The Double Pendulum

Muhammad Shatla

May 14, 2018

Contents

1 Introduction

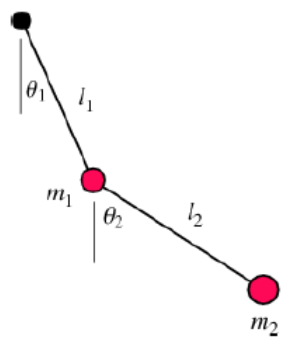
Dynamical Systems that exhibit chaotic behavior have gained interest of scientists at the end of 19th century and the beginning of 20th century. these systems are extremely sensitive to initial conditions such that we cannot completely predict their behavior. The double pendulum system is considered a chaotic system despite of its simple structure; For small angles, a pendulum behaves like a linear system. When the angles are small in the Double Pendulum, the system behaves like the linear Double Spring; For large angles, the pendulum is non-linear, and the phase graph becomes much more complex. in this article, we are going to develop the equations of motion for the double pendulum based on the methods of Calculus of variations and solve these differential equations numerically using MATLAB. The numerical solution of the equations is based on Rung-Kutta method that will give us a sufficient approximation for the actual behavior of the double pendulum.

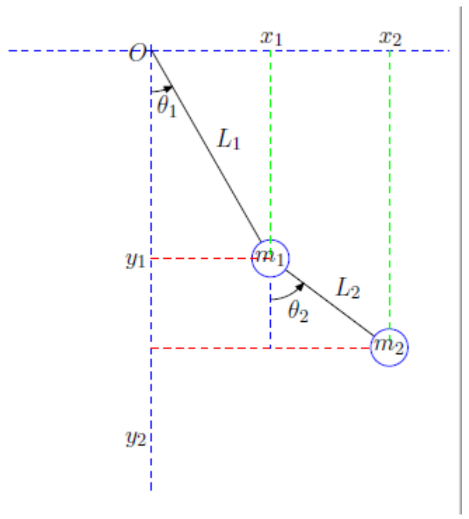
2 The Lagrangian ($\mathcal{L}$) of the Double Pendulum

In Analytical Mechanics the Lagrangian $\mathcal{L}$ is defined as

$$\mathcal{L} \equiv T - V$$

Where T is the kinetic energy and V is the potential energy. In order to set the lagrangian of the double pendulum system, we will use the two angles θ_1 and θ_2 as the generalized coordinates (Fig.1), since the system has two degrees of freedom.





The positions of the two bobs in cartesian coordinates (Fig.2) are given by:

$$x_1 = L_1 \sin\theta_1 \tag{1}$$
$$y_1 = -L_1 \cos\theta_1 \tag{2}$$
$$x_2 = L_1 \sin\theta_1 + L_2 \sin\theta_2 \tag{3}$$
$$y_2 = -L_1 \cos\theta_1 - L_2 \cos\theta_2 \tag{4}$$

The Potential Energy of the system:

$$V = m_1 g y_1 + m_2 g y_2 \tag{5}$$
$$= -m_1 g L_1 \cos\theta_1 - m_2 g \left(L_1 \cos\theta_1 + L_2 \cos\theta_2\right) \tag{6}$$

The Kinetic Energy:

$$T = \frac{1}{2} m_1 \left(\dot{x_1}^2 + \dot{y_1}^2\right) + \frac{1}{2} m_2 \left(\dot{x_2}^2 + \dot{y_2}^2\right) \tag{7}$$

Differentiating Equations (1), (2), (3) and (4):

$$\dot{x_1} = L_1 \cos\theta_1 \dot{\theta_1} \tag{8}$$

$$\dot{x_2} = L_1 \cos\theta_1 \dot{\theta_1} + L_2 \cos\theta_2 \dot{\theta_2} \tag{9}$$

$$\dot{y_1} = L_1 \sin\theta_1 \dot{\theta_1} \tag{10}$$

$$\dot{y_2} = L_1 \sin\theta_1 \dot{\theta_1} + L_2 \sin\theta_2 \dot{\theta_2} \tag{11}$$

So, The total kinetic energy of the system will be:

$$T = \frac{1}{2} m_1 \dot{\theta_1}^2 L_1^2 + \frac{1}{2} m_2 \left[\dot{\theta_1}^2 L_1^2 + \dot{\theta_2}^2 L_2^2 + 2\dot{\theta_1} L_1 \dot{\theta_2} \cos(\theta_1 - \theta_2 \right] \tag{12}$$

Finally, we can get the lagrangian $\mathcal{L}$:

$$\mathcal{L} = \frac{1}{2}(m_1 + m_2) L_1^2 \dot{\theta_1}^2 + \frac{1}{2} m_2 L_2^2 \dot{\theta_2}^2 + m_2 L_1 L_2 \dot{\theta_1} \dot{\theta_2} \cos(\theta_1 - \theta_2) + (m_1 + m_2) g L_1 \cos\theta_1 + m_2 g L_2 \cos\theta_2 \tag{13}$$

