The Effect of Rotational Speed on the Horizontal Distance that a Tennis Ball Travels Until Its Second Bounce

Subject: Physics

Research Question: How does the rotational speed (Hz) of a tennis ball affect the horizontal distance (m) that it travels until its second bounce, when it is launched parallel to the ground from a height of 1 meter with a speed of $5ms^{-1}$?

Word Count: 3989

Table of Contents

Introduction

Tennis is a game rich in dynamics. The tennis ball is constantly acted upon by three forces (Gravitational, Drag, Lift) while it is in the air and three forces (Gravitational, Normal, Friction) while it is bouncing. The rotational speed of the ball affects both how it moves in air and how it interacts with the ground. There is various research on the aerodynamics of a tennis ball (Štěpánek) (Cross, "Ball Trajectories") (Alam) (Metha), and three main papers regarding the bounce of a spinning ball (Garwin) (Cross, "Bounce of a Spinning Ball") (Brody). Rather surprisingly, no study combined these two parts and inspected the distance that a tennis ball travels until its second bounce. There are two main reasons for investigating the effect of rotational speed of a tennis ball on the horizontal distance that it travels until its second bounce. The first reason is that the relationship between the rotational speed of the tennis ball and the horizontal distance that it travels until its second bounce is complex and the answer can't be found intuitively. The second reason is the lack of research investigating this relationship.

In this investigation, a mathematical model will be formed by combining and expanding upon previous theoretical models regarding tennis ball's movement in air and its collision with the ground. This mathematical model will be turned into code using iterative methods such as Euler and Runge-Kutta approximations to make predictions. Then the predictions will be compared with the experimental results to see the validity of the mathematical model.

Research Question: How does the rotational speed (Hz) of a tennis ball affect the horizontal distance (m) that it travels until its second bounce, when it is launched parallel to the ground from a height of 1 meter with a speed of $5ms^{-1}$?

Mathematical Model

Motion of the tennis ball is divided into two parts: the movement in air and the bounce (Figure 1).

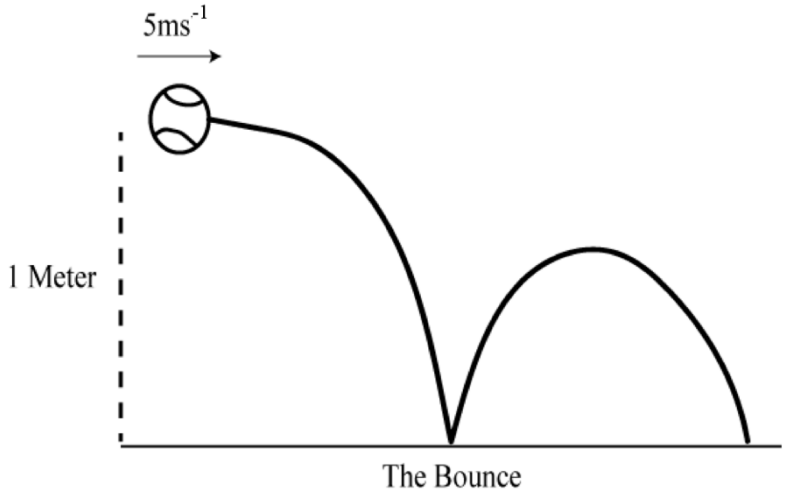


Figure 1: The Trajectory of the Tennis Ball

The Movement in Air

There are 3 forces acting on the ball while it is in the air. Gravitational Force (F_G) perpendicular to the ground, Drag Force (F_D) antiparallel to the direction of movement, and Magnus Force (F_M) perpendicular to the direction of movement.

Gravitational Force is caused by the attraction of the mass of Earth and mass of the ball and can be written as $F_G = g * m_{ball}$ ($g = 9.81ms^{-2}$)

Drag Force is caused by the difference in velocity between a solid object and a fluid (Hall). The origin of the force is the collisions between the molecules in the fluid and the object. The formula for it is $F_D = \frac{1}{2} * C_D * A * p * v^2$ (where C_D is the drag coefficient, A is the cross sectional area, p is the density of the medium, and v is the difference in velocity) (Cross ,"Ball Trajectories " 370).

Magnus Force is caused by the difference of pressure around a spinning object. For example, for a tennis ball with backspin, the magnus force is directed upwards (Figure 2). The velocity at the bottom of the ball going with backspin is v+wr while the velocity at the top of the ball is v-wr. (Where v is velocity, w is angular velocity, r is radius) Therefore, the air molecules at the bottom of the ball are getting slowed down more than the air molecules at the top of the ball. The slower air creates a higher pressure according to Bernoulli's Principle and a lift force forms. When the direction of the spin reverses, the direction of the Magnus Force reverses as well. The formula for the lift force created by the Magnus Effect is very similar to the drag force formula $F_M = \frac{1}{2} * C_L * A * p * v^2$ (the only difference being C_L, the lift coefficient) (Cross ,"Ball Trajectories " 371)

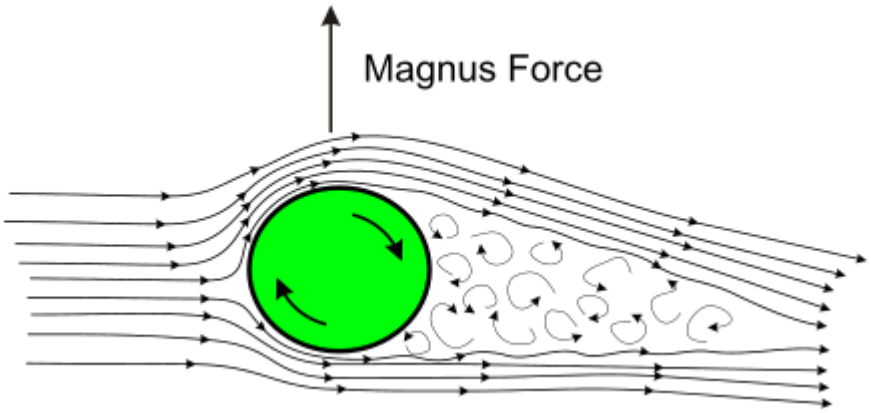


Figure 2: Rdurkacz. *Sketch of Magnus Effect with Streamlines and Turbulent Wake.* 2017. *Wikimedia Commons, the Free Media Repository*

The equations for the drag coefficient C_D and lift coefficient C_L of a tennis ball were found to be

$$C_D = 0.508 + \frac{1}{(2.503+\frac{4.196}{(\frac{wr}{v})^{2.5}})^{0.4}} \qquad C_L = \frac{1}{(2.022+\frac{0.981}{(\frac{wr}{v})})}$$

by experimental data collected in an open-type aerodynamics tunnel at the Research Institute for Aeronautics in Prague (Štěpánek 140). (v is velocity, w is angular velocity, r is the radius of the tennis ball)

The model for the movement in the air was expanded from Cross' Ball Trajectories

The Case for the Ball with the Backspin:

When the ball is rising, it forms an angle θ with the ground (Figure 3). The Gravitational Force is always perpendicular to the ground. The Drag Force is always antiparallel to the direction of motion while the Magnus Force is always perpendicular to it.

