CHAPTER 3

3.1 The M-file can be written as

```
function Vol = tankvolume(R, d)
if d < R
  Vol = pi * d ^ 3 / 3;
elseif d <= 3 * R
  V1 = pi * R ^ 3 / 3;
  V2 = pi * R ^ 2 * (d - R);
  Vol = V1 + V2;
else
  Vol = 'overtop';
end
```

This function can be used to evaluate the test cases,

```
>> tankvolume(0.9,1)
ans =
    1.0179
>> tankvolume(1.5,1.25)
ans =
    2.0453
>> tankvolume(1.3,3.8)
ans =
   15.5739
>> tankvolume(1.3,4)
ans =
overtop
```

3.2 The M-file can be written as

```
function futureworth(P, i, n)
nn=0:n;
F=P*(1+i).^nn;
y=[nn;F];
fprintf('\n  year   future worth\n');
fprintf('%5d %14.2f\n',y);
```

This function can be used to evaluate the test case,

```
>> futureworth(100000,0.05,10)

  year   future worth
    0      100000.00
    1      105000.00
    2      110250.00
    3      115762.50
    4      121550.63
    5      127628.16
    6      134009.56
    7      140710.04
    8      147745.54
    9      155132.82
   10      162889.46
```

3.3 The M-file can be written as

```
function annualpayment(P, i, n)
```

```
nn = 1:n;
A = P*i*(1+i).^nn./((1+i).^nn-1);
y = [nn;A];
fprintf('\n  year   annual payment\n');
fprintf('%5d %14.2f\n',y);
```

This function can be used to evaluate the test case,

```
>> annualpayment(100000,.033,5)

  year   annual payment
    1      103300.00
    2       52488.39
    3       35557.14
    4       27095.97
    5       22022.84
```

3.4 The M-file can be written as

```
function Ta = avgtemp(Tm, Tp, ts, te)
w = 2*pi/365;
t = ts:te;
T = Tm + (Tp-Tm)*cos(w*(t-205));
Ta = mean(T);
```

This function can be used to evaluate the test cases,

```
>> avgtemp(23.1,33.6,0,59)
ans =
   13.1332
>> avgtemp(10.6,17.6,180,242)
ans =
   17.2265
```

3.5 The M-file can be written as

```
function sincomp(x,n)
i = 1;
tru = sin(x);
ser = 0;
fprintf('\n');
fprintf('order  true value     approximation     error\n');
while (1)
  if i > n, break, end
  ser = ser + (-1)^(i - 1) * x^(2*i-1) / factorial(2*i-1);
  er = (tru - ser) / tru * 100;
  fprintf('%3d  %14.10f  %14.10f  %12.7f\n',i,tru,ser,er);
  i = i + 1;
end
```

This function can be used to evaluate the test case,

```
>> sincomp(0.9,8)

order  true value     approximation     error
  1    0.7833269096    0.9000000000  -14.89455921
  2    0.7833269096    0.7785000000    0.61620628
  3    0.7833269096    0.7834207500   -0.01197972
  4    0.7833269096    0.7833258498    0.00013530
  5    0.7833269096    0.7833269174   -0.00000100
```

```
  6    0.7833269096    0.7833269096    0.00000001
  7    0.7833269096    0.7833269096   -0.00000000
  8    0.7833269096    0.7833269096    0.00000000
```

3.6 The M-file can be written as

```
function [r, th] = polar(x, y)
r = sqrt(x .^ 2 + y .^ 2);
if x > 0
  th = atan(y/x);
elseif x < 0
  if y > 0
    th = atan(y / x) + pi;
  elseif y < 0
    th = atan(y / x) - pi;
  else
    th = pi;
  end
else
  if y > 0
    th = pi / 2;
  elseif y < 0
    th = -pi / 2;
  else
    th = 0;
  end
end
th = th * 180 / pi;
```

