------------------------------------------------
 1001        hammer             2.58      38

 1002        plier              1.20      38

 1003        screw driver       1.56      42

 1004        soldering iron     3.70      40

 1005        wrench             2.60      45

The items listed above are for sale
Do you wish to make a purchase?
Enter Y for yes and N for no
Y
```

```
enter your first name
John
enter your last name
Doe
Enter the catalog number of the item that you wish to purchase
1001
Enter the amount of this item to be purchased
3
Do you wish to make another purchase?
Enter Y for yes and N for no
Y
Catalog     Item description    cost($)  quantity
number                                   available
-------------------------------------------------
 1001        hammer             2.58      35

 1002        plier              1.20      38

 1003        screw driver       1.56      42

 1004        soldering iron     3.70      40

 1005        wrench             2.60      45

Enter the line number of the item that you wish to purchase
1002
Enter the amount of this item to be purchased
3
Do you wish to make another purchase?
Enter Y for yes and N for no
Y
Catalog     Item description    cost($)  quantity
```

```
number                                    available
-------------------------------------------------
 1001        hammer             2.58      35

 1002        plier              1.20      35

 1003        screw driver       1.56      42

 1004        soldering iron     3.70      40

 1005        wrench             2.60      45

Enter the line number of the item that you wish to purchase
1005
Enter the amount of this item to be purchased
4
Do you wish to make another purchase?
Enter Y for yes and N for no
N

               Bob Retail store
               779 Glades Road
               Boca Raton, FL 33431

-------------------------------------------------------------

Customer Name: John  Doe
-----------------------------------------

Catalog #  item description     unit cost   quantity   sub-total
-----------------------------------------------------------------
 1001        hammer               2.58            3        7.74
```

```
 1002        plier                1.20            3        3.60
 1005        wrench               2.60            4       10.40
-----------------------------------------------------------------
                                 Total cost:            $  21.74
```