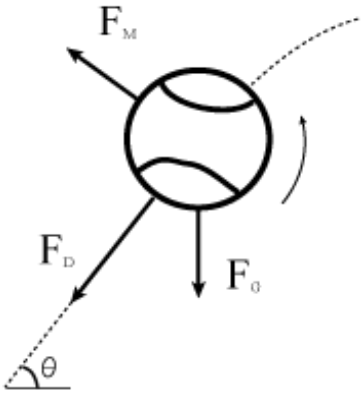


Figure 3: Tennis Ball with Backspin Rising

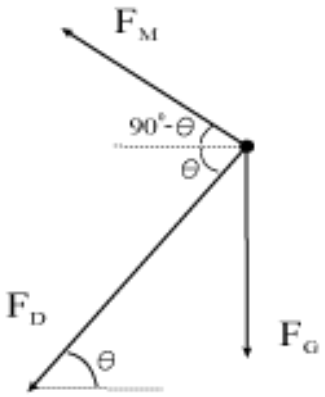


Figure 4: Free Body Diagram of a Tennis Ball with Backspin Rising

From the free body diagram of this movement (Figure 4), equations of the net force on the ball can be derived by dividing the force vectors into their x and y components.

In the horizontal direction: $m\frac{dv_x}{dt} = -F_M\, cos(90° - \theta) - F_D\, cos(\theta)$

In the vertical direction: $m\frac{dv_y}{dt} = F_M\, sin(90° - \theta) - F_D\, sin(\theta) - F_G$

Simplifying to get:

$$m\frac{dv_x}{dt} = -F_M\, sin(\theta) - F_D\, cos(\theta)$$

$$m\frac{dv_y}{dt} = F_M\, cos(\theta) - F_D\, sin(\theta) - F_G$$

The Case for the Ball with the Topspin:

When the ball is rising with a topspin, every force is the same with the backspin except the direction of the Magnus Force is reversed (Figure 5).

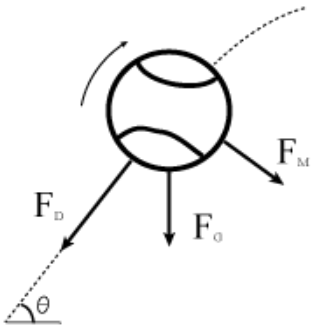


Figure 5: Tennis Ball with Topspin Rising

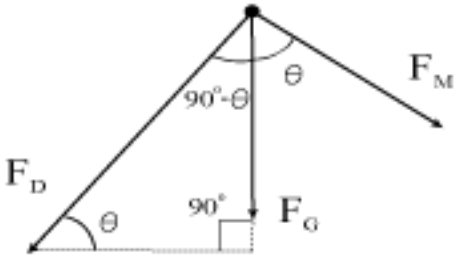


Figure 6: Free Body Diagram of a Tennis Ball with Topspin Rising

From the free body diagram of this movement (Figure 6), equations of the net force on the ball can be derived again by dividing the force vectors into their x and y components.

In the horizontal direction: $m\frac{dv_x}{dt} = F_M\ sin(\theta)\ - F_D\ sin(90^\circ - \theta)$

In the vertical direction: $m\frac{dv_y}{dt} = -F_M\ cos(\theta)\ - F_D\ cos(90^\circ - \theta) - F_G$

Simplifying to get:

$m\frac{dv_x}{dt} = F_M \sin(\theta) - F_D\ \cos(\theta)$

$m\frac{dv_y}{dt} = -F_M\ cos(\theta)\ - F_D\ sin(\theta) - F_G$

Inserting $F_D = \frac{1}{2} * C_D * A * p * v^2$, $F_M = \frac{1}{2} * C_L * A * p * v^2$, $F_G = g * m$,

$cos(\theta) = \frac{v_x}{v}$, $sin(\theta) = \frac{v_y}{v}$ for both the topspin and backspin equations to get

Topspin:

$$\frac{dv_x}{dt} = \frac{A * p * v}{2 * m}(C_L * v_y - C_D * v_x) \qquad (1)$$

$$\frac{dv_y}{dt} = -\frac{A * p * v}{2 * m}(C_L * v_x + C_D * v_y) - g \qquad (2)$$

Backspin:

$$\frac{dv_x}{dt} = -\frac{A * p * v}{2 * m}(C_L * v_y + C_D * v_x) \qquad (3)$$

$$\frac{dv_y}{dt} = \frac{A * p * v}{2 * m}(C_L * v_x - C_D * v_y) - g \qquad (4)$$

These equations are valid for the ball falling down as well since the v_y becomes negative and changes the direction of the drag and magnus forces.

Since the rate of change of velocity and the initial velocity is known, the velocities throughout the movement in air are calculated by the fourth-order Runge-Kutta method (Press 710-711) and the positions of the ball are calculated using the Euler's method (Wazir 1463-1466).

The limitation of this model is that there is no algebraic description of how rotational speed of a tennis ball changes while it is moving in air. The fuzz of the tennis ball makes its surface rough and porous (Metha, "Review of Tennis Ball Aerodynamics" 14). Therefore, the flow around the tennis ball is always turbulent (Cross, "Sports ball aerodynamics" 2) and the only algebraic description of viscous torque on a sphere is derived for low Reynolds numbers (Lei 1). The validity of this assumption will be investigated.

The Bounce

Out of three theoretical models (Garwin) (Cross, "Bounce of a Spinning Ball") (Brody), Brody's was chosen because Garwin's model assumes that the ball is perfectly elastic (which is a limited description of a tennis ball[1]) and Cross' model can only predict the outcome if a measurement of the bounce was made before[2]. Brody's model assumes that the ball does not deform during the bounce, the normal force acts through the center of the ball, the ball does not slow down when it rolls, and neglects gravitational force.

