Lab 5 - Grapefruit

```
In[1]:= Needs["PlotLegends`"]
        dir = NotebookDirectory[];
        SetDirectory[dir];

In[4]:= v0test = 10

Out[4]= 10

In[5]:= thetaMax = NMaximize[{v0test * Cos[th] * t,
            v0test * Sin[th] * t - .5 * 9.8 * t^2 == 0 , t ≥ 0}, {th, t}][[2, 1, 2]]

Out[5]= 0.785396

In[6]:= N[Pi / 4]

Out[6]= 0.785398
```

To show the graphs easily, we will paramiterize the functions and remove t. Let (x0,y0) = (0,0)

```
In[7]:= yX[x_, v0_, th_] := v0 * Sin[th] * x / (v0 * Cos[th]) - 1 / 2 * 9.8 * (x / (v0 * Cos[th])) ^2
```

Optimal firing angle is Pi/4. To reach a distance of 1000m, we need:

```
In[8]:= optV = Solve[yX[1000, v0, Pi / 4] == 0 && v0 > 0, {v0} ][[1, 1, 2]]

Out[8]= 98.9949

In[9]:= xOpt = optV * Cos[Pi / 4]

Out[9]= 70.

In[10]:= yOpt = optV * Sin[Pi / 4]

Out[10]= 70.

In[11]:= d[x0_, v0_, t_, th_] := x0 + v0 * Cos[th] * t

In[12]:= h[y0_, v0_, t_, th_] := y0 + v0 * Sin[th] * t - 1 / 2 * 9.8 * t^2
```

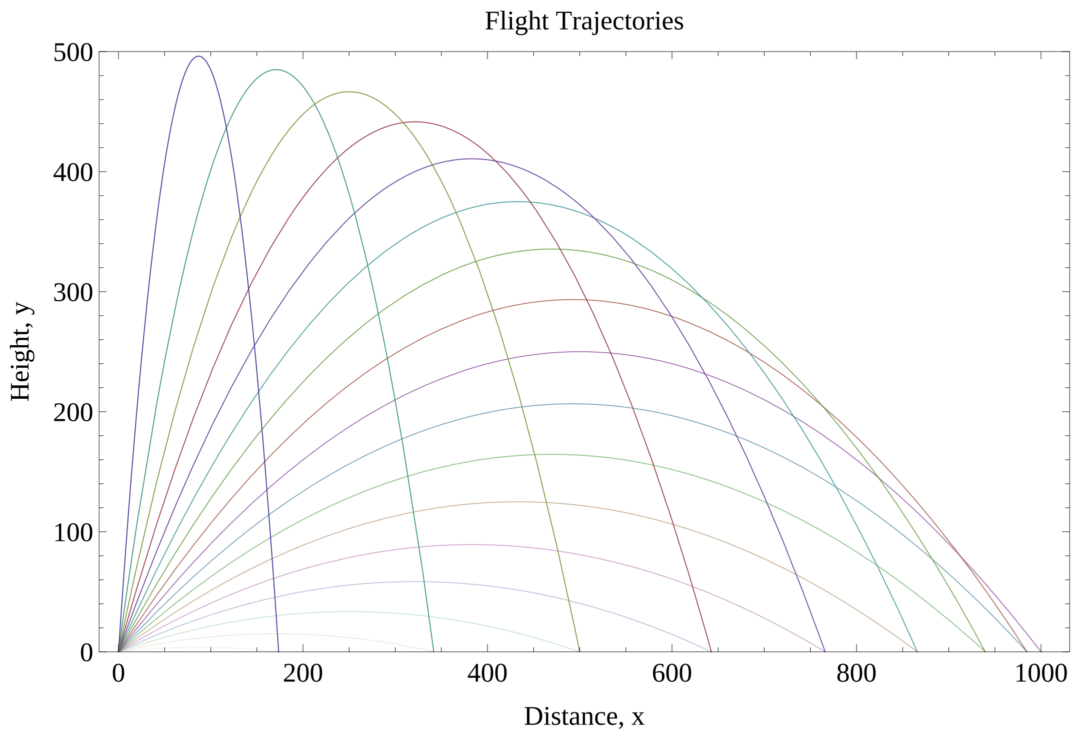
Flight distance in the abscence of drag is only dependent on the x component of the velocity, which is v0 times the cosine of the firing angle, and the time of flight, which is dependent on the y components v0 times sine of firing angle.

```
In[13]:= thetas = Table[th, {th, 0, Pi / 2 - Pi / 36, Pi / 36}]
```

