Mathematical Modeling – Project 1 – Term 1 LC1 AM

Oscar Barros-Nogueira

Modeling Brief:

Throwing the javelin as sport evolved from the everyday use of the spear in hunting and warfare. It was widely practiced in Ancient Greece and incorporated in the Olympic Games in 708BC as part of the pentathlon. It has been part of the modern Olympic Games program since 1908 for men and 1932 for women.

Individual techniques vary but the goal of every javelin athlete is the same, to achieve the largest possible range from their point of projection. By using relevant historical data and through appropriate calculations examine how athletes can optimize their javelin throws.

Formulating the Problem

After reading the brief it is clear that the objective of this project is to discover which variables accumulate in the furthest possible distance that a javelin could be theoretically thrown by a human being.

Before diving into the calculations there are a number of tasks we can complete that can make our lives easier in the future:

Research:

By conducting research either in person or online we can find data that would be extremely difficult to find otherwise. Some topics we should research:

- What is the mass of the javelin?
- What are the best javelin throwing techniques?
- How fast can a human throw an object?
- What is the maximum velocity of a human?
- How much drag does the javelin generate?

Variables:

By listing any possible variables now, we can decide which ones we should exclude in future calculations:

- Velocity of the javelin when:
  - Being held by the athlete
  - Thrown by the athlete
  - It impacts the ground
- Velocity of the athlete when:
  - They begin their run
  - When they release the javelin
- The air resistance of:
  - The athlete

    - The javelin

- The acceleration due to gravity as the altitude of the javelin varies.
- The angle that the javelin is launched at.
- The launch force of the javelin.
- The length of the javelin.
- The center of gravity of the javelin.

To simplify the calculations the following assumptions will be made:

- The javelin will be treated as a particle with a streamlined body for drag calculations.
- Wind resistance is a consistent force and only moves horizontally towards the front of the javelin.
- During the athlete's run up, they will travel at a constant acceleration.
- The existence of lift will be ignored for all calculations.

Due to these assumptions if this experiment is ever conducted in the real-world, the results from said experiment will most likely vary from the results calculated here.

Translating the Problem to Mathematics

If we take take Usain Bolt's 2009 world record 100m sprint as the basis for the peak of human athletic performance we can calculate that with an acceleration of 9.5 m/s^2 (science.org), an athlete could run the javelin running distance of 19m in 2s, with a final velocity of 19m/s.

$u=0\,m/s$	$v^2=u^2+2as$	$v=u+at$
$v=vm/s$	$v=\sqrt{2(9.5)(19)}$	$19=9.5t$
$a=9.5\,m/s^2$	$v=19\,m/s$	$t=2s$
$s=19\,m$		
$t=ts$		

The athlete will also increase the acceleration of the javelin by over-arm throwing the javelin, this converts rotational momentum into angular acceleration, resulting in an increase in overall velocity. To keep calculations simple we will ignore this for the first and second set of calculations, and attempt to include in the third set of calculations (I don't think this will be possible as I believe I currently lack the mathematical ability to do so, but this does not mean I won't try my best to include it).

To find the most accurate values for the best speed and launch angle, I will be using a custom made script written in the coding language python (python 3.10). All relevant files can be found at

https://github.com/TheBanditOfRed/Applied-Maths-Project-2022/, although I will also include the code used to produce the results for each set of calculations in this booklet.
It is important to note that the code uses one external code library created by J.D.Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science and Engineering, vol.9, no.3, pp.90-95, 2007. More information about this library can be found at https://matplotlib.org/.