In this investigation, the mathematical model expands upon Brody's model by not neglecting gravitational force, solving for angular velocity, analyzing one more topspin case, and analyzing the bounce of a ball with backspin.

The motion of the ball is divided into vertical and horizontal. In the vertical direction, there are two forces acting on the ball: Normal Force (F_N) and Gravitational Force (F_G).

Since the ball's acceleration is pointed upwards, the net force $m\frac{dv_y}{dt} = F_N - F_G$

and therefore $F_N = mg + m\frac{dv_y}{dt}$ (this description of the normal force will be used later)

In both cases the velocity of the ball after the bounce is determined by a constant coefficient of restitution[3] (e_y) . Coefficient of restitution equals the ratio of velocities after and before the bounce (Ferreira Da Silva 1221): $e_y = -\frac{v_{fy}}{v_{iy}}$

[1] Garwin's model was originally intended for the bounce of an ultraelastic roughball, i.e. a superball.

[2] Furthermore, the constant used in his model varies with angle. Since the angle of incidence will change with changing spin in this experiment, Cross' model can't be used.

[3] Check Appendix 1 to see how it is determined

$$v_{fy} = -e_y * v_{iy} \quad (5)$$

The horizontal part of the interaction determines both the final horizontal velocity (v_{fx}) and final angular velocity (w_f)

The Case for the Ball with the Backspin:

For the ball with backspin, the magnitude of the tangential velocity (wr) doesn't affect the direction and type of friction force (Figure 7 and 8). It slides throughout the bounce.

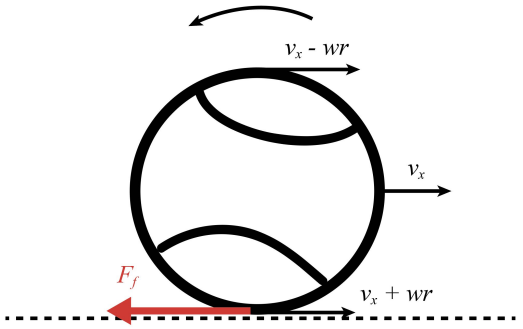


Figure 7: The Friction Force when $v_x > wr$

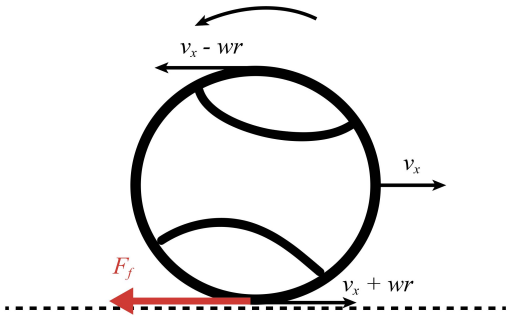


Figure 8: The Friction Force when $v_x < wr$

The impulse created by the friction force (F_f) equals the change of linear momentum.

$$\int_0^t F_f \, dt = m\,(v_{fx} - v_{ix})$$

v_{fx} is the final horizontal velocity, v_{ix} the initial horizontal velocity and t is the duration of the bounce.

Since $F_f = \mu F_N$ and $F_N = mg + m\frac{dv_y}{dt}$,

$$\int_0^t \mu m(\frac{dv_y}{dt} + g)dt = m\,(v_{fx} - v_{ix})$$

μ and m are taken out of the integral since they are constants and m cancels out at both sides

$\mu \int_0^t (\frac{dv_y}{dt} + g)dt = v_{fx} - v_{ix}$ the integral is separated and performed

$$\mu(\int_0^t \frac{dv_y}{dt}\, dt + \int_0^t g\, dt) = v_{fx} - v_{ix} \quad \rightarrow \quad \mu(\int_0^t dv_y + gt) = v_{fx} - v_{ix}$$

$\mu(v_{fy} - v_{iy} + gt) = v_{fx} - v_{ix}$ inserting $-e_y * v_{iy}$ for v_{fy}

$\mu((-e_y * v_{iy}) - v_{iy} + gt) = v_{fx} - v_{ix}$ reorganizing to get:

$$v_{fx} = v_{ix} + \mu(-v_{iy}(e_y + 1) + gt) \qquad (6)$$

The Frictional Force also does work to create angular impulse which equals the change in rotational momentum.

$r \int_0^t F_f \, dt = I \, (w_f - w_i)$, inserting $\mu m(-v_{iy}(e_y + 1) + gt)$ for $\int_0^t F_f \, dt$ from the previous part

$\mu mr(-v_{iy}(e_y + 1) + gt) = I \, (w_f - w_i)$

$I = kmr^2$ where k is a constant to be determined (Appendix 1)

$\mu(-v_{iy}(e_y + 1) + gt) = kr(w_f - w_i)$ reorganizing

$\frac{\mu}{kr}(-v_{iy}(e_y + 1) + gt) = w_f - w_i$ reorganizing

$$w_f = w_i + \frac{\mu}{kr}(-v_{iy}(e_y + 1) + gt) \qquad (7)$$

The Case for the Ball with the Topspin:

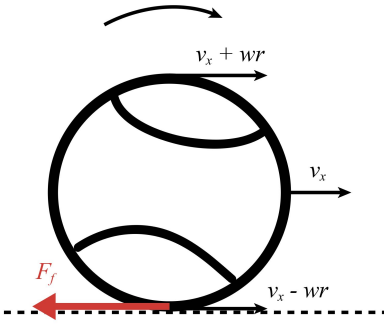


Figure 9: The Friction Force when $v_x > wr$

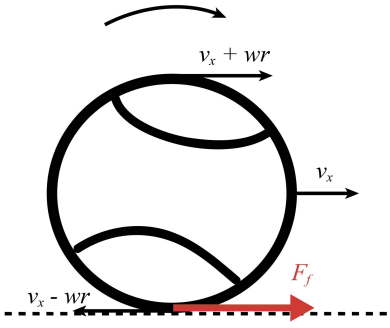


Figure 10: The Friction Force when $v_x < wr$

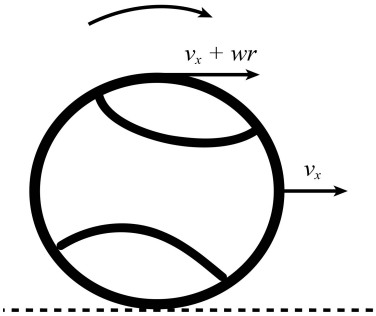


Figure 11: The Friction Force when $v_x = wr$

In the situation depicted in Figure 9,

$$v_{fx} = v_{ix} + \mu(-v_{iy}(e_y + 1) + gt) \qquad (6)$$

$$w_f = w_i + \frac{\mu}{kr}(-v_{iy}(e_y + 1) + gt) \qquad (7)$$

Hold true again.

