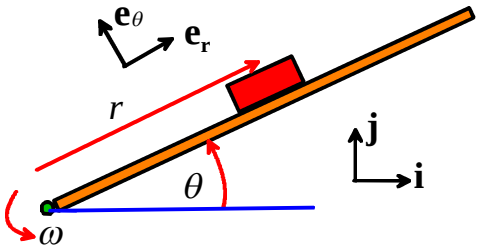
School of Engineering
Brown University

EN40: Dynamics and Vibrations

Homework 3: Solving equations of motion for particles
Solutions
MAX SCORE 52 POINTS + 14 EXTRA CREDIT

1. The figure shows a small mass *m* on a rigid rod. The system starts at rest with $\theta = 0$ and *r=r0*, and then the rod begins to rotate with constant angular speed ω. The system rotates in the vertical plane, and gravity should be included. Friction may be neglected.



1.1 Write down a formula for the angle θ as a function of time.

Constant angular velocity, so $\theta = \omega t$

[1 POINT]

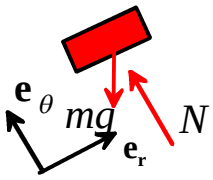
1.2 Write down a formula for the acceleration vector of the mass *m* in terms of *r* and ω, expressing your answer as radial-polar components in the basis $\mathbf{e}_r, \mathbf{e}_\theta$ (you don't need to derive the result – just use the formula).

From notes, $\mathbf{a} = \left(\frac{d^2r}{dt^2} - r\omega^2\right)\mathbf{e}_r + \left(2\omega\frac{dr}{dt}\right)\mathbf{e}_\theta$

[1 POINT]

1.3 Draw a free body diagram for the mass.

[2 POINTS]



1.4 Using Newton's laws, show that *r* satisfies the differential equation

$$\frac{d^2r}{dt^2} - \omega^2 r = -g\sin\omega t$$

Newton's law gives

$$m\left(\frac{d^2r}{dt^2} - r\omega^2\right)\mathbf{e}_r + m\left(2\omega\frac{dr}{dt}\right)\mathbf{e}_\theta = \left(N - mg\cos\theta\right)\mathbf{e}_\theta - mg\sin\theta\,\mathbf{e}_r$$

The radial component of this equation gives the required answer.

[2 POINTS]

1.5 Use Mathematica to solve the differential equation.

```
In[21]:= ODE =
           {r''[t] - w^2 * r[t] == -g * Sin[w * t]};

In[22]:= IC = {r[0] == r0, r'[0] == 0};

In[23]:= DSolve[Join[ODE, IC], r, t]
```

$$\text{Out[23]=}\ \left\{\left\{\mathrm{r} \to \mathrm{Function}\left[\{\mathrm{t}\},\ \frac{1}{4\,\mathrm{w}^2}\,\mathrm{e}^{-\mathrm{t\,w}}\left(\mathrm{g} - \mathrm{e}^{2\,\mathrm{t\,w}}\,\mathrm{g} + 2\,\mathrm{r0}\,\mathrm{w}^2 + 2\,\mathrm{e}^{2\,\mathrm{t\,w}}\,\mathrm{r0}\,\mathrm{w}^2 + 2\,\mathrm{e}^{\mathrm{t\,w}}\,\mathrm{g}\,\mathrm{Sin}[\mathrm{t\,w}]\right)\right]\right\}\right\}$$

The result can be simplified a bit by hand (not necessary, however)

$$r = \frac{1}{2} r_0 \left(e^{\omega t} + e^{-\omega t}\right) - \frac{g}{4\omega^2}\left(e^{\omega t} - e^{-\omega t}\right) + \frac{g}{2\omega^2}\sin\omega t$$

[3 POINTS]

1.4 Show that the differential equation can be expressed in MATLAB form as

$$\frac{d}{dt}\begin{bmatrix} r \\ v_r \end{bmatrix} = \begin{bmatrix} v_r \\ -g\sin\omega t + \omega^2 r \end{bmatrix}$$

Follow the procedure discussed in class – let $\frac{dr}{dt} = v_r$. With this definition, the differential equation becomes

$$\frac{dv_r}{dt} = -g\sin\omega t + \omega^2 r$$

This equation can be rearranged into the form given.

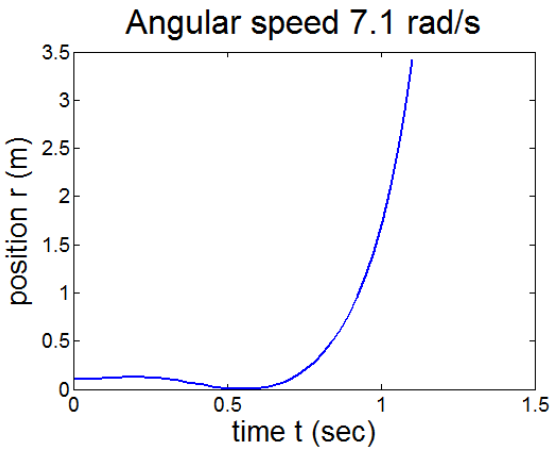
[2 POINTS]

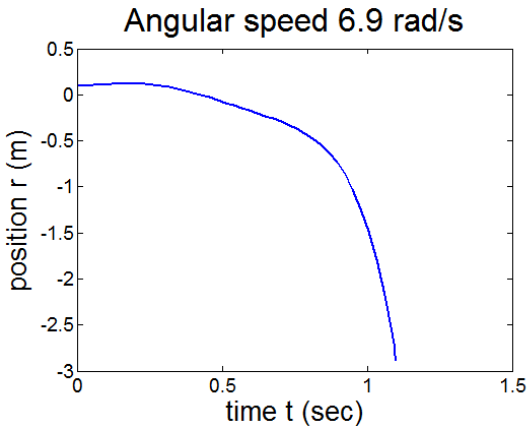
1.5 Write a MATLAB script that will plot r as a function of time t given appropriate values for ω and the values of r and v_r at time $t=0$. **There is no need to submit a solution to this problem**

```
function hw3prob1
    close all
    g = 9.81;
    omega = 7.1;
    r0 = 0.1;
    v0 = 0;
    initial_w = [r0;v0];
    start = 0;
    stop = 1.1;
    [time_vals,sol_vals] = ode45(@eom,[start,stop],initial_w);
```

