Homework #7

Table of Contents

Ian Blackstone, Helena-Nikolai Fujishin
Math 365, Fall 2016

Problems

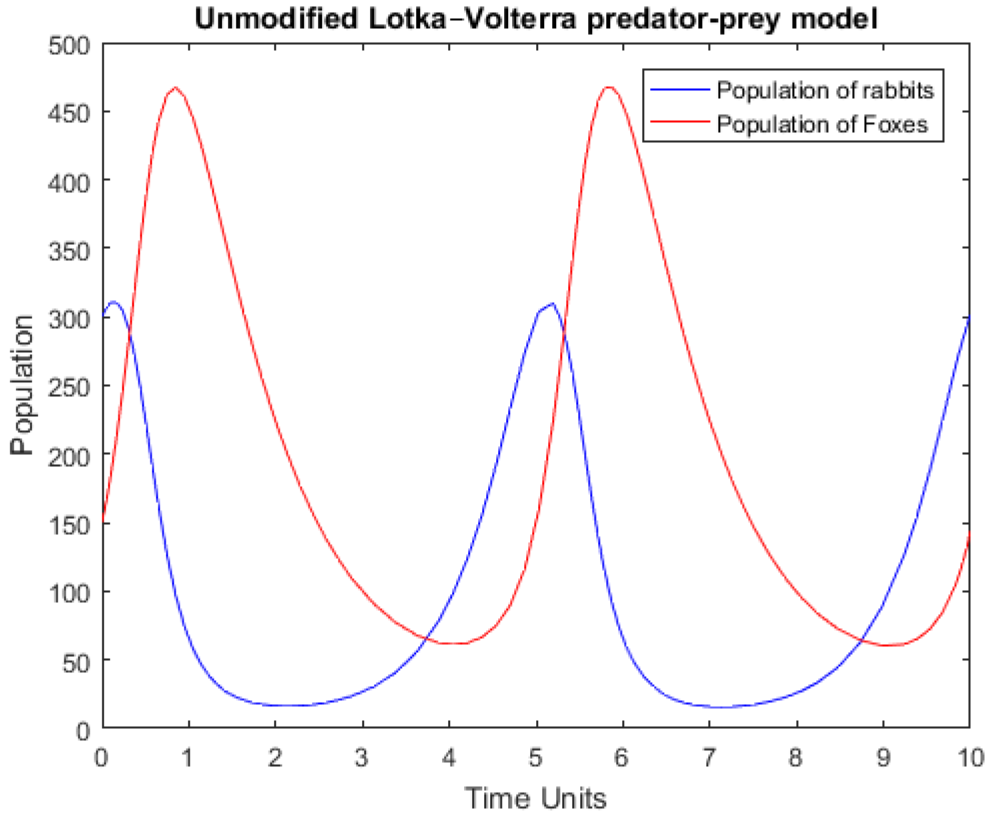
```
function hmwk1()
hmwk_problem(@prob1,'prob1');
hmwk_problem(@prob2,'prob2');
hmwk_problem(@prob3,'prob3');
end
function hmwk_problem(prob,msg)
try
 prob()
 fprintf('% s : Success!\n',msg);
catch me
 fprintf('% s : Something went wrong.\n',msg);
 fprintf('% s\n',me.message);
end
fprintf('\n');
end
```

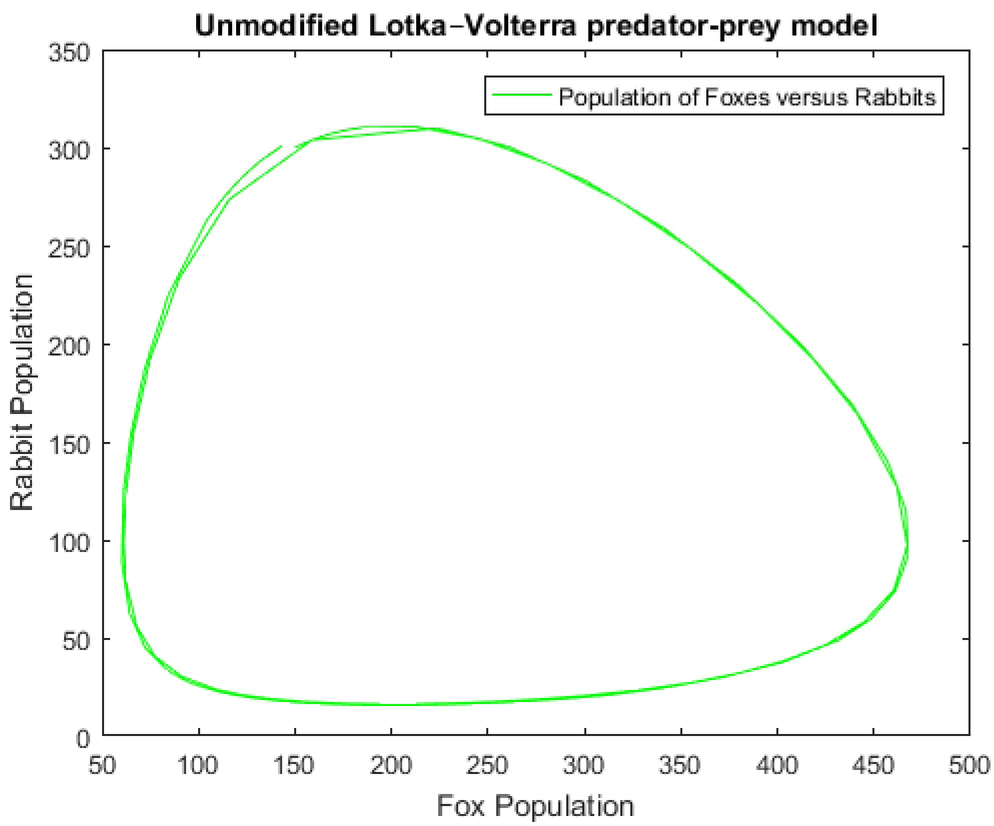
Problem #1 : Lotka–Volterra predator-prey model NCM 7.16