Out[13]= $\left\{0, \frac{\pi}{36}, \frac{\pi}{18}, \frac{\pi}{12}, \frac{\pi}{9}, \frac{5\pi}{36}, \frac{\pi}{6}, \frac{7\pi}{36}, \frac{2\pi}{9}, \frac{\pi}{4}, \frac{5\pi}{18}, \frac{11\pi}{36}, \frac{\pi}{3}, \frac{13\pi}{36}, \frac{7\pi}{18}, \frac{5\pi}{12}, \frac{4\pi}{9}, \frac{17\pi}{36}\right\}$

```
In[14]:= data = Map[Function[th, yX[x, optV, th]], thetas];

In[15]:= ops = Table[
           Solve[optV * Sin[thetas[[i]]] * t - .5 * 9.8 * t^2 == 0 && t >= 0, t], {i, Length[thetas]}]

Out[15]= {{{t → 0.}, {t → 0.}, {t → 0.}, {t → 0.}}, {{t → 0.}, {t → 0.}, {t → 1.76081}},
          {{t → 0.}, {t → 0.}, {t → 3.50822}}, {{t → 0.}, {t → 0.}, {t → 5.22893}},
          {{t → 0.}, {t → 0.}, {t → 6.90985}}, {{t → 0.}, {t → 0.}, {t → 8.53818}},
          {{t → 0.}, {t → 0.}, {t → 10.1015}}, {{t → 0.}, {t → 0.}, {t → 11.588}},
          {{t → 0.}, {t → 0.}, {t → 12.9863}}, {{t → 0.}, {t → 0.}, {t → 14.2857}},
          {{t → 0.}, {t → 0.}, {t → 15.4764}}, {{t → 0.}, {t → 0.}, {t → 16.5494}},
          {{t → 0.}, {t → 0.}, {t → 17.4964}}, {{t → 0.}, {t → 0.}, {t → 18.3102}},
          {{t → 0.}, {t → 0.}, {t → 18.9847}}, {{t → 0.}, {t → 0.}, {t → 19.5146}},
          {{t → 0.}, {t → 0.}, {t → 19.8961}}, {{t → 0.}, {t → 0.}, {t → 20.1262}}}
```

```
In[16]:= ops = Map[Function[t, t[[3, 1, 2]]], ops]

Out[16]= {0., 1.76081, 3.50822, 5.22893, 6.90985, 8.53818, 10.1015, 11.588, 12.9863,
          14.2857, 15.4764, 16.5494, 17.4964, 18.3102, 18.9847, 19.5146, 19.8961, 20.1262}

In[17]:= ops = ops / Max[ops]

Out[17]= {0., 0.0874887, 0.174311, 0.259808, 0.343327, 0.424233, 0.50191, 0.575767, 0.645243,
          0.709808, 0.768971, 0.822281, 0.869333, 0.90977, 0.943282, 0.969616, 0.98857, 1.}

In[18]:= ops = Map[Opacity, ops]

Out[18]= {Opacity[0.], Opacity[0.0874887], Opacity[0.174311], Opacity[0.259808], Opacity[0.343327],
          Opacity[0.424233], Opacity[0.50191], Opacity[0.575767], Opacity[0.645243],
          Opacity[0.709808], Opacity[0.768971], Opacity[0.822281], Opacity[0.869333],
          Opacity[0.90977], Opacity[0.943282], Opacity[0.969616], Opacity[0.98857], Opacity[1.]}

In[19]:= graph = Plot[data, {x, 0, 1010}, PlotStyle → ops, PlotRange -> {0, 500}, Frame → True,
           FrameLabel → {{"Height, y", ""}, {"Distance, x", "Flight Trajectories"}},
           ImageSize → Large, LabelStyle → Larger ]

Out[19]=
```