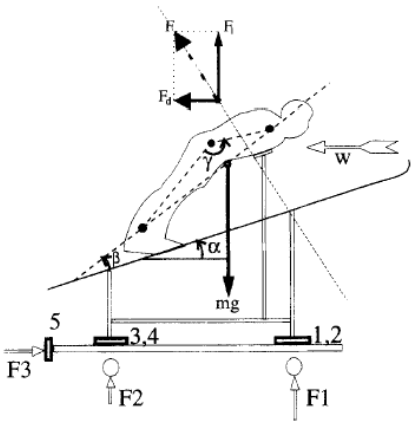
```
    figure1 = figure
    axes('Parent',figure1,'FontSize',14);
    box('on');
    plot(time_vals,sol_vals(:,1),'LineWidth',2);

    xlabel('time t (sec)','FontSize',20);
    ylabel('position r (m)','FontSize',20);
    title('Angular speed 7.1 rad/s','FontSize',24)

    animate_impeller(time_vals,sol_vals,omega);

    function dwdt = eom(t,w)
        r = w(1); vr = w(2);
        drdt = vr;
        dvrdt = -g*sin(omega*t)+omega^2*r;
        dwdt = [drdt;dvrdt];

    end
end
```

1.6 Use your MATLAB script to plot r as a function of time t for $0<t<1.1$ sec, and initial conditions $r = 0.1\,\text{m}$, $v_r = 0$ at time $t=0$. Try solutions with (a) $\omega = 7.1\,\text{rad/s}$ and (b) $\omega = 6.9$

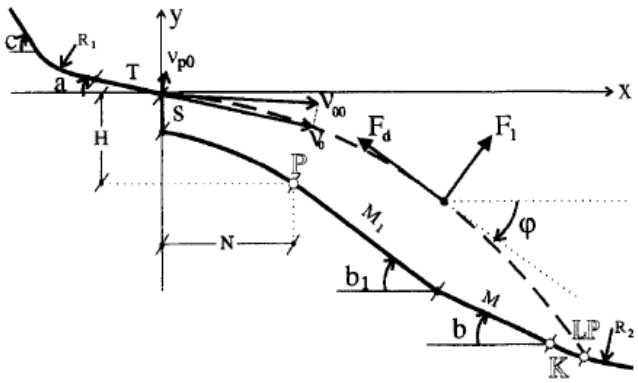




[4 POINTS]



Skier in a wind-tunnel



Flight-path of a typical ski jump

2. In this problem you will implement a computer-model of a ski-jumper in flight. For a detailed background to this problem see Wolfram Müller, Dieter Platzer, Bernhard Schmölzer, "Dynamics of human flight on skis: Improvements in safety and fairness in ski jumping, Journal of Biomechanics," **29**, 1996, pp.1061-1068, (http://www.sciencedirect.com/science/article/pii/0021929095001697). Both figures shown above are taken from this article.

During flight, three forces act on the athlete (they are shown on the figure):

(i) An aerodynamic lift force, which acts perpendicular to her path
(ii) A drag force, which acts parallel to the path, opposite to the direction of motion
(iii) Gravity

Wind tunnel experiments show that the lift and drag forces depend on the angle of attack α (the angle between the skis and the airflow relative to the skier). Muller *et al* show that the measured lift and drag forces can be approximated by

$$F_L = \frac{1}{2}\rho\left(-0.8468 + 0.08963\alpha - 0.001292\alpha^2\right)V^2$$

$$F_D = \frac{1}{2}\rho\left(-0.5072 + 0.04398\alpha - 2.861\times10^{-4}\alpha^2\right)V^2$$

where ρ is the air density, V is the skier's airspeed, α is the angle of attack in degrees (SI units must be used for all quantities). Both expressions are valid only for $\alpha > 15^0$ (the formula predicts negative drag and lift for small α, which is nonsense)

2.1 Let $\mathbf{v} = v_x\mathbf{i} + v_y\mathbf{j}$ denote the velocity vector of the jumper while in flight. Find expressions for unit vectors parallel to the lift and drag forces (which act parallel and perpendicular to the jumper's path) in terms of v_x, v_y. Hint – for the vector parallel to lift, think about cross products…

Dividing **v** by its magnitude creates a unit vector parallel to the path. Changing its sign makes it opposite to the direction of motion – parallel to the drag. So $\mathbf{e}_D = -(v_x\mathbf{i} + v_y\mathbf{j})/\sqrt{v_x^2 + v_y^2}$

To create a vector normal to the path, we can just take a cross product with the **k** vector. Thus

$$\mathbf{e}_L = \mathbf{k}\times(v_x\mathbf{i} + v_y\mathbf{j})/\sqrt{v_x^2 + v_y^2} = (v_x\mathbf{j} - v_y\mathbf{i})/\sqrt{v_x^2 + v_y^2}$$ **[2 POINTS]**

2.2 Write down Newton's law of motion for the skier, and show that they can be re-arranged in MATLAB form as

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ -F_D v_x/(mV) - F_L v_y/(mV) \\ -F_D v_y/(mV) + F_L v_x/(mV) - g \end{bmatrix}$$

where $V = \sqrt{v_x^2 + v_y^2}$.

The resultant force acting on the skier can be expressed as

$$\mathbf{F} = F_D \mathbf{e}_D + F_L \mathbf{e}_L - mg\mathbf{j} = -F_D(v_x\mathbf{i} + v_y\mathbf{j})/\sqrt{v_x^2 + v_y^2} + F_L(v_x\mathbf{j} - v_y\mathbf{i})/\sqrt{v_x^2 + v_y^2}$$

The acceleration is

$$\mathbf{a} = \frac{d^2x}{dt^2}\mathbf{i} + \frac{d^2y}{dt^2}\mathbf{j}$$

Introducing

$$\frac{dx}{dt} = v_x \qquad \frac{dy}{dt} = v_y$$

writing **F**=m**a** and reading off the **i** and **j** components of the equations of motion then gives the required ODEs.