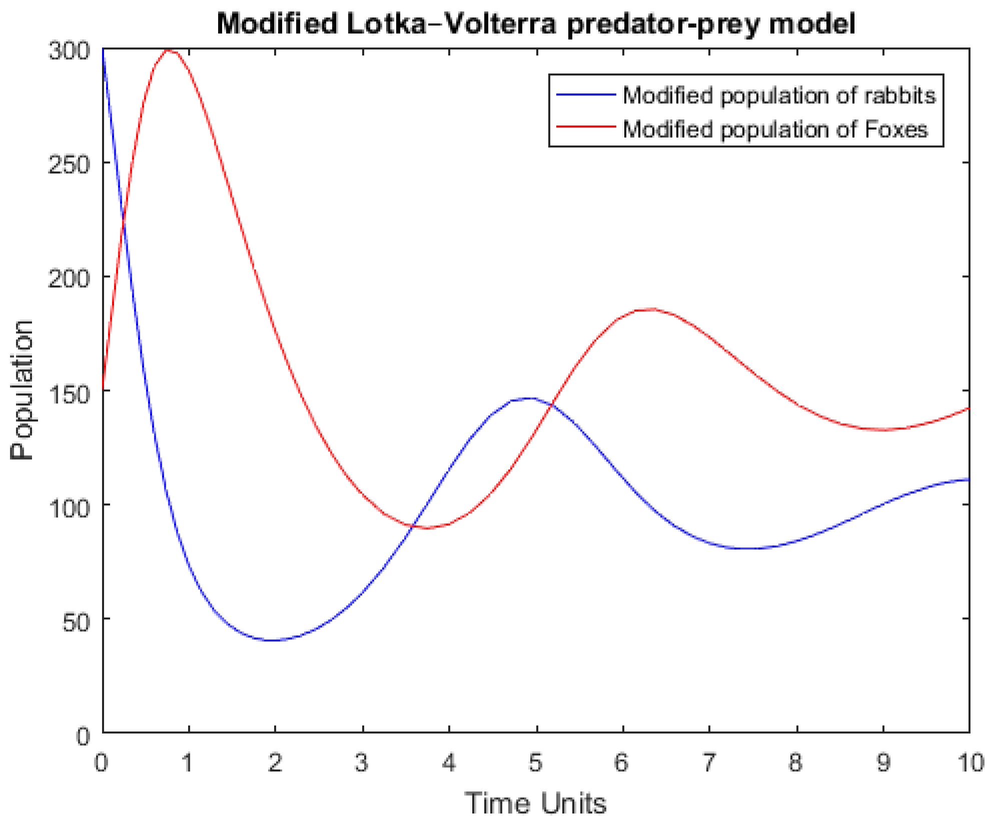
```
function prob1()

% For Nonmodified Predator versus prey model:
alpha = 0.01;

% Here we have our system where
% F(1) = dr/dt, Change in population of rabbits
% F(2) = df/dt, Change in population of foxes
% y(1) = population of rabbits and y(2) = population of foxes
F = @(t,y) [(2.*y(1))-alpha.*(y(1).*y(2)); alpha.*y(1).*y(2)-y(2)];
[t y] = ode45(F,[0 10],[300 150]);
% This is a plot of the population changes over time
figure
plot(t,y(:,1),'b',t,y(:,2),'r')
```

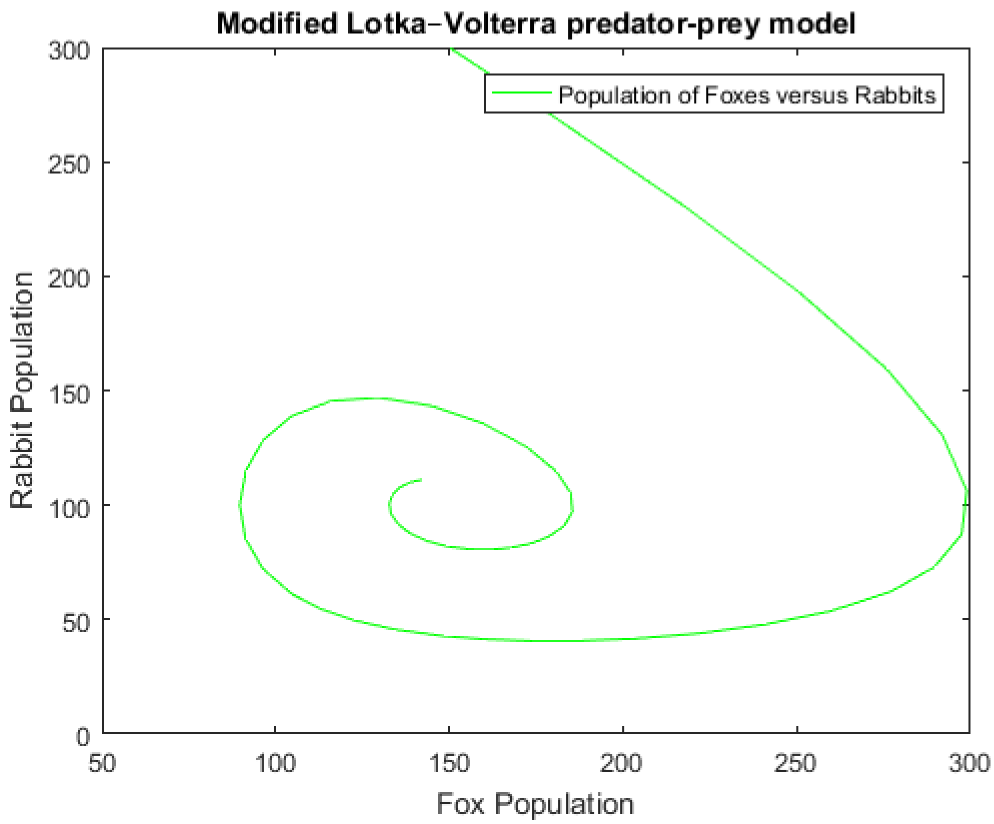
```
legend('Population of rabbits','Population of Foxes')
title('Unmodified Lotka-Volterra predator-prey model')
xlabel('Time Units')
ylabel('Population')

figure
plot(y(:,2),y(:,1),'g')
legend('Population of Foxes versus Rabbits')
title('Unmodified Lotka-Volterra predator-prey model')
xlabel('Fox Population')
ylabel('Rabbit Population')

% For Modified Predator versus prey model:
alpha = 0.01;
R = 400;

% Here we have our system where
% F(1) = dr/dt, Change in population of rabbits
% F(2) = df/dt, Change in population of foxes
% y(1) = population of rabbits and y(2) = population of foxes
F = @(t,y) [(2.*(1-(y(1)./R)).*y(1))-alpha.*(y(1).*y(2));
 alpha.*y(1).*y(2)-y(2)];
[t y] = ode45(F,[0 10],[300 150]);

% This is a plot of the population changes over time
figure
plot(t,y(:,1),'b',t,y(:,2),'r')
legend('Modified population of rabbits','Modified population of
 Foxes')
title('Modified Lotka-Volterra predator-prey model')
xlabel('Time Units')
ylabel('Population')

figure
plot(y(:,2),y(:,1),'g')
legend('Population of Foxes versus Rabbits')
title('Modified Lotka-Volterra predator-prey model')
xlabel('Fox Population')
ylabel('Rabbit Population')
end
```

prob1 : Success!