Computing the Solution – First Calculation

The script used to calculate the first set of calculations was version 7 of the code and was completed on the 04/02/2022 at 01:53am.
In the first set of calculations my main objective was to get a the basic projectile projection complete and working, therefore I ignored the height of the athlete, and I have also used the formula for gravity based on an objects altitude instead of using 9.8m/s^2 for the acceleration due to gravity.

$$g = \frac{G * M}{(R + y)^2}$$

g = Gravity
G = Universal Gravitational Constant
M = Mass of Earth
R = Radius of Earth
y = Altitude

This is the code for the first set of calculations:

```
# Import libraries
import matplotlib.pyplot as plt
import numpy as np
import math as m


# Base Vector Figures
u = 19

y = 0
A = 0                                   # Angle
G = 6.67430 * (10 ** -11)               # Universal gravitation constant
M = 5.972 * (10 ** 24)                  # Mass of the Earth
R = 6.371 * (10 ** 6)                   # Radius of the Earth
g = (G * M) / ((R + y) ** 2)            # Gravity as altitude increases


while A != 90:
    A = A + 1

    print('------------  Angle of ', A,'  ------------')


    # Vector Components
    Ux = u * m.cos(m.radians(A))
    Uy = u * m.sin(m.radians(A))

    print('Ux:              ', Ux)
    print('Uy:              ', Uy)
    print('Gravity:         ', g,'m/s^2')
```

```
    # Time of Flight
    t = Uy / (g / 2)

    print('Time of Flight:  ', t, 's')


    # Range
    r = Ux * t

    print('Range:           ', r, 'm')


    # Max Height
    T = t / 2
    H = (Uy * T) - ((g / 2) * (T ** 2))

    print('Max Height:      ', H, 'm')


    # Creating X and Y Vectors
    x = np.linspace(0, t)

    y = (Uy * x) - ((g / 2) * (x ** 2))


    # Creating the Plot
    plt.plot(x, y)


# File Saving
plt.savefig('test.png')


# Showing the Plot
plt.show()
```

So what is happening here? Everything with a # before it is a comment (ie. Has no effect on the code. Comments are usually used to keep code organized and easier to read). Lets break the code down by comment category:

Import Libraries

This section imports any internal and external libraries into the coding environment so that they may be used as needed. In this case we import:

- Matplotlib.pyplot => Used to graph the graphs
- Numpy => An advanced math library
- Math => A simple math library

Base Vector Figures

All constant variables and start figures are stored here, such as ‘u’ which is the initial velocity of the javelin which we calculated in the previous section to be 19m/s.

while A != 90:

While this is not a comment it is important to explain that this represents the beginning of a loop, and that everything which is indented after this line will loop as long as ‘A’ is not equal to 90.

Vector Components

Calculates ‘Ux’ and ‘Uy’, and prints the values for ‘Ux’, ‘Uy’, and ‘g’ in the console

Time of Flight

Calculates ‘t’ and prints the value for ‘t’ in the console

Range

Calculates ‘r’ and prints the value for ‘r’ in the console

Max Height

Calculates ‘T’ and ‘H’, and prints the values for ‘T’ and ‘H’ in the console

Creating the X and Y Variables

The line of code ‘x = np.linspace’ exists purely to ensure when ‘y’ is graphed, that it is fully visible to the viewer. ‘y’ is also calculated.

Creating the Plot

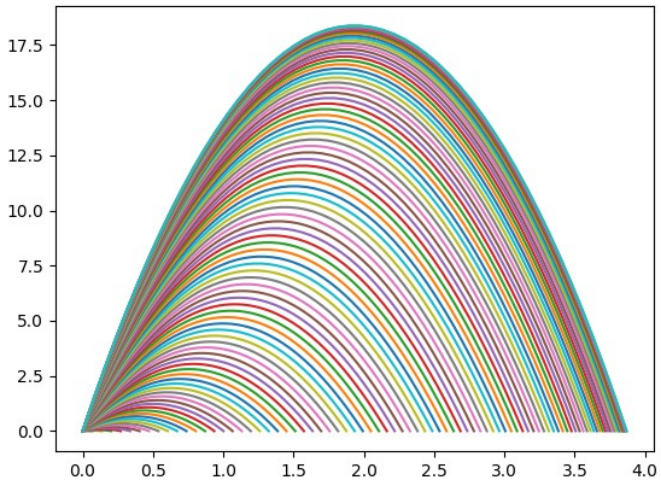
The graph is plotted based on ‘x’ and ‘y’

Showing the Plot

The plot and all related information is shown in the console

The Data

Having Ran the script we now have 90 data points to check and see which has the largest range.