[3 POINTS]

2.3 Write a MATLAB script to calculate x,y, v_x,v_y as functions of time (**There is no need to submit a solution to this part).**

```
function skijump

    m = 70;      % ski jumper mass (kg)
    psi = 0;    % Ski angle to horizontal (deg)
    rho = 1.03;  % air density kg/m^3
    vx0 = 28.7;    % initial horizontal velocity m/s
    vy0 = -1.2;   % initial vertical velocity m/s
    g = 9.81;     % gravitational accel m/s^2
    initial_w = [0;0;vx0;vy0]; % initial solution vector

    options = odeset('Events',@event);
    [times,sols] = ode45(@eom,[0,8],initial_w,options);

    figure1 = figure; % Plot the trajectory
    plot(sols(:,1),sols(:,2),'LineWidth',2,'Color',[1 0 0]);
    figure2 = figure; % Plot the velocity
    vel = sqrt(sols(:,3).^2+sols(:,4).^2);
    plot(times,vel)
     function dwdt =  eom(t,w)
         % The equations for the rates of change of x,y,vx,vy
         x = w(1); y = w(2); vx = w(3); vy = w(4);
         phi = atan(-vy/vx)*180/pi;
         alpha = psi + phi; % Angle of attack (degrees)
         Fl = 0.5*rho*(-0.8468 + 0.08963*alpha - 0.001292*alpha^2);
         Fd = 0.5*rho*(-0.5072 + 0.04398*alpha - 2.861e-4*alpha^2);
         dxdt = vx; dydt = vy;
```

```
        V = sqrt(vx^2+vy^2); % Magnitude of velocity
        dvxdt = -Fd*vx*V/m - Fl*vy*V/m;
        dvydt = -Fd*vy*V/m + Fl*vx*V/m - g;
        dwdt = [dxdt;dydt;dvxdt;dvydt];
    end
    function [ev,s,dir] = event(t,w)
        % Function to detect when skier lands
        x = w(1); y = w(2);
        q = atan(-y/x)*180/pi; % Angle of skier below start
        ev = q-32; % Hits slope when angle-32 degrees = 0
        s = 1; % Stop when the event occurs
        dir = 0; % Either sign of zero crossing is OK.
    end
end
```

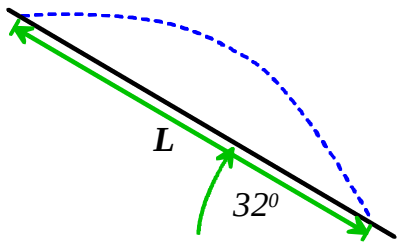
2.4 Run your simulation with the following parameters and initial conditions (taken from Muller et al):

(i) Mass m = 70 kg

(ii) Initial velocity $v_x = 28.7 \quad v_y = -1.2$ m/s and initial position $x=y=0$.

(iii) Air density 1.03 kg/m^3

(iv) Ski orientation (assume that the skier does not rotate during flight – this is not quite correct, as expert skiers rotate their skis to maximize lift) $\psi = 0^0$

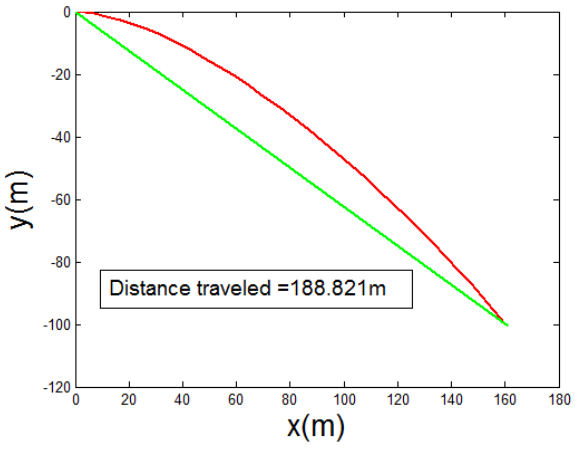


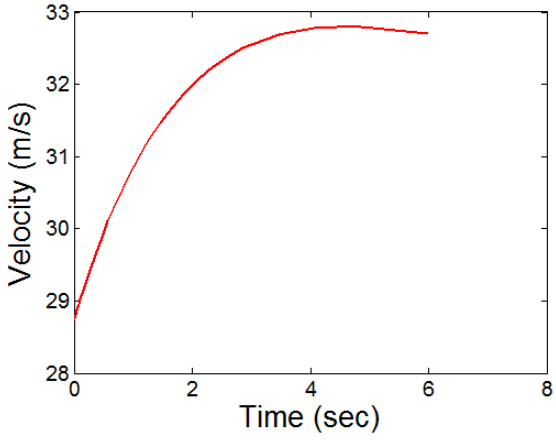
Plot the trajectory (*y as a function of x*) for $0<t<5$ sec. Once your code is working, add an 'event' function that will stop the calculation when the skier lands on the slope (for simplicity, assume that the hillside is just a straight line with slope 32^0). Hand in:

(i) A plot of the trajectory predicted by your final code (with the event function)

(ii) A plot of the magnitude of the skier's velocity as a function of time

(iii) The predicted length L of the jump





[4 POINTS]

2.5 For comparison, compute (by hand) the predicted length L of the jump without air resistance (you need to use the trajectory equations to do this – follow the procedure used for the 'shoot the elmo' demo in class).