Unmodified Lotka−Volterra predator-prey model
Population of rabbits
Population of Foxes
Population
Time Units
0
50
100
150
200
250
300
350
400
450
500
1
2
3
4
5
6
7
8
9
10


Unmodified Lotka−Volterra predator-prey model
Population of Foxes versus Rabbits
Rabbit Population
Fox Population
0
50
100
150
200
250
300
350
400
450
500


Modified Lotka−Volterra predator-prey model
Modified population of rabbits
Modified population of Foxes
Population
Time Units
0
50
100
150
200
250
300
0
1
2
3
4
5
6
7
8
9
10


Modified Lotka−Volterra predator-prey model
Population of Foxes versus Rabbits
Rabbit Population
Fox Population
0
50
100
150
200
250
300
50
100
150
200
250
300

Problem #2 : Trajectory of a spherical cannon-ball NCM 7.18

```
function prob2()

% This problem uses several functions stored in external files.

% function f = cannonball0(t,u)

% % Declare constants.
% g = 9.81;
% m = 15;
% c = 0.02;
% p = 1.29;
% s = 0.25;

% % Return the derivative of the vector u.
% f = [u(3);u(4); -c*p*s/(2*m)*u(3)^2; -c*p*s/(2*m)*(u(4))^2-g];

% end

% function f = cannonballneg10(t,u)

% % Declare constants.
% g = 9.81;
% m = 15;
% c = 0.02;
% p = 1.29;
% s = 0.25;

% % Return the derivative of the vector u.
% f = [u(3);u(4); -c*p*s/(2*m)*(u(3)-(-10))^2; -c*p*s/(2*m)*(u(4))^2-
g];

% end

% function f = cannonballvar(t,u)

% % Declare constants.
% g = 9.81;
% m = 15;
% c = 0.02;
% p = 1.29;
% s = 0.25;

% % Set the wind to 10 if the time (rounded down) is even.
% if mod(floor(t),2) =  = 0
%     w = 10;
% else
%     w = 0;
% end
```

```
% % Return the derivative of the vector u.
% f = [u(3);u(4); -c*p*s/(2*m)*(u(3)-(w))^2; -c*p*s/(2*m)*(u(4))^2-g];

% end

% function f = cannonballrand(t,u)

% % Declare constants.
% g = 9.81;
% m = 15;
% c = 0.02;
% p = 1.29;
% s = 0.25;

% % Return the derivative of the vector u.
% f = [u(3);u(4); -c*p*s/(2*m)*(u(3)-(10*randn))^2; -c*p*s/
(2*m)*(u(4))^2-g];

% end

% Declare the initial speed, timespan, and options for the ODE solver.
v = 50;
t1 = linspace(0,10);
opt = odeset('RelTol',10e-2,'event',@ground);

% For our cannon ball without wind:
% I'd like to set different colors to use for legend:
colors =
 ['y-' 'm.' 'c-' 'r.' 'g-' 'b.' 'k-' 'y.' 'm-' 'c.' 'r-' 'g.' 'b-' 'k.' 'y*' 'mx'
+'];
labels = []; % empty matrix for angles
impactspeed = []; % Create empty matrix for my impact speeds to go
 into
flighttime = []; % Empty matrix for flight times
distance = []; % Empty matrix for downrange distances

% Using my for loop to loop over 17 values of theta.
for i = 1:17

 % Determine theta for this loop.
 theta = i*(pi/36);

 [t u] = ode45(@cannonball0,t1,[0 , 0 , v*cos(theta) ,
 v*sin(theta)],opt); % Calling Ode with initials

 % Calculate the final speed and flight time for this loop.
 impactspeed(i) = sqrt(u(end,3)^2+u(end,4)^2);
 flighttime(i) = t(end);

 % Add the data for this loop to the plot.
 distance(i) = u(end,1);
 plot(u(:,1),u(:,2),colors(i))
 axis([0,250,0,140])
 title('Graph of CannonBall without wind (theta in degrees)')
```

```
 ylabel('y position')
 xlabel('x position')
 legendInfo{i} = ['theta = ' num2str(i*(pi/36)*(180/pi))];
 labels{i} = ['theta(in degrees) = ' num2str(i*(pi/36)*(180/pi))];
 hold on
end

% Last, set my legend up:
legend(legendInfo)

% Report table of values:
impactspeed = impactspeed.';
xdistance = distance.';
flighttime = flighttime.';
InitialDegrees = labels.';
NoWind =
 table(impactspeed,xdistance ,flighttime,'RowNames',InitialDegrees)

figure % So we can start a new figure

% For our cannon ball with wind -10m/s:
labels = []; % empty matrix for angles
impactspeed = []; % Create empty matrix for my impact speeds to go
 into
flighttime = []; % Empty matrix for flight times
distance = []; % Empty matrix for downrange distances

% Using my for loop to loop over 17 values of theta.
for i = 1:17
 % Set theta for this loop
 theta = i*(pi/36);

 [t u] = ode45(@cannonballneg10,t1,[0 , 0 , v*cos(theta) ,
 v*sin(theta)],opt); % Calling Ode with initials

 % Calculate the final speed and time of flight for this loop.
 impactspeed(i) = sqrt(u(end,3)^2+u(end,4)^2);
 flighttime(i) = t(end);

 % Add the data for this loop to the plot
 distance(i) = u(end,1);
 plot(u(:,1),u(:,2),colors(i))
 axis([0,250,0,140])
 title('Graph of CannonBall with wind -10m/s (theta in degrees)')
 ylabel('y position')
 xlabel('x position')
 legendInfo{i} = ['theta = ' num2str(i*(pi/36)*(180/pi))];
 labels{i} = ['theta(in degrees) = ' num2str(i*(pi/36)*(180/pi))];
 hold on
end

% Last, set my legend up:
legend(legendInfo)
```

```
% Report table of values:
impactspeed = impactspeed.';
xdistance = distance.';
flighttime = flighttime.';
InitialDegrees = labels.';
Wind10 =
 table(impactspeed,xdistance ,flighttime,'RowNames',InitialDegrees)

figure % So we can start a new figure

% For our cannon ball with wind 10 m/s if the integer part of t is
 even, and zero otherwise:
labels = []; % empty matrix for angles
impactspeed = []; % Create empty matrix for my impact speeds to go
 into
flighttime = []; % Empty matrix for flight times
distance = []; % Empty matrix for downrange distances

% Using my for loop to loop over 17 values of theta.
for i = 1:17
 % Set theta for this loop
 theta = i*(pi/36);

 [t u] = ode45(@cannonballvar,t1,[0 , 0 , v*cos(theta) ,
 v*sin(theta)],opt); % Calling Ode with initials

 % Calculate the final speed and time of flight for this loop.
 impactspeed(i) = sqrt(u(end,3)^2+u(end,4)^2);
 flighttime(i) = t(end);

 % Add the data for this loop to the plot
 distance(i) = u(end,1);
 plot(u(:,1),u(:,2),colors(i))
 axis([0,250,0,140])
 title('Graph of CannonBall with Variable Wind (0 or 10m/s)(theta in
 degrees)')
 ylabel('y position')
 xlabel('x position')
 legendInfo{i} = ['theta = ' num2str(i*(pi/36)*(180/pi))];
 labels{i} = ['theta(in degrees) = ' num2str(i*(pi/36)*(180/pi))];
 hold on
end

% Last, set my legend up:
legend(legendInfo)

% Report table of values:
impactspeed = impactspeed.';
xdistance = distance.';
flighttime = flighttime.';
InitialDegrees = labels.';
VariableWind =
 table(impactspeed,xdistance ,flighttime,'RowNames',InitialDegrees)
```

```
figure % So we can start a new figure

% For our cannon ball with wind 10 m/s*randn:
labels = []; % empty matrix for angles
impactspeed = []; % Create empty matrix for my impact speeds to go
 into
flighttime = []; % Empty matrix for flight times
distance = []; % Empty matrix for downrange distances

% Using my for loop to loop over 17 values of theta.
for i = 1:17
 % Set theta for this loop
 theta = i*(pi/36);

 [t u] = ode45(@cannonballrand,t1,[0 , 0 , v*cos(theta) ,
 v*sin(theta)],opt); % Calling Ode with initials

 % Calculate the final speed and time of flight for this loop.
 impactspeed(i) = sqrt(u(end,3)^2+u(end,4)^2);
 flighttime(i) = t(end);

 % Add the data for this loop to the plot
 distance(i) = u(end,1);
 plot(u(:,1),u(:,2),colors(i))
 axis([0,250,0,140])
 title('Graph of CannonBall with Random Wind(theta in degrees)')
 ylabel('y position')
 xlabel('x position')
 legendInfo{i} = ['theta = ' num2str(i*(pi/36)*(180/pi))];
 labels{i} = ['theta(in degrees) = ' num2str(i*(pi/36)*(180/pi))];
 hold on
end

% Last, set my legend up:
legend(legendInfo)

% Report table of values:
impactspeed = impactspeed.';
xdistance = distance.';
flighttime = flighttime.';
InitialDegrees = labels.';
RandomWind =
 table(impactspeed,xdistance ,flighttime,'RowNames',InitialDegrees)

end
```