3 The Equations of Motion

The Euler-Lagrange equation can be implemented to get the equations of motion for the system since in our analysis, we do not consider non-conservative constraint forces. Although, it is possible to extend the analysis beyond this to conclude these non-conservative forces.

$$\frac{\partial \mathcal{L}}{\partial \theta_i} - \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta_i}}\right) = 0$$

Where $i = 1, 2, 3, ...n$ and n is the number of degrees of freedom.

$$\therefore \frac{\partial \mathcal{L}}{\partial \theta_1} = -L_1 g (m_1 + m_2) \sin\theta_1 - m_2 L_1 L_2 \dot{\theta_1} \dot{\theta_2} \sin(\theta_1 - \theta_2) \tag{14}$$

$$\frac{\partial \mathcal{L}}{\partial \dot{\theta_1}} = (m_1 + m_2) L_1^2 \dot{\theta_1} + m_2 L_1 L_2 \dot{\theta_2} \cos(\theta_1 - \theta_2) \tag{15}$$

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta_1}}\right) = (m_1 + m_2) L_1^2 \ddot{\theta_1} + m_2 L_1 L_2 \ddot{\theta_2} \cos(\theta_1 - \theta_2) - m_2 L_1 L_2 \dot{\theta_2} \sin(\theta_1 - \theta_2)(\dot{\theta_1} - \dot{\theta_2}) \tag{16}$$

$$\frac{\partial \mathcal{L}}{\partial \theta_2} = m_2 L_1 L_2 \dot{\theta_1} \dot{\theta_2} \sin(\theta_1 - \theta_2) - L_2 m_2 g \sin\theta_2 \tag{17}$$

$$\frac{\partial \mathcal{L}}{\partial \dot{\theta_2}} = m_2 L_2^2 \dot{\theta_2} + m_2 L_1 L_2 \dot{\theta_1} \cos(\theta_1 - \theta_2) \tag{18}$$

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta_2}}\right) = m_2 L_2^2 \ddot{\theta_2} + m_2 L_1 L_2 \ddot{\theta_1} \cos(\theta_1 - \theta_2) - m_2 L_1 L_2 \dot{\theta_1} \sin(\theta_1 - \theta_2)(\dot{\theta_1} - \dot{\theta_2}) \tag{19}$$

After simplification:

$$\ddot{\theta_1} = \frac{-m_2 L_2 \ddot{\theta_2} \cos(\theta_1 - \theta_2) - m_2 L_2 \dot{\theta_2}^2 \sin(\theta_1 - \theta_2) - (m_1 + m_2) g \sin\theta_1}{(m_1 + m_2) L_1}$$

$$\ddot{\theta_2} = \frac{-L_1 \ddot{\theta_1} \cos(\theta_1 - \theta_2) + L_1 \dot{\theta_1}^2 \sin(\theta_1 - \theta_2 - g \sin\theta_2)}{L_2}$$

4 The Numerical analysis of The Equations of Motion

The previous two equations are second order partial differential equations. we can turn it into a system of firt order ordinary differential equations.

Using the substitutions:

$$z_1 = \theta_1 \qquad \Rightarrow \qquad \dot{z_1} = \dot{\theta_1} \tag{20}$$
$$z_2 = \theta_2 \qquad \Rightarrow \qquad \dot{z_2} = \dot{\theta_2} \tag{21}$$
$$z_3 = \dot{\theta_1} \qquad \Rightarrow \qquad \dot{z_3} = \ddot{\theta_1} \tag{22}$$
$$z_4 = \dot{\theta_2} \qquad \Rightarrow \qquad \dot{z_4} = \ddot{\theta_2} \tag{23}$$

Therefore,

$$\dot{z_1} = \dot{\theta_1} \tag{24}$$
$$\dot{z_2} = \dot{\theta_2} \tag{25}$$
$$\tag{26}$$

$$\dot{z_3} = \frac{-m_2 L_1 z_4^2 \sin(z_1 - z_2)\cos(z_1 - z_2) + m_2 g \sin z_2 \cos(z_1 - z_2) - m_2 L_2 z_4^2 \sin(z_1 - z_2) - (m_1 + m_2) g \sin z_1}{L_1(m_1 + m_2) - m_2 L_1 \cos(z_1 - z_2)^2} \tag{27}$$

$$\dot{z_4} = \frac{m_2 L_2 z_4^2 \sin(z_1 - z_2)\cos(z_1 - z_2) + g \sin z_1 \cos(z_1 - z_2)(m_1 + m_2) + L_1 z_4^2 \sin(z_1 - z_2)(m_1 + m_2) - g \sin z_2 (m_1 + m_2)}{L_2(m_1 + m_2) - m_2 L_2 \cos(z_1 - z_2)^2} \tag{28}$$

Solving these four equations numerically ,using the MATLAB Function **ode45** which uses the Runge-Kutta method, yields approximate solutions for θ_1 , θ_2 , $\dot{\theta_1}$, $\dot{\theta_2}$.