The trajectory equations give

$$x = V_{x0}t \qquad y = V_{y0}t - gt^2/2$$

At the landing point $x = L\cos(32)$ $y = -L\sin(32)$. The time of flight follows as $t = L\cos(32)/V_{x0}$, and the remaining equation gives

$$-L\sin(32) = V_{y0}L\cos(32)/V_{x0} - g(L\cos(32)/V_{x0})^2/2$$

$$\Rightarrow L = 2V_{x0}\left(V_{x0}\sin(32) + V_{y0}\cos(32)\right)/(g\cos(32)^2)$$

Substituting numbers gives 115.4 m (you can easily check this with MATLAB by setting the air density to zero). Surprisingly the actual jump is much longer than this – the lift force is extending the jump (drag reduces it). The skier really is flying!
(Making the initial vertical velocity of the skier positive instead of negative is a common error – this gives L=132m. Deduct a point for this)

[4 POINTS]

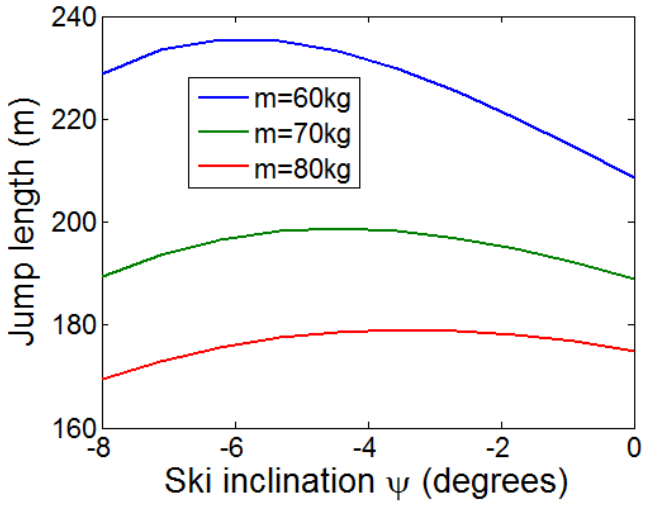
2.6 Repeat (1.5) for skiers with masses of *m=60kg* and *m=80*kg. (report the predicted jump lengths only, there is no need to hand in plots of the trajectories or velocities).

- 60kg skier – 209m
- 80kg skier - 175m

There is clearly a very significant advantage to minimizing body weight – in fact the rules of the sport were changed recently to reduce the pressure on athletes to lose weight (see http://www.nytimes.com/2010/02/12/sports/olympics/12skijump.html?pagewanted=all)

[2 POINTS]

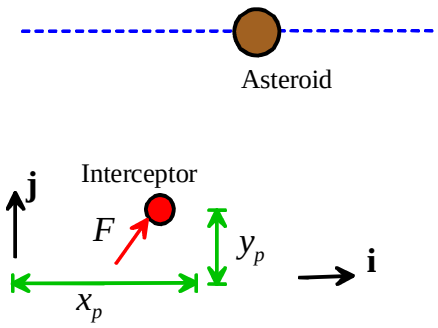
2.7 **OPTIONAL – extra credit problem** Find the value of ψ that maximizes the jump length.



The figure shows a graphical solution – the optimal angle depends on the weight of the skier, varying between about -3.5 degrees for an 80kg jumper to -6 degrees for a 60 kg skier. The predictions may not be very reliable for the 80 and 60 kg cases because the simulations use wind tunnel measurements for a 70kg skier to compute the lift and drag forces. In practice skiers vary their orientation throughout the jump to maximize lift. More sophisticated solutions might use the MATLAB optimizer.

[3 POINTS]

3. The goal of this problem is to test a proposed guidance system for an interceptor, whose purpose is to impact an asteroid. For simplicity, we will neglect gravity in this problem, and assume that both asteroid and interceptor move in the *(x,y)* plane.



3.1 When the asteroid is first detected, it has position (x_0, y_0) and has constant velocity (V_{x0}, V_{y0}). Write down formulas for the position vector of the asteroid as a function of time

Straight line motion with constant velocity $x_a = x_0 + V_{x0}t \qquad y_a = y_0 + V_{y0}t$

[1 POINT]

3.2 At time *t=0* the interceptor is at rest and at the origin. The interceptor is powered by a rocket exerting a constant thrust *F*. The interceptor will be steered by altering the direction of the thrust. The guidance system will detect the instantaneous position of the asteroid $(x_a(t), y_a(t))$, and adjust the direction of the thrust so that it always points towards the asteroid. The goal of this problem is to determine whether this procedure will result in an impact.

Let (x, y) and (v_x, v_y) denote the components of the position vector of the interceptor and its velocity. Write down the resultant force vector acting on the interceptor, in terms of *F*, $(x_a(t), y_a(t))$ and (x, y) (start by writing down a unit vector parallel to the thrust vector).

A unit vector pointing from the interceptor to the asteroid is

$$\mathbf{n} = \left((x_a - x)\mathbf{i} + (y_a - y)\mathbf{j}\right) / \sqrt{\left(x_a - x\right)^2 + \left(y_a - y\right)^2}$$

The force (magnitude and direction) is $F\mathbf{n}$

[2 POINTS]

3.3 Using Newton's laws, show that the equations of motion for the interceptor can be re-arranged into the standard MATLAB form as follows

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ F(x_a - x)/(dm) \\ F(y_a - y)/(dm) \end{bmatrix}$$

Where $d = \sqrt{(x_a - x)^2 + (y_a - y)^2}$ and m is the interceptor mass. Note that x_a, y_a are known functions of time – you will have to enter formulas for these functions in your MATLAB script.

The acceleration vector is $\mathbf{a} = \frac{d^2x}{dt^2}\mathbf{i} + \frac{d^2y}{dt^2}\mathbf{j}$. The two components of **F**=m**a**, with **F** from the previous problem, and writing

$$\frac{dx}{dt} = v_x \qquad \frac{dy}{dt} = v_y$$

Gives

$$m\frac{dv_x}{dt} = F(x_a - x)/(d) \qquad m\frac{dv_y}{dt} = F(y_a - y)/(d)$$

[2 POINTS]

3.4 Write a MATLAB script to compute the path of the interceptor. Add an 'event' function to your code that will stop the calculation if the asteroid and interceptor are less than 1km apart (work with units of km and s). You can download a matlab .m file that will animate the trajectory of both the interceptor and the asteroid. To the script, put the .m file in the same directory as your homework MATLAB. Insert a line that looks like

```
animate_projectiles(time_vals,sol_vals,[X0,Y0],[Vx0,Vy0])
```

after your ODE solver, where 'time_vals', and 'sol_vals' are the solution found in your ODE solver, and X0,Y0, Vx0,Vy0 are the initial position and velocity of the asteroid. **THERE IS NO NEED TO SUBMIT A SOLUTION TO THIS PROBLEM.**

