STUDENT VERSION

Euler and How Far Can You Throw a Ball?

Chris McCarthy
Department of Mathematics
BMCC City University of New York
New York, NY USA

STATEMENT

If a tennis ball of mass m is thrown through the air it will eventually hit the ground due to gravity. If you can throw a tennis ball 12 meters/second (about 26.8 mph) how far can you throw it; meaning how far away from you can you make it land? Assume that when the ball leaves your hand it is at a height of 2 meters (about 6 feet). With the help of Euler's Method, write a short script (Python, Matlab, R, etc.) to find the ideal launch angle θ to throw the ball, so that it will result in the ball landing as far as possible from you. Also find that maximal distance and plot that trajectory. Make sure to include air resistance (drag) in your model. Don't worry about rotational effects.

Physics Background and Euler's Method Review. Let $x(t)$ and $y(t)$ represent a ball's position at time t, in the x, y plane, with y being height. Vectors will be denoted by a **bold** font. The ball's velocity is:

$$\mathbf{v} = \frac{d}{dt}\langle x, y\rangle = \langle \dot{x}, \dot{y}\rangle = \langle v_x, v_y\rangle. \tag{1}$$

The ball's acceleration is:

$$\mathbf{a} = \frac{d}{dt}\langle v_x, v_y\rangle = \langle \ddot{x}, \ddot{y}\rangle = \langle \dot{v}_x, \dot{v}_y\rangle. \tag{2}$$

From Newton's Second Law of Motion we know that Force = mass × acceleration:

$$\mathbf{F} = m\mathbf{a} \tag{3}$$

For our models we will approximate the earth as being flat with gravity pointing straight down (in the negative y direction) and so we will approximate the force of gravity acting on a mass m as:

$$\mathbf{F_g} = \langle 0, -mg\rangle \tag{4}$$

with $g = 9.8$ meters/second^2 in the MKS system.

Simple Model for a Thrown Ball (no drag). To start with, we will ignore drag. Once we understand this simple model, we can make it more realistic by adding drag.

Combining (2), (3), and (4) we get the following second order ODE (which neglects drag):

$$\mathbf{F_g} = \langle 0, -mg \rangle = m\langle \ddot{x}, \ddot{y} \rangle = m\mathbf{a} \tag{5}$$

Writing the $\ddot{x}$ and $\ddot{y}$ parts of (5) on separate lines we get the 2nd order system of ODE's:

$$\begin{aligned} m\ddot{x} &= 0 \\ m\ddot{y} &= -mg \end{aligned} \tag{6}$$

To apply Euler's method, we need to convert (6) to a first order system of ODE's. We do the usual substitution. As in (1) and (2), we let $\dot{x} = v_x$, so that $\ddot{x} = \dot{v}_x$; we let $\dot{y} = v_y$, so that $\ddot{y} = \dot{v}_y$; and we divide by m. This gives us the first order version of (6):

$$\begin{aligned} \dot{x} &= v_x \\ \dot{y} &= v_y \\ \dot{v}_x &= 0 \\ \dot{v}_y &= -g \end{aligned} \tag{7}$$

We can rewrite (7) in vector form as:

$$\frac{d}{dt}\begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix} = \begin{pmatrix} v_x \\ v_y \\ 0 \\ -g \end{pmatrix} \tag{8}$$

We can linearly approximate the state of the system $\begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix}$ at time $t + \Delta t$ using information about the system at time t:

$$\begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix}_{\text{at } t+\Delta t} \approx \begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix}_{\text{at } t} + \frac{d}{dt}\begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix}_{\text{at } t} \cdot \Delta t \tag{9}$$

It is convenient[1] to explicitly include time t in the linear approximation. When we explicitly include

[1]Explicitly including time allows us to easily keep track of time while doing Euler's method calculations in vector form. It is especially convenient for when we have to deal with non-autonomous ODE's. The ODE model discussed in this paper is autonomous.

time t (9) becomes:

$$\begin{pmatrix} x \\ y \\ v_x \\ v_y \\ t \end{pmatrix}_{\text{at } t+\Delta t} \approx \begin{pmatrix} x \\ y \\ v_x \\ v_y \\ t \end{pmatrix}_{\text{at } t} + \frac{d}{dt}\begin{pmatrix} x \\ y \\ v_x \\ v_y \\ t \end{pmatrix}_{\text{at } t} \cdot \Delta t \tag{10}$$