NoWind =

	impactspeed	*xdistance*	*flighttime*
	___________	_________	__________
theta(in degrees) = 5	*49.532*	*44.038*	*0.88831*
theta(in degrees) = 10	*49.109*	*86.307*	*1.7691*

theta(in degrees) = 15	48.759	125.56	2.6351
theta(in degrees) = 20	48.504	160.65	3.479
theta(in degrees) = 25	48.357	190.63	4.2941
theta(in degrees) = 30	48.318	214.67	5.0738
theta(in degrees) = 35	48.38	232.16	5.8121
theta(in degrees) = 40	48.525	242.66	6.5036
theta(in degrees) = 45	48.729	245.93	7.1433
theta(in degrees) = 50	48.969	241.93	7.7267
theta(in degrees) = 55	49.217	230.78	8.25
theta(in degrees) = 60	49.45	212.81	8.71
theta(in degrees) = 65	49.651	188.5	9.1036
theta(in degrees) = 70	49.807	158.51	9.4288
theta(in degrees) = 75	49.913	123.66	9.6835
theta(in degrees) = 80	49.973	84.885	9.8665
theta(in degrees) = 85	49.996	43.274	9.9766

Wind10 =

	impactspeed	xdistance	flighttime
theta(in degrees) = 5	49.327	43.946	0.88831
theta(in degrees) = 10	48.715	85.95	1.7691
theta(in degrees) = 15	48.202	124.78	2.6351
theta(in degrees) = 20	47.818	159.35	3.479
theta(in degrees) = 25	47.578	188.72	4.2941
theta(in degrees) = 30	47.486	212.13	5.0738
theta(in degrees) = 35	47.533	229	5.8121
theta(in degrees) = 40	47.699	238.95	6.5036
theta(in degrees) = 45	47.957	241.77	7.1433
theta(in degrees) = 50	48.274	237.44	7.7267
theta(in degrees) = 55	48.618	226.13	8.25
theta(in degrees) = 60	48.959	208.17	8.71
theta(in degrees) = 65	49.271	184.07	9.1036
theta(in degrees) = 70	49.535	154.47	9.4288
theta(in degrees) = 75	49.74	120.17	9.6835
theta(in degrees) = 80	49.881	82.106	9.8665
theta(in degrees) = 85	49.964	41.315	9.9766

VariableWind =

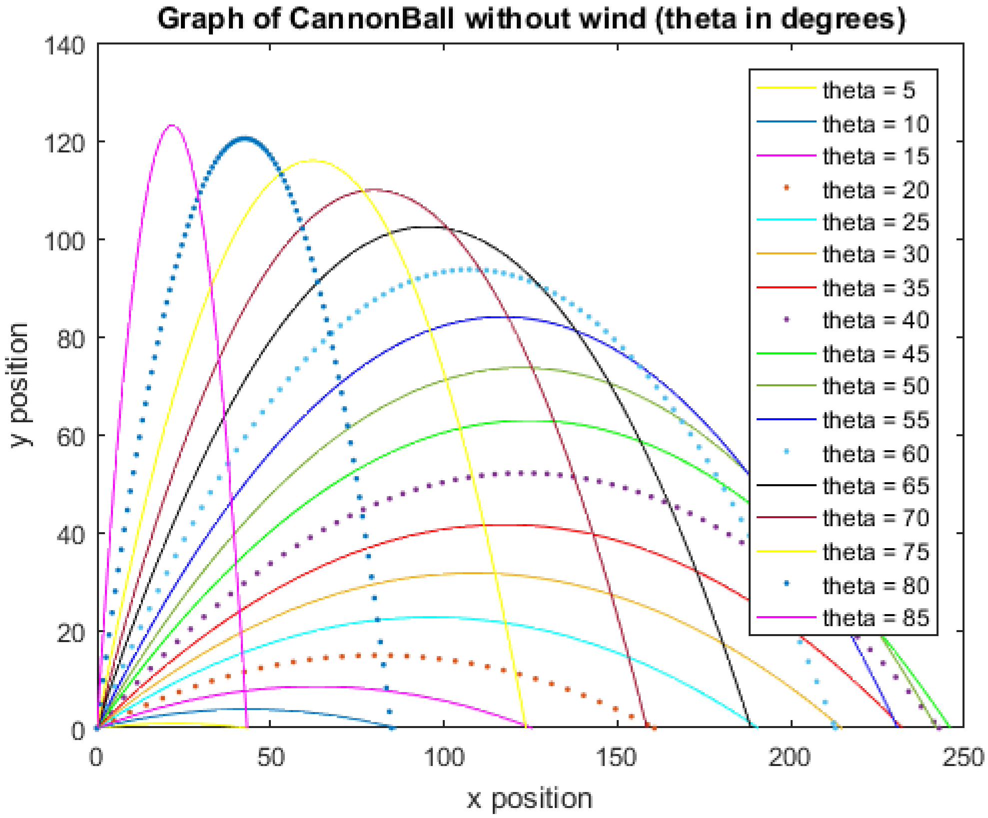
	impactspeed	xdistance	flighttime
theta(in degrees) = 5	49.682	44.113	0.88831
theta(in degrees) = 10	49.255	86.511	1.7691
theta(in degrees) = 15	49.046	125.94	2.6351
theta(in degrees) = 20	48.789	161.29	3.479
theta(in degrees) = 25	48.694	191.5	4.2941
theta(in degrees) = 30	48.694	215.83	5.0738
theta(in degrees) = 35	48.71	233.56	5.8121
theta(in degrees) = 40	48.876	244.26	6.5036

