1a. Solving:

Substituting for r2 :

Solve for r: r = ±

The solution takes the form:

Solve for C1 and C2 using x(0) = x0 and = v0:

Substituting the constants back in and simplifying gives the solution:

1b. The system is:

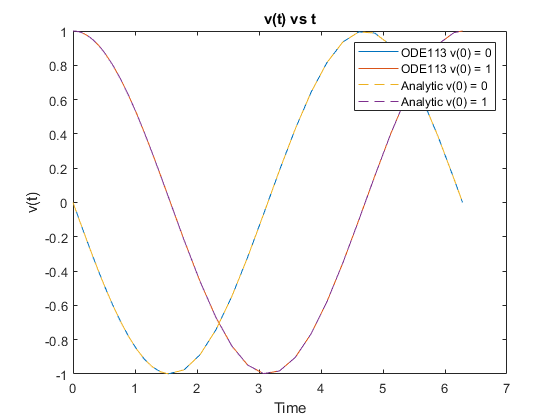
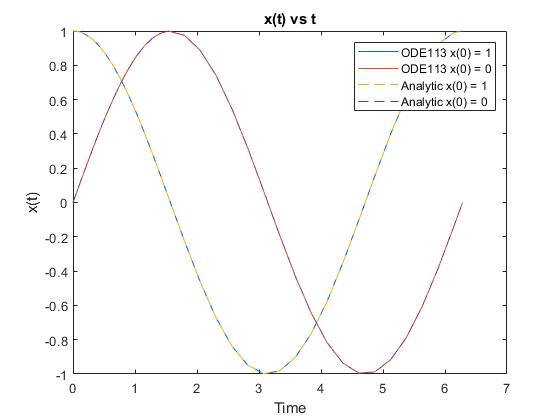
1c.

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

1d.



1 Code:

%% Defines all of the variables used in the differential equation

clc;clear;close all;

omegan = 1;

trange = [0 2\*pi/omegan];

conditions = [1 0 0 1];

options = odeset('RelTol',1e-8);

%% Loops through both sets of inital conditions and gets a solution and plots it

for i=1:2:length(conditions)

po = [conditions(i) conditions(i+1)];

[t p] = ode113(@harmonicOscillator,trange,po,options,omegan);

x(:,i) = p(:,1);

x(:,i+1) = p(:,2);

end

%% Creates Plots for part c

figure

plot(t,x(:,1),t,x(:,3))

xlabel('Time')

ylabel('x(t)')

title("x(t) vs t")

legend('x(0) = 1','x(0) = 0')

figure

plot(t,x(:,2),t,x(:,4))

xlabel('Time')

ylabel('v(t)')

title("v(0) vs t")

legend('v(0) = 0','v(0) = 1')

%% Defines analytic solution and its derivative

analytic = @(t,x0,omegan) x0+2\*(sin(omegan\*t).^2);

danalytic= @(t,x0,omegan) 4\*omegan\*x0.\*cos(omegan\*t).\*sin(omegan\*t);

%% Plots the analytic solution and the matlab solution

%the postion plot

figure

plot(t,x(:,1),t,x(:,3),t,analytic(t,conditions(1),omegan),t,analytic(t,conditions(3),omegan))

xlabel('Time')

ylabel('x(t)')

title("x(t) vs t")

legend('ODE113 x(0) = 1','ODE113 x(0) = 0','Analytic x(0) = 1','Analytic x(0) = 0')

%the velocity plot

figure

plot(t,x(:,2),t,x(:,4),t,danalytic(t,conditions(1),omegan),t,danalytic(t,conditions(3),omegan))

xlabel('Time')

ylabel('v(t)')

title("v(t) vs t")

legend('ODE113 v(0) = 0','ODE113 v(0) = 1','Analytic v(0) = 0','Analytic v(0) = 1')

%% Funtion to turn the second order ODE into first order system

function pdot = harmonicOscillator(t,p,omegan)

x1=p(1);

x2=p(2);

pdot=zeros(size(p));

x1dot=x2;

x2dot=-omegan^2\*x1;

pdot(1) = x1dot;

pdot(2) = x2dot;

end

2. Proving that:

First define these derivatives:

and

We can substitute these into the original equation:

The dx on the right-side cancel simplifying to:

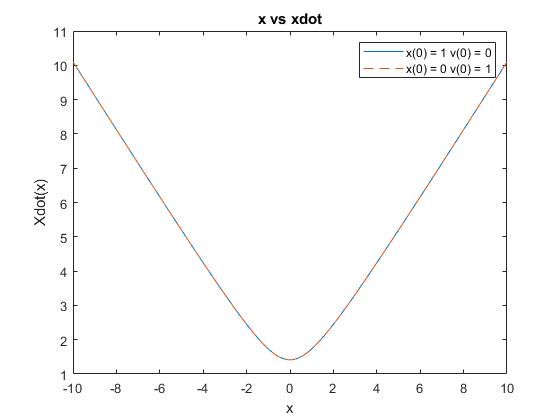
Substituting into the original equation:

Subtracting and multiplying:

Integrating both sides:

Solving for C using initial conditions, x(0) = and = v0

Substituting back in for C and solving for results in the equation:



%% Defines variables and range for x values

clc;clear;close all;

omegan = 1;

conditions = [1 0 0 1];

x = -10:0.01:10;

%% defines the function

xdot = @(x,x0,v0,omegan) sqrt((omegan)^2\*x.^2+2\*v0+2\*omegan\*x0);

%% Computes the equation for each set of initial conditions and x

for i=1:2:length(conditions)

x0 = conditions(i);

v0 = conditions(i+1);

v(:,i) = xdot(x,x0,v0,omegan);

end

%% plots the function for both sets of initial conditions over x

plot((x),(v(:,1)),x,v(:,3),'--')

xlabel('x')

ylabel('Xdot(x)')

title("x vs xdot")

legend('x(0) = 1 v(0) = 0','x(0) = 0 v(0) = 1')

3.

--------Newton's Method------

Initial Guess: 0.000000

0.000000 6.613186

6.613186 -1.668447

-1.668447 1.945357

1.945357 3.045255

3.045255 2.885278

2.885278 2.884565

2.884565 2.884565

2.884565 2.884565

2.884565 2.884565

2.884565 2.884565

Converges to 2.884565

Initial Guess: 1.570796

1.570796 3.321865

3.321865 2.883341

2.883341 2.884565

2.884565 2.884565

2.884565 2.884565

2.884565 2.884565

2.884565 2.884565

2.884565 2.884565

2.884565 2.884565

2.884565 2.884565

Converges to 2.884565

Initial Guess: 3.141593

3.141593 2.885608

2.885608 2.884565

2.884565 2.884565

2.884565 2.884565

2.884565 2.884565

2.884565 2.884565

2.884565 2.884565

2.884565 2.884565

2.884565 2.884565

2.884565 2.884565

Converges to 2.884565

Initial Guess: 4.712389

4.712389 2.149154

2.149154 2.964998

2.964998 2.884808

2.884808 2.884565

2.884565 2.884565

2.884565 2.884565

2.884565 2.884565

2.884565 2.884565

2.884565 2.884565

2.884565 2.884565

Converges to 2.884565

---------Fixed Point-----------

Initial Guess: 0.000000

0.000000 2.735510

2.735510 2.967128

2.967128 2.837290

2.837290 2.911198

2.911198 2.869411

2.869411 2.893142

2.893142 2.879696

2.879696 2.887325

2.887325 2.883000

2.883000 2.885453

Converges to 2.885453

Initial Guess: 1.570796

1.570796 3.321865

3.321865 2.630378

2.630378 3.022376

3.022376 2.805247

2.805247 2.929030

2.929030 2.859210

2.859210 2.898894

2.898894 2.876424

2.876424 2.889177

2.889177 2.881948

Converges to 2.881948

Initial Guess: 3.141593

3.141593 2.735510

2.735510 2.967128

2.967128 2.837290

2.837290 2.911198

2.911198 2.869411

2.869411 2.893142

2.893142 2.879696

2.879696 2.887325

2.887325 2.883000

2.883000 2.885453

Converges to 2.885453

Initial Guess: 4.712389

4.712389 2.149154

2.149154 3.226501

3.226501 2.685783

2.685783 2.993617

2.993617 2.821960

2.821960 2.919753

2.919753 2.864522

2.864522 2.895901

2.895901 2.878127

2.878127 2.888213

Converges to 2.888213

%% Defines variables used in the equaions and the four initial guesses

clc;clear;

a = 0.5863552428728520;

C = -2.735509517401657;

x0 = [0 pi/2 pi 3\*pi/2];

%% Defines the three functions used, orginal, derivative, and solved for x

f = @(x) x-a\*sin(x)+C;

g = @(x) 1-a\*cos(x);

b = @(x) a\*sin(x)-C;

fprintf('--------Newton''s Method---------- \n');

%% loops through initial conditions and applies newton's method for 10 iterations

for j=1:length(x0)

guess = x0(j);

fprintf('Initial Guess: %f\n',guess);

for i=1:10

nextg = guess - (f(guess)/g(guess));

fprintf('%f %f \n',guess,nextg);

guess = nextg;

end

fprintf('\n Converges to %f \n\n\n',guess);

end

%% loops through initial conditions and applies fixed point method for 10 iterations

fprintf('---------Fixed Point----------- \n');

for j=1:length(x0)

guess = x0(j);

fprintf('Initial Guess: %f\n',guess);

for i=1:10

nextg = b(guess);

fprintf('%f %f \n',guess,nextg);