However, if $v_x = wr$ at some point during the bounce, the ball will start rolling and F_f will be zero. The ball won't slow down anymore.

In that case:

$r \int_0^{t_r} F_f \, dt = I \, (w_f - w_i)$ where t_r is the time the ball starts rolling.

Inserting $m(v_{fx} - v_{ix})$ for $\int_0^{t_r} F_f \, dt$ and $I = kmr^2$ to get:

$mr(v_{fx} - v_{ix}) = kmr^2(w_f - w_i) \quad \rightarrow \quad (v_{fx} - v_{ix}) = kr(w_f - w_i)$

Since $\frac{v_{fx}}{r} = w_f$ during rolling,

$(v_{fx} - v_{ix}) = kr(\frac{v_{fx}}{r} - w_i)$ simplifying,

$(1 - k)v_{fx} = v_{ix} - krw_i$ reorganizing,

$$v_{fx} = \frac{v_{ix} - krw_i}{1 - k} \quad (8)$$

and due to the $\frac{v_{fx}}{r} = w_f$ identity

$$w_f = r\frac{v_{ix} - krw_i}{1 - k} \quad (9)$$

In the situation depicted in Figure 10, the direction of the friction force is reversed.

That means:

$-\int_0^t F_f \, dt = m \, (v_{fx} - v_{ix}) \rightarrow \int_0^t F_f \, dt = m \, (v_{ix} - v_{fx})$

and $\mu(-v_{iy}(e_y + 1) + gt) = v_{ix} - v_{fx}$

Reorganizing to get:

$$v_{fx} = v_{ix} - \mu(-v_{iy}(e_y + 1) + gt) \qquad (10)$$

The same process is repeated for the angular velocity:

$-r \int_0^t F_f \, dt = I \, (w_f - w_i) \rightarrow r \int_0^t F_f \, dt = I \, (w_i - w_f)$

and $\frac{\mu}{kr}(-v_{iy}(e_y + 1) + gt) = w_i - w_f$

Reorganizing to get:

$$w_f = w_i - \frac{\mu}{kr}(-v_{iy}(e_y + 1) + gt) \qquad (11)$$

If $v_x = wr$ at some point during the bounce, the ball will start rolling and

$$v_{fx} = \frac{v_{ix} - krw_i}{1 - k} \quad (8)$$

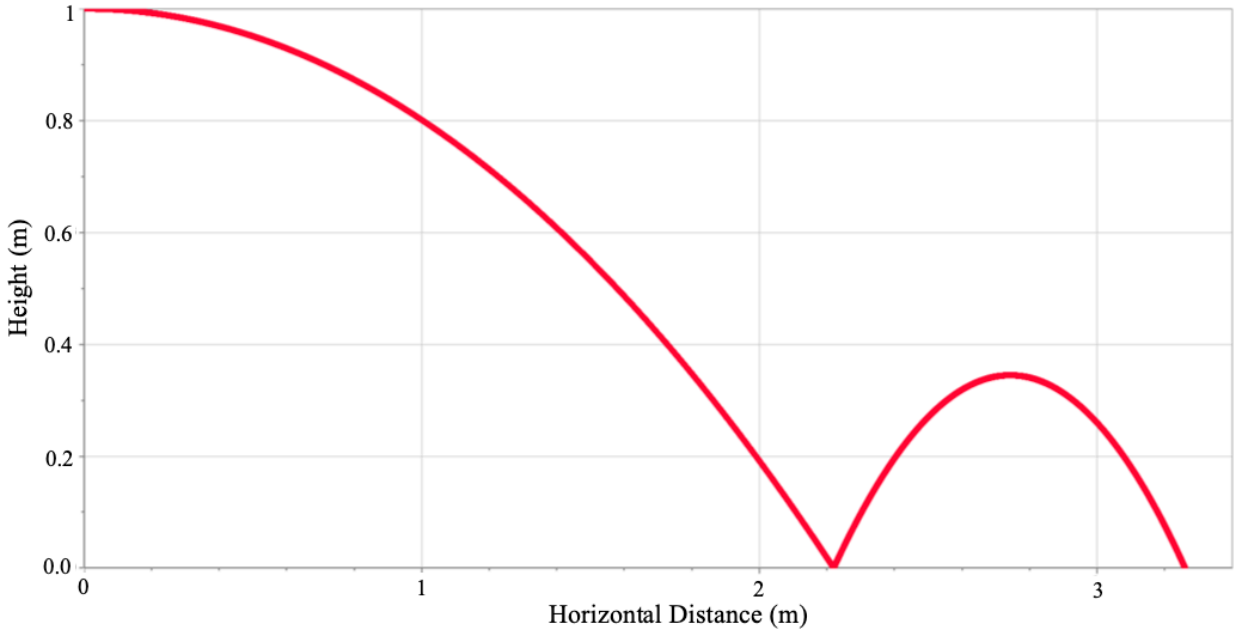
$$w_f = r\frac{v_{ix} - krw_i}{1 - k} \quad (9)$$

Hold true again.

In the situation depicted in Figure 11, the ball commences rolling immediately and both the horizontal velocity and the angular velocity stays the same.

The derived equations are integrated into Python[4] and the theoretical distances are graphed in Logger Pro. A theoretical trajectory can be seen in Graph 1.

Graph 1: The Theoretical Trajectory of the Tennis Ball with a Rotational Speed of 0Hz



[4] The constants and their uncertainties that are used in the code is in Appendix 1, the code itself is in Appendix 2

Experimental Design

Variables

Independent Variable: The rotational speed of the tennis ball (Hz).

Dependent Variable: The distance that the tennis ball travels until its second bounce (m).

Controlled Variables: The factors that would affect the distance significantly were tried to be controlled. The experimental setup for achieving this can be seen in Figure 12.

The initial velocity of the ball was kept at $5ms^{-1}$ (measured by LoggerPro video analysis) by using a pneumatic cylinder fixed parallel to the ground always with the pressure of $(6 \pm 0.125)kPa$ (Figure 13). However, as the rotational speed of the tennis ball increased, the reaction force between the cylinder and the ball increased, slightly changing the magnitude and direction of velocity. This effect became more pronounced at higher rotational speeds.

