

# Range of a projectile, including air resistance.

---

## Introduction

Here we study the motion of a projectile thrown through the air, including the important effects of air resistance. We will investigate how the maximum distance the projectile travels before hitting the ground (optimized with respect to the initial angle at a fixed initial speed) varies as a function of the friction coefficient.

Neglecting air resistance, it is easy to show (elementary physics classes) that if we throw a projectile with a speed  $v$  at an angle  $\theta$  to the horizontal (angle of throw), that its trajectory is a parabola, it reaches the ground after a time  $t_0$ , and it has then traveled a horizontal distance  $x_{\max}$  where

$$t_0 = \frac{2 v \sin \theta}{g}, \quad x_{\text{final}} = \frac{v^2 \sin 2 \theta}{g}.$$

The maximum distance traveled by the ball is clearly for  $\theta = \pi/4$ , and is equal to  $x_{\max} = v^2/g$ . Here we wish to see how the optimal angle changes from  $\pi/4$  if there is air resistance, and how the maximum distance is reduced from  $v^2/g$ .

Clearly we need to have a model for the force due to air resistance, which is a complicated problem. For an object moving slowly through a viscous medium (e.g. molasses) the flow is smooth (laminar) and the force is proportional to the velocity. However, in air, except for a really tiny velocity, the flow of the air is turbulent and a better approximation is that the force is proportional to the square of the velocity (note that this is still an approximation). Clearly the direction of the force is opposite to the velocity. Hence we write:

$$\mathbf{F} = -k |\mathbf{v}| \mathbf{v},$$

or

$$F_x = -k \sqrt{v_x^2 + v_y^2} v_x, \quad F_y = -k \sqrt{v_x^2 + v_y^2} v_y$$

where  $k$  is a parameter. I emphasize that this is only a rough model for the friction of a projectile in air, but it should give us an idea of the effects of friction.

First of all we clear any variables previously defined, give the value of  $g$ , 9.81, and initial values for  $v$ ,  $\theta$ , and the friction parameter  $k$  (20,  $\pi/4$ , and 0.3 respectively):

```
In[1]:= Clear["Global`*"]
```

```
In[2]:= g = 9.81;
```

```
In[3]:= v = 20; theta = Pi / 4;
```

```
In[4]:= k = 0.3;
```

---

## Fixed initial speed and angle of throw $\theta$ .

First we will fix the initial speed and angle of throw, and see how far the particle travels before it hits the ground. This distance is call the **range**.

Next, we integrate the equations of motion using `NDSolve` up to time  $t = 10$ , which is sufficient for the projectile to hit the ground with initial speed  $v = 20$ .

```
In[5]:= sol = NDSolve [ { x''[t] == -k x'[t] Sqrt[y'[t]^2 + x'[t]^2] ,
  y''[t] == -k y'[t] Sqrt[y'[t]^2 + x'[t]^2] - g, x[0] == 0, y[0] == 0,
  x'[0] == v Cos[theta], y'[0] == v Sin[theta] }, {x, y}, {t, 0, 10}]
```

```
Out[5]= {{x -> InterpolatingFunction[{{0., 10.}}, <>], y -> InterpolatingFunction[{{0., 10.}}, <>]}}
```

It is convenient to determine the time  $t_{\text{final}}$  when the particle hits the ground, i.e.  $y[t] == 0$ . This is done using the `FindRoot` command. However, it is convenient to first convert  $y[t]$  to a function rather than the replacement rule in the last output. This can be done for both  $x$  and  $y$  by

```
In[6]:= yy[t_] = y[t] /. sol[[1]]; xx[t_] = x[t] /. sol[[1]];
```

in which we removed a curly bracket by taking the first element (even though there's only 1). We also needed to take different variable names, so we use `xx` and `yy` rather than `x` and `y`. As an example, for  $t = 1$  we get

```
In[7]:= yy[1]
```

```
Out[7]= 1.74936
```

`tfinal` is given by the root of `yy[t]`. However, the result of `FindRoot[yy[t] == 0, {t, 0.1, 4}]`, is again a replacement rule, whereas we want the numerical value. Hence the desired command is:

```
In[8]:= tfinal = t /. FindRoot[yy[t] == 0, {t, 0.1, 4}]
```

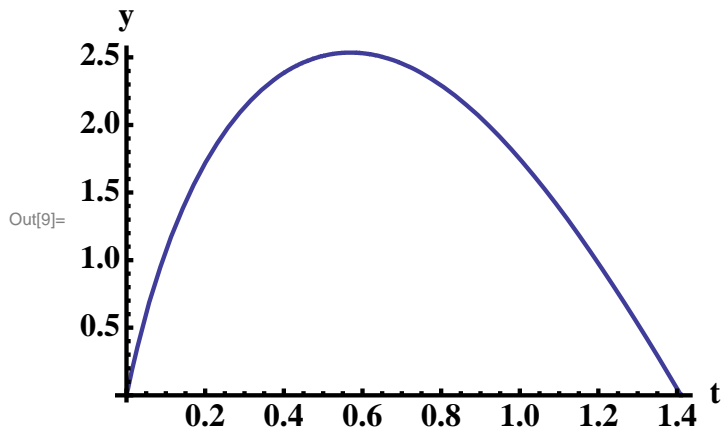
```
Out[8]= 1.41016
```

This example is showing us that, for all its power, *Mathematica* is not as intuitive as one would like.