This function can be used to evaluate the test cases. For example, for the first case,

```
>> [r,th]=polar(2,0)
r =
     2
th =
     0
```

All the cases are summarized as

x	y	r	θ
2	0	2	0
2	1	2.236068	26.56505
0	3	3	90
-3	1	3.162278	161.5651
-2	0	2	180
-1	-2	2.236068	-116.565
0	0	0	0
0	-2	2	-90
2	2	2.828427	45

3.7 The M-file can be written as

```
function polar2(x, y)
r = sqrt(x .^ 2 + y .^ 2);
n = length(x);
for i = 1:n
  if x(i) > 0
    th(i) = atan(y(i) / x(i));
  elseif x(i) < 0
```

```
    if y(i) > 0
      th(i) = atan(y(i) / x(i)) + pi;
    elseif y(i) < 0
      th(i) = atan(y(i) / x(i)) - pi;
    else
      th(i) = pi;
    end
  else
    if y(i) > 0
      th(i) = pi / 2;
    elseif y(i) < 0
      th(i) = -pi / 2;
    else
      th(i) = 0;
    end
  end
  th(i) = th(i) * 180 / pi;
end
ou=[x;y;r;th];
fprintf('\n     x        y       radius     angle\n');
fprintf('%8.2f %8.2f %10.4f %10.4f\n',ou);
```

This function can be used to evaluate the test cases and display the results in tabular form,

```
>> x=[2 2 0 -3 -2 -1 0 0 2];
>> y=[0 1 3 1 0 -2 0 -2 2];
>> polar2(x,y)

     x        y       radius     angle
    2.00     0.00     2.0000     0.0000
    2.00     1.00     2.2361    26.5651
    0.00     3.00     3.0000    90.0000
   -3.00     1.00     3.1623   161.5651
   -2.00     0.00     2.0000   180.0000
   -1.00    -2.00     2.2361  -116.5651
    0.00     0.00     0.0000     0.0000
    0.00    -2.00     2.0000   -90.0000
    2.00     2.00     2.8284    45.0000
```

3.8 The M-file can be written as

```
function grade = lettergrade(score)
if score <0 | score>100
  error('Value must be >= 0 and <= 100')
elseif score >= 90
  grade = 'A';
elseif score >= 80
  grade = 'B';
elseif score >= 70
  grade = 'C';
elseif score >= 60
  grade = 'D';
else
  grade = 'F';
end
```

This function can be tested with a few cases,

```
>> grade = lettergrade(89.9999)
grade =
```

```
B
>> grade = lettergrade(90)
grade =
A
>> grade = lettergrade(45)
grade =
F
>> grade = lettergrade(120)
??? Error using ==> lettergrade
Incorrect value entered
```

3.9 The M-file can be written as

```
function Manning(A)
A(:,5)=sqrt(A(:,2))./A(:,1).*(A(:,3).*A(:,4)./(A(:,3)+2*A(:,4))).^(2/3);
fprintf('\n    n         S          B          H          U\n');
fprintf('%8.3f %8.4f %10.2f %10.2f %10.4f\n',A');
```

This function can be run to create the table,

```
>> A=[.036 .0001 10 2
.020 .0002 8 1
.015 .0012 20 1.5
.03 .0007 25 3
.022 .0003 15 2.6];
>> Manning(A)

    n         S          B          H          U
   0.036   0.0001      10.00       2.00     0.3523
   0.020   0.0002       8.00       1.00     0.6094
   0.015   0.0012      20.00       1.50     2.7569
   0.030   0.0007      25.00       3.00     1.5894
   0.022   0.0003      15.00       2.60     1.2207
```