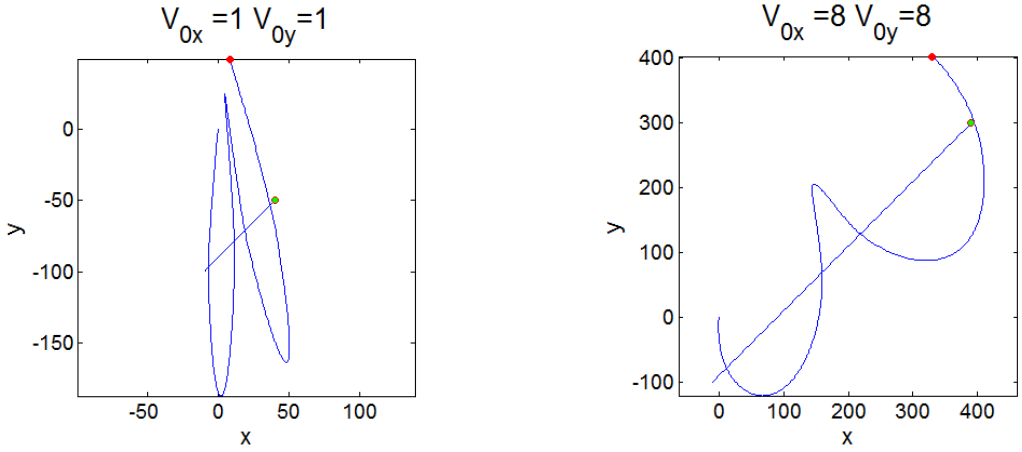
```
function interceptor

    close all

    X0 = -10;
    Y0 = -100;
    Vx0 = 1;
    Vy0 = 1;
    m = 50;
    F = 250;
    initial_w = [0,0,0,0];
    t_vals = 0:0.1:50;
    options = odeset('Events',@event,'RelTol',0.0001);
    [time_vals,sol_vals] = ode45(@eom,t_vals,initial_w,options);

    animate_projectiles(time_vals,sol_vals,[X0,Y0],[Vx0,Vy0])

    function dwdt = eom(t,w)
        xa = X0+Vx0*t; ya = Y0+Vy0*t;
        x=w(1); y=w(2); vx=w(3); vy=w(4);
```

```
        d = sqrt((xa-x)^2+(ya-y)^2);
        dxdt = vx; dydt = vy;
        dvxdt = F*(xa-x)/(d*m);
        dvydt = F*(ya-y)/(d*m);
        dwdt = [dxdt; dydt; dvxdt; dvydt];

    end
    function [ev,s,dir] = event(t,w)
        xa = X0+Vx0*t; ya = Y0+Vy0*t;
        x=w(1); y=w(2);
        ev = sqrt((xa-x)^2 + (ya-y)^2)-1.;
        s = 1;
        dir = -1;
    end

end
```

3.5 Run your code (for a time interval of 50 sec) with the following parameters:

$$X_0 = -10km, Y_0 = -100km, V_{x0} = 1km/s, V_{y0} = 1km/s$$
$$m = 50kg, F = 250kN,$$
$$x = y = 0 \quad \text{at} \quad t = 0$$

$$X_0 = -10km, Y_0 = -100km, V_{x0} = 8km/s, V_{y0} = 8km/s$$
$$m = 50kg, F = 250kN,$$
$$x = y = 0 \quad \text{at} \quad t = 0$$



[4 POINTS]

(Graphs may differ a bit because of scaling of axes, etc – anything looking along right lines is OK)

3.6 **[OPTIONAL – extra credit problem]** The guidance system proposed here is clearly unsatisfactory. We can improve it by making the direction of the force depend on the relative velocity of the two particles, in as well as on their relative position. As a first attempt, we could try directing the thrust along a unit vector

$$\mathbf{n} = \left(\mathbf{r}_a - \mathbf{r} + c(\mathbf{v}_a - \mathbf{v})\right) / \left|\mathbf{r}_a - \mathbf{r} + c(\mathbf{v}_a - \mathbf{v})\right|$$

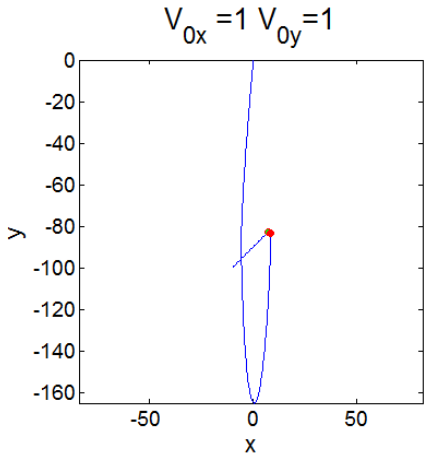
where c is a constant that can be adjusted to give the best performance. Modify your code and test this idea. Use the same parameters as in problem 3.5, and try $c=0.4$. You could try other values of c as well if you are curious.

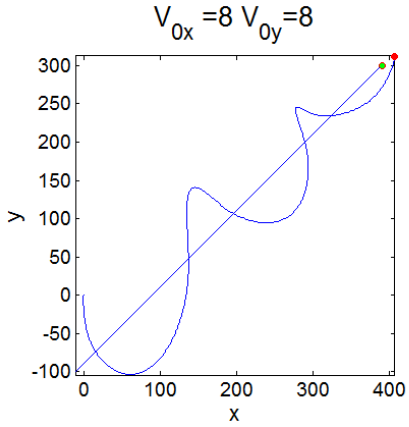
Adding the following lines to the eom function accomplishes this

```
vrel = w(3:4)-[Vx0;Vy0];
dvxdt = xa-x - c*vrel(1);
dvydt = ya-y - c*vrel(2);
dvxdt = F*dvxdt/sqrt(dvxdt^2+dvydt^2)/m;
dvydt = F*dvydt/sqrt(dvxdt^2+dvydt^2)/m;
```

You also need to define the value of c at the top of the code.

