Introduction

This assignment explores further some of the symbolic and numerical capabilities of Mathematica. So far, the Ph20 assignments have dealt with particles undergoing free motion in a potential. This assignment has as its subject the motion of particles when a dissipative force (drag) is present.

Where did that grapefruit come from?

Rumor (history) has it that Caltech undergrads used a kerosene cannon to lob grapefruits onto Pasadena City College, at a distance of \( \sim 1000 \) m. Of course, Caltech students would never do something like this, at least not the students we get these days, but the physics of this venerable legend is a good problem to feed to Mathematica.

Mathematically (and Mathematically), the problem is that of a projectile subject to gravity, and to the resistance of air (atmospheric drag). Leaving apart drag for the moment, the equations of motion should be quite familiar to you:

\[
\frac{dv_x}{dt} = 0, \quad \frac{dv_y}{dt} = -g; \quad (1)
\]

\[v_x = \frac{dx}{dt}, \quad v_y = \frac{dy}{dt}; \quad (2)\]

which (as you know well) can be solved to yield

\[x = x_0 + v_{x0} t, \quad y = y_0 + v_{y0} t - \frac{1}{2} gt^2. \quad (3)\]

Of course, real projectiles don’t move quite so simply, because of drag. It turns out that the appropriate expression for the drag force in the case of spherical projectiles, with reasonable velocities, is (see Box 1)

\[F_{drag} \sim -\frac{1}{2} \rho r^2 v^2, \quad (4)\]

where \( \rho \) is the density of air at sea level, or approximately 1.3 kg/m\(^3\), and where \( r \) is the radius of the projectile. This drag force can then be incorporated into the equations of motion,

\[\frac{dv_x}{dt} = -\frac{|F_{drag}|}{m} \frac{v_x}{v}, \quad \frac{dv_y}{dt} = -g - \frac{|F_{drag}|}{m} \frac{v_y}{v}, \quad \text{where} \quad v = \sqrt{v_x^2 + v_y^2}. \quad (5)\]

These equations are much harder to solve by pencil and paper; we will use Mathematica.

The Assignment

1. Using Mathematica, prove the well known result that, in the absence of drag, the optimal firing angle (the angle that yields the longer range for a given initial velocity) is 45°. Then compute the velocity components necessary to reach PCC from Caltech using the optimal firing angle, still not including drag.
As you can imagine, the physics of atmospheric drag is very complicated. Enter approximations! We know for a fact that the drag force is null for an object at rest, and that it grows with velocity. Under certain mathematical assumptions (of smoothness, for instance), we can then write

\[ F_{\text{drag}} = -B_1 v - B_2 v^2 - B_3 v^3 - \cdots \]

The linear term (and other odd terms) vanish because drag does not depend on the sign of the velocity. For reasonable velocities, it turns out that the quadratic term is dominant. The resulting differential equation for the velocity is

\[ \frac{dv}{dt} = \frac{P}{mv} - \frac{B_2 v^2}{m}. \]

We estimate the coefficient \( B_2 \) by the following argument: to overcome atmospheric drag, the projectile must push out of the way the volume of air directly in front of it. In a time \( \Delta t \), the mass of air moved is \( m_{\text{air}} \approx \rho A v \Delta t \), where \( \rho \) is the density of air, and \( A \) is the projectile’s frontal area. This air is given a velocity of order \( v \), and therefore a momentum \( m_{\text{air}} v \) over a time \( \Delta t \). It follows that the instantaneous force exerted by the projectile on the air (and therefore, because of Newton’s third law, by the air on the projectile as a drag force) is approximately

\[ F_{\text{drag}} = -\rho A v^2. \]

For a spherical projectile, one finds empirically that the coefficient \( r^2/2 \) works better than the nominal area \( 4\pi r^2 \).

Box 1: A justification of the quadratic dependence of atmospheric drag.

2. In Mathematica, write a routine to integrate numerically the equations for motion without drag, and verify the above result. Superimpose several plots of the trajectory, with the same initial velocity but different firing angles, to show visually that \( 45^\circ \) is optimal. This is your first Ph20 Beautiful Plot™: try to arrange it so that it makes your point as clearly and boldly as possible.

3. In Mathematica, assume the grapefruit has mass 0.5 kg and radius 0.05 m. Include the acceleration due to drag in the equations of motion, and now predict where the grapefruit will land if you use the initial velocity you have just found.

4. In Mathematica, try increasing the initial velocity components and changing the firing angle until you can reach the range calculated in the drag-free case. Find the optimal firing angle (an approximate solution obtained by trial and error is acceptable). Show a visual comparison of trajectories with the same range (Caltech to PCC), but different firing angles and correspondingly different initial velocities. Again, make this graph beautiful and information-dense.

5. To go beyond trial and error, implement the following hierarchy of Mathematica functions (if needed, enlist the substantial help of your TA): (a) write a function that returns the time when the grapefruit lands \( (y = 0) \) as a function of initial angle and speed; (b) using the result of item a, write another function that returns the range \( (x \text{ at } y = 0 \text{ and } t \neq 0) \), given the initial angle and speed, and the corresponding landing time; (c) using the results of items a and b, write a function that iterates to find the initial velocity required to obtain a given range, for a certain initial angle;
(d) using the results of items a to c, write a function that iterates to find the angle with the least initial velocity required to yield a certain range.

**Mathematica Hints for Part 2**

How do we solve differential equations in Mathematica? First we define a list containing the equations. Mind the different “equal” signs: “=” is an assignment operator (...let eqs be the equation...), whereas “==” is the test operator used to state that the left side of the equation equals its right side.

```mathematica
eqs = {x''[t] == 0, y''[t] == -g}
```

Notice the “apostrophe” syntax used to denote derivatives. Let’s then define the initial conditions, say

```mathematica
ini = {x[0] == 0, y[0] == 0, x'[0] == 10, y'[0] == 10}
```

The system of differential equations can be solved with the NDSolve Mathematica function, which has the following syntax:

```mathematica
rules = NDSolve[Join[eqs, ini], {x, y}, {t, tinit, tend}]
```

When you invoke NDSolve, the parameters of your equation (in this case, g) and the initial time tinit and final time tend should have numerical values. You can assign these beforehand, or enclose NDSolve in a function and pass the parameters as arguments. The result returned by NDSolve is not a function, but rather a Mathematica rule. You can however define functions from the rule:

```mathematica
xx[t_] := x[t] /. rules[[1]];
yy[t_] := y[t] /. rules[[1]]
```

The “[[1]]” is needed because NDSolve returns the rules enclosed in a list (the braces), and we have to get rid of that first. You can then plot xx[t] and yy[t] normally, with Plot, and evaluate them directly, as in “xx[2]”.

---

3