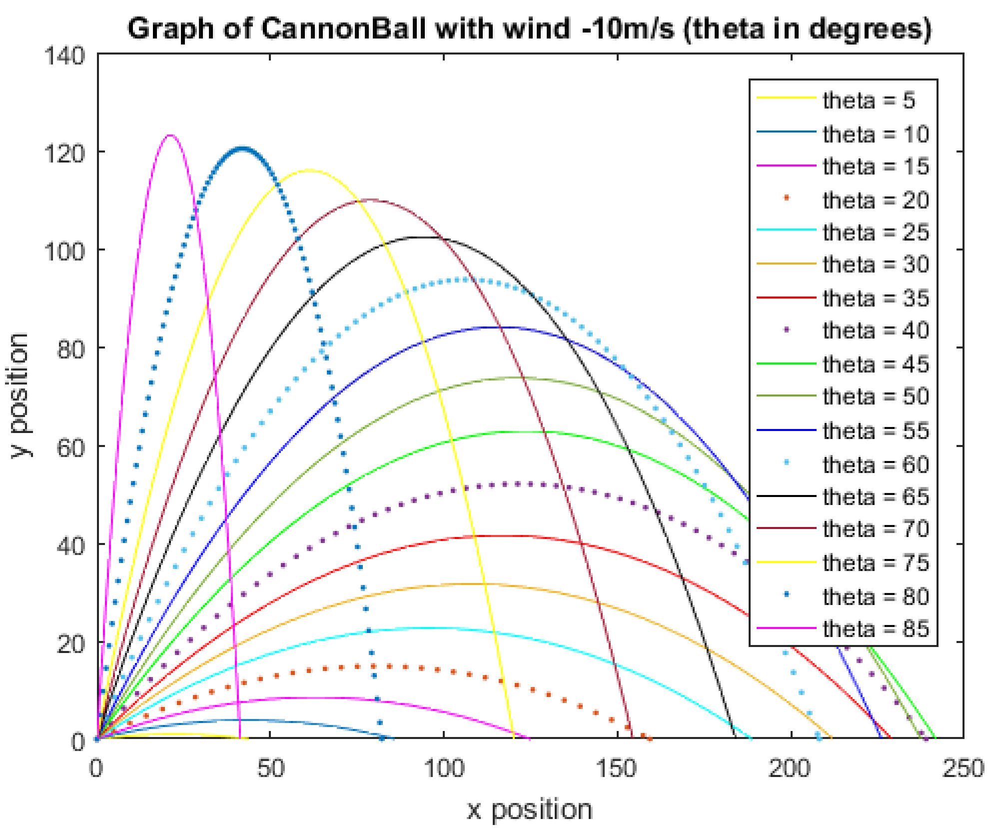
theta(in degrees) = 45	*49.056*	*247.7*	*7.1433*
theta(in degrees) = 50	*49.228*	*243.75*	*7.7267*
theta(in degrees) = 55	*49.441*	*232.57*	*8.25*
theta(in degrees) = 60	*49.636*	*214.49*	*8.71*
theta(in degrees) = 65	*49.781*	*189.98*	*9.1036*
theta(in degrees) = 70	*49.886*	*159.71*	*9.4288*
theta(in degrees) = 75	*49.953*	*124.49*	*9.6835*
theta(in degrees) = 80	*49.986*	*85.286*	*9.8665*
theta(in degrees) = 85	*49.995*	*43.2*	*9.9766*

