Assignment 6

1.

We find an expression for the range of a projectile fired at an initial velocity v.

```
In[1]:= it = Solve[vy0 * t - 1/2 * g * t^2 == 0 && t ≠ 0, t]
```

$$\text{Out[1]=}\ \left\{\left\{t \to \frac{2\,vy0}{g}\right\}\right\}$$

```
In[2]:= vy0 = √(v^2 - vx0^2);

In[3]:= range = vx0 * t /. it[[1]][[1]]
```

$$\text{Out[3]=}\ \frac{2\,vx0\,\sqrt{v^2 - vx0^2}}{g}$$

```
In[4]:= frange[x_] := range /. vx0 → x

In[5]:= diffrange[x_] := frange'[x]

In[6]:= maxrange = Solve[diffrange[vx0] == 0, v]
```

$$\text{Out[6]=}\ \left\{\left\{v \to -\sqrt{2}\,vx0\right\}, \left\{v \to \sqrt{2}\,vx0\right\}\right\}$$

This shows that range is maximized for $v = \pm\sqrt{2}\,v_{x_0} \Rightarrow v_{y_0} = \sqrt{v^2 - v_{x_0}{}^2} = v_{x_0}$. Since the vertical and horizontal components of the initial velocity are equal, the optimal firing angle is 45° as stated. We now find the velocity components to reach PCC from Caltech (1000m):

```
In[7]:= g = 9.81;
        fmaxrange[x_] := frange[x] /. maxrange[[1]][[1]] /. vx0 → x

In[9]:= Solve[fmaxrange[vx0] == 1000, vx0, Reals]

        ··· Solve: Solve was unable to solve the system with inexact coefficients. The answer was obtained by solving a
              corresponding exact system and numericizing the result.

Out[9]= {{vx0 → 70.0357}}
```

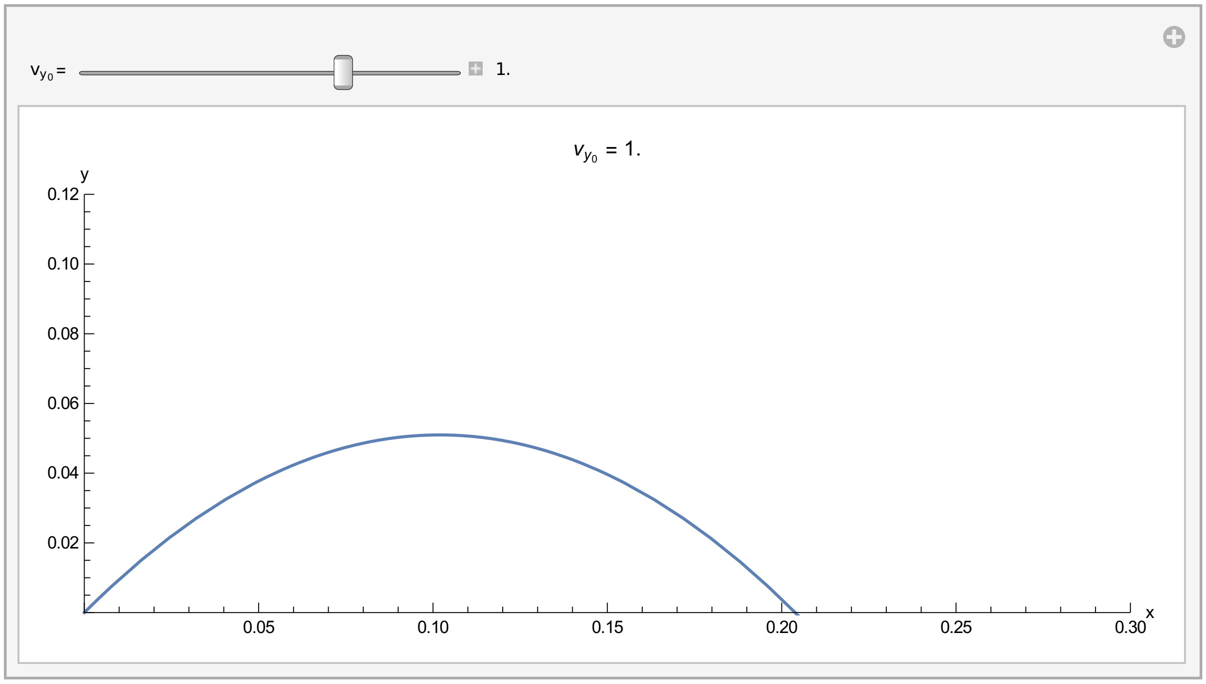
This gives us that in order to reach PCC from Caltech, we need to have $v_{x_0} = v_{y_0} = 70.03\,\text{ms}^{-1}$.

2.

We set $x_0 = y_0 = 0$ since we can always do this by a change of coordinates. We also set $v = \sqrt{2}$ since we only need to concern ourselves with the firing angle. If our previous result is correct, we expect the maximum range to be obtained for $v_{y_0} = 1$.

In[75]:=
```
g = 9.81;
Manipulate[
 ParametricPlot[Evaluate[{x[t], y[t]} /. NDSolve[{x''[t] == 0, y''[t] == -g, x[0] == 0,
      x'[0] == Sqrt[2 - vy0^2], y[0] == 0, y'[0] == vy0}, {x[t], y[t]}, {t, 0, 1}]],
   {t, 0, 0.4}, ImageSize → Large, AxesLabel → {"x", "y"},
   PlotRange → {{0, 0.3}, {0, 0.12}}, PlotLabel → Row@{"v_{y₀} = ", vy0}],
  {{vy0, 0, "v_{y₀}="}, 0, 1.4, Appearance → "Labeled"}]
```

Out[76]=



By manipulating the slider for v_{y_0} above, we can observe that the maximum range occurs when $v_{y_0} = 1 \Rightarrow v_{y_0} = v_{x_0}$ as expected (we use Manipulate rather than superimposing several plots because in this case it gives a clearer picture of the situation).

3.

We set the parameters, then simplify the equations with drag and solve them numerically using the firing velocity found in part 1 as initial conditions.

In[12]:=
```
r = 0.05;
m = 0.5;
ρ = 1.3;
g = 9.81;
```

```
In[16]:= rule = NDSolve[{x''[t] == -(ρ * r^2)/(2 * m) (x'[t] * Sqrt[x'[t]^2 + y'[t]^2]),
           y''[t] == -g - (ρ * r^2)/(2 * m) (y'[t] * Sqrt[x'[t]^2 + y'[t]^2]), x[0] == 0,
           x'[0] == 70.03, y[0] == 0, y'[0] == 70.03}, {x, y}, {t, 0, 9}]

Out[16]= {{x → InterpolatingFunction[ Domain: {{0., 9.}} Output: scalar ],
           y → InterpolatingFunction[ Domain: {{0., 9.}} Output: scalar ]}}

In[17]:= xx[t_] := x[t] /. rule[[1]]
         yy[t_] := y[t] /. rule[[1]]

In[24]:= impact = FindRoot[yy[t] == 0, {t, 7}] // Quiet

Out[24]= {t → 9.92805}

In[25]:= xx[t] /. impact

         InterpolatingFunction: Input value {9.92805} lies outside the range of data in the interpolating function.
            Extrapolation will be used.

Out[25]= 333.308
```

This gives us that the grapefruit lands 333m away from Caltech in the direction of PCC if fired with initial velocity components as calculated without accounting for drag.

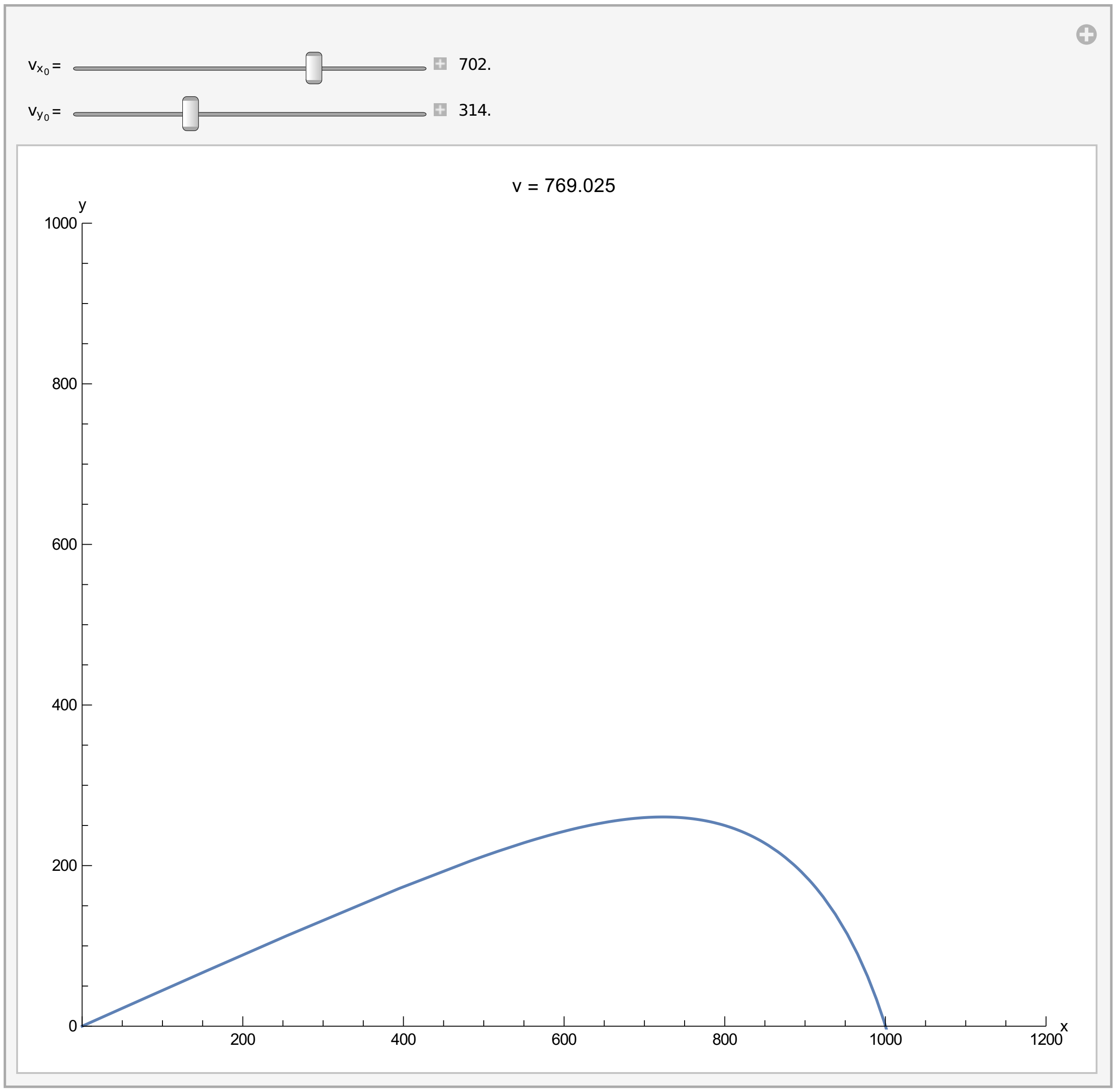
4.

```
In[1]:= Clear[vx0, vy0]
        sol = ParametricNDSolve[{x''[t] == -(ρ * r^2)/(2 * m) (x'[t] * Sqrt[x'[t]^2 + y'[t]^2]),
            y''[t] == -g - (ρ * r^2)/(2 * m) (y'[t] * Sqrt[x'[t]^2 + y'[t]^2]), x[0] == 0, x'[0] == vx0,
            y[0] == 0, y'[0] == vy0, WhenEvent[y[t] < 0, "StopIntegration"]},
          {x, y}, {t, 0, 50}, {vx0, vy0}, Method → "StiffnessSwitching"]

Out[2]= {x → ParametricFunction[ Expression: x Parameters: {vx0, vy0} ],
         y → ParametricFunction[ Expression: y Parameters: {vx0, vy0} ]}
```