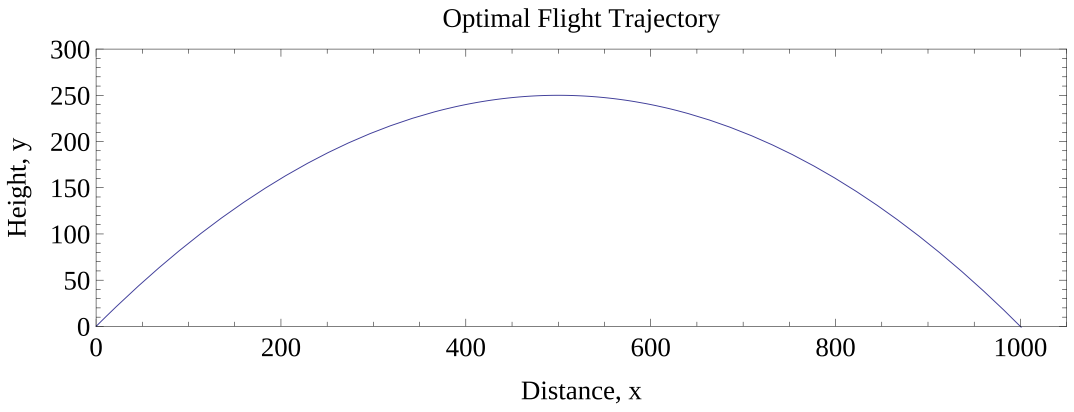
```
In[20]:= Export["trajectories.png", graph]

Out[20]= trajectories.png
```

Numerical Integration of DiffEQs:

```
In[21]:= eqs = {x''[t] == 0, y''[t] == -9.8}

Out[21]= {x″[t] == 0, y″[t] == -9.8}

In[22]:= ini = {x[0] == 0, y[0] == 0, x'[0] == 70, y'[0] == 70}

Out[22]= {x[0] == 0, y[0] == 0, x′[0] == 70, y′[0] == 70}

In[23]:= rules = NDSolve[Join[eqs, ini], {x, y}, {t, 0, 20}][[1]]

Out[23]= {x → InterpolatingFunction[{{0., 20.}}, <>], y → InterpolatingFunction[{{0., 20.}}, <>]}

In[24]:= xx[t_] := x[t] /. rules
```

```
In[25]:= yy[t_] := y[t] /. rules

In[26]:= graph = ParametricPlot[{xx[t], yy[t]},
           {t, 0, 16}, Frame → True, PlotRange → {{0, 1050}, {0, 300}},
           FrameLabel → {{"Height, y", ""}, {"Distance, x", "Optimal Flight Trajectory"}},
           ImageSize → Large, LabelStyle → Larger ]
```

Out[26]=



```
In[27]:= Export["trajectory.png", graph]

Out[27]= trajectory.png
```

With Drag

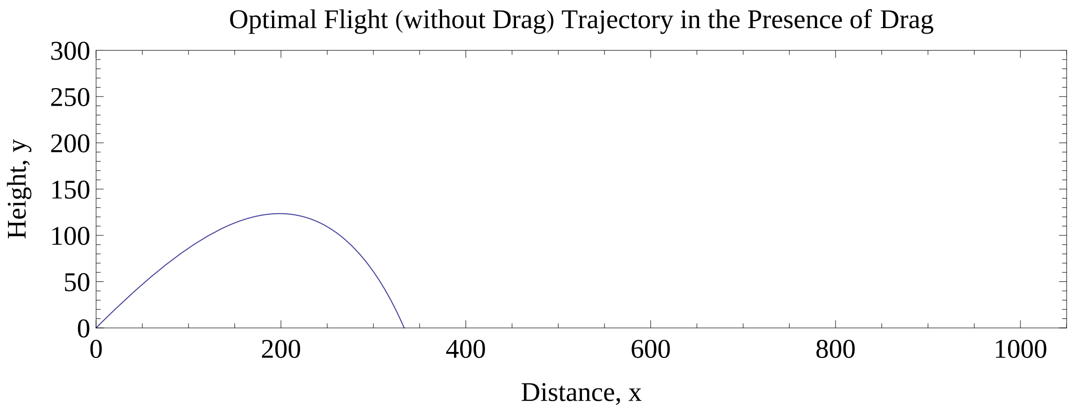
```
In[28]:= m = .5

Out[28]= 0.5

In[29]:= r = .05

Out[29]= 0.05

In[30]:= FDrag[v_] := -.5 * 1.3 * r^2 * v^2

In[31]:= dragOpt = FDrag[optV]

Out[31]= -15.925

In[32]:= eqsD = {x''[t] ==
            -Abs[FDrag[Sqrt[x'[t]^2 + y'[t]^2]]] * x'[t] / (m * Sqrt[x'[t]^2 + y'[t]^2]), y''[t] ==
            -9.8 - Abs[FDrag[Sqrt[x'[t]^2 + y'[t]^2]]] * y'[t] / (m * Sqrt[x'[t]^2 + y'[t]^2])}
```