After looking through all data points, the javelin thrown at 45° with a velocity of 19m/s gave us out best results:

---------- Angle of 44 ----------
Ux: 13.667456206434373
Uy: 13.198509038720948
Gravity: 9.819973426224687 m/s^2
Time of Flight: 2.688094654813164 s
Range: 36.739415973409244 m
Max Height: 8.869710399622253 m

---------- Angle of 45 ----------
Ux: 13.435028842544403
Uy: 13.435028842544403
Gravity: 9.819973426224687 m/s^2
Time of Flight: 2.736265824643791 s
Range: 36.76181027495788 m
Max Height: 9.19045256873947 m

---------- Angle of 46 ----------
Ux: 13.198509038720948
Uy: 13.667456206434373
Gravity: 9.819973426224687 m/s^2
Time of Flight: 2.783603501397429 s
Range: 36.739415973409244 m
Max Height: 9.511194737856684 m

When considering that the world record javelin throw is 98.48m, the over-arm throw, which was left out from this set of calculations, must create a lot of additional acceleration seeing that our best throw is still 61.8005840266m away from the world record.

NOTE: I have just come to the realization that I have coded the gravity based on an objects altitude incorrectly and it in fact only calculates the gravity when y = 0. This will be fixed for the second and third set of calculations.

Computing the Solution – Second Calculation

In the second calculation I'm going to implement drag into my equation. But before I do that lets fix the error I discovered with the gravity while writing up the first calculation.

Due to the nature of code it wont be as simple as changing the line of code:

```
y = (Uy * x) - ((g / 2) * (x ** 2))
```

To:

```
y = (Uy * x) - (((G * M) / ((R + y) ** 2) / 2) * (x ** 2))
```

But because 'y' is defined inside and outside the variable and 'y' can't be assigned a value before this we must derive a formula were 'y' is on the left hand side of the equals and everything else is on the right:

$$y=(Uy*x)-\left(\left(\frac{\frac{G*M}{(R+y)^2}}{2}\right)x^2\right)$$

$$y=(Uy*x)-\left(\frac{2GM}{(R+y)^2}\right)x^2$$

$$y=c-\left(\frac{a}{by^2}\right)x^2$$

$$y=c-\left(\frac{ax^2}{by^2}\right)$$

$$y=c-\left(\frac{ax}{by}\right)^2$$

$$\sqrt{y}=\sqrt{c}-\left(\frac{ax}{by}\right)$$

$$\sqrt{y}(by)=\sqrt{c}-ax$$

$$b^2y^3=c-(ax)^2$$

$$y^3=\frac{c}{b^2}-\frac{(ax)^2}{b^2}$$

$$y=\sqrt[3]{\frac{c}{b^2}}-\sqrt[3]{\left(\frac{(ax)}{b}\right)^2}$$

$$a=2GM \qquad b=R^2 \qquad c=Uy*x$$

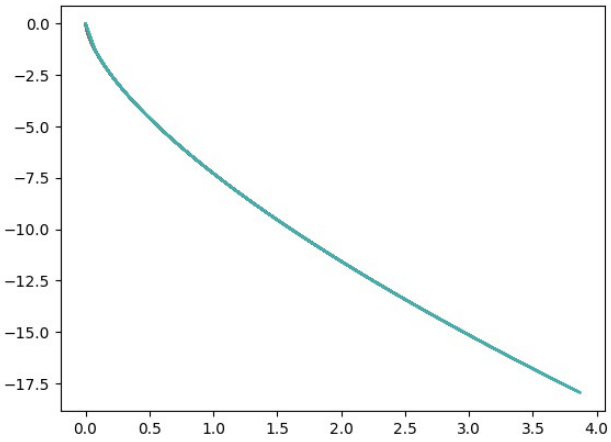
$$y=\left(\frac{c}{b^2}\right)^{\left(\frac{1}{3}\right)}-\left(\frac{ax}{b}\right)^{\left(\frac{2}{3}\right)}$$

$$y=\left(\frac{Uy*x}{R^4}\right)^{\left(\frac{1}{3}\right)}-\left(\frac{2*G*M*x^2}{R^2}\right)^{\left(\frac{2}{3}\right)}$$

After converting the formula into code the line of code shown above becomes:

```
y = (((Uy * x) / (R ** 4)) ** (1 / 3)) - (((2 * G * M * x) / (R ** 2)) ** (2 / 3))
```

But once we substitute the old code for the new code a new issue arises, the graph no longer forms a parabolic shape, rising up in the positive and lowering back to 0 on the y-axis, but instead a negative curve descending in the negative y-axis as seen below.