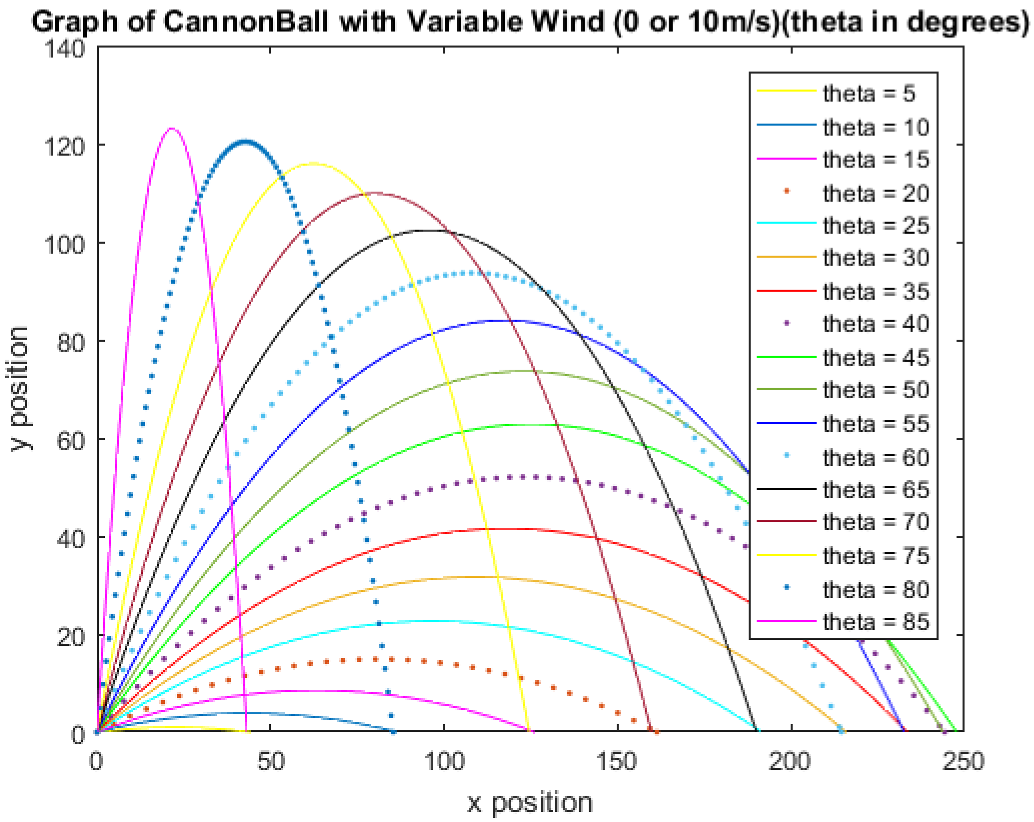
RandomWind =

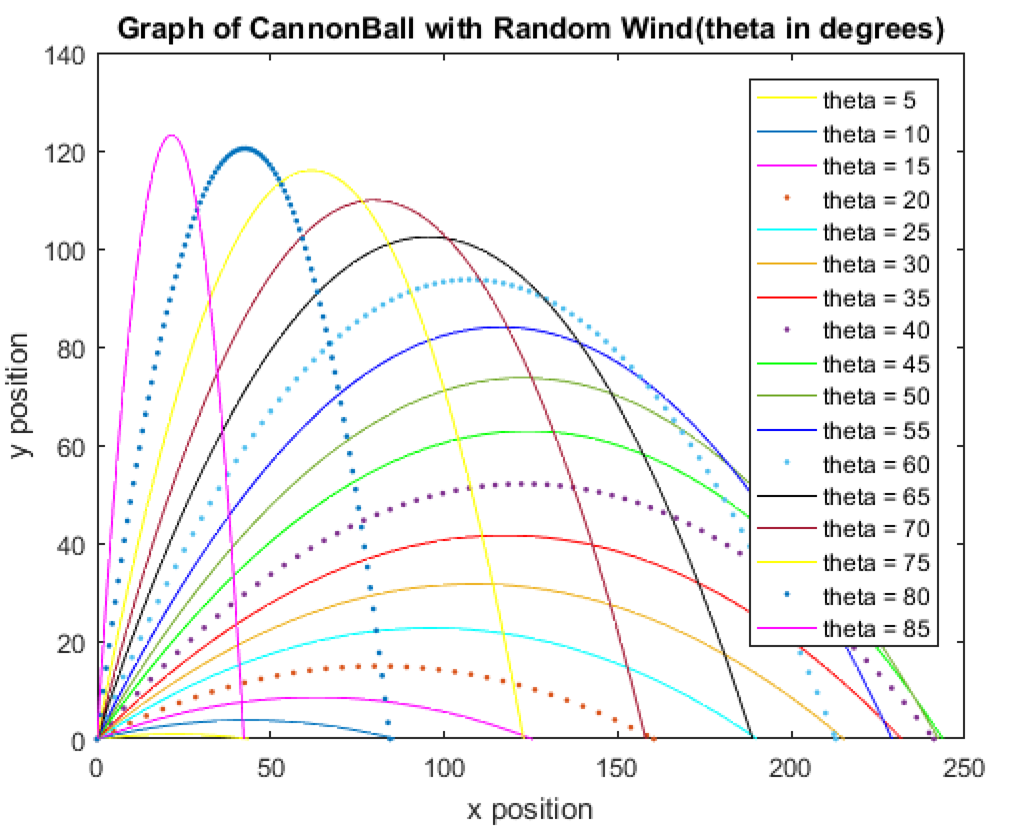
	impactspeed	*xdistance*	*flighttime*
theta(in degrees) = 5	*49.55*	*44.056*	*0.88831*
theta(in degrees) = 10	*49.034*	*86.108*	*1.7691*
theta(in degrees) = 15	*48.556*	*125.33*	*2.6351*
theta(in degrees) = 20	*48.629*	*160.4*	*3.479*
theta(in degrees) = 25	*47.91*	*190.17*	*4.2941*
theta(in degrees) = 30	*48.5*	*215.18*	*5.0738*
theta(in degrees) = 35	*48.268*	*231.76*	*5.8121*
theta(in degrees) = 40	*47.963*	*241.07*	*6.5036*
theta(in degrees) = 45	*48.432*	*243.69*	*7.1433*
theta(in degrees) = 50	*48.886*	*242.34*	*7.7267*
theta(in degrees) = 55	*49.037*	*229.01*	*8.25*
theta(in degrees) = 60	*49.379*	*212.79*	*8.71*
theta(in degrees) = 65	*49.65*	*188.81*	*9.1036*
theta(in degrees) = 70	*49.739*	*158.06*	*9.4288*
theta(in degrees) = 75	*49.879*	*122.86*	*9.6835*
theta(in degrees) = 80	*49.953*	*84.411*	*9.8665*
theta(in degrees) = 85	*49.979*	*42.307*	*9.9766*

prob2 : Success!


Graph of CannonBall without wind (theta in degrees)
y position
x position
0
50
100
150
200
250
0
20
40
60
80
100
120
140
theta = 5
theta = 10
theta = 15
theta = 20
theta = 25
theta = 30
theta = 35
theta = 40
theta = 45
theta = 50
theta = 55
theta = 60
theta = 65
theta = 70
theta = 75
theta = 80
theta = 85


Graph of CannonBall with wind -10m/s (theta in degrees)
y position
x position
0
50
100
150
200
250
0
20
40
60
80
100
120
140
theta = 5
theta = 10
theta = 15
theta = 20
theta = 25
theta = 30
theta = 35
theta = 40
theta = 45
theta = 50
theta = 55
theta = 60
theta = 65
theta = 70
theta = 75
theta = 80
theta = 85


Graph of CannonBall with Variable Wind (0 or 10m/s)(theta in degrees)
y position
x position
0
20
40
60
80
100
120
140
0
50
100
150
200
250
theta = 5
theta = 10
theta = 15
theta = 20
theta = 25
theta = 30
theta = 35
theta = 40
theta = 45
theta = 50
theta = 55
theta = 60
theta = 65
theta = 70
theta = 75
theta = 80
theta = 85

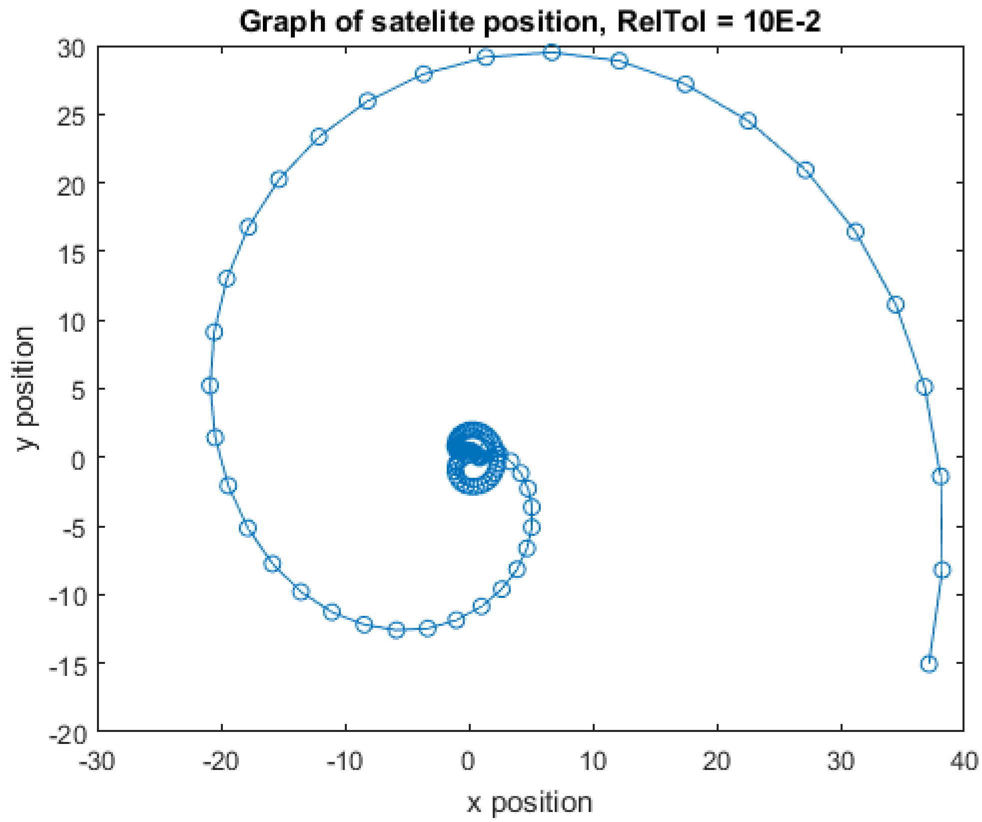

Graph of CannonBall with Random Wind(theta in degrees)
y position
x position
0
20
40
60
80
100
120
140
0
50
100
150
200
250
theta = 5
theta = 10
theta = 15
theta = 20
theta = 25
theta = 30
theta = 35
theta = 40
theta = 45
theta = 50
theta = 55
theta = 60
theta = 65
theta = 70
theta = 75
theta = 80
theta = 85