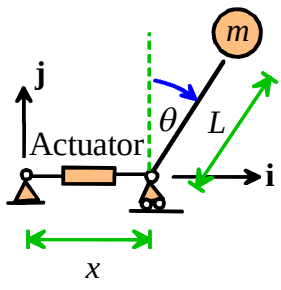
The solution (with $c=0.4$) for the two cases considered earlier is shown below. The interceptor reaches the 1km cutoff distance for the slower asteroid. It will eventually catch the faster one too, but not in the 50 sec time interval. A larger value of c gives better behavior – with c=2.4 it catches both





[4 POINTS]

4. A class demonstration showed that an inverted pendulum can be stabilized by shaking its pivot vertically at a correct frequency (for a computer model see HW3, 2011). The goal of this problem is to model stabilization using a feedback control mechanism. An actuator is attached to the pivot of the pendulum. The length of the actuator $x(t)$ varies with time.



4.1 Write down the position vector of the mass m, in terms of x and other relevant variables. Hence, calculate an expression for the acceleration of the mass, in terms of time derivatives of x, θ and other relevant variables.

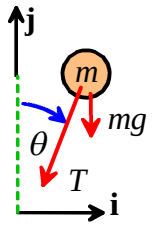
$$\mathbf{r} = (x + L\sin\theta)\mathbf{i} + (L\cos\theta)\mathbf{j}$$

$$\mathbf{v} = \left(\frac{dx}{dt} + L\cos\theta\frac{d\theta}{dt}\right)\mathbf{i} + (-L\sin\theta\frac{d\theta}{dt})\mathbf{j}$$

$$\mathbf{a} = \left[\frac{d^2x}{dt^2} - L\sin\theta\left(\frac{d\theta}{dt}\right)^2 + L\cos\theta\frac{d^2\theta}{dt^2}\right]\mathbf{i} + \left[-L\cos\theta\left(\frac{d\theta}{dt}\right)^2 - L\sin\theta\frac{d^2\theta}{dt^2}\right]\mathbf{j}$$

[3 POINTS]

3.2 Draw a free body diagram showing the forces acting on the mass *m*



[2 POINTS]

4.3 Write down Newton's law of motion of the mass, and hence show that the angle θ satisfies the equation

$$\frac{d^2\theta}{dt^2} - \frac{g}{L}\sin\theta + \frac{\cos\theta}{L}\frac{d^2x}{dt^2} = 0$$

F=m**a** gives

$$-T\sin\theta\mathbf{i} - T\cos\theta\mathbf{j} - mg\mathbf{j}$$

$$= m\left[\frac{d^2x}{dt^2} - L\sin\theta\left(\frac{d\theta}{dt}\right)^2 + L\cos\theta\frac{d^2\theta}{dt^2}\right]\mathbf{i} + m\left[-L\cos\theta\left(\frac{d\theta}{dt}\right)^2 - L\sin\theta\frac{d^2\theta}{dt^2}\right]\mathbf{j}$$

The **i** and **j** components of this equation yield two equations. We can eliminate *T* by multiplying the **i** component by $\cos\theta$ and the **j** component by $\cos\theta$ and subtracting **i** component from the **j**. This gives

$$-mg\sin\theta = -m\cos\theta\left[\frac{d^2x}{dt^2} - L\sin\theta\left(\frac{d\theta}{dt}\right)^2 + L\cos\theta\frac{d^2\theta}{dt^2}\right] + m\sin\theta\left[-L\cos\theta\left(\frac{d\theta}{dt}\right)^2 - L\sin\theta\frac{d^2\theta}{dt^2}\right]$$

$$mg\sin\theta = m\cos\theta\frac{d^2x}{dt^2} + mL\frac{d^2\theta}{dt^2}\left(\sin^2\theta + \cos^2\theta\right)$$

This equation can be rearranged into the form required.

[3 POINTS]

4.4 One approach to stabilizing the pendulum is to add a sensor to the system that will measure the angle θ, and then use a computer-control system to extend or contract the actuator so as to try to reduce θ. As a particularly simple scheme, we could try making *x* proportional to θ - for example by setting $x = -K\theta$, where *K* is a constant. (The negative sign here is counter-intuitive – it means that if the

pendulum swings to the right, the actuator must displace the pivot to the left. The vibrations section of EN40 will give some more mathematical insight into why this is necessary)

Show that, with this scheme, the governing equations for θ and x can be expressed in the form

$$\frac{d}{dt}\begin{bmatrix}\theta\\ \omega\end{bmatrix}=\begin{bmatrix}\omega\\ (g/L)\sin\theta/(1-(K/L)\cos\theta)\end{bmatrix}$$

(The equation of motion for x is not really necessary, since we already know $x=-K\theta$, but is included to make the code easy to modify later)

The equation of motion for θ is (from 3.2)

$$g\sin\theta=\cos\theta\frac{d^2x}{dt^2}+L\frac{d^2\theta}{dt^2}$$

and since $x=-K\theta\Rightarrow\frac{d^2x}{dt^2}=-K\frac{d^2\theta}{dt^2}$, we can re-write this as

$$\frac{g}{L}\sin\theta=\frac{d^2\theta}{dt^2}(1-\frac{K}{L}\cos\theta)$$

Introducing $\omega=\frac{d\theta}{dt}$ then yields

$$\frac{d\omega}{dt}=\frac{g}{L(1-\frac{K}{L}\cos\theta)}\sin\theta$$

[3 POINTS]

4.5 Write a MATLAB script that will integrate these equations of motion. If you would like to see an animation of the motion of pendulum, you can download a MATLAB script called animate_pendulum.m. Store this in the same directory as your MATLAB homework file, and add a line

```
animate_pendulum(times,sols,L);
```

after the call to the ODE solver in your homework. Here times,sols are the time values and solution variable values computed by the ODE solver, and L is the pendulum length. The animation looks most realistic if the solution is computed at equally spaced time intervals. **THERE IS NO NEED TO SUBMIT A SOLUTION TO THIS PROBLEM.**