At least one of two things must be happening:

- I have derived the formula incorrectly

And/Or

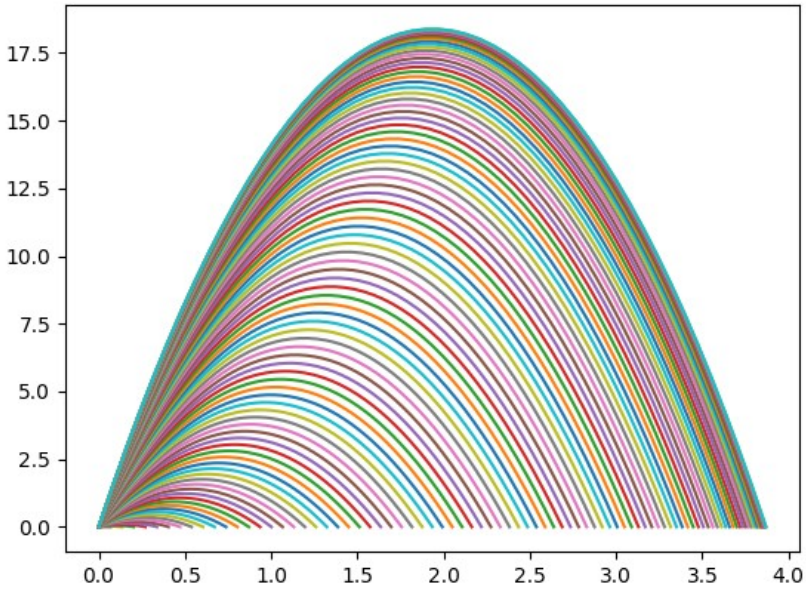
- This formula cannot be used in the way I am try to use it

Either way, from here on forward I'm going to continue to use the formula but the altitude will always be based on the maximum height the javelin has achieved, although as the max height formula also requires a value for gravity all future calculations which calculate gravity based on altitude are fundamentally flawed.

Nevertheless, the new line of code now looks like this:

```
y = (Uy * x) - (((G * M) / ((R + H) ** 2) / 2) * (x ** 2))
```

And produces a properly look graph like in the first set of calculations:



It should also be noted that after completing some rough calculations, the difference in height is so minute that it would make little to no difference if we did not include this formula.

The Code

The script used to calculate the first set of calculations was version 12 of the code and was completed on the 07/03/2022 at 03:35am.
In the second set of calculations I implemented the changes seen above and also incorporated the drag of the javelin into the calculations.

$$F_d = \frac{1}{2} p u^2 c_d A$$

Fd = Drag Force
p = Mass Density of Fluid
u = Flow Velocity
A = Reference Area
Cd = Drag Coefficient

This is the code for the second set of calculations:

```
# Import libraries
import matplotlib.pyplot as plt
import numpy as np
import math as m


# Base Vector Figures
u = 19

y = 0
A = 0                                                   # Angle
G = 6.67430 * (10 ** -11)                               # Universal Gravitation Constant
M = 5.972 * (10 ** 24)                                  # Mass of the Earth
R = 6.371 * (10 ** 6)                                   # Radius of the Earth
```

```
g = (G * M) / ((R + y) ** 2)                          # Gravity as Altitude Increases
DC = 0.1                                              # Drag Coefficient
RA = 0.70686                                          # Reference Area
p = 1.225                                             # Mass Density of Air
FD = (1/2) * p * (u ** 2) * DC * RA                   # Drag
a = (FD / 800)                                        # Deceleration due to Drag
u = u - (a * 0.25)                                    # Acceleration after Drag

print('Drag Force:      ', FD,'N')
print('Acceleration:      ', u,'m/s')

while A != 90:
    A = A + 1

    print('------------  Angle of ', A,'  ------------')


    # Vector Components
    Ux = u * m.cos(m.radians(A))
    Uy = u * m.sin(m.radians(A))


    print('Ux:               ', Ux)
    print('Uy:               ', Uy)


    # Time of Flight
    t = Uy / (g / 2)

    print('Time of Flight:  ', t, 's')


    # Range
    r = Ux * t

    print('Range:            ', r, 'm')


    # Max Height
    T = t / 2
    H = (Uy * T) - ((g / 2) * (T ** 2))

    print('Max Height:       ', H, 'm')


    # Creating X and Y Vectors
    x = np.linspace(0, t)

    y = (Uy * x) - (((G * M) / ((R + H) ** 2) / 2) * (x ** 2))


    # Creating the Plot
    plt.plot(x, y)


# File Saving
plt.savefig('test.png')


# Showing the Plot
plt.show()
```