5 The MATLAB code

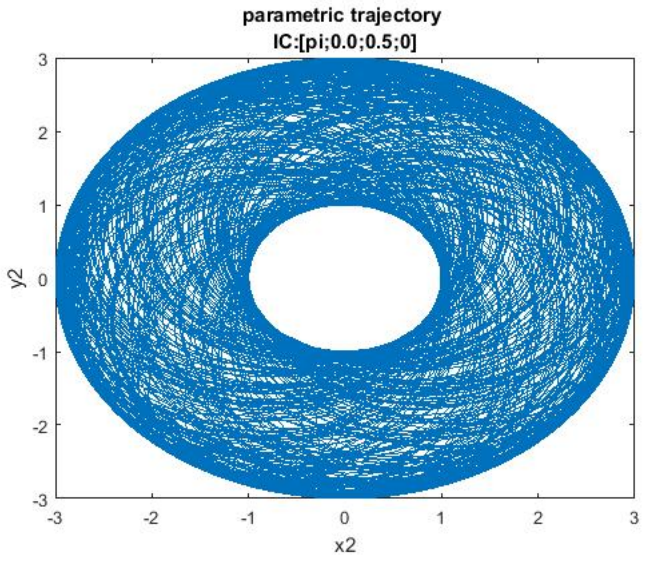
At first, we define a function that contains the four first order equations:

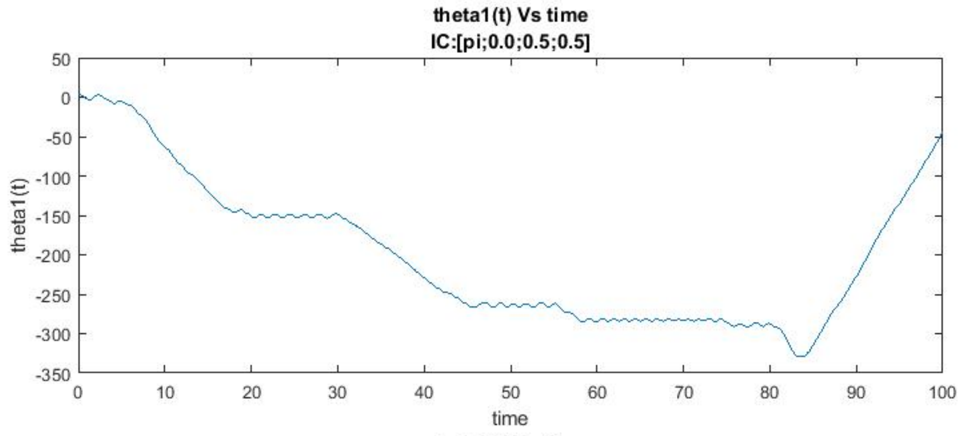
```
function [ yprime ] = double_pendulum(t,y)
yprime = zeros(4,1);
m1 = 2;
m2 = 1;
L1 = 1;
L2 = 2;
g = 32.0;
a = (m1+m2)*L1;
b = m2*L2*cos(y(1)-y(3));
c = m2*L1*cos(y(1)-y(3));
d = m2*L2;
e = -m2*L2*y(4)*sin(y(1)-y(3))-g*(m1+m2)*sin(y(1));
f = m2*L1*y(2)*y(2)*sin(y(1)-y(3))-m2*g*sin(y(3));

yprime(1) = y(2);
yprime(3) = y(4);
yprime(2) = (e*d-b*f)/(a*d-c*b);
yprime(4) = (a*f-c*e)/(a*d-c*b);

end
```