Out[32]= $\left\{x''[t] == -\frac{0.00325\,\text{Abs}\left[x'[t]^2 + y'[t]^2\right]\,x'[t]}{\sqrt{x'[t]^2 + y'[t]^2}},\ y''[t] == -9.8 - \frac{0.00325\,\text{Abs}\left[x'[t]^2 + y'[t]^2\right]\,y'[t]}{\sqrt{x'[t]^2 + y'[t]^2}}\right\}$

```
In[33]:= iniD = {x[0] == 0, y[0] == 0, x'[0] == 70, y'[0] == 70}

Out[33]= {x[0] == 0, y[0] == 0, x′[0] == 70, y′[0] == 70}

In[34]:= rulesD = NDSolve[Join[eqsD, iniD], {x, y}, {t, 0, 20}][[1]]

Out[34]= {x → InterpolatingFunction[{{0., 20.}}, <>], y → InterpolatingFunction[{{0., 20.}}, <>]}

In[35]:= xx[t_] := x[t] /. rulesD

In[36]:= yy[t_] := y[t] /. rulesD
```

```
In[37]:= graph = ParametricPlot[{xx[t], yy[t]}, {t, 0, 16}, Frame → True,
           PlotRange → {{0, 1050}, {0, 300}}, FrameLabel → {{"Height, y", ""}, {"Distance, x",
              "Optimal Flight (without Drag) Trajectory in the Presence of Drag"}},
           ImageSize → Large, LabelStyle → Larger ]
```

Out[37]=



We get nowhere close to the 1000m target, instead falling to the ground at about 330m.

```
In[38]:= Export["drag.png", graph]

Out[38]= drag.png
```

Guess and Check (change the initial Conditions)

```
In[39]:= iniD = {x[0] == 0, y[0] == 0, x'[0] == vX, y'[0] == vY}

Out[39]= {x[0] == 0, y[0] == 0, x′[0] == vX, y′[0] == vY}

In[40]:= V[th_, v_] := {vX =  Cos[th] * v, vY = Sin[th] * v}

In[41]:= v0 = 800

Out[41]= 800

In[42]:= th0 = Pi / 4
```

