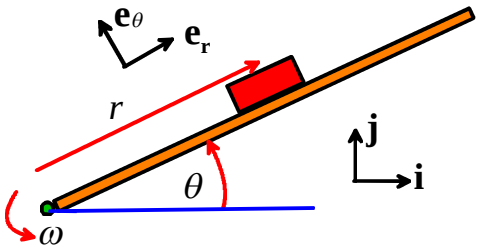
School of Engineering
Brown University

# EN40: Dynamics and Vibrations

## Homework 3: Solving equations of motion for particles

**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 $\omega$. 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 $\theta$ as a function of time.

1.2 Write down a formula for the acceleration vector of the mass *m* in terms of *r* and $\omega$, 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).

1.3 Draw a free body diagram for the mass.

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

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

1.5 Use Mathematica to solve the differential equation (Mathematica will give the exact, analytical solution)

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

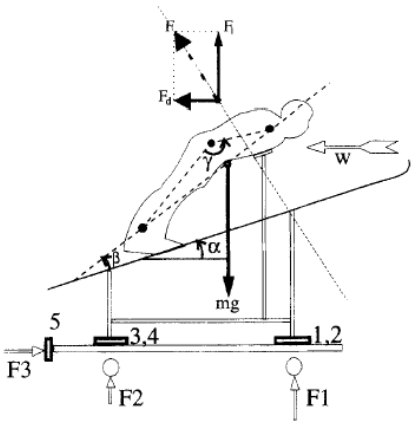
$$\frac{d}{dt}\begin{bmatrix} r \\ v_r \end{bmatrix} = \begin{bmatrix} v_r \\ -g \sin \omega t + \omega^2 r \end{bmatrix}$$

1.7 Write a MATLAB script that will plot $r$ as a function of time *t* given appropriate values for $\omega$ and the values of $r$ and $v_r$ at time *t=0*. If you like, you can download the .m file used in class that will animate the motion of the system from the HW web page. To use the script, put the .m file in the same directory as your homework MATLAB, and make sure the file is named animate_impeller.m Then, insert a line in your code that looks like

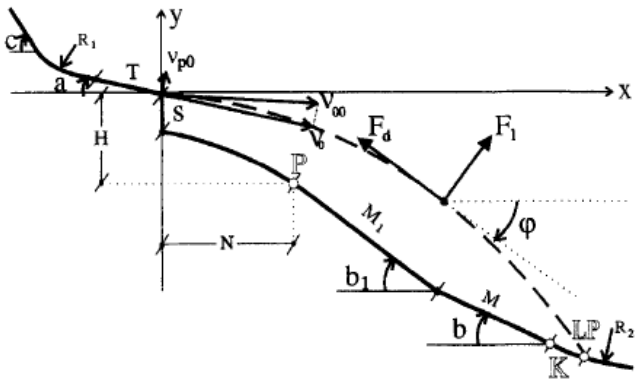
```
animate_impeller(time_vals,sol_vals,omega)
```

after your ODE solver, where 'time_vals', and 'sol_vals' are the solution found in your ODE solver, and omega is the value of the angular velocity of the bar. **There is no need to submit a solution to this problem**

1.8 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) $\omega = 6.9$




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 $\alpha$ (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 $\rho$ is the air density, $V$ is the skier's airspeed, $\alpha$ is the angle of attack in degrees (SI units must be used for all other quantities). Both expressions are valid only for $\alpha > 15^0$ (the formula predicts negative drag and lift for small $\alpha$, 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 – to calculate the vector parallel to lift, you could use a cross product…

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 $V = \sqrt{v_x^2 + v_y^2}$ and $F_L, F_D$ are computed from the complicated formulas given earlier.

2.3 Write a MATLAB script to calculate $x$,$y$, $v_x, v_y$ as functions of time, given values for the following quantities:

- The angle $\psi$ between the skis and the **i** direction
- Skier mass $m$
- Air density $\rho$
- The gravitational acceleration $g$
- The initial position of the skier and the initial velocity of the skier.