In order to calculate drag we need a few variables:

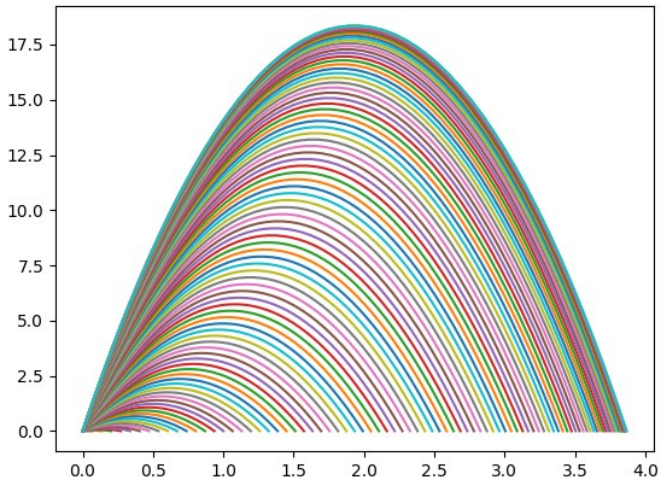
- As a javelin is a streamline body we can assume its drag coefficient is 0.1
- The reference area is the area of an object collides with a medium, in this case air, as per Olympic regulations the diameter of a javelin must be 30mm, which results in a total area of 0.70686 m^2
- We will also need the density of the medium we are traveling through, which for air is 1.225 kg/m^3

After calculating drag we a given a resistive force of αN which must then be converted to a deceleration using the formula $f = ma$ were f = drag, m = mass of the javelin (800g), and a = deceleration due to drag.

We then calculate the new speed of the javelin using the formula $v = u + at$ were v = final velocity, u = initial velocity, a = deceleration due to drag, and t = time (0.25s).

The Data

Having ran the script again we now have 90 new data points to compare and see which has the largest range.



After looking through all data points, the javelin thrown at 45° with a velocity of 19m/s is still giving us the best results:

---------- Angle of 44 ----------
Ux: 13.659381678274363
Uy: 13.190711557514962
Time of Flight: 2.686506568803652 s
Range: 36.69601860448032 m
Max Height: 8.859233311614549 m
---------- Angle of 45 ----------
Ux: 13.427091628985337
Uy: 13.427091628985337
Time of Flight: 2.7346492798295516 s
Range: 36.718386453410155 m
Max Height: 9.179596613352539 m
---------- Angle of 46 ----------
Ux: 13.190711557514962
Uy: 13.659381678274363
Time of Flight: 2.7819589901936723 s
Range: 36.69601860448033 m
Max Height: 9.499959915090521 m

If we compare our results from the first set of calculations, we do see a reduction in the distance traveled by 0.04342382154 but it is so small that it can be considered negligible data. I suspect that there might be an error in my math, but at this time I have no way to know for certain if this is the case.