3.10 The M-file can be written as

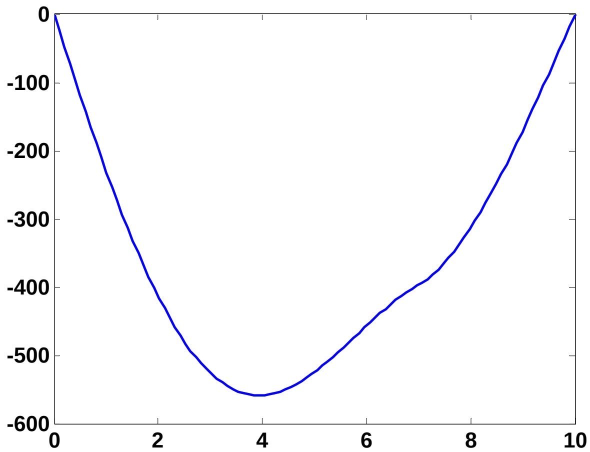
```
function beamProb(x)
xx = linspace(0,x);
n=length(xx);
for i=1:n
  uy(i) = -5/6.*(sing(xx(i),0,4)-sing(xx(i),5,4));
  uy(i) = uy(i) + 15/6.*sing(xx(i),8,3) + 75*sing(xx(i),7,2);
  uy(i) = uy(i) + 57/6.*xx(i)^3 - 238.25.*xx(i);
end
clf,plot(xx,uy,'--')
```

The M-file uses the following function

```
function s = sing(xxx,a,n)
if xxx > a
  s = (xxx - a).^n;
else
  s=0;
end
```

The plot can then be created as,

```
>> beam(10)
```

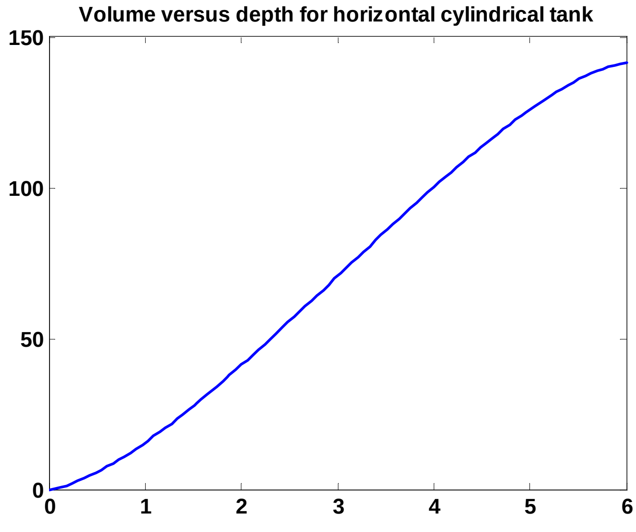


3.11 The M-file can be written as

```
function cylinder(r, L, plot_title)
% volume of horizontal cylinder
% inputs:
% r = radius
% L = length
% plot_title = string holding plot title
h = linspace(0,2*r);
V = (r^2*acos((r-h)./r)-(r-h).*sqrt(2*r*h-h.^2))*L;
clf,plot(h, V)
```

This function can be run to generate the plot,

```
>> cylinder(3,5,...
'Volume versus depth for horizontal cylindrical tank')
```



3.12 Errata for first printing: The loop should be:

```
for i=2:ni+1
  t(i)=t(i-1)+(tend-tstart)/ni;
  y(i)=12 + 6*cos(2*pi*t(i)/ ...
```

```
        (tend-tstart));
end
```

A vectorized version can be written as

```
tt=tstart:(tend-tstart)/ni:tend
yy=12+6*cos(2*pi*tt/(tend-tstart))
```

Both generate the following values for t and y:

```
t =
  0 2.5000 5.0000 7.5000 10.0000 12.5000 15.0000 17.5000 20.0000
y =
 18.0000 16.2426 12.0000 7.7574 6.0000 7.7574 12.0000 16.2426 18.0000
```

3.13

```
function s=SquareRoot(a,eps)
ind=1;
if a ~= 0
  if a < 0
    a=-a;ind=j;
  end
  x = a / 2;
  while(1)
    y = (x + a / x) / 2;
    e = abs((y - x) / y);
    x = y;
    if e < eps, break, end
  end
  s = x;
else
  s = 0;
end
s=s*ind;
```