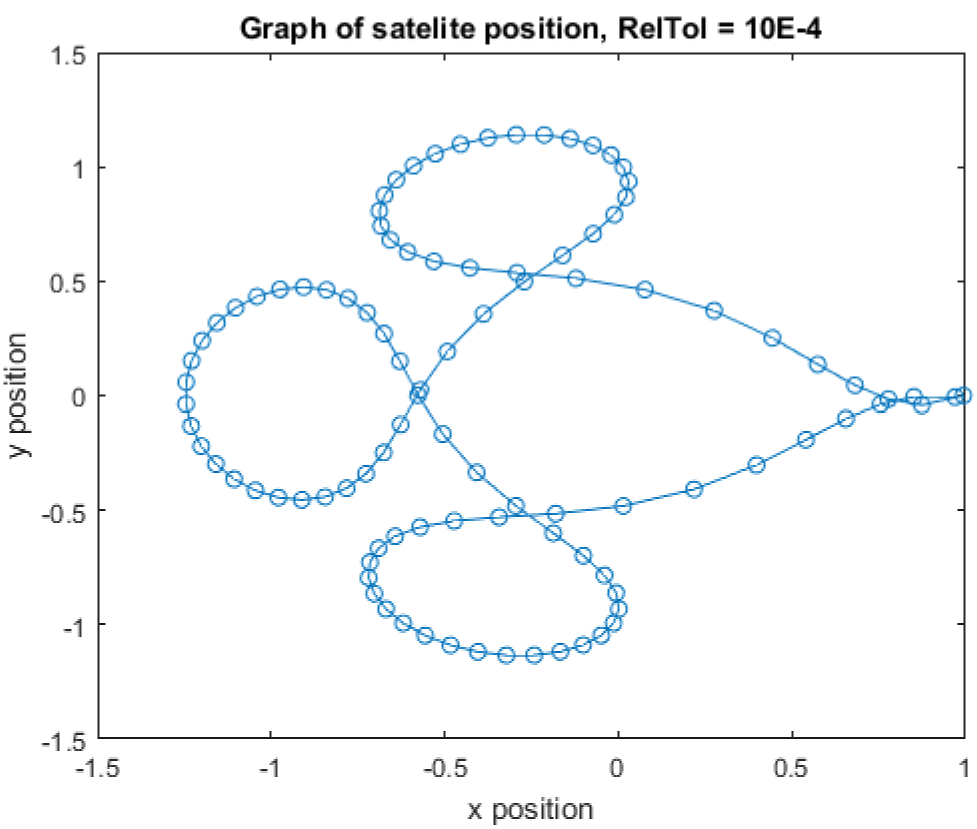
Problem #3 : Satellite Orbit

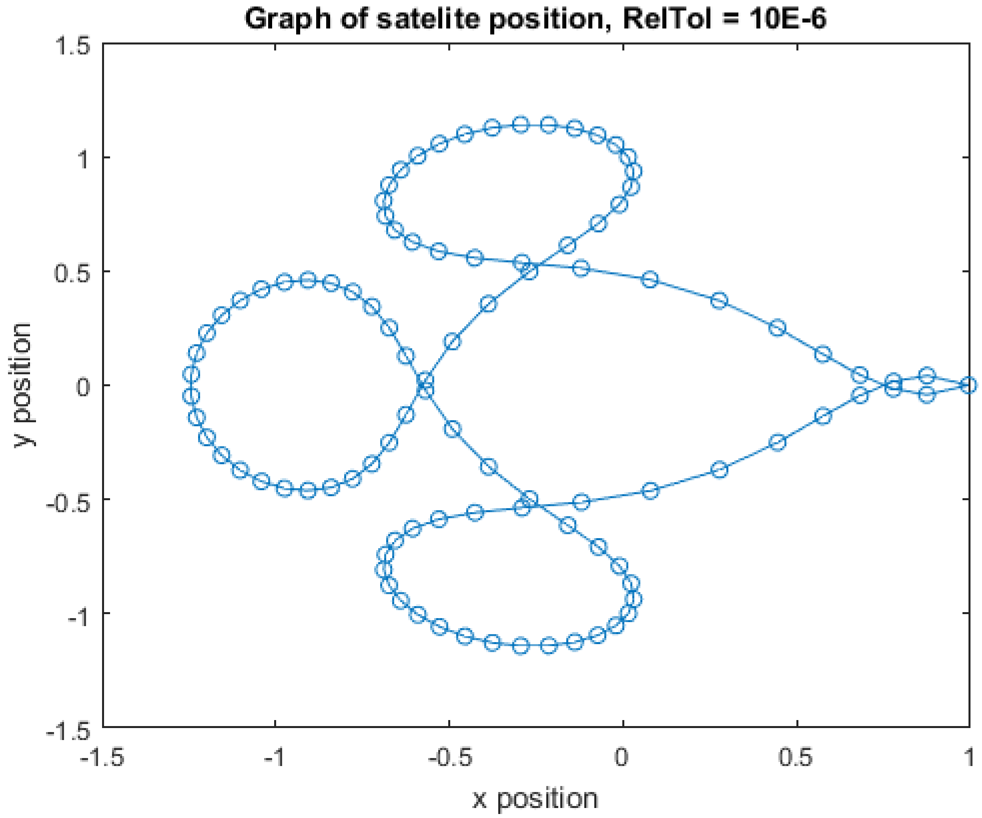
```
function prob3()

% Part a

% The following function is called from an external file.
% function f = Orbit(t,u)

% mu = 0.012277471;
% mus = 1 - mu;

% D1 = ((u(1)+mu)^2 + u(2)^2)^(3/2);
% D2 = ((u(1)-mus)^2 + u(2)^2)^(3/2);

% f = [u(3); u(4); u(1)+2*u(4)-mus*(u(1)+mu)/D1-mu*(u(1)-mus)/D2;
 u(2)-2*u(3)-mus*u(2)/D1-mu*u(2)/D2];

% end

% Generate a list of time points for one full cycle.
b = 17.06521656015796;
t = linspace(0,b);

% Initial conditions
u0 = [0.994,0,0,-2.0015851063790825];

% Call the ODE solver using a given relative tolerance and output a
 phase diagram.
figure
opt = odeset('RelTol',10e-2,'OutputFcn',@odephas2);
[t1,u1] = ode45(@Orbit,t,u0,opt);
title('Graph of satelite position, RelTol = 10E-2')
ylabel('y position')
xlabel('x position')

figure
opt = odeset('RelTol',10e-4,'OutputFcn',@odephas2);
[t2,u2] = ode45(@Orbit,t,u0,opt);
title('Graph of satelite position, RelTol = 10E-4')
ylabel('y position')
xlabel('x position')

figure
opt = odeset('RelTol',10e-6,'OutputFcn',@odephas2);
[t3,u3] = ode45(@Orbit,t,u0,opt);
title('Graph of satelite position, RelTol = 10E-6')
ylabel('y position')
xlabel('x position')

% Part b

% Run the ODE solver for 3 periods using the same relative tolerance.
```