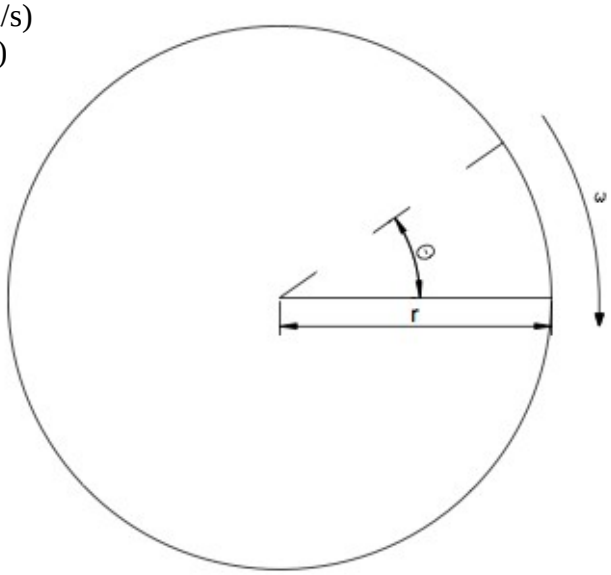
Computing the Solution – Final Calculation

In the final calculation I have implemented angular velocity into the calculations. This is necessary as the athlete preforms an overarm throw with the javelin before releasing it. It is in this over arm throw where most of the javelin's velocity is generated and this can be clearly seen in the results.

I used two formulas to find the angular velocity of the javelin:

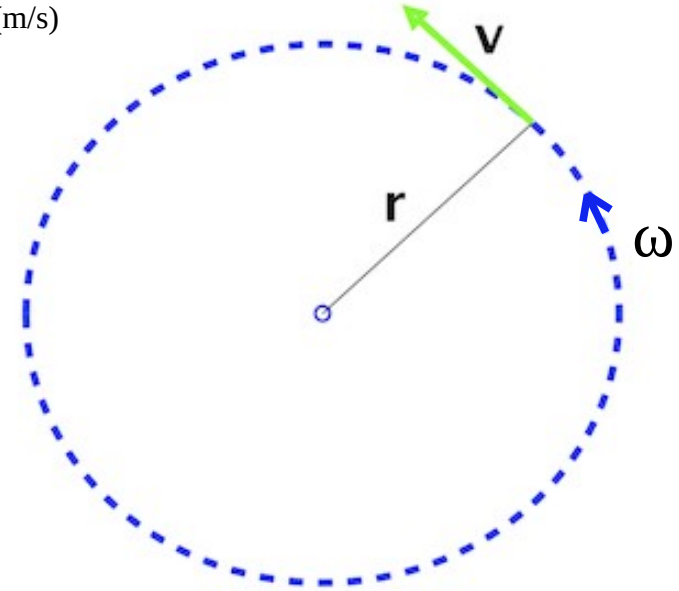
$\omega = \frac{\theta}{t}$ This formula finds the angular velocity of the javelin between two points of an angle. The SI unit for the solution of this formula is rad/s (radians per second).

ω = angular velocity (rad/s)
θ = angular distance (rad)
t = time (s)



$v = \omega r$ This formula finds the tangential velocity of a point in angular velocity, or put more simply the velocity that the javelin travels once released by the athlete. The SI unit for the solution of this formula is m/s.

r = distance from center to the point (m)
v = tangential velocity (m/s)



The Code

The script used to calculate the final calculation was version 14 of the code and was completed on the 10/03/2022 at 04:30am.