Out[42]= $\frac{\pi}{4}$

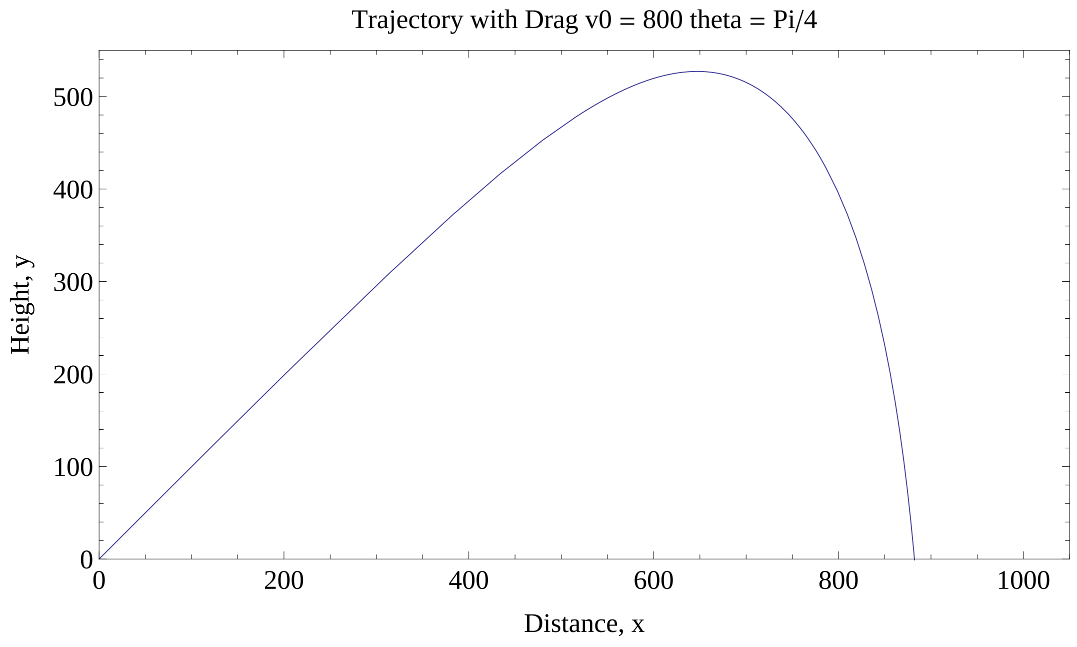
```
In[43]:= V[th0, v0]
```

Out[43]= $\left\{400\sqrt{2},\ 400\sqrt{2}\right\}$

```
In[44]:= rulesD = NDSolve[Join[eqsD, iniD], {x, y}, {t, 0, 75}][[1]]

Out[44]= {x → InterpolatingFunction[{{0., 75.}}, <>], y → InterpolatingFunction[{{0., 75.}}, <>]}

In[45]:= xx[t_] := x[t] /. rulesD

In[46]:= yy[t_] := y[t] /. rulesD
```

```
In[47]:= graph = ParametricPlot[{xx[t], yy[t]}, {t, 0, 30},
           Frame → True, PlotRange → {{0, 1050}, {0, 550}}, FrameLabel →
            {{"Height, y", ""}, {"Distance, x", "Trajectory with Drag v0 = " <> ToString[v0] <>
               " theta = " <> ToString[th0, InputForm]}}, ImageSize → Large, LabelStyle → Larger ]
```

Out[47]=



```
In[48]:= Export["dragGuess1.png", graph]

Out[48]= dragGuess1.png

In[49]:= v0 = 800

Out[49]= 800

In[50]:= th0 = Pi / 8
```

Out[50]= $\frac{\pi}{8}$

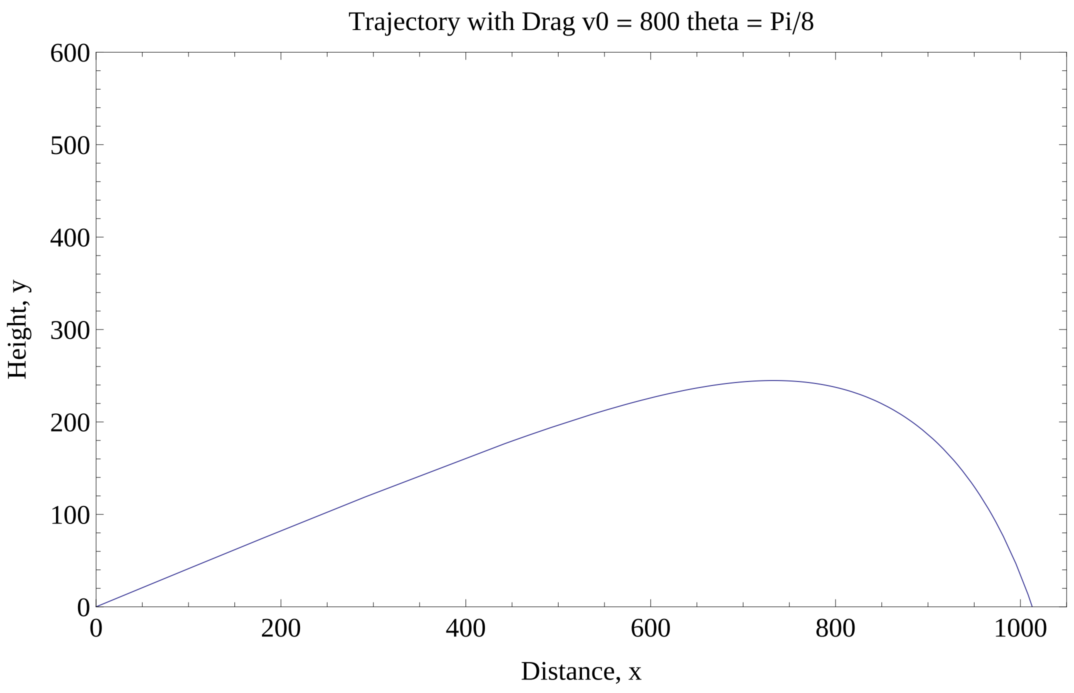
```
In[51]:= V[th0, v0]
```

Out[51]= $\left\{800\ \mathrm{Cos}\left[\frac{\pi}{8}\right],\ 800\ \mathrm{Sin}\left[\frac{\pi}{8}\right]\right\}$

```
In[52]:= rulesD = NDSolve[Join[eqsD, iniD], {x, y}, {t, 0, 75}][[1]]

Out[52]= {x → InterpolatingFunction[{{0., 75.}}, <>], y → InterpolatingFunction[{{0., 75.}}, <>]}

In[53]:= xx[t_] := x[t] /. rulesD

In[54]:= yy[t_] := y[t] /. rulesD
```

```
In[55]:= graph = ParametricPlot[{xx[t], yy[t]}, {t, 0, 35},
           Frame → True, PlotRange → {{0, 1050}, {0, 600}}, FrameLabel →
            {{"Height, y", ""}, {"Distance, x", "Trajectory with Drag v0 = " <> ToString[v0] <>
               " theta = " <> ToString[th0, InputForm]}}, ImageSize → Large, LabelStyle → Larger]
```