The function can be tested:

```
>> SquareRoot(0,1e-4)
ans =
     0
>> SquareRoot(2,1e-4)
ans =
    1.4142
>> SquareRoot(10,1e-4)
ans =
    3.1623
>> SquareRoot(-4,1e-4)
ans =
        0 + 2.0000i
```

3.14 Errata: On first printing, change function for $8 \le t < 16$ to `v = 624 – 3*t;`

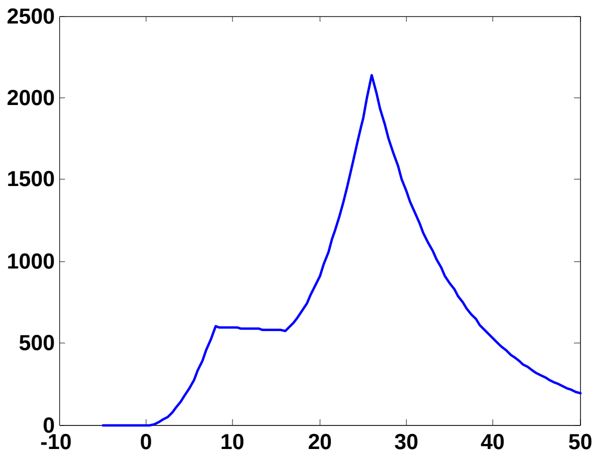
The following function implements the piecewise function:

```
function v = vpiece(t)
if t<0
  v = 0;
elseif t<8
  v = 10*t^2 - 5*t;
```

```
elseif t<16
  v = 624 - 3*t;
elseif t<26
  v = 36*t + 12*(t - 16)^2;
else
  v = 2136*exp(-0.1*(t-26));
end
```

Here is a script that uses `vpiece` to generate the plot

```
k=0;
for i = -5:.5:50
  k=k+1;
  t(k)=i;
  v(k)=vpiece(t(k));
end
plot(t,v)
```



3.15 This following function employs the round to larger method. For this approach, if the number is exactly halfway between two possible rounded values, it is always rounded to the larger number.

```
function xr=rounder(x, n)
if n < 0,error('negative number of integers illegal'),end
xr=round(x*10^n)/10^n;
```

Here are the test cases:

```
>> rounder(477.9587,2)
ans =
  477.9600
>> rounder(-477.9587,2)
ans =
 -477.9600
>> rounder(0.125,2)
ans =
    0.1300
>> rounder(0.135,2)
ans =
    0.1400
>> rounder(-0.125,2)
ans =
   -0.1300
```

```
>> rounder(-0.135,2)
ans =
   -0.1400
```

A preferable approach is called *banker's rounding* or *round to even*. In this method, a number exactly midway between two possible rounded values returns the value whose rightmost significant digit is even. Here is as function that implements banker's rounding along with the test cases that illustrate how it differs from `rounder`:

```
function xr=rounderbank(x, n)
if n < 0,error('negative number of integers illegal'),end
x=x*10^n;
if mod(floor(abs(x)),2)==0 & abs(x-floor(x))==0.5
  xr=round(x/2)*2;
else
  xr=round(x);
end
xr=xr/10^n;
```

```
>> rounder(477.9587,2)
ans =
  477.9600
>> rounder(-477.9587,2)
ans =
 -477.9600
>> rounderbank(0.125,2)
ans =
    0.1200
>> rounderbank(0.135,2)
ans =
    0.1400
>> rounderbank(-0.125,2)
ans =
   -0.1200
>> rounderbank(-0.135,2)
ans =
   -0.1400
```