We rewrite (10) using (8):

$$\begin{pmatrix} x \\ y \\ v_x \\ v_y \\ t \end{pmatrix}_{\text{at } t+\Delta t} \approx \begin{pmatrix} x \\ y \\ v_x \\ v_y \\ t \end{pmatrix}_{\text{at } t} + \begin{pmatrix} v_x \\ v_y \\ 0 \\ -g \\ 1 \end{pmatrix}_{\text{at } t} \cdot \Delta t \tag{11}$$

Note, the “1” appearing in the bottom of the third vector in (11) is because $\dfrac{dt}{dt} = 1$.
Recall Euler’s method produces a sequence of Euler points (or vectors)

$$\begin{pmatrix} x \\ y \\ v_x \\ v_y \\ t \end{pmatrix}_0, \begin{pmatrix} x \\ y \\ v_x \\ v_y \\ t \end{pmatrix}_1, \begin{pmatrix} x \\ y \\ v_x \\ v_y \\ t \end{pmatrix}_2, \dots$$

which approximate the state of the system (and the solution) at times $t = 0,\ \Delta t,\ 2\Delta t \dots$.

So, for example, $\begin{pmatrix} x \\ y \\ v_x \\ v_y \\ t \end{pmatrix}_2$ is the Euler approximation of the solution at $t = 2\Delta t$.

Euler’s Method Example. Suppose the ball is thrown at time $t = 0$ seconds from $\langle x, y\rangle = \langle 5, 10\rangle$ meters with initial velocity $\langle v_x, v_y\rangle = \langle 1, 2\rangle$ meters/second. Using a step size of $\Delta t = 0.1$ seconds, approximate the location of the ball at time $t = 0.2$ seconds using the Euler method. Ignore air resistance (drag).

Answer. The 0^{th} Euler point is the initial conditions.

$$\begin{pmatrix} x \\ y \\ v_x \\ v_y \\ t \end{pmatrix}_0 = \begin{pmatrix} 5 \\ 10 \\ 1 \\ 2 \\ 0 \end{pmatrix} \tag{12}$$

The 1^{st} Euler point approximates the state of the system after 0.1 seconds. By (11) and (12) it is given by:

$$\begin{aligned}
\begin{pmatrix} x \\ y \\ v_x \\ v_y \\ t \end{pmatrix}_1 &= \begin{pmatrix} x \\ y \\ v_x \\ v_y \\ t \end{pmatrix}_0 + \begin{pmatrix} v_x \\ v_y \\ 0 \\ -g \\ 1 \end{pmatrix}_0 \Delta t \\
&= \begin{pmatrix} 5 \\ 10 \\ 1 \\ 2 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 2 \\ 0 \\ -9.8 \\ 1 \end{pmatrix} (.1) \\
&= \begin{pmatrix} 5.1 \\ 10.2 \\ 1.0 \\ 1.02 \\ .1 \end{pmatrix} \qquad (13)
\end{aligned}$$

The 2^{nd} Euler point approximates the state of the system after 0.2 seconds. By (11) and (13) it is given by:

$$\begin{aligned}
\begin{pmatrix} x \\ y \\ v_x \\ v_y \\ t \end{pmatrix}_2 &= \begin{pmatrix} x \\ y \\ v_x \\ v_y \\ t \end{pmatrix}_1 + \begin{pmatrix} v_x \\ v_y \\ 0 \\ -g \\ 1 \end{pmatrix}_1 \Delta t \\
&= \begin{pmatrix} 5.1 \\ 10.2 \\ 1.0 \\ 1.02 \\ .1 \end{pmatrix} + \begin{pmatrix} 1.0 \\ 1.02 \\ 0 \\ -9.8 \\ 1 \end{pmatrix} (.1) \\
&= \begin{pmatrix} 5.2 \\ 10.302 \\ 1.0 \\ 0.04 \\ .2 \end{pmatrix} \qquad (14)
\end{aligned}$$

So, at time $t = 0.2$ seconds the ball's position is $\langle x(0.2),\ y(0.2)\rangle \approx \langle 5.2,\ 10.302\rangle$ meters and the ball's velocity is $\langle v_x(0.2),\ v_y(0.2)\rangle \approx \langle 1.0,\ 0.04\rangle$ meters/second.

Including Air Resistance. The standard formula [1, 2] for the magnitude of the drag force F_D on a ball is given by

$$F_D = \frac{C_D\ d\ A\ ||\mathbf{v}||^2}{2} \tag{15}$$

where C_D is the drag coefficient, for a tennis ball $C_D = 0.5$; d is the density of air, which at 20 degrees Centigrade (68° F) is 1.21 kg/meter^3; A is the cross sectional area of the ball, which for a ball of radius r is πr^2, a tennis ball's radius is $r = 0.0265$ meters; the velocity squared $||\mathbf{v}||^2 = v_x^2 + v_y^2$. The mass of a tennis ball is 0.058 kg. However, it is important to note that drag force does not depend on the mass of the ball.