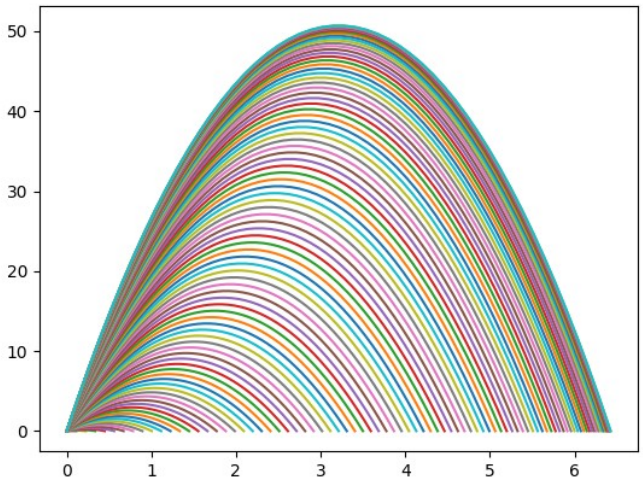
```
# Import libraries
import matplotlib.pyplot as plt
import numpy as np
import math


# Base Figures
u = 19
y = 0
DC = 0.1                                                # Drag Coefficient
RA = 0.70686                                            # Reference Area
p = 1.225                                               # Mass Density of Air
A = 0                                                   # Angle
G = 6.67430 * (10 ** -11)                               # Universal Gravitation Constant
M = 5.972 * (10 ** 24)                                  # Mass of the Earth
R = 6.371 * (10 ** 6)                                   # Radius of the Earth
O = (math.pi) / 2                                       # Angular Distance (rad)
tl = 0.25                                               # Time
m = 800                                                 # Mass of Javelin
rc = 1                                                  # Radius / Length of Arm (m)
print('Initial Velocity:              ', u,'m/s')


# Gravity as Altitude Increases
g = (G * M) / ((R + y) ** 2)


# Angular Velocity
w = (O / tl) * 2                                        # Angular Velocity
print('Angular Velocity (rad):        ', w,'rad/s')

v = w * rc                                              # Tangential Velocity of a Point
in Angular Velocity
print('Angular Velocity (m):          ', v,'m/s')

u = u + v
print('Initial + Angular Velocity:    ', u,'m/s')


# Drag
FD = (1/2) * p * (28.8036 ** 2) * DC * RA               # Drag
print('Drag Force:                    ', FD,'N')

a = (FD / m)                                            # Deceleration due to Drag
u = u - (a * tl)                                        # Velocity after Drag
print('Velocity:                      ', u,'m/s')


while A != 90:
    A = A + 1

    print('------------  Angle of ', A,'  ------------')


    # Vector Components
    Ux = u * math.cos(math.radians(A))
```

```
    Uy = u * math.sin(math.radians(A))


    print('Ux:                ', Ux)
    print('Uy:                ', Uy)


    # Time of Flight
    t = Uy / (g / 2)

    print('Time of Flight:  ', t, 's')


    # Range
    r = Ux * t

    print('Range:             ', r, 'm')


    # Max Height
    T = t / 2
    H = (Uy * T) - ((g / 2) * (T ** 2))

    print('Max Height:        ', H, 'm')


    # Creating X and Y Vectors
    x = np.linspace(0, t)


    y = (Uy * x) - (((G * M) / ((R + H) ** 2) / 2) * (x ** 2))


    # Creating the Plot
    plt.plot(x, y)


# File Saving
plt.savefig('test.png')


# Showing the Plot
plt.show()
```

Besides from adding angular velocity I also organized the code and fixed some of the labels as they were incorrect. Also the line `w = (O / tl) * 2` where you see formula for angular velocity being multiplied by two instead of not being multiplied at all is due to a weird and annoying python bug which refuses to do proper radian based math calculations, but I discovered this little cheat which gives me the correct answer.

The Data

Having ran the script for the last time we now have 90 new data points to compare and see which has the largest range.



X = Time (s) *Y = Height (m)*

Unsurprisingly after looking through all data points, the javelin thrown at 45° is still traveling the furthest:

---------- Angle of 44 ----------
Ux: 22.698872206988984
Uy: 21.920046091069533
Time of Flight: 4.464379920322605 s
Range: 101.33638929485048 m
Max Height: 24.464853405379213 m
---------- Angle of 45 ----------
Ux: 22.31285750530207
Uy: 22.31285750530207
Time of Flight: 4.544382461507394 s
Range: 101.39815831320834 m
Max Height: 25.349539578302085 m
---------- Angle of 46 ----------
Ux: 21.920046091069533
Uy: 22.698872206988984
Time of Flight: 4.623000739771986 s
Range: 101.33638929485049 m
Max Height: 26.23422575122496 m

Finally we have broken the world record of 98.48m, not by much but enough to make a difference. As I predicted at the end of my first calculation the over arm throw generates a lot of velocity.

Conclusion

After many hours of coding and writing I can finally say with confidence that the two most important variables when throwing a javelin is to be able to run fast and be able to throw fast. The best launch angle for the javelin, as per my results, was 45 degrees.