3.16

```
function nd = days(mo, da, leap)
nd = 0;
for m=1:mo-1
  switch m
    case {1, 3, 5, 7, 8, 10, 12}
      nday = 31;
    case {4, 6, 9, 11}
      nday = 30;
    case 2
      nday = 28+leap;
  end
  nd=nd+nday;
end
nd = nd + da;
```

```
>> days(1,1,0)
ans =
     1
>> days(2,29,1)
```

```
ans =
    60
>> days(3,1,0)
ans =
    60
>> days(6,21,0)
ans =
   173
>> days(12,31,1)
ans =
   366
```

3.17

```
function nd = days(mo, da, year)
leap = 0;
if year / 4 - fix(year / 4) == 0, leap = 1; end
nd = 0;
for m=1:mo-1
  switch m
    case {1, 3, 5, 7, 8, 10, 12}
      nday = 31;
    case {4, 6, 9, 11}
      nday = 30;
    case 2
      nday = 28+leap;
  end
  nd=nd+nday;
end
nd = nd + da;

>> days(1,1,1997)
ans =
     1
>> days(2,29,2004)
ans =
    60
>> days(3,1,2001)
ans =
    60
>> days(6,21,2004)
ans =
   173
>> days(12,31,2008)
ans =
   366
```

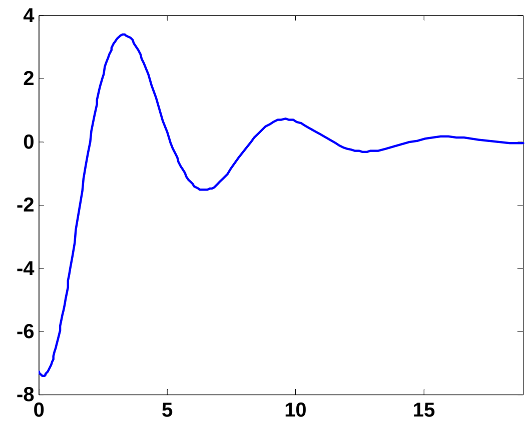
3.18

```
function fr = funcrange(f,a,b,n,varargin)
% funcrange: function range and plot
%   fr=funcrange(f,a,b,n,varargin): computes difference
%      between maximum and minimum value of function over
%      a range. In addition, generates a plot of the function.
% input:
%   f = function to be evaluated
%   a = lower bound of range
```

```
%   b = upper bound of range
%   n = number of intervals
% output:
%   fr = maximum - minimum
x = linspace(a,b,n);
y = f(x,varargin{:});
fr = max(y)-min(y);
fplot(f,[a b],varargin{:})
end
```

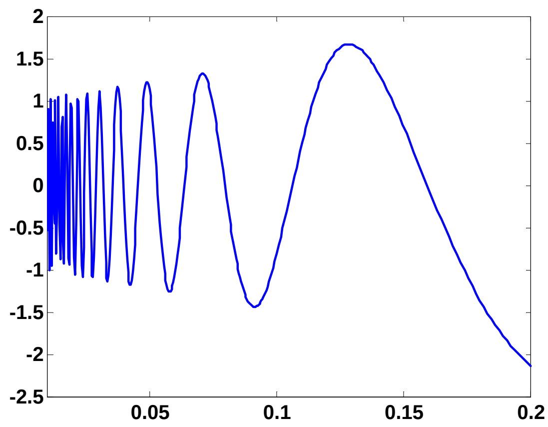
(a)

```
>> f=@(t) 8*exp(-0.25*t).*sin(t-2);
>> funcrange(f,0,6*pi,1000)
ans =
   10.7910
```