The initial rotational speed of the ball was adjusted by controlling the rpm's of the dc motors using a L298n DC motor driver.

The initial height and horizontal distance of the ball were controlled by fixing the spinning mechanism to the wooden platform.

The coefficient of restitution and coefficient of sliding friction were controlled by using the same tennis ball in the same tennis court.

To prevent wind causing any random errors by stirring the trajectory of the tennis ball, the experiment was performed in an indoor tennis court.

To keep the air density relatively constant, the experiment was performed in 4 hours at the same place to prevent a significant temperature gradient.

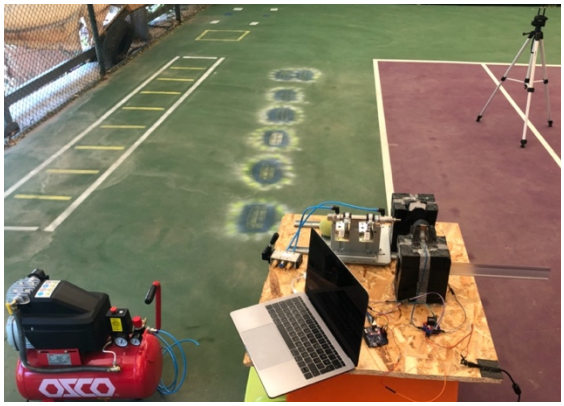


Figure 12: The Experimental Setup

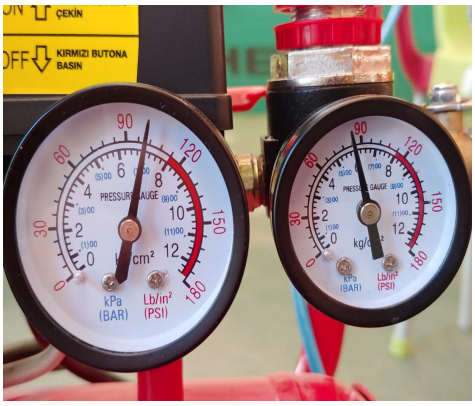


Figure 13: Controlling the Pressure via an Air Regulator

The experimental setup consists of three parts: The spinning mechanism, the shooting mechanism, and the recording apparatus.

The Spinning Mechanism

The annotations of the spinning mechanism can be seen in Figure 14 and 15. The DC motors are fixed to the motor towers with duct tape and the motor towers are fixed to the wooden base with a glue gun. The DC motors are connected to the L298n DC motor driver which is connected to Arduino Uno. The rpm's of the DC motors are controlled from the Arduino Uno which is controlled from the computer by the code written by the author (Appendix 3). The back of the silicone half spheres are inserted into the DC motors while their fronts are covered with anti-slip tape to increase friction and prevent the tennis ball from falling.

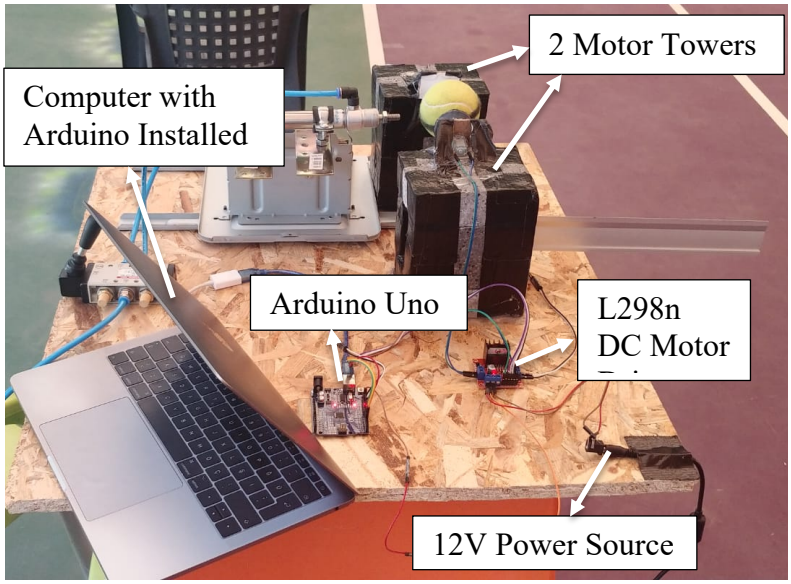


Figure 14: The Spinning Mechanism Annotated in the Experimental Setup

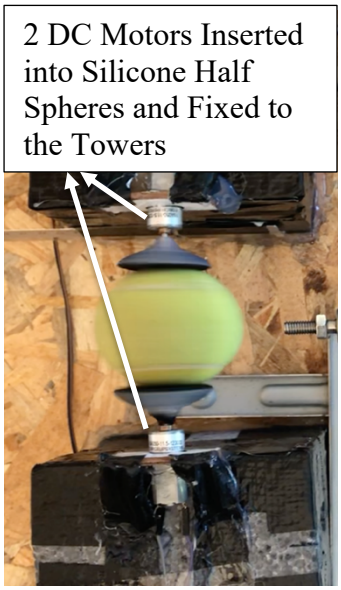


Figure 15: A Closer Look at the Motor Towers

The Shooting Mechanism

The annotations of the shooting mechanism can be seen in Figure 16 and 17. The pressure of the air coming from the compressor is controlled via an air regulator. The double acting pneumatic cylinder is connected to a manual control valve. The pneumatic cylinder is fixed into position by using pipe clamps. The pipe clamps are connected to corner braces and the corner braces are connected to the metal base which is screwed into the wooden base.

