

# EULER' S METHOD APPLIED TO TRAJECTORY PROBLEMS

Now that we are familiar with using Euler's method and recursion techniques to solve differential equations, let's see how to apply this to trajectory problems. Here, we will start with the very simple case of motion in a uniform gravitational field with no friction. After we set up the basic code to solve this problem, we will learn a "trick" to allow us to end our numerical integration when the projectile hits the ground.

Consider a projectile launched at an angle  $\theta$  with initial velocity  $v_o$  on a level plane. We know the equations of motion are:

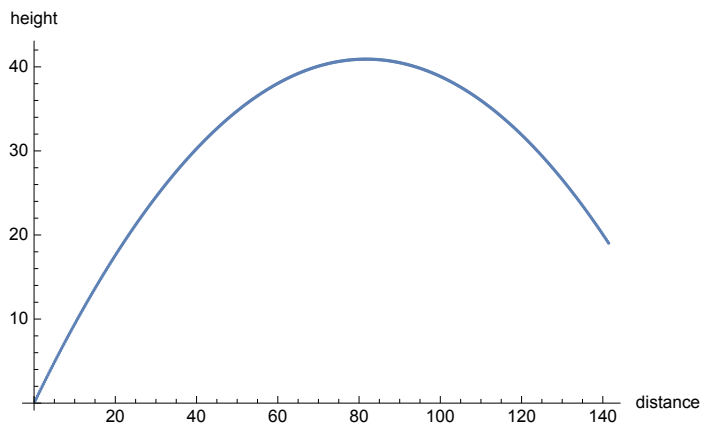
$$\begin{aligned}x(t) &= v_o \cos \theta t \\ y(t) &= v_o \sin \theta t - \frac{1}{2} g t^2\end{aligned}$$

where these symbols have familiar meanings. To solve the trajectory using Euler's method and recursion relations:

```

Clear[x, y, vx, vy, h, g]
x[0] = 0;
y[0] = 0; g = 9.81; h = 0.01; v0 = 40;
 $\theta = \pi / 4$ ;
vy[0] = v0 Sin[ $\theta$ ];
(* I am setting the launch point at the origin, and launching the
   projectile with an initial velocity of 40m/s at an angle of 45° *)
x[n_] := x[n] = x[n - 1] + v0 Cos[ $\theta$ ] h
vy[n_] := vy[n] = vy[n - 1] - h g
y[n_] := y[n] = y[n - 1] + h vy[n - 1]
ListPlot[Table[{x[n], y[n]}, {n, 500}], AxesLabel → {"distance", "height"}]

```



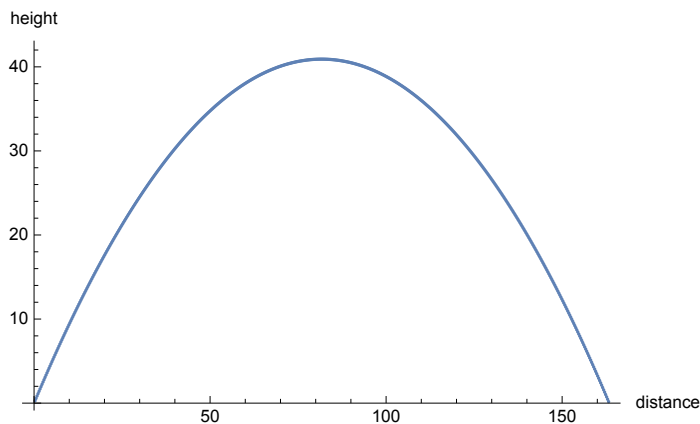
Notice how  $x[n]$  and  $y[n]$  are handled in the code. In the absence of friction, the motion in the  $x$  direction is constant, so I do not include an acceleration term in  $x$ . However, the motion in  $y$  is accelerated, so I need to include an acceleration term and cannot treat the velocity in the  $y$  direction as a constant. Note the inclusion of a function to calculate  $vy$  at different points.

The plot above shows what looks like a trajectory, but we were off a bit in our guess of how many data points to plot. How could we write our code to ensure that the trajectory goes all the way to the horizon without going below it. We could test various values of  $n$  by trial and error, but this is tedious and requires a new set of trials every time we change launch parameters. What we want to do is figure out a way to break out of the Do loop (or For loop or While loop) as soon as our projectile reaches the ground. Read the doc center descriptions of Catch and Throw, and see why this addition to our code provides the solution we seek:

```

Clear[x, y, vx, vy, h, g, nterms]
x[0] = 0;
y[0] = 0; g = 9.81; h = 0.01; v0 = 40;
θ = π / 4;
vy[0] = v0 Sin[θ];
(* I am setting the launch point at the origin, and launching the
   projectile with an initial velocity of 40m/s at an angle of 45° *)
x[n_] := x[n] = x[n - 1] + v0 Cos[θ] h
vy[n_] := vy[n] = vy[n - 1] - h g
y[n_] := y[n] = y[n - 1] + h vy[n - 1]
nterms = Catch[Do[If[y[n] < 0, Throw[n - 1]], {n, 10 000}]];
ListPlot[Table[{x[n], y[n]}, {n, nterms}], AxesLabel → {"distance", "height"}]

```



I define a variable “nterms” that specifies the number of data points that will be plotted. The Do loop that defines nterms checks each value of y, and halts calculation when  $y < 0$ , in other words, when the projectile falls below the horizon. We can use nterms to find the time of flight; since we know each time step is 0.01s, we can easily compute:

```
Print["time of flight = ", h nterms, " seconds"]
```

```
time of flight = 5.77 seconds
```

Which is the value you would get from the standard kinematics equation  $t = 2 v_o \sin \theta / g$

Read this note carefully, on the next homework, I will ask you to amend this code to include the effects of linear friction, i.e., a frictional force that is proportional to the velocity.