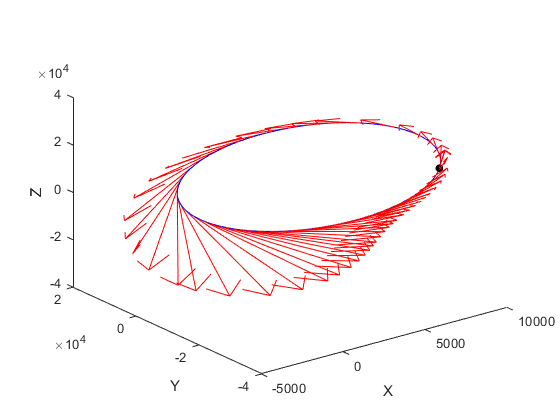
guess = nextg;

end

fprintf('\nConverges to %f \n\n\n',guess);

end

4.



%% Defines variables and initial conditions

clc;clear;close all;

mu = 398600;

trange = [0 38032];

xconditions = [6997.56; -34108; 20765.49];

vconditions = [0.1559; 0.25517; 1.80763];

po = [xconditions vconditions];

options = odeset('RelTol',1e-8);

%% Solves and plots the solution to the ODE, Includes velocity vectors and point at t = 0

[t p] = ode113(@twoBodyOde,trange,po,options,mu);

figure

plot3(p(:,1),p(:,2),p(:,3),'b')

hold on

plot3(p(1,1),p(1,2),p(1,3),'.k','MarkerSize',20)

hold on

quiver3(p(1:3:153,1),p(1:3:153,2),p(1:3:153,3),p(1:3:153,4),p(1:3:153,5),p(1:3:153,6),4,'r')

xlabel('X')

ylabel('Y')

zlabel('Z')

%% Function which defines the ODE

function pdot = twoBodyOde(t,p,mu)

pos = p(1:3);

vel = p(4:6);

rad = norm(pos,2);

posdot = vel;

veldot = -mu/(rad^3)\*pos;

pdot = [posdot; veldot];

end

5a. r is the distance from O to P. The direction, **e**r, is in the direction of P from O. Therefore, the position of P relative to O can be expressed as r **e**r

5b. To take the derivative of r **e**r in reference frame I, the transport theorem must be used since the vector is expressed in a different reference frame.

Computing the cross product and simplifying results in

5c. Using the substitutions in the previous equation:

and

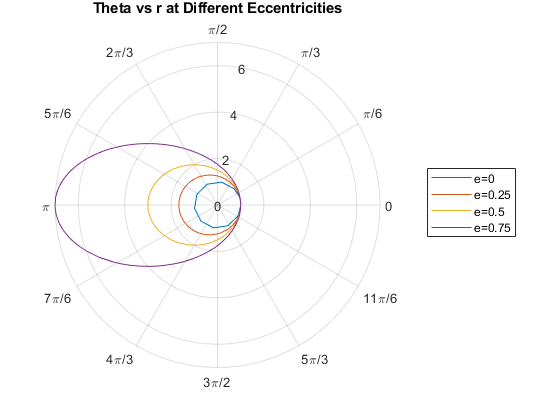
5d. Differentiating to find acceleration of P in I using the transport theorem results in:

5e. Using Newton’s second law:

Separating each vector:

5f. The final system is:

5g-f.



%% Defines the initial conditions and variables needed

clc;clear;close all;

mu = 1;

options = odeset('RelTol',1e-10);

e = [0 1/4 1/2 3/4];

conditions = [1 0 0];

j=1;

%% Loops through all values of e. Calculates the new vtheta0 each time. Solves the ODE and stores the results in a cell array

for i=1:length(e)

r0 = conditions(1);

theta0=conditions(2);

vr0=conditions(3);

eval = e(i);

pval=r0\*(1+eval);

ra = pval/(1-eval);

a = (r0+ra)/2;

vtheta0= sqrt(mu\*pval)/r0;

trange=[0 2\*pi\*sqrt(a^3/mu)];

p0 = [r0 theta0 vr0 vtheta0];

[t p] = ode113(@twoBody,trange,p0,options,mu);

z{j}=p(:,1);

z{j+1}=p(:,2);

j=j+2;

end

%% Plots theta vs r and adds labels

polarplot(z{2},z{1},z{4},z{3},z{6},z{5},z{8},z{7})

legend('e=0','e=0.25','e=0.5','e=0.75')

polaraxis = gca;

polaraxis.ThetaAxisUnits = 'radians';

title('Theta vs r at Different Eccentricities')

%% the function which defines the ODE

function pdot = twoBody(t,p,mu)

r = p(1);

theta = p(2);

vr = p(3);

vtheta = p(4);

rdot = vr;

thetadot = vtheta/r;

vrdot = (vtheta^2)/r-mu/(r^2);

vthetadot = -vr\*vtheta/r;

pdot=zeros(size(p));

pdot(1)=rdot;

pdot(2)=thetadot;

pdot(3)=vrdot;

pdot(4)=vthetadot;

end