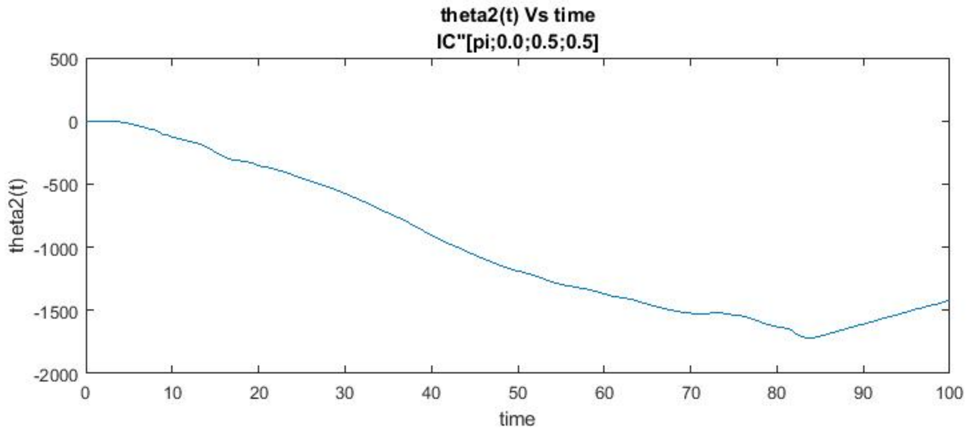
Solving the system:

```
function [t,x2,y2] = double_pendulum_demo(time)

%set up parameters
m1 = 2; m2 = 1; L1 = 1; L2 = 2; g=32.0;

%redefine relative tolerance
options = odeset('RelTol',1.0e-6);

%calling Runge-Kutta solver
[t,y] = ode45('double_pendulum',[0 time],[pi;0.0;0.5;0.5],options);
x2 = L1*sin(y(:,1))+L2*sin(y(:,3));
y2 = -L1*cos(y(:,1))-L2*cos(y(:,3));
plot(x2,y2);   %parametric trajectory of the outer bob
pause;
plot(t,y(:,1)); %theta1 Vs time
pause;
plot(t,y(:,3)); %theta2 Vs time
end
```

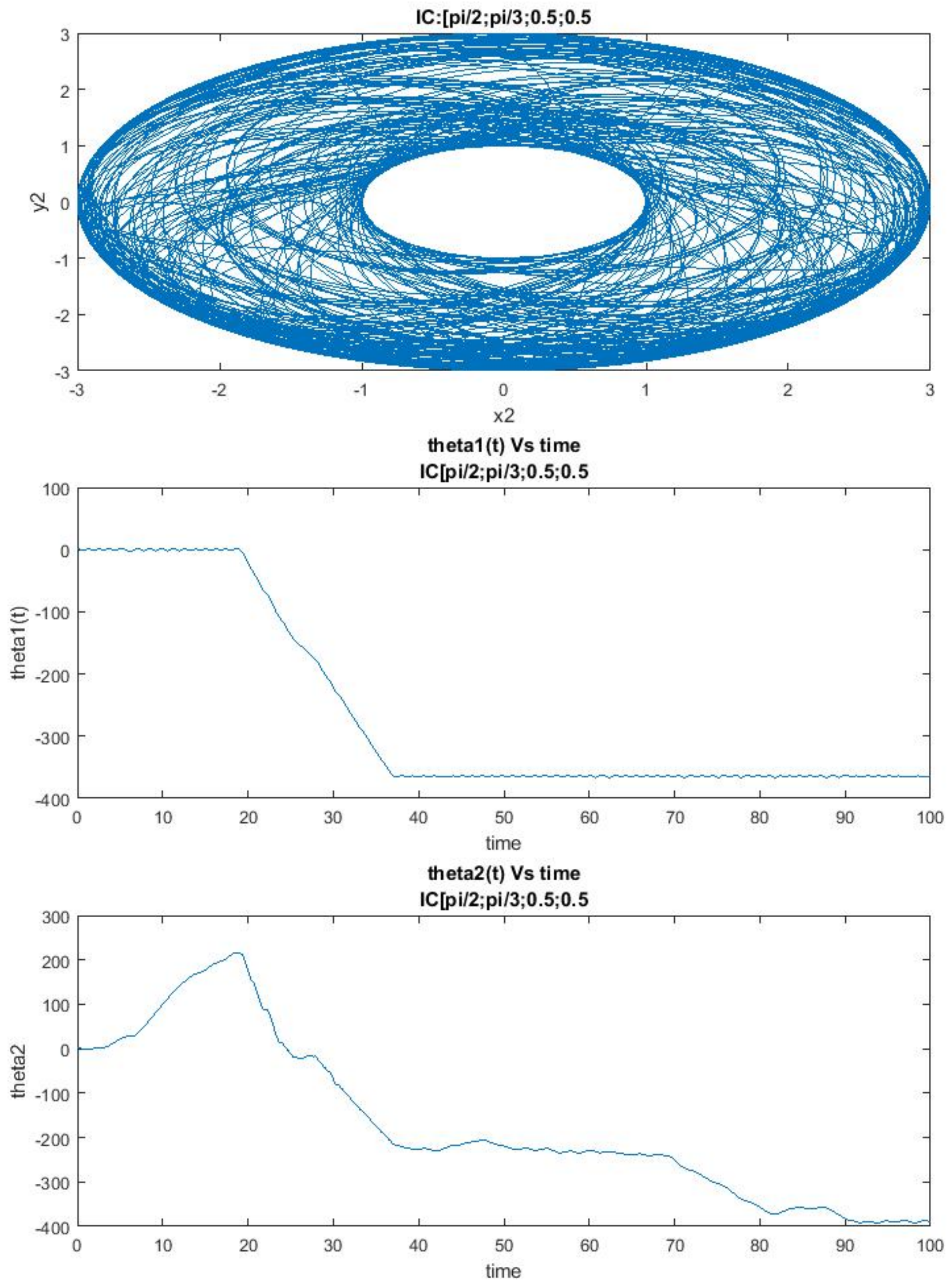
For Initial Conditions $[\pi; 0.0; 0.5; 0.0]$ **and time** $= 100$