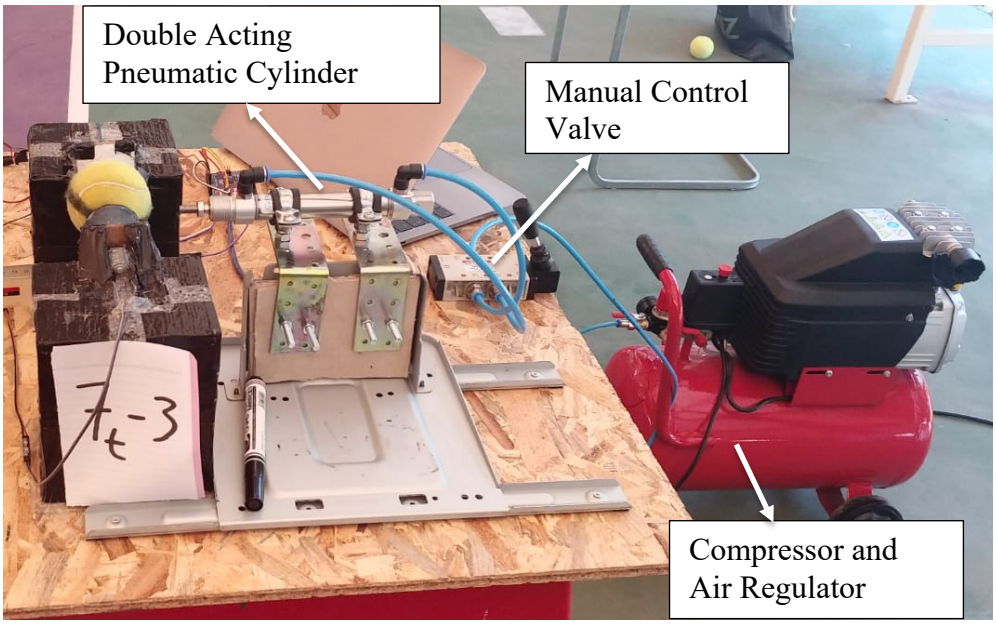


Figure 16: The Shooting Mechanism Annotated in the Experimental Setup

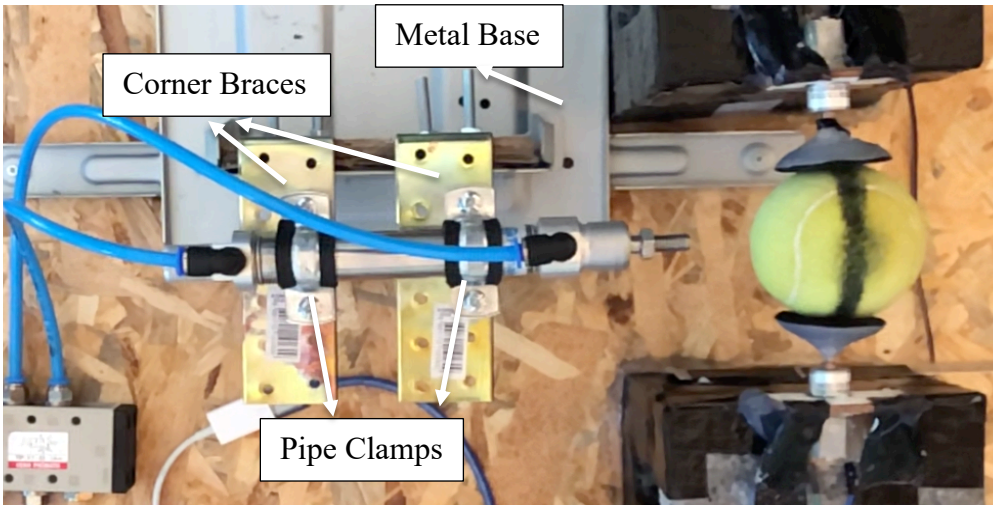


Figure 17: A Closer Look at the Pneumatic Cylinder

The Recording Apparatus

The annotations of the recording apparatus can be seen in Figure 18 and 19. The tennis ball is marked along its circumference with a black board marker to measure its rotational speed. A metal ruler is connected to a motor tower and marked every 10cm to be used as a reference length during video analysis. The numbers and letter on the paper indicate the supposed rotational speed (7Hz), type of spin (t for topspin) and the number of trial (3). A camera filming at 240fps in 1080p is attached to a tripod which enables it to oversee the entire trajectory of the tennis ball until its second bounce. The position of the tripod is unchanged throughout the experiment.

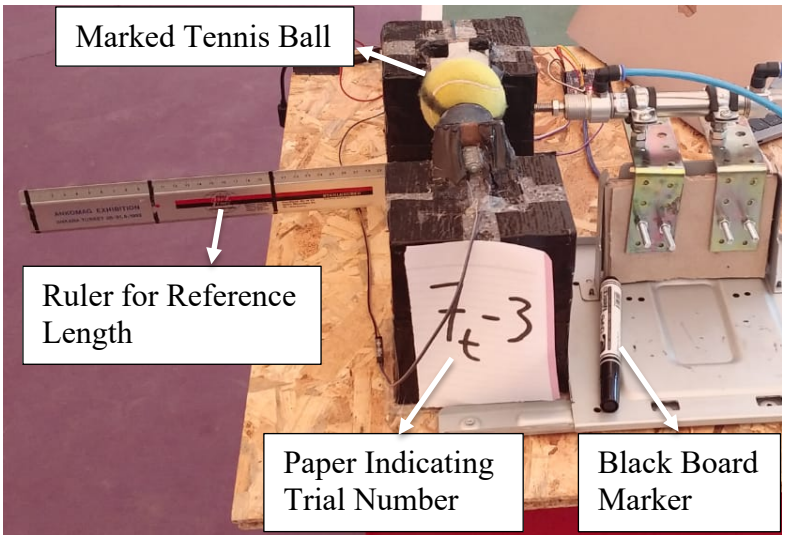


Figure 18: The Recording Apparatus Annotated in the Experimental Setup

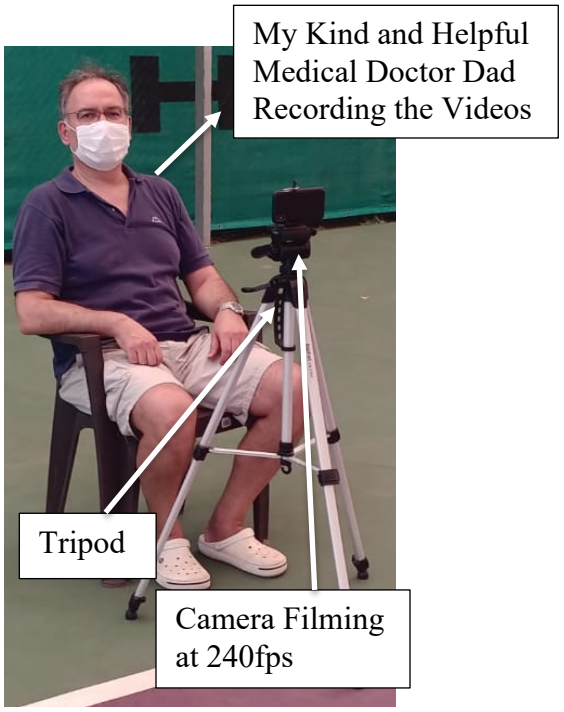


Figure 19: The Placement of the Camera

The Procedure

1) Prepare the setup as shown in the Figures 14-19. Control that the air regulator shows $6kPa$.
2) Adjust the speed of the DC motors from the computer so that when the ball is launched, it has a rotational speed of 4Hz with topspin. The example given in Figure 20 shows how the rotational speed can be determined using the black mark on the tennis ball.