The drag force vector $\mathbf{F_D}$ points in the opposite direction of the ball's motion. So:

$$\mathbf{F_D} = -F_D \frac{\mathbf{v}}{||\mathbf{v}||} = -\frac{C_D\ d\ A\ ||\mathbf{v}||^2}{2}\ \frac{\mathbf{v}}{||\mathbf{v}||} = -\frac{C_D\ d\ \pi r^2\ ||\mathbf{v}||}{2}\ \mathbf{v} = -c\,||\mathbf{v}||\ \mathbf{v} \tag{16}$$

is the drag force vector on the ball due to air resistance. Note, we let $c = \dfrac{C_D\ d\ \pi r^2}{2}$.

Forces are vectors so they add together. So, to include the drag force in your model, you add Equation (16) to (5), which will give you the total combined force (gravity and drag) acting on the ball:

$$\mathbf{F_g} + \mathbf{F_D} = m\mathbf{a}. \tag{17}$$

You can now Euler's method to (17), in the same way as it was applied to (5).

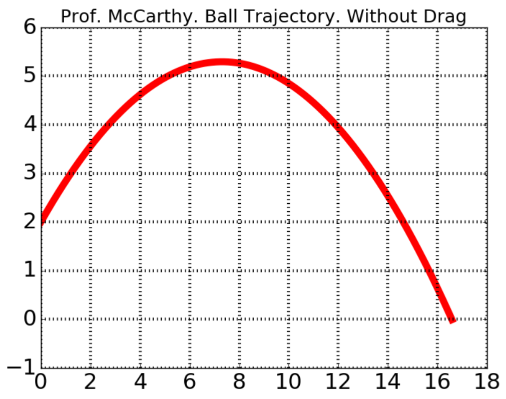
At this point you have everything you need to solve the problem and find the launch angle θ which will maximize how far the ball will land from you.

Python Script

Accompanying this modeling scenario is a Python script called `ThrownBallEulerStudent.py`. It solves the above problem, but assumes there is no air resistance. You can "easily" modify this script to include drag.

Details. The accompanying Python script, `ThrownBallEulerStudent.py`, computes the tennis ball's trajectories from launch to hitting the ground, for launch angles $\theta = 0, 1, 2, \ldots, 90$ degrees using Euler's method. It stores the x coordinate for where each trajectory hits the ground, along with the corresponding launch angle θ, as well as outputting this to the console. It then applies Euler's method to that launch angle θ which gave the "best (maximal)" result to recalculate the optimal (best) trajectory. It outputs and saves a plot of the best trajectory, see below, and prints to the console a paragraph that gives some information about the scenario, including the optimal launch angle and maximum x distance.

From Output of `ThrownBallEulerStudent.py`:



```
Assuming drag = 0.  If when you throw a tennis ball you release it at a height of 2
meters and a speed of 12 meters/second, and you want it to land furthest from you, you
should throw the ball at an angle of 42 degrees:  it will land about 16.6 meters away.
```

Notes about Python

1. You don't have to download Python to run or edit Python scripts. There are free, online Python compilers. A good one is at `https://repl.it/languages/python3`. You just copy and paste your script into the online compiler window and hit RUN.
2. The fastest and best way to run Python is to install it on you computer. A free and widely used Python IDE, especially in the sciences and engineering, is called Anaconda [3].
3. `ThrownBallEulerStudent.py` is written for Python 3. There are two versions of Python: Python 2 and the newer Python 3. They are almost exactly the same. Unfortunately, they are not 100% the same. So, sometimes a script written for one version of Python can't be compiled by the other version of Python.
4. To do 5^2 in Python use $5 **2$
5. To do $\sqrt{7}$ in Python use np.sqrt(7)
6. For π use np.pi

Possible Scenario Extensions

1. Wind.
2. After the ball hits the ground, model how it bounces.
3. Instead of level ground, suppose the ground is an inclined plane [4].

REFERENCES

[1] Cross, R. 2011. *Ball Trajectories*. `http://www.physics.usyd.edu.au/~cross/TRAJECTORIES/Trajectories.html` (accessed 18 August 2019).

[2] Brody, H. and Cross, R. and Lindsey, C. 2002. *The Physics and Technology of Tennis*. Suwanee GA: Racquet Tech Publishing.

[3] Anaconda. 2019. *Anaconda Distribution*. `https://www.anaconda.com/distribution/` (accessed 18 August 2019).

[4] Winkel, Brian. 2015. *3-045-T-BallBounce*. `https://www.simiode.org/resources/1131`.