For Initial Conditions $[\pi/2; \pi/3; 0.5; 0.0]$ **and time** $= 100$



For the phase space trajectory with different initial conditions:

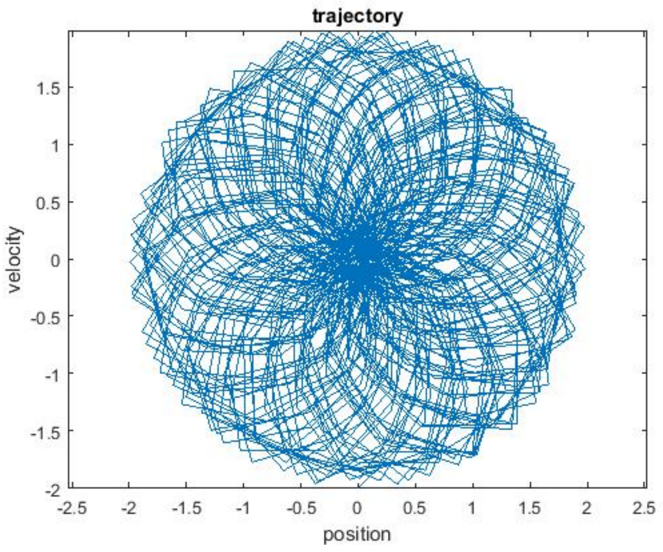
System of equations Doublependulum2.m

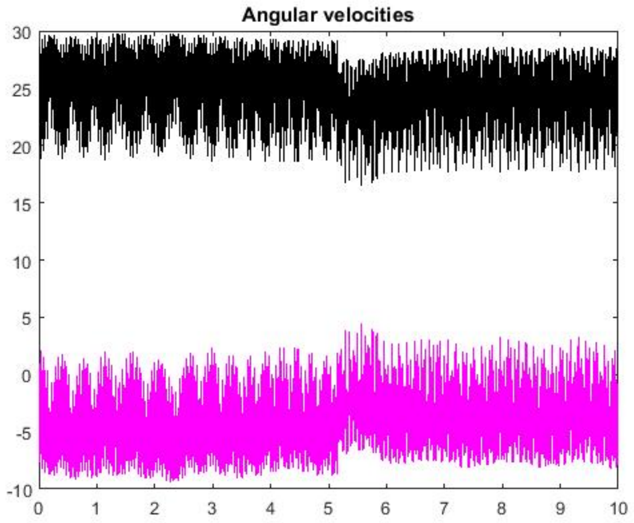
```
%define a function that calculates the dynamics of the double pendulum
%flag determines initial positions and velocities of the inner and outer
%bob.
function xprime=doublependulum2(t,x,flag,g,l1,l2,m1,m2)
xprime=zeros(4,1);
xprime(1) = 6*(2*x(3)-3*cos(x(1)-x(2))*x(4))/(16-9*cos(x(1)-x(2))^2);
xprime(2) = 6*(8*x(4)-3*cos(x(1)-x(2))*x(3))/(16-9*cos(x(1)-x(2))^2);
xprime(3) = -(xprime(1)*xprime(2)*sin(x(1)-x(2))+3*g*sin(x(1)))/2;
xprime(4) = -(-xprime(1)*xprime(2)*sin(x(1)-x(2))+g*sin(x(2)))/2;
end
```