**I emphasize that it is generally convenient to turn the replacement rule coming from the solution of a differential equation into a function, and the replacement rule from the solution of an equation into a number.**

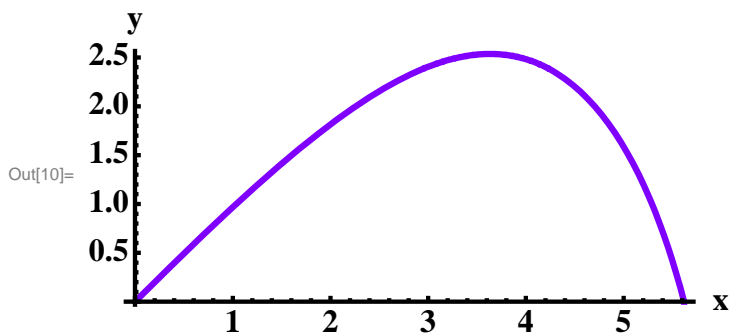
Now we can, for example, plot the height,  $y$ , as a function of time up to the time  $t_{\text{final}}$  at which the projectile hits the ground:

```
In[9]:= Plot[yy[t], {t, 0, tfinal}, AxesLabel -> {"t", "y"}]
```



and also y against x (using `ParametricPlot`, in which, just for amusement, I've defined a non-standard `PlotStyle` for the line)

```
In[10]:= ParametricPlot[{xx[t], yy[t]}, {t, 0, tfinal},
  PlotStyle -> {{Hue[0.75], AbsoluteThickness[3]}}, AxesLabel -> {"x", "y"}]
```



Without air resistance this curve would be a parabola which is symmetric about the maximum. However, we see that with air resistance the curve is no longer symmetric about the maximum, and the particle drops to the ground almost vertically. This is in agreement with our experience.

The horizontal distance traveled when the particle hits the ground is called the **range**. We can determine it from `xx[tfinal]`:

```
In[11]:= xx[tfinal]
```

Out[11]= 5.61992

As expected this is less than the distance traveled without friction  $v^2/g$ .

```
In[21]:= v^2 / g
```

Out[21]= 40.7747

---

## Fixed initial speed but vary the angle of throw $\theta$ .

Next we are going to vary the initial angle  $\theta$  to determine the maximum range of the projectile, for a given initial speed, in the presence of air resistance. To do this we first determine the time,  $t_{\text{final}}(\theta)$  at which the projectile hits the ground as a function of  $\theta$ . We define a function for doing this, which is just a combination of the above commands for integrating the equations and determining the time at which the particle hits the ground. Before that, however, we need to remove the previous expressions for `tfinal` and `theta`:

```
In[12]:= Clear[tfinal, theta]

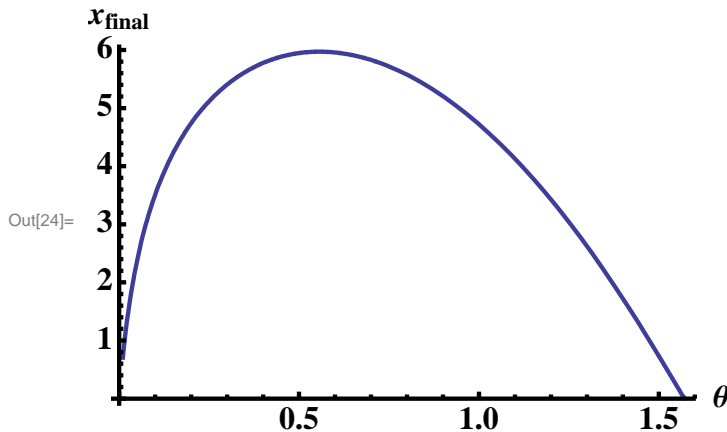
In[22]:= tfinal[theta_] := (sol = NDSolve[{x'[t] == -k x'[t] Sqrt[y'[t]^2 + x'[t]^2],
    y'[t] == -k y'[t] Sqrt[y'[t]^2 + x'[t]^2] - g, x[0] == 0, y[0] == 0,
    x'[0] == v Cos[theta], y'[0] == v Sin[theta]}, {x, y}, {t, 0, 10}];
  yy[t_] = y[t] /. sol[[1]]; xx[t_] = x[t] /. sol[[1]];
  t /. FindRoot[yy[t], {t, 1, 4}, MaxIterations -> 50])
```