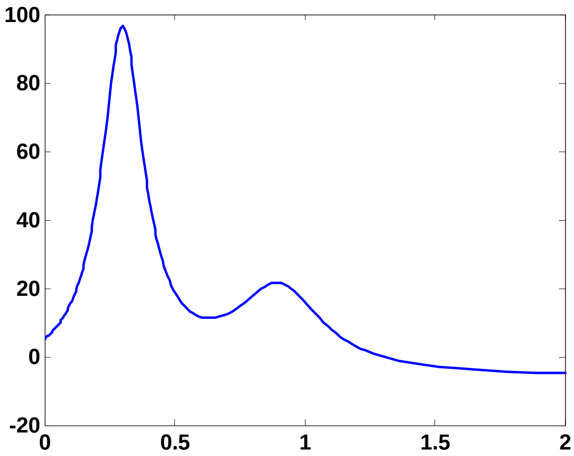
(b)

```
>> f=@(x) exp(4*x).*sin(1./x);
>> funcrange(f,0.01,0.2,1000)
ans =
    3.8018
```



(c)

```
>> funcrange(@humps,0,2,1000)
ans =
  101.3565
```



3.19
```
function yend = odesimp(dydt, dt, ti, tf, yi, varargin)
% odesimp: Euler ode solver
% yend = odesimp(dydt, dt, ti, tf, yi, varargin):
%   Euler's method solution of a single ode
% input:
%   dydt = function defining ode
%   dt = time step
%   ti = initial time
%   tf = final time
%   yi = initial value of dependent variable
% output:
%   yend = dependent variable at final time
t = ti; y = yi; h = dt;
while (1)
  if t + dt > tf, h = tf - t; end
  y = y + dydt(y,varargin{:}) * h;
  t = t + h;
  if t >= tf, break, end
end
yend = y;
```

test run:
```
>> dvdt=@(v,m,cd) 9.81-(cd/m)*v^2;
>> odesimp(dvdt,0.5,0,12,-10,70,0.23)

ans =
   51.1932
```

3.20 Here is a function to solve this problem:

```
function [theta,c,mag]=vector(a,b)
amag=norm(a);
bmag=norm(b);
adotb=dot(a,b);
theta=acos(adotb/amag/bmag)*180/pi;
c=cross(a,b);
mag=norm(c);
x1=[0 a(1)];y1=[0 a(2)];z1=[0 a(3)];
x2=[0 b(1)];y2=[0 b(2)];z2=[0 b(3)];
x3=[0 c(1)];y3=[0 c(2)];z3=[0 c(3)];
```

```
x4=[0 0];y4=[0 0];z4=[0 0];
plot3(x1,y1,z1,'--b',x2,y2,z2,'--r',x3,y3,z3,'-k',x4,y4,z4,'o')
xlabel='x';ylabel='y';zlabel='z';
```

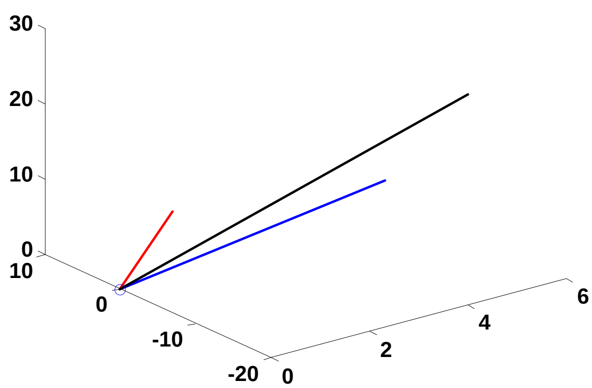
Here is a script to run the three cases

```
a = [6 4 2]; b = [2 6 4];
[th,c,m]=vector(a,b)
pause
a = [3 2 -6]; b = [4 -3 1];
[th,c,m]=vector(a,b)
pause
a = [2 -2 1]; b = [4 2 -4];
[th,c,m]=vector(a,b)
pause
a = [-1 0 0]; b = [0 -1 0];
[th,c,m]=vector(a,b)
```