Producing the trajectory of the outer bob and angular velocity graph.m

```
clear
%declare time step
T = 10;
%y0 = [0;0;-2;0]; % close to regular
%y0 = [pi;pi;0;0];
% y0 = [pi;pi;.5;0]; % chaotic
% y0=[0.5233;0;0.5233;0]%periodic
%y0 = [0.2,0.2828,0,0]%perfect periodic with energy = 0.7809
%y0 = [0.2,-0.2828,0,0]%perfect quasiperiodic with energy = 0.7809
%y0 = [27.8;0;1.22;2.62] %perfect KAM scenario
%y0 = [0,0,2,20]
y0 = [0,0,1,20];
%y0 =[pi;0;.5;.5];%high energy
s=ode45(@doublependulum2,[0,T],y0,[],0,1,1,1,1);
t = 0:0.01:T; x = deval(s,t)’;
x1=sin(x(:,1));
y1=-cos(x(:,1));
x2=x1+sin(x(:,2));
y2=y1-cos(x(:,2));
figure; plot(x2,y2); axis equal; xlabel(’position’);ylabel(’velocity’);title(’trajectory’)
pause;
plot(t,x(:,3),’magenta’,t,x(:,4),’black’);title(’Angular velocities’);
```

If we take $y0 = [0, 0, 1, 20]$; for example, The result will be:





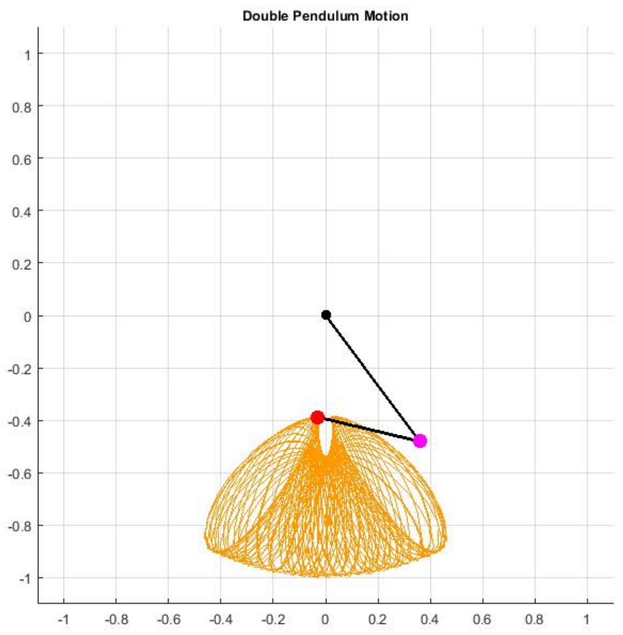
For a simulation:

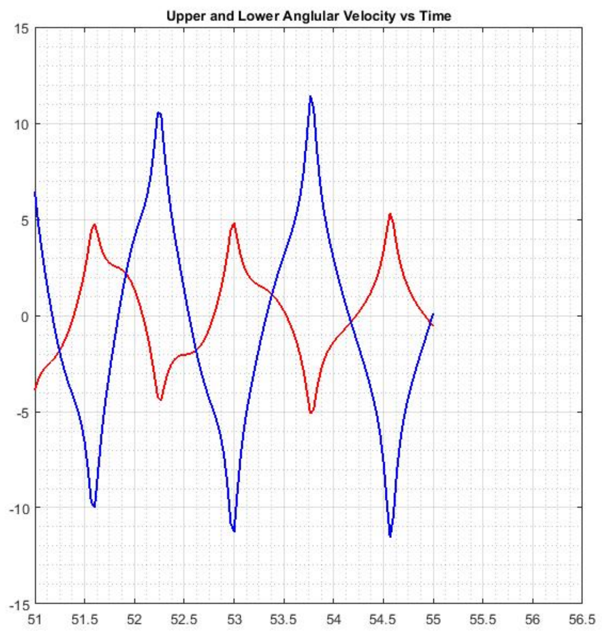
```
%
%   This file holds the differential equations that represent the motion
%   of a double pendulum.
%
%   @param ic
%   ic= [theta1, dtheta1, theta2, dtheta2, grav, m1, m2, len1, len2]
%   ic is a vector of initial conditions needed to properly set up the
%   equations of the double pendulum
%
%   -------------------------------------------------------------------

function xdot = DoublePendEquations(t, ic)

%extract the initial conditions from the ic vector
grav=ic(5); m1=ic(6); m2=ic(7); len1=ic(8); len2=ic(9);
theta1 = ic(1); theta2 = ic(3);
dtheta1 = ic(2); dtheta2 = ic(4);


xdot=zeros(9,1);

%theta1 prime = angular velocity1
xdot(1)=dtheta1;

% angularvelocity1 prime = this equation
xdot(2)= -(grav.*(2*m1+m2)*sin(theta1) + m2*grav*sin(theta1-2*theta2)...
         + 2*sin(theta1-theta2)*m2*((dtheta2.^2)*len2 + (dtheta1.^2).*len1*cos(theta1-theta
         (len1*(2*m1+m2-m2*cos(2*theta1-2*theta2)));

%theta2 prime = angular velocity2
xdot(3)=dtheta2;

%angularvelocity2 prime = this equation
xdot(4)= (2*sin(theta1-theta2)*((dtheta1.^2)*len1*(m1+m2)+grav*(m1+m2)*cos(theta1)...
         +(dtheta2.^2)*len2*m2*cos(theta1-theta2)))/...
         (len2*(2*m1+m2-m2*cos(2*theta1-2*theta2)));


end

% This solves and animates the motion, angle, or angular velocity of a double
% pendulum system as it progresses through time
%
%
%   -------------------------------------------------------------------
%
%   @params ic
%   ic = [theta1; angvel1; theta2; angvel2; grav; mass1; mass2; len1; len2;]
%   ic - Initial Conditions, is a 9 length row vector that represents the
%   initial conditions of the pendulm system.
%   a 1 after the parameter name means upper pendulum, 2 means lower pendulum.
%
%   @param time
%   The length in seconds of the simulation. It must always start at zero,
%   and it’s value will be the length of the simulation at 100% of normal
%   running speed
%
%   @param simspeed
%   This number is a double that represents a percentage in a decimal
%   format. This percentage is the simulation speed as a percentage of the
%   normal speed. So 50% speed would have simspeed = 0.5.
%
%   @param angorangvel
%   This is either a 1 or a 2, do not pass in a value besides these. If it
%   is a 1, then the graph on the right will show the angles of the two
%   pendulums as they move throughout time. If this value is a 2, then the
%   graph on the right will be the angular velocities of the two pendulums
%   as they move throughout time.

%%  ----------------------------------------------------------------------

function DoublePendSimulation(ic, time, simspeed, angorangvel)
clear All;

opengl software;

%define the normal frames per second of the animation and the adjusted
%frames per second based on the simspeed variable.
fpsnormal = 30;
fps = fpsnormal*simspeed;
numframes=time*fps;

%define the tolerances for the Runge-Kutta method of the differential
%equation
options = odeset(’Refine’,6,’RelTol’,1e-5,’AbsTol’,1e-7);

%solve the differential equations defined in the file @DoublePEndEquations,
%over the interval t = 0 to t = time, with the initial conditions specified
%in the vector ic, according to the options defined directly above.
solutionsstruct=ode45(@DoublePendEquations,[0 time], ic, options);
%define a discrete vector of points that we want to obtain the solutions on
t = linspace(0,time,numframes);
%obtain the values of the differential equations defined on the linespace
%above
solutionsvector=deval(solutionsstruct,t);

% get the individual components of the solution vector
theta1=solutionsvector(1,:)’; angvel1=solutionsvector(2,:)’;
theta2=solutionsvector(3,:)’; angvel2=solutionsvector(4,:)’;
%get the individual initial conditions and constants passed in by the user
len1=ic(8); len2=ic(9);
m1 = ic(6); m2 = ic(7);
moment1 = 0.5.*m1.*len1.^2;
moment2 = 0.5*m2.*len2.^2;
grav = ic(5);

%initialize vectors to hold the x and y coordinated of the lines we are
```

```
%going to be plotting
linex1 = zeros(0, numframes-1);
linex2 = zeros(0,numframes-1); % x/y coordinates of trailing line for position
liney2 = zeros(0,numframes-1);
liney1 = zeros(0, numframes-1);
omega1 = zeros(0, numframes-1);
omega2 = zeros(0, numframes-1);
ang1 = zeros(0,numframes-1);
ang2 = zeros(0,numframes-1);
timearr = zeros(0,numframes-1);

%you can ignore these next 4 lines, they were used for an energy analysis
%of the system
maxval= max(omega1(:));
maxval2 = max(omega2(:));
maxv = [moment1.*maxval.^2 moment2.*maxval2.^2];
[maxe junk] = max(maxv);


%create the figure window, set it to outer edges of screen
figure('units','normalized','outerposition',[0 0 1 1]);