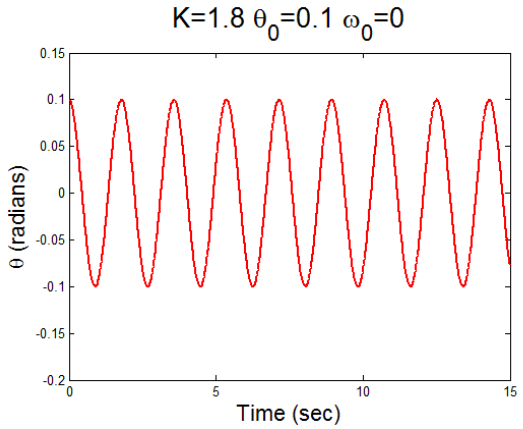
```
function pendulum

    close all
    K = 1.8;
    g = 9.81;
    L = 1;

    initial_w = [.1,0];
    tvals = 0:0.1:15;
    [times,sols] = ode45(@eom,tvals,initial_w);

    animate_pendulum(times,sols,L,K);
    figure
    plot(times,sols(:,1));

    function dwdt = eom(t,w)
       q = w(1); o = w(2);
```

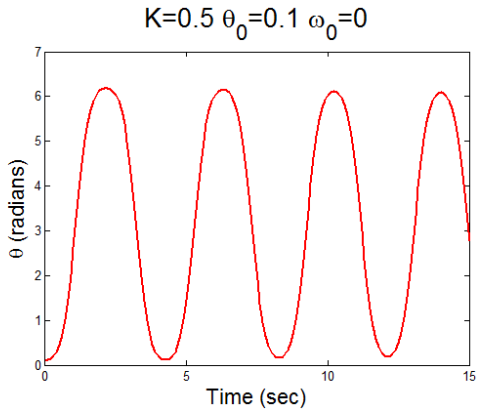
```
        dqdt = o;
        dodt = g*sin(q)/(L-K*cos(q));
        dwdt = [dqdt;dodt];
    end

end
```

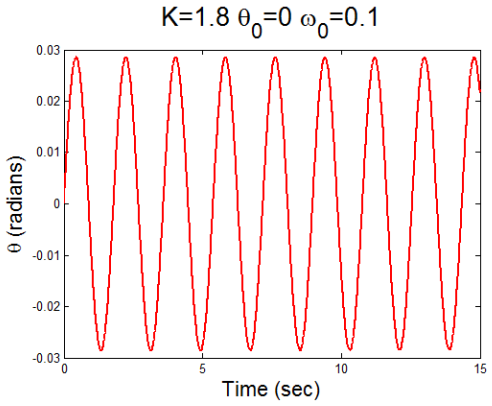
4.6 Use your MATLAB code to plot a graph of θ as a function of time, with a time interval of 15 sec, and a pendulum length of 1m. Try solutions with *K=1.8* and *K=0.5* for the following initial conditions

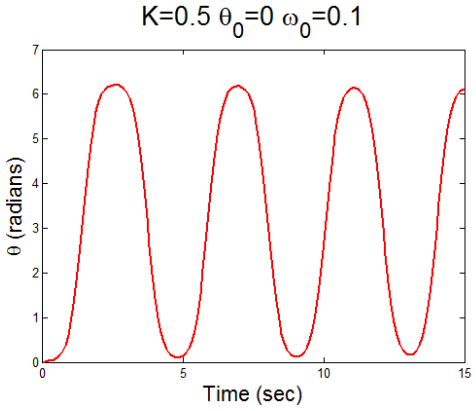
- $\theta = 0.1, \frac{d\theta}{dt} = 0$
- $\theta = 0., \frac{d\theta}{dt} = 0.1$
- $\theta = 0.9\pi, \frac{d\theta}{dt} = 0$

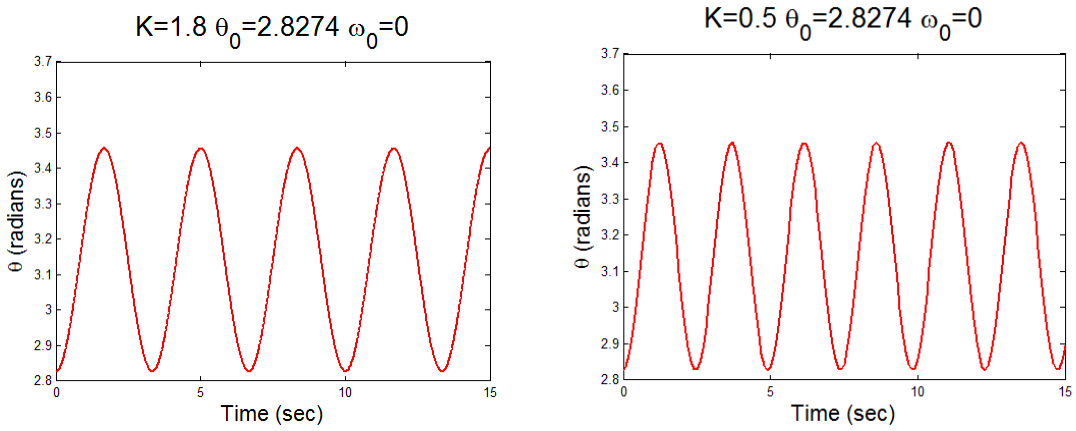
Note that the pendulum is stabilized in inverted position for $K>L$ – but if the pendulum is slightly displaced it will oscillate forever. In practice a bit of air resistance or friction would damp out the oscillations, but it is better to design the control system to avoid this oscillatory behavior. Note also that the pendulum is not self-righting – in fact $\theta = \pi$ is a stable configuration.











[4 POINTS]

4.7 **[OPTIONAL – extra credit problem]** Better behavior is achieved if the controller is designed so that its *velocity* is controlled, rather than its length. Specifically, we could try $\frac{dx}{dt} = -K\frac{d\theta}{dt} - \lambda_1\theta - \lambda_2(x - x_0)$. Here, $K, \lambda_1, \lambda_2, x_0$ are constants. Show that with this choice, the equations of motion for θ and *x* can be arranged into the form

$$\frac{d}{dt}\begin{bmatrix}\theta\\ \omega\\ x\end{bmatrix} = \begin{bmatrix}\omega\\ \alpha\\ v\end{bmatrix} \qquad \begin{aligned} v &= -K\omega - \lambda_1\theta - \lambda_2(x - x_0)\\ \alpha &= g\sin\theta/(L - K\cos\theta) + (\lambda_1\omega + \lambda_2 v)\cos\theta/(L - K\cos\theta)\end{aligned}$$

Now, we have that

$$\frac{dx}{dt} = -K\frac{d\theta}{dt} - \lambda_1\theta - \lambda_2(x - x_0) \Rightarrow \frac{d^2x}{dt^2} = -K\frac{d^2\theta}{dt^2} - \lambda_1\frac{d\theta}{dt} - \lambda_2\frac{dx}{dt}$$

Substituting this into the equation of motion gives

$$g\sin\theta = \cos\theta\left(-\lambda_1\frac{d\theta}{dt} - \lambda_2\frac{dx}{dt}\right) + (L - K\cos\theta)\frac{d^2\theta}{dt^2}$$

Hence, introducing $\omega = d\theta/dt$ and $v = dx/dt$

$$\frac{d\omega}{dt} = \frac{g\sin\theta}{(L - K\cos\theta)} + \frac{\cos\theta}{(L - K\cos\theta)}(\lambda_1\omega + \lambda_2 v)$$

[3 POINTS]

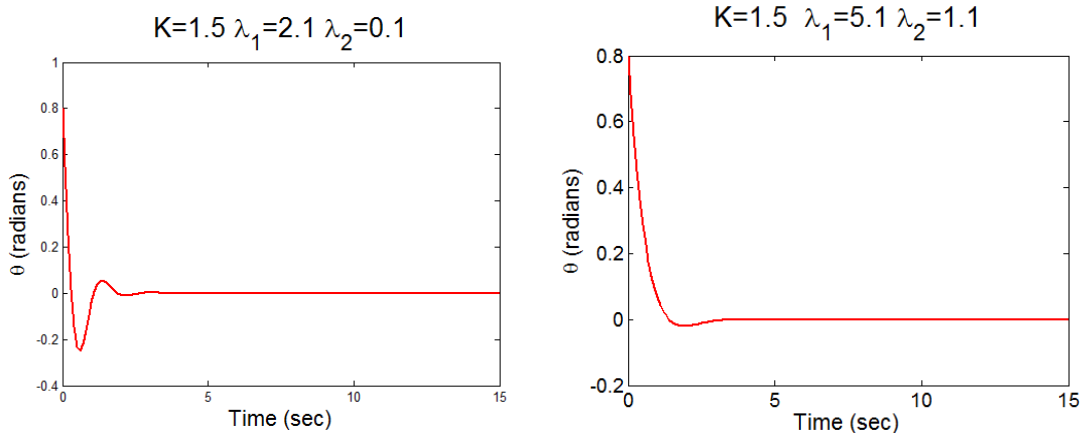
4.8 **[OPTIONAL – extra credit problem]** Modify your MATLAB code to use the control scheme described in 3.6, and use it to plot a graph of θ as a function of time, for a time interval 0<t<15 sec, and a pendulum length of 1m. Try solutions with $\theta = 0.8, d\theta/dt = 0, x = x_0 = 0.5$ at time *t=0*, with the following parameter values

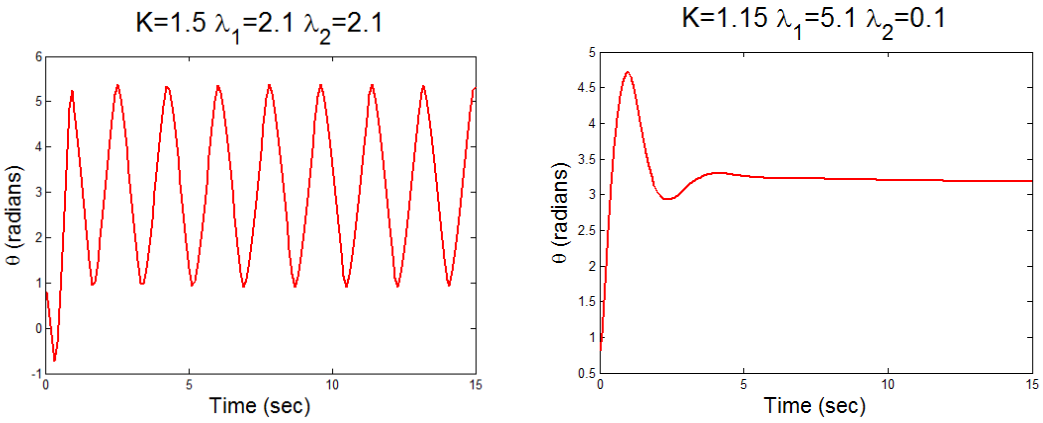
- $K = 1.5, \lambda_1 = 2.1, \lambda_2 = 0.1$
- $K = 1.5, \lambda_1 = 5.1, \lambda_2 = 0.1$

- $K = 1.5, \lambda_1 = 2.1, \lambda_2 = 2.1$
- $K = 1.15, \lambda_1 = 5.1, \lambda_2 = 0.1$

Designing optimal control systems is a concern in a wide range of engineering applications. Cruise control and autopilots are two obvious examples, but control systems are also needed in chemical plants, nuclear reactors, robotics, and so on. Sophisticated mathematical methods have been developed to help design control systems – you would not normally use a trial-and-error approach. But it is often helpful to check the performance of a design by integrating the equations of motion, as done here.

```
function pendulum

    close all
    K = 1.5;
    lam = 5.1;
    lam2 = 1.1;
    x0 = 0.50;
    g = 9.81;
    L = 1;

    initial_w = [.8,0,x0];
    tvals = 0:0.1:15;
    [times,sols] = ode45(@eom,tvals,initial_w);

    animate_pendulum(times,sols,L);
    figure
    plot(times,sols(:,1));

    function dwdt = eom(t,w)
       q = w(1); o = w(2); x = w(3);
       dqdt = o;
       v = -K*o-lam*q-lam2*(x-x0);
       dodt = g*sin(q)/(L-K*cos(q))+(lam*o+lam2*v)*cos(q)/(L-K*cos(q));
       dxdt = v;
       dwdt = [dqdt;dodt;dxdt];
    end

end
```





The first two cases both give good stabilization. The last two cases are unstable (in the last case the pendulum settles to its standard vertical configuration).

(Graphs in student solutions may differ a bit because of different initial conditions, etc – anything that looks sensible should get credit)

[4 POINTS]