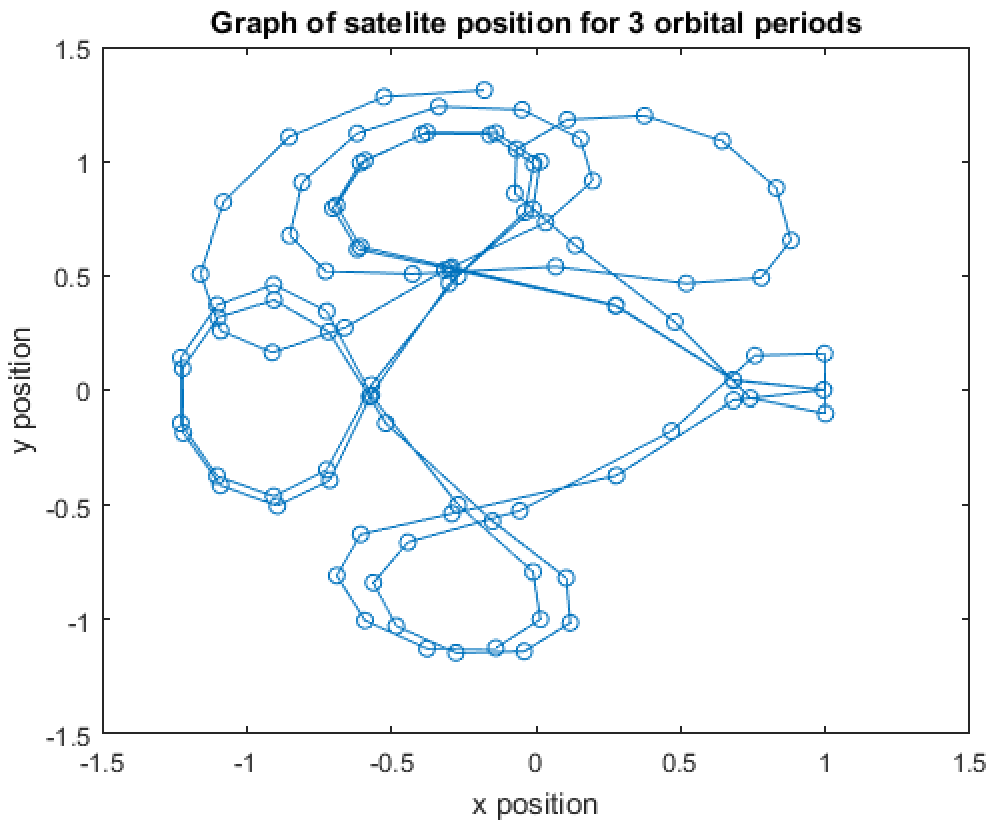
```
figure
[t4,u4] = ode45(@Orbit,3*t,u0,opt);
title('Graph of satelite position for 3 orbital periods')
ylabel('y position')
xlabel('x position')

% These results show the satelite to precess in its orbit, which is
 not the actual behavior.
% Changing this to a larger tolerance causes the orbit to be less
 table.
% Even the smallest possible tolerance (~2E-14) does not give accurate
 results over three periods.

% Part c

% Run the 113 ODE solver.
figure
[t5,u5] = ode113(@Orbit,3*t,u0,opt);
title('Graph of satelite position using ODE113 solver')
ylabel('y position')
xlabel('x position')

% These results are much more accurate, though there is still
 innacuracy in the last portion of the third orbit.

end
```

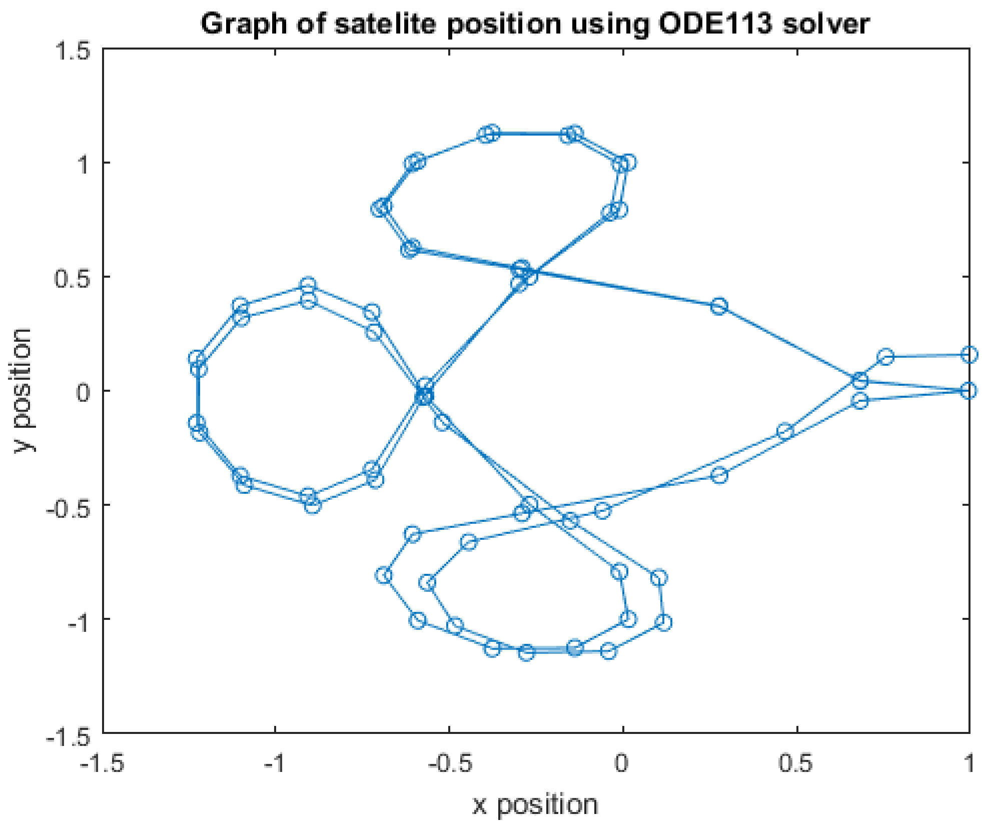
prob3 : Success!


Graph of satelite position, RelTol = 10E-2
30
25
20
15
10
5
0
-5
-10
-15
-20
-30
-20
-10
0
10
20
30
40
y position
x position


Graph of satelite position, RelTol = 10E-4
1.5
1
0.5
0
-0.5
-1
-1.5
-1.5
-1
-0.5
0
0.5
1
y position
x position


Graph of satelite position, RelTol = 10E-6
y position
x position
1.5
1
0.5
0
-0.5
-1
-1.5
-1.5
-1
-0.5
0
0.5
1


Graph of satelite position for 3 orbital periods
y position
x position
1.5
1
0.5
0
-0.5
-1
-1.5
-1.5
-1
-0.5
0
0.5
1
1.5



Published with MATLAB® R2016a