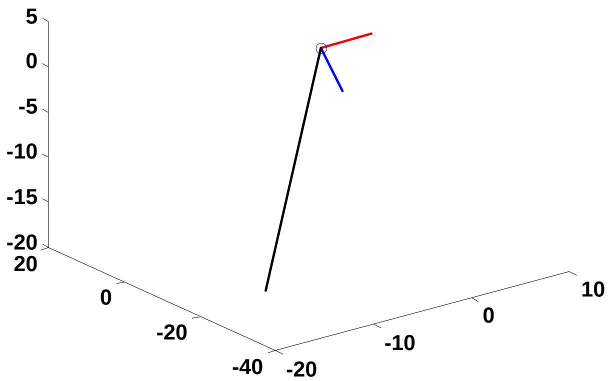
When this is run, the following output is generated

(a)

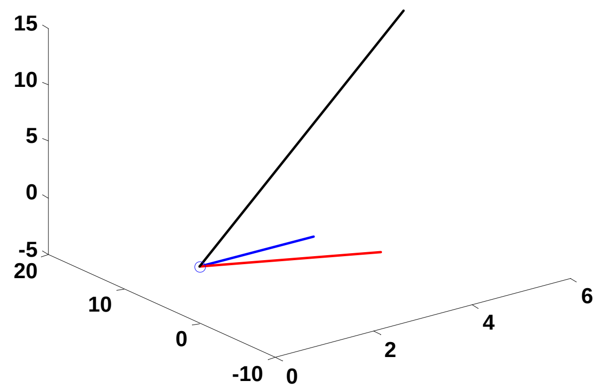
```
th =
   38.2132
c =
     4   -20    28
m =
   34.6410
```



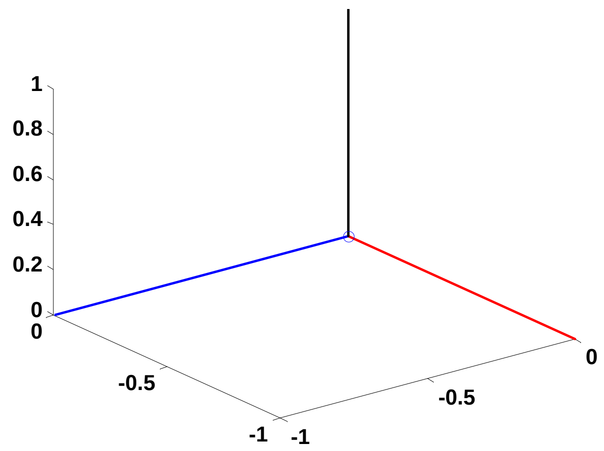

```
(b)
th =
    90
c =
   -16   -27   -17
m =
   35.6931
```




```
(c)
th =
    90
c =
     6    12    12
m =
    18
```



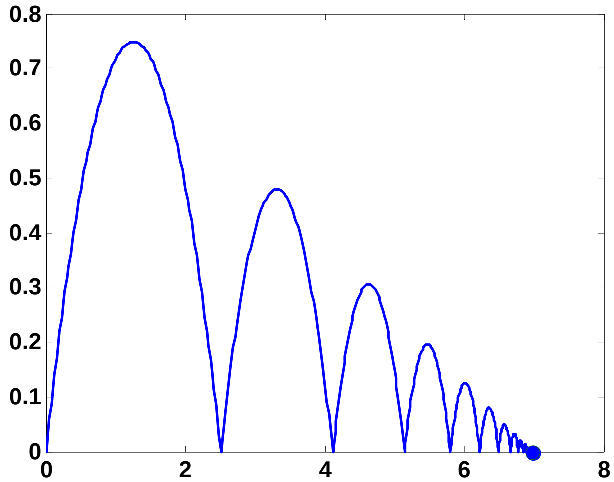

```
(d)
th =
    90
c =
     0     0     1
m =
     1
```