In[55]:=
```
Clear[vx0, vy0]
r = 0.05;
m = 0.5;
ρ = 1.3;
g = 9.81;
Manipulate[ParametricPlot[Evaluate[{x[vx0, vy0][t], y[vx0, vy0][t]} /. sol],
   {t, 0, 30}, ImageSize → Large, AxesLabel → {"x", "y"},
   PlotRange → {{0, 1200}, {0, 1000}}, PlotLabel → Row@{"v = ", Sqrt[vx0^2 + vy0^2]}],
  {{vx0, 0, "vx₀="}, 0, 1000, Appearance → "Labeled"},
  {{vy0, 0, "vy₀="}, 0, 1000, Appearance → "Labeled"}]
```

Out[60]=



By playing around with sliders for v_{x_0} and v_{y_0} above, we observe that the components that minimize the initial velocity are $v_{x_0} \approx 702$, $v_{y_0} \approx 314$, giving v ≈ 769 and a firing angle of $\theta \approx 0.42_{\text{rad}}$.

5.

```
In[10]:= Clear[vx0, vy0]

In[11]:= yfnt[vx0_, vy0_, t_] := y[vx0, vy0][t] /. sol
         xfnt[vx0_, vy0_, t_] := x[vx0, vy0][t] /. sol

In[13]:= tfninit[vx0_, vy0_] := t /. FindRoot[yfnt[vx0, vy0, t] == 0, {t, 0.1, 30}] // Quiet

In[14]:= range[vx0_, vy0_] := xfnt[vx0, vy0, tfninit[vx0, vy0]]

In[15]:= range2[v_?NumericQ, θ_?NumericQ] := range[v * Cos[θ], v * Sin[θ]]

In[37]:= initv[x_, θ_] := Catch[Do[If[x - 0.5 < range2[v, θ] < x + 0.5, Throw[v]], {v, 100, 1000, 1}]]

In[67]:= minv[x_] :=
          Catch[Do[If[initv[x, θ] < initv[x, θ + 0.01], Throw[θ]], {θ, 0.20, 0.80, 0.01}]] // Quiet

In[38]:= initv2[x_, θ_] := v /. FindRoot[range2[v, θ] == x, {v, 100, 1000}]

In[45]:= minv2[x_] := θ /. FindMinimum[initv2[x, θ], {θ, 0.2}][[2]] // Quiet

In[69]:= {timing, vmin} = Timing[minv[1000]]

Out[69]= {109.67, 0.42}

In[70]:= initv[1000, vmin]

Out[70]= 768
```

We use the range for the optimal firing angle and velocity found by trial and error to define our iteration over a domain for which our functions defined above behave nicely (we can change the parameters of the functions above if we ever need to find the optimal firing angle and velocity for other ranges). Our iterations then give us an optimal firing angle of 0.42_{rad} and firing velocity of 768 ms^{-1}, not too far from our guess in part 4. We can easily confirm this using another implementation of the optimization function using built-in functions rather than defining new iterations. This implementation are the functions initv2 and minv2:

```
In[71]:= {timing2, vmin2} = Timing[minv2[1000]]

Out[71]= {0.847996, 0.410454}

In[72]:= initv2[1000, vmin2]

Out[72]= 768.2
```