The range, i.e. the distance traveled,  $x_{\text{final}}(\theta)$ , is easily obtained as  $x[t_{\text{final}}(\theta)]$ . We need to indicate that  $\theta$ , the argument of `xfinal`, must be numeric to avoid problems with the subsequent **FindMaximum** command (see below) when using Version 5 or later (this is most annoying):

```
In[23]:= xfinal[theta_?NumericQ] := xx[tfinal[theta]]
```

We can now plot the range,  $x_{\text{final}}$ , versus  $\theta$ :

```
In[24]:= Plot[xfinal[theta], {theta, 0.01, 1.57}, AxesLabel -> {" $\theta$ ", " $x_{\text{final}}$ "}]
```



Note that the range increases rapidly as  $\theta$  increases and reaches a maximum at a value *less* than  $\pi/4 = 0.784\dots$  (which is the value without air resistance). I would say that this is also in agreement with our experience.

---

## Optimize with respect to $\theta$ .

We would like to know what is the choice of  $\theta$  which maximises the range of the projectile. We will call the maximum range `xmax`. We locate the maximum with the *Mathematica* function **FindMaximum**

```
In[16]:= FindMaximum[xfinal[theta], {theta, 0.1, 1.3}]
```

```
Out[16]= {5.971, {theta -> 0.556149}}
```

Note that we have to give two starting values because the derivative of **xfinal[theta]** is not known. The maximum range, **xmax**, obtained by optimizing with respect to the initial angle  $\theta$ , is the first element of this list:

```
In[17]:= FindMaximum[xfinal[theta], {theta, 0.5, 0.6}][[1]]
```

```
Out[17]= 5.971
```

Now lets combine everything together to determine **xmax** as a function of the friction parameter **k** (assuming the same initial speed  $v = 20$ ).

```
In[18]:= xmax[k_] :=
  (tfinal[theta_] := (sol = NDSolve[{x'[t] == -k x'[t] Sqrt[y'[t]^2 + x'[t]^2}, y'[t] ==
    -k y'[t] Sqrt[y'[t]^2 + x'[t]^2] - g, x[0] == 0, y[0] == 0, x'[0] == v Cos[theta],
    y'[0] == v Sin[theta]}, {x, y}, {t, 0, 10}]; yy[t_] = y[t] /. sol[[1]];
    xx[t_] = x[t] /. sol[[1]]; t /. FindRoot[yy[t], {t, 1, 4}, MaxIterations -> 50]);
  xfinal[theta_?NumericQ] := xx[tfinal[theta]];
  FindMaximum[xfinal[theta], {theta, 0.1, 1.3}][[1]])
```

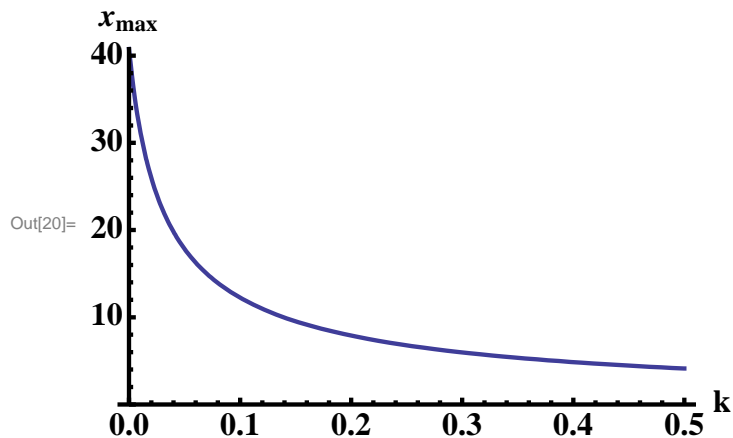
For example we can recover the exact result,  $v^2/g$  for  $k = 0$  (remember  $v$ , the initial speed, is 20 and  $g = 9.81$ )

```
In[19]:= {xmax[0], v^2/g}
```

```
Out[19]= {40.7747, 40.7747}
```

Finally we plot the **xmax** for a certain range of **k**:

```
In[20]:= Plot[xmax[k], {k, 0, 0.5}, AxesLabel -> {"k", "x_max"}, PlotRange -> {0, 41}]
```



This plot, which shows the maximum distance traveled by the projectile optimized with respect to  $\theta$ , as a function of the friction constant **k**, summarizes the results of this handout. The maximum distance falls off quite rapidly as the friction coefficient **k** increases. With more time it would be interesting to study this behavior in greater detail.

This notebook shows that non-trivial results can be obtained in *Mathematica*, and then plotted, with a few quite simple commands.