3.21 The script for this problem can be written as

```
clc,clf,clear
maxit=1000;
g=9.81; theta0=50*pi/180; v0=5; CR=0.83;
j=1;t(j)=0;x=0;y=0;
xx=x;yy=y;
plot(x,y,'o','MarkerFaceColor','b','MarkerSize',8)
xmax=8; axis([0 xmax 0 0.8])
M(1)=getframe;
dt=1/128;
j=1; xxx=0; iter=0;
while(1)
  tt=0;
  timpact=2*v0*sin(theta0)/g;
  ximpact=v0*cos(theta0)*timpact;
  while(1)
    j=j+1;
    h=dt;
    if tt+h>timpact,h=timpact-tt;end
    t(j)=t(j-1)+h;
    tt=tt+h;
    x=xxx+v0*cos(theta0)*tt;
    y=v0*sin(theta0)*tt-0.5*g*tt^2;
    xx=[xx x];yy=[yy y];
    plot(xx,yy,':',x,y,'o','MarkerFaceColor','b','MarkerSize',8)
    axis([0 xmax 0 0.8])
    M(j)=getframe;
    iter=iter+1;
    if tt>=timpact, break, end
  end
  v0=CR*v0;
  xxx=x;
  if x>=xmax|iter>=maxit,break,end
end
pause
clf
axis([0 xmax 0 0.8])
movie(M,1,36)
```

Here's the plot that will be generated:

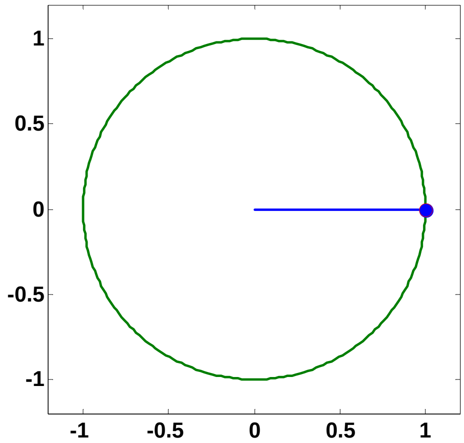


3.22 The function for this problem can be written as

```
function phasor(r, nt, nm)
% function to show the orbit of a phasor
% r = radius
% nt = number of increments for theta
% nm = number of movies
clc;clf
dtheta=2*pi/nt;
th=0;
fac=1.2;
xx=r;yy=0;
for i=1:nt+1
  x=r*cos(th);y=r*sin(th);
  xx=[xx x];yy=[yy y];
  plot([0 x],[0 y],xx,yy,':',...
        x,y,'o','MarkerFaceColor','b','MarkerSize',8)
  axis([-fac*r fac*r -fac*r fac*r]);
  axis square
  M(i)=getframe;
  th=th+dtheta;
end
pause
clf
axis([-fac*r fac*r -fac*r fac*r]);
axis square
movie(M,1,36)
```

When it is run, the result is

```
>> phasor(1, 256, 10)
```



3.23 A script to solve this problem based on the parametric equations described in Prob. 2.22:

```
clc;clf
t=[0:1/16:128];
x(1)=sin(t(1)).*(exp(cos(t(1)))-2*cos(4*t(1))-sin(t(1)/12).^5);
y(1)=cos(t(1)).*(exp(cos(t(1)))-2*cos(4*t(1))-sin(t(1)/12).^5);
xx=x;yy=y;
plot(x,y,xx,yy,':',x,y,'o','MarkerFaceColor','b','MarkerSize',8)
axis([-4 4 -4 4]); axis square
M(1)=getframe;
for i = 2:length(t)
  x=sin(t(i)).*(exp(cos(t(i)))-2*cos(4*t(i))-sin(t(i)/12).^5);
  y=cos(t(i)).*(exp(cos(t(i)))-2*cos(4*t(i))-sin(t(i)/12).^5);
  xx=[xx x];yy=[yy y];
  plot(x,y,xx,yy,':',x,y,'o','MarkerFaceColor','b','MarkerSize',8)
  axis([-4 4 -4 4]); axis square
  M(i)=getframe;
end
```