% this subplot defnes the coordinates and size of the pendulum pos. plot
subplot('Position',[.03 .1 .52 .8]);hold on;

% this plots the line that shows where the bottom pendulum has been
pendline = plot([0 0], [0 0], 'Color', [1 153/255 0]);
%h is the handle to the actual plot of the 2 pendulums x and y coordinates
rods = plot(0,0,'k', 'LineWidth',2);
%this plot places the objects on the ends up the pendulums
ColorSet = [0 0 0; 1 0 1; 1 0 0];
pendobjects = scatter([0 0 0], [0 0 0], [50, 100, 100], ColorSet, 'filled');
axis equal;grid on;
title('Double Pendulum Motion', 'fontweight', 'bold', 'fontsize',10); hold off;

range=1.1*(len1+len2); axis([-range range -range range]); %pos plot axis limits

%dfine the subplot and plot for the second plot (theta plot or angvel plot)
subplot('Position', [.56 .1 .38 .8]);
%this is for the upper pendulum
plot2l1 = plot(0, 0,'r', 'LineWidth', 1.2);grid minor; hold on;
%this is for the lower pendulum
plot2l2 = plot(0, 0, 'b', 'LineWidth', 1.2, 'MarkerSize', 50);
if(angorangvel == 1)
    title('Upper and Lower Angle vs Time','fontweight','bold','fontsize',10);grid on;
end;
if(angorangvel == 2)
    title('Upper and Lower Anglular Velocity vs Time','fontweight','bold','fontsize',10);
end;
grid on;
hold off;

%put the data from the ODE solution into vector to be plotted
linex1 = len1*sin(theta1);
liney1 = -len1*cos(theta1);
```

```
linex2 = linex1+len2*sin(theta2);
liney2 = liney1-len2*cos(theta2);

%IGNORE all of these next commented lines, they were used for an energy
%analysis of the system
%height = len1+len2;
%en = .5.*(m1+m2).*(len1.^2).*angvel1.^2 + .5.*m2.*(len2.^2).*(angvel2.^2)...
%    + m2.*len1.*len2.*(angvel1).*angvel2.*cos(theta2-theta1)+(m1+m2).*grav.*len1.*(1-cos(
%en1 = zeros(0, numframes-1);
%pot1 = m1.*grav.*(range+liney1);
%pot2 = m2.*grav.*(range+liney2);
%kin1 = .5*m1.*(angvel1.^2.*len1.^2);
%kin2 = en-pot1-pot2-kin1;
%enmax = max(en);
%ploten = bar([0 0], .5); ylim([0 enmax]); grid on;



%simulation loop
    for i=1:numframes
            timearr(i) = i/fps;
            Xcoord=[0,linex1(i),linex2(i)];
            Ycoord=[0,liney1(i),liney2(i)];

            %moment2 = 0.5*m2.*(linex2(i).^2 + liney2(i).^2);

            ang1(i) = theta1(i);
            ang2(i) = theta2(i);
            omega1(i) = angvel1(i);
            omega2(i) = angvel2(i);

            %ignore these next lines, they were used for an energy analysis
            k1 = .5.*moment1*(angvel1(i).^2);
            k2 = .5.*moment2*(angvel2(i).^2)+ .5*m2*angvel1(i).^2/len1.^2;
            u1 = (Ycoord(2)).*grav.*m1;
            u2 = (Ycoord(3)).*grav.*m2;

            %pendulum position simulation
            set(pendline,'XData',linex2(1:i),'YData',liney2(1:i));
            %uncomment the following if-statements if you want the line to
            %only last for the last 200 frames
            %if(i > 200)
            %    set(pendline,'XData',linex2(i-200:i),'YData',liney2(i-200:i));
            %end
            %if(i < 200)
            %    set(pendline,'XData',linex2(1:i),'YData',liney2(1:i));
            %end
            set(rods,'XData',Xcoord,'YData',Ycoord);
            set(pendobjects,'Xdata', Xcoord, 'YData', Ycoord);

            ax1 = get(plot2l1,'Parent');
            ax2 = get(plot2l2,'Parent');
            %these next two plot the two angles or angular velocities vs
            %time on the same graph, depending on the value of angorangvel
```

```
        if(angorangvel == 1)
            set(plot2l1,'XData',timearr,'YData',ang1);
            set(ax1,'XLIM',[timearr(i)-4 timearr(i)+1.5])
            set(plot2l2,'XData',timearr,'YData', ang2);
            set(ax2,'XLIM',[timearr(i)-4 timearr(i)+1.5]);
        end
        if(angorangvel==2)
            set(plot2l1,'XData',timearr,'YData',omega1);
            set(ax1,'XLIM',[timearr(i)-4 timearr(i)+1.5])
            set(plot2l2,'XData',timearr,'YData', omega2);
            set(ax2,'XLIM',[timearr(i)-4 timearr(i)+1.5]);
        end

        %ENERGY
        %en1(i) = en(i);
        %enarr = [kin1(i); kin2(i); pot1(i); pot2(i)];
        %set(ploten,'YData', enarr);

        %3D plot of position and angvel
        %set(plot2l2, 'XData',linex1(1:i),'YData',liney1(1:i),'ZData', angvel1(1:i));
        %set(plot2l3, 'XData',linex2(1:i),'YData',liney2(1:i),'ZData', angvel2(1:i));

        %set(plot2l2, 'XData',linex1(1:i),'YData',liney1(1:i),'ZData',zeros(1,i));
        %set(plot2l3, 'XData',linex1(1:i)+linex2(1:i),'YData',linex1(1:i),'ZData', line


        drawnow;
        %F(i) = getframe;
end
```


Double Pendulum Motion


Upper and Lower Anglular Velocity vs Time