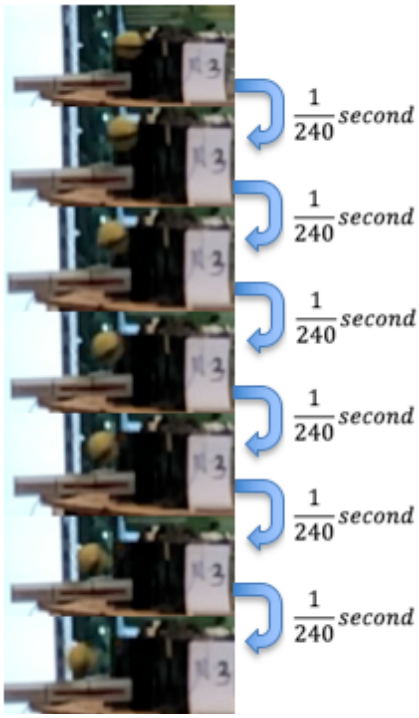


Figure 20: Measuring the Rotational Speed of the Tennis Ball

* First find a frame just after the ball is launched that has a horizontal or vertical line.

* Count the frames that the line takes to become orthogonal to the line in the first frame that was chosen. Multiplying that number with $\frac{1}{240}s$ (as the frame rate is 240fps, one frame takes $\frac{1}{240}s$) gives the amount of time needed to complete $\frac{1}{4}th$ of a cycle.

* Multiplying that with 4 gives the time it takes to complete 1 cycle.

* Dividing 1 by that number gives the rotational speed of the tennis ball.

The rotational speed of the tennis ball shown in Figure 20: $\frac{1}{\frac{1}{240}s \cdot 6 \cdot 4} = 10Hz$

3) Measure the distance that the ball travels until its second bounce using LoggerPro Video Analysis. An example measurement is given in Figure 21.

*Assign 0.2m to the marked length of 20cm on the ruler.

*Find the frame in which the tennis ball undergoes its second bounce

*Measure the distance from the center of the tennis ball to the start of the white line.

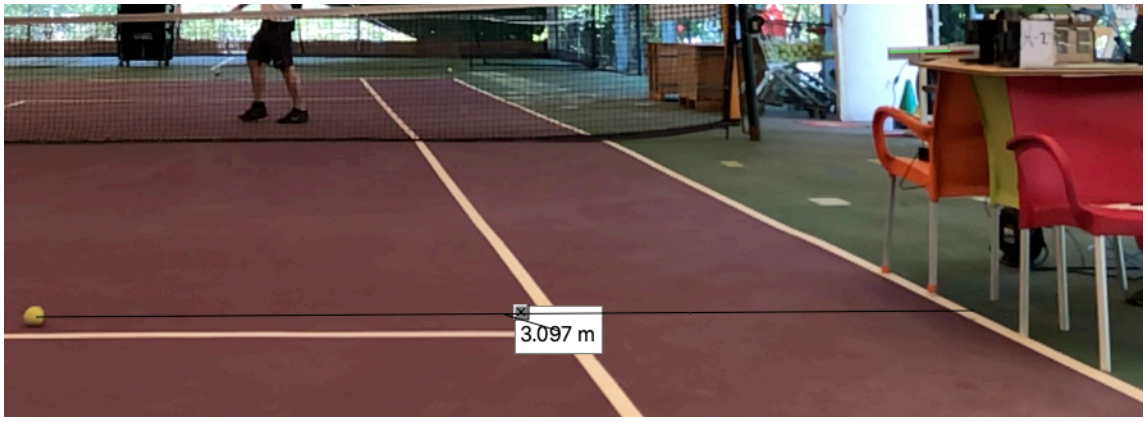


Figure 21: Measuring the Distance that the Tennis Ball Travels Until its Second Bounce

4) Take 3 trials using the same process and repeat for the rotational speeds of -12Hz, -10 Hz, -7Hz, -4 Hz, 0Hz, 7Hz, 10Hz, 12Hz (the negative sign indicates that the ball has backspin)

Qualitative Observations

When the ball reached high rotational speeds in backspin and was thrown, it started having a greater angle with the horizontal. This effect increased with higher rotational speeds.

When the ball reached high rotational speeds in topspin and was thrown, it started having a lower angle with the horizontal. This effect increased with higher rotational speeds.

The pneumatic cylinder recoiled after hitting tennis balls with high rotational speeds.

Data Processing

The theoretical distances that the ball travels are tabulated in Table 1. Their maximums and minimums were found using the uncertainties of the constants. The negative sign before rotational speed indicates that the ball is going with backspin.

Sample Calculation:

The theoretical value is from the code (Appendix 2)

Uncertainty:

$$\frac{Maximum\ Value - Minimum\ Value}{2} \quad \frac{3.320m - 3.187m}{2} = \pm 0.066m$$ [5]

[5] The actual values are to 14 decimal places and that is the reason why this calculation seems wrong but isn't