Out[55]=



```
In[56]:= Export["dragGuess2.png", graph]

Out[56]= dragGuess2.png

In[57]:= v0 = 780

Out[57]= 780

In[58]:= th0 = Pi / 9
```

Out[58]= $\frac{\pi}{9}$

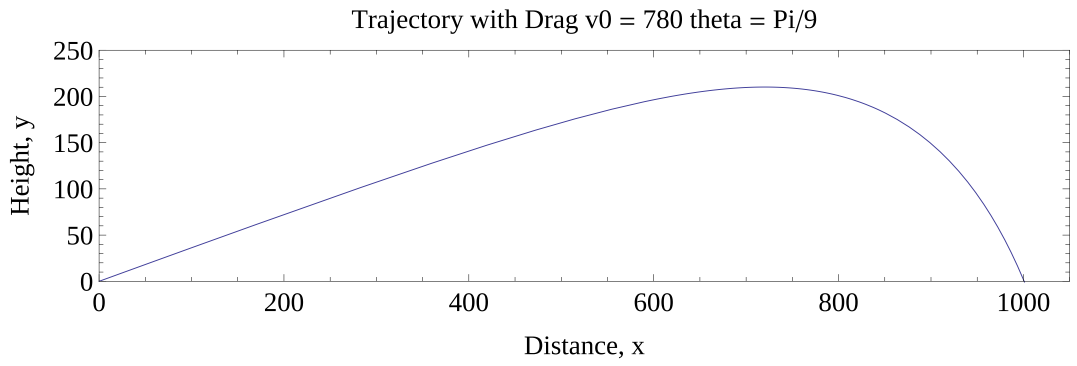
```
In[59]:= V[th0, v0]
```

Out[59]= $\left\{780 \operatorname{Cos}\left[\frac{\pi}{9}\right], 780 \operatorname{Sin}\left[\frac{\pi}{9}\right]\right\}$

```
In[60]:= rulesD = NDSolve[Join[eqsD, iniD], {x, y}, {t, 0, 75}][[1]]

Out[60]= {x → InterpolatingFunction[{{0., 75.}}, <>], y → InterpolatingFunction[{{0., 75.}}, <>]}

In[61]:= xx[t_] := x[t] /. rulesD

In[62]:= yy[t_] := y[t] /. rulesD
```

```
In[63]:= graph = ParametricPlot[{xx[t], yy[t]}, {t, 0, 16},
           Frame → True, PlotRange → {{0, 1050}, {0, 250}}, FrameLabel →
            {{"Height, y", ""}, {"Distance, x", "Trajectory with Drag v0 = " <> ToString[v0] <>
               " theta = " <> ToString[th0, InputForm]}}, ImageSize → Large, LabelStyle → Larger]
```