Note that the angle of attack can be calculated from $\alpha = (180/\pi)\left[\psi + \tan^{-1}(-v_y/v_x)\right]$, where $\psi$ is the angle between the skis and the **i** direction in radians. You should find that you can just modify the script shown in class (also in online notes) that calculated the motion of a projectile with air resistance (**There is no need to submit a solution to this part).**

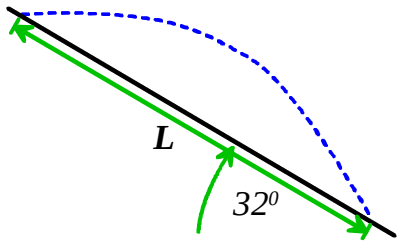
2.4 Run your simulation with the following parameters and initial conditions (taken from Muller et al):

(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 $32^0$). Hint: note that $\tan^{-1}(-y/x) - 32\pi/180 = 0$ when the skier lands.

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

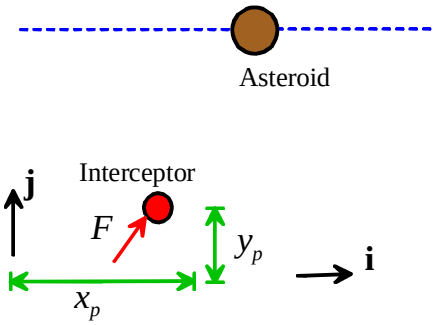
(iii) The predicted length $L$ of the jump

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).

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).

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

**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 (just use the straight line motion formulas…)

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).

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 (from part 3.1) – you will have to enter formulas for these functions in your MATLAB script.

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, kN 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 hand in a solution to this problem**

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

$$X_0 = -10km, Y_0 = -100km, V_{x0} = 1km/s, V_{y0} = 1km/s$$

1 $m = 50kg, F = 250kN,$

$$x = y = 0 \quad \frac{dx}{dt} = \frac{dy}{dt} = 0 \text{ at } t = 0$$

$$X_0 = -10km, Y_0 = -100km, V_{x0} = 8km/s, V_{y0} = 8km/s$$

**2** $m = 50kg, F = 250kN,$

$$x = y = 0 \quad \frac{dx}{dt} = \frac{dy}{dt} = 0 \quad \text{at} \quad t = 0$$

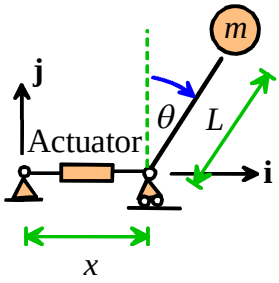
Plot the trajectory of both the interceptor and asteroid for each case.

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 4.5, and try *c=0.4*. You could try other values of *c* as well if you are curious. Hand in plots of the trajectories of the asteroid and interceptor for each case.

**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 $x, \theta$ and other relevant variables.

4.2 Draw a free body diagram showing the forces acting on the mass *m* (note that the pendulum shaft is a two-force member)

4.3 Write down Newton's law of motion of the mass, and hence show that the angle $\theta$ satisfies the equation (Hint – eliminate the unknown force in the shaft of the pendulum)

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

4.4 One approach to stabilizing the pendulum is to add a sensor to the system that will measure the angle $\theta$, and then use a computer-control system to extend or contract the actuator so as to try to reduce $\theta$. As a particularly simple scheme, we could try making *x* proportional to $\theta$ - for example by setting $x = -K\theta$, where *K* is a positive 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 $\theta$ and *x* can be expressed in the form

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

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 (make sure you name the file animate_pendulum.m), and add a line

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

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, *L* is the pendulum length, and *K* is the gain in the feedback control. 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.**

4.6 Use your MATLAB code to plot a graph of $\theta$ 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

- $\theta = 0.1, \frac{d\theta}{dt} = 0$
- $\theta = 0., \frac{d\theta}{dt} = 0.1$
- $\theta = 0.9\pi, \frac{d\theta}{dt} = 0$ (this case will blow up for K=1.8 so don't try that one!)

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.

4.7 **[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 $\theta$ and *x* can be arranged into the form

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

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 $\theta$ 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.