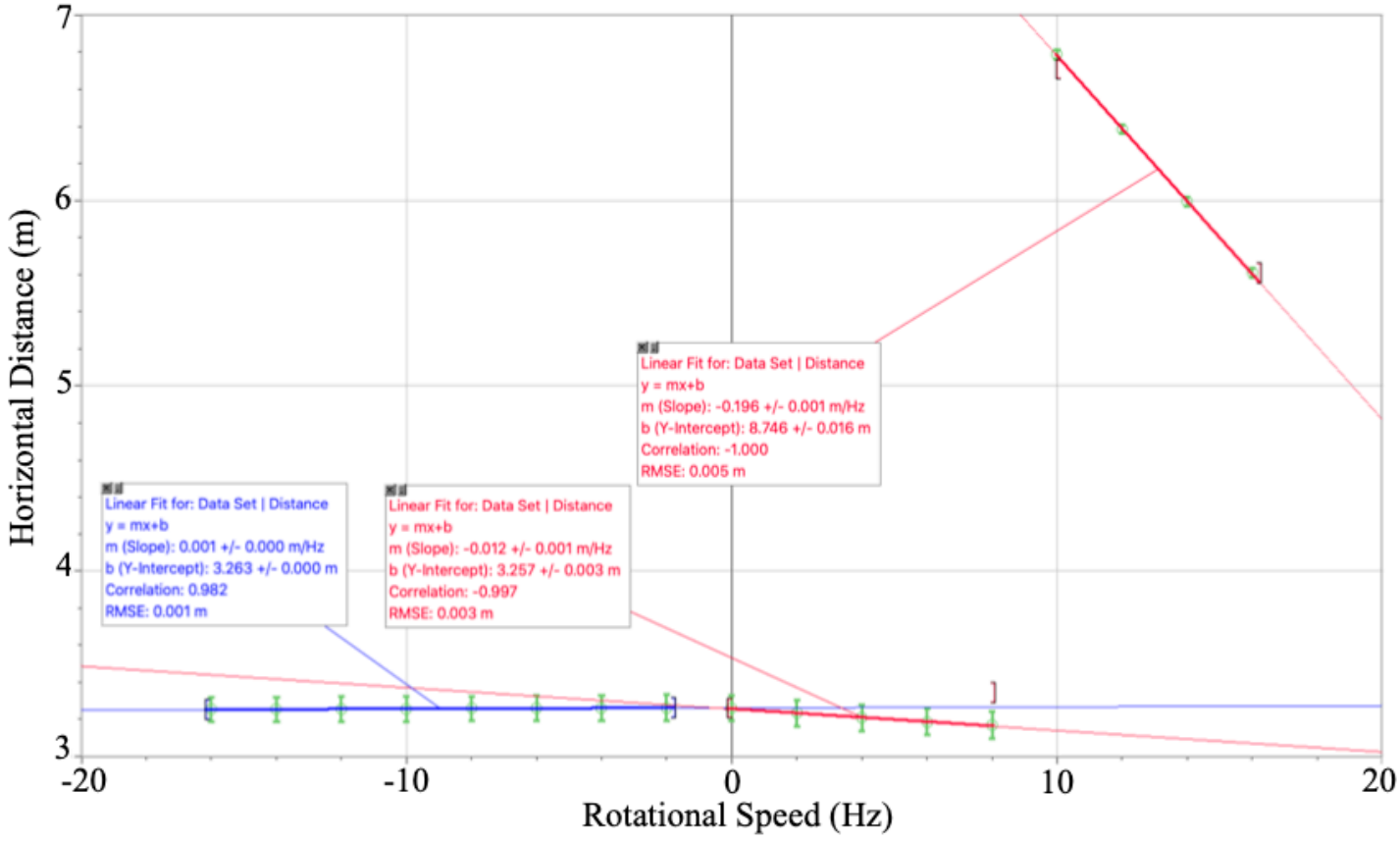
Table 1: The Theoretical Distances and Their Absolute Uncertainties Corresponding to Their Rotational Speeds

Rotational Speed (Hz)	Theoretical Distance (m)	Minimum Distance (m)	Maximum Distance (m)	Uncertainty (m)
-16	3.254	3.187	3.320	0.066
-14	3.254	3.187	3.321	0.067
-12	3.255	3.188	3.322	0.067
-10	3.256	3.188	3.323	0.067
-8	3.258	3.190	3.326	0.068
-6	3.260	3.191	3.328	0.069
-4	3.261	3.192	3.330	0.069
-2	3.262	3.192	3.332	0.070
0	3.260	3.191	3.330	0.070
2	3.233	3.160	3.305	0.072
4	3.208	3.135	3.281	0.073
6	3.187	3.113	3.260	0.074
8	3.168	3.094	3.242	0.074
10	6.787	6.757	6.816	0.029
12	6.386	6.356	6.415	0.030
14	5.994	5.964	6.024	0.030
16	5.609	5.578	5.639	0.031

The data from Table 1 is graphed in Graph 2.

Backspin region is made blue while topspin region is made red for easier understanding.

Graph 2: The Theoretical Horizontal Distance Versus the Rotational Speed of the Tennis Ball



For the backspin region, the ball with around 0Hz and -4Hz is predicted to travel further than a ball with no spin. After that, distance that the ball travels is predicted to decrease linearly with increasing rotational speed.

For the topspin region, the balls with rotational speeds between 0-10Hz are predicted to travel less than a ball with no spin. This linear decrease is faster as it can be seen from the slopes in the Graph 1 (-0.012 for topspin and 0.001 for backspin). The balls with rotational speeds exceeding 10Hz start rolling during the bounce and therefore acquire a much faster speed leaving the ground. This causes them to traverse a much longer distance. In this region, the distance that the ball travels is predicted to decrease linearly with increasing backspin faster than the 0-10Hz region (slope of -0.196 for the above 10Hz region).

The raw data of the distances that the ball travels corresponding to different rotational speeds are tabulated in Table 2 and Table 3.

Table 2: The Distances Corresponding to Their Rotational Speeds for balls going with Topspin

Supposed Rotational Speed (Hz)	Trial	Rotational Speed (Hz)	Distance (m)
0.00	1	0.00	3.277
	2	0.00	3.315
	3	0.00	3.297
4.00	1	4.29	3.155
	2	3.75	3.097
	3	4.00	3.312
7.00	1	6.67	2.987
	2	6.67	3.204
	3	7.50	3.182
10.00	1	10.00	3.020
	2	8.57	3.064
	3	10.00	3.111
12.00	1	12.00	2.757
	2	12.00	2.925
	3	12.00	2.816

Table 3: The Distances Corresponding to Their Rotational Speeds for balls going with Backspin

Supposed Rotational Speed (Hz)	Trial	Rotational Speed (Hz)	Distance (m)
-4.00	1	-4.62	3.358
	2	-4.00	3.471
	3	-4.29	3.313
-7.00	1	-7.50	3.249
	2	-6.67	3.434
	3	-6.67	3.260
-10.00	1	-10.00	3.495
	2	-8.57	3.543
	3	-10.00	3.482
-12.00	1	-12.00	3.509
	2	-12.00	3.569
	3	-12.00	3.549

The data from Table 2 and Table 3 are used to construct Table 4.

Sample Calculation:

$$Average\ value = \frac{trial\ 1 + trial\ 2 + trial\ 3}{3}$$

$$\frac{(-12.00)Hz+(-12.00)Hz+(-12.00)Hz}{3} = -12.00Hz \qquad \frac{3.509m+3.569m+3.549m}{3} = 3.542m$$

Uncertainty:

$$\frac{Maximum\ Value - Minimum\ Value}{2}$$