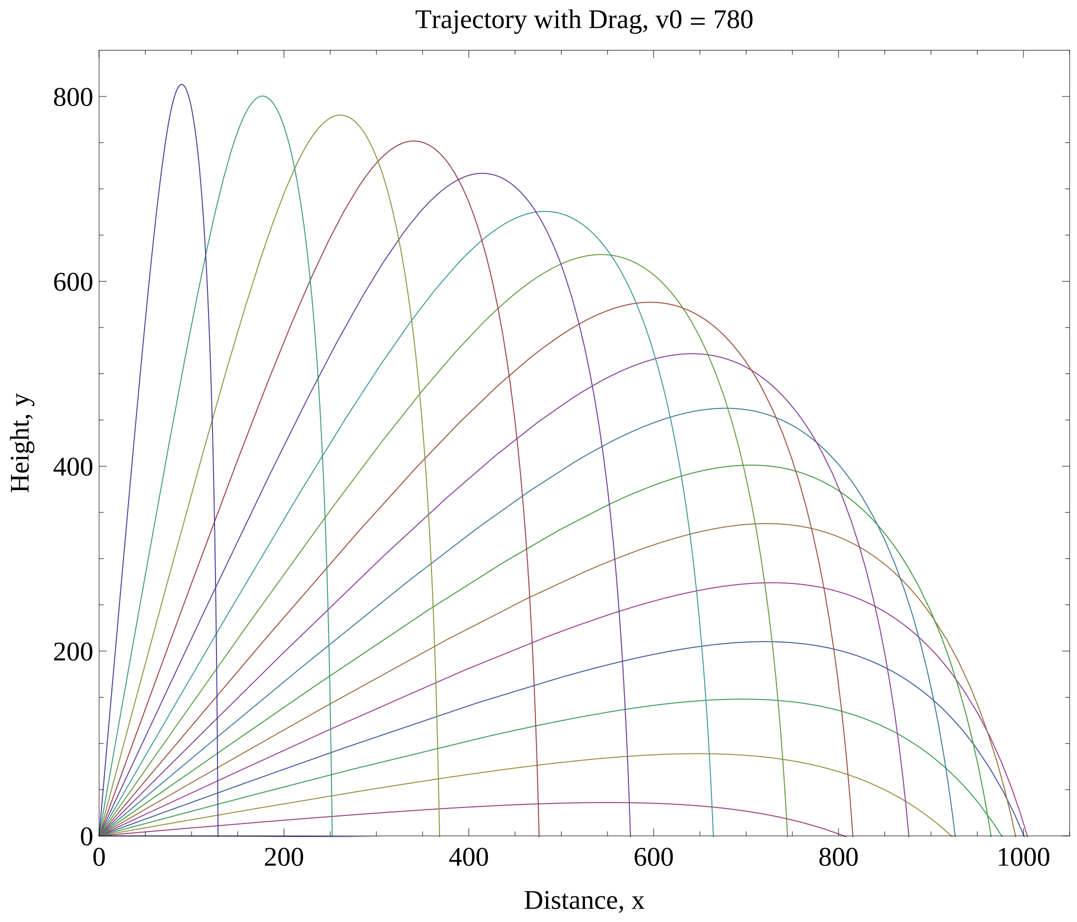
Out[63]=



```
In[64]:= Export["dragGuess3.png", graph]

Out[64]= dragGuess3.png

In[65]:= initD[th_] := {x[0] == 0, y[0] == 0, x'[0] == v0 * Cos[th], y'[0] == v0 * Sin[th]}

In[66]:= initDs = Map[initD, thetas];

In[67]:= conds = Map[Function[x, Join[eqsD, x]], initDs];

In[68]:= rulesDs = Map[Function[c, NDSolve[c, {x, y}, {t, 0, 200}][[1]]], conds];

In[69]:= xxs[t_] := Table[x[t] /. rulesDs[[i]], {i, Length[rulesDs]}]

In[70]:= yys[t_] := Table[y[t] /. rulesDs[[i]], {i, Length[rulesDs]}]

In[71]:= paraPlot = Transpose[{xxs[t], yys[t]}];
```

```
In[72]:= graph = ParametricPlot[paraPlot, {t, 0, 30},
           Frame → True, PlotRange → {{0, 1050}, {0, 850}}, FrameLabel →
            {{"Height, y", ""}, {"Distance, x", "Trajectory with Drag, v0 = " <> ToString[v0]}},
           ImageSize → Large, LabelStyle → Larger]
```

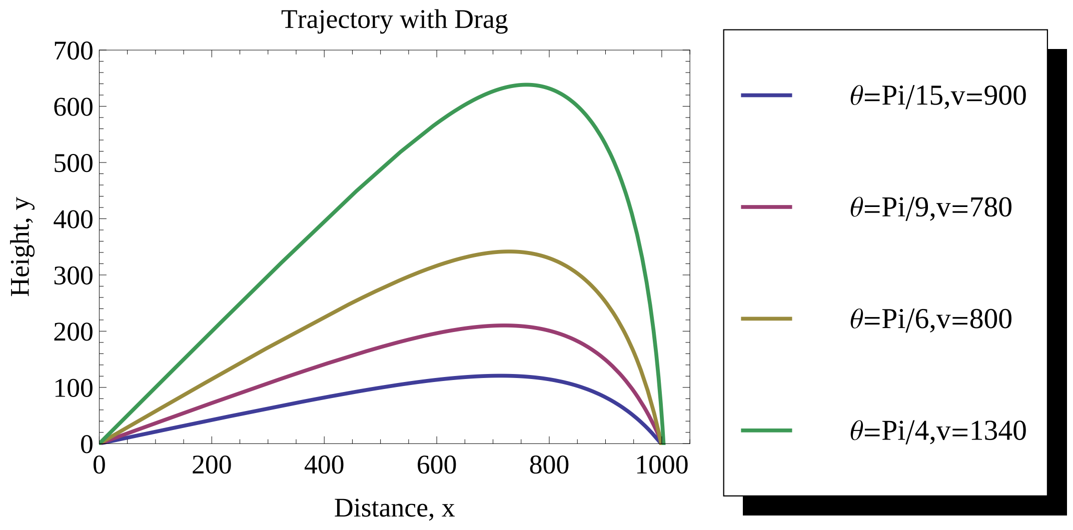
Out[72]=



```
In[73]:= Export["dragTrajectories.png", graph]

Out[73]= dragTrajectories.png

In[74]:= init1000[th_, v0_] := {x[0] == 0, y[0] == 0, x'[0] == v0 * Cos[th], y'[0] == v0 * Sin[th]}

In[75]:= inits = Map[Function[x, init1000[x[[1]], x[[2]]]],
            {{Pi / 15, 900}, {Pi / 9, 780}, {Pi / 6, 800}, {Pi / 4, 1340}}];

In[76]:= cond1000 = Map[Function[x, Join[eqsD, x]], inits];

In[77]:= rules1000 = Map[Function[c, NDSolve[c, {x, y}, {t, 0, 300}][[1]]], cond1000];

In[78]:= xx1000[t_] := Table[x[t] /. rules1000[[i]], {i, Length[rules1000]}]

In[79]:= yy1000[t_] := Table[y[t] /. rules1000[[i]], {i, Length[rules1000]}]

In[80]:= paraPlot = Transpose[{xx1000[t], yy1000[t]}];
```

```
In[81]:= graph = ParametricPlot[{paraPlot[[1]], paraPlot[[2]], paraPlot[[3]], paraPlot[[4]]},
           {t, 0, 40}, Frame → True, PlotRange → {{0, 1050}, {0, 700}},
           FrameLabel → {{"Height, y", ""}, {"Distance, x", "Trajectory with Drag" }},
           ImageSize → Large, LabelStyle → Larger, PlotLegend → {Style["θ=Pi/15,v=900", 15],
             Style["θ=Pi/9,v=780", 15], Style["θ=Pi/6,v=800", 15], Style["θ=Pi/4,v=1340", 15]},
           LegendPosition → {.85, -.6}, LegendSize → 1.2, PlotStyle → Thick]
```

Out[81]=



```
In[82]:= Export["1000Drag.png", graph]

Out[82]= 1000Drag.png
```

Better Methods

a)

```
In[83]:= eqsD = {x''[t] ==
             -Abs[FDrag[Sqrt[x'[t]^2 + y'[t]^2]]] * x'[t] / (m * Sqrt[x'[t]^2 + y'[t]^2]), y''[t] ==
             -9.8 - Abs[FDrag[Sqrt[x'[t]^2 + y'[t]^2]]] * y'[t] / (m * Sqrt[x'[t]^2 + y'[t]^2])};

In[84]:= init[v0_, th_] := {x[0] == 0, y[0] == 0, x'[0] == v0 * Cos[th], y'[0] == v0 * Sin[th]}

In[85]:= rule[v0_, th_] := NDSolve[Join[eqsD, init[v0, th]], {x, y}, {t, 0, 100}][[1]]

In[86]:= xxFunc[t_, v0_, th_] := x[t] /. rule[v0, th]

In[87]:= yyFunc[t_, v0_, th_] := y[t] /. rule[v0, th]

In[88]:= time[v0_, th_] := t /. FindRoot[yyFunc[t, v0, th], {t, 20}]
```

b)

```
In[89]:= range[v0_, th_] := xxFunc[time[v0, th], v0, th]
```

c)

```
In[90]:= initVel[th_, rng_] := Module[{nextRng = 0},
           For[v = 1, nextRng < rng, v = v + 1,
            nextRng = range[v + 1, th];
            If[nextRng > rng, Return[v + 1]]]]
```

d)

```
In[143]:= initAngle[rng_] := Module[{prevVel = initVel[Pi / 180, rng], nextVel = 0, th2 = 0},
            For[th = 5 * Pi / 180, th < Pi / 2, th = th + Pi / 180,
             nextVel = initVel[th + Pi / 180, rng];
             If[prevVel ≤ nextVel, Return[th + Pi / 180]];
                prevVel = nextVel;]]
```

```
In[92]:=  time[780, Pi / 9]

Out[92]=  12.1243

In[93]:=  range[780, Pi / 9]

Out[93]=  1000.77

In[127]:= initVel[Pi / 180, 1000]

Out[127]= 2958

In[144]:= initAngle[1000]
```

Out[144]= $\frac{5\pi}{36}$

```
In[149]:= range[770, 5 * Pi / 36]

Out[149]= 1000.17

In[150]:= Export["grapefruit.pdf", EvaluationNotebook[]]

Out[95]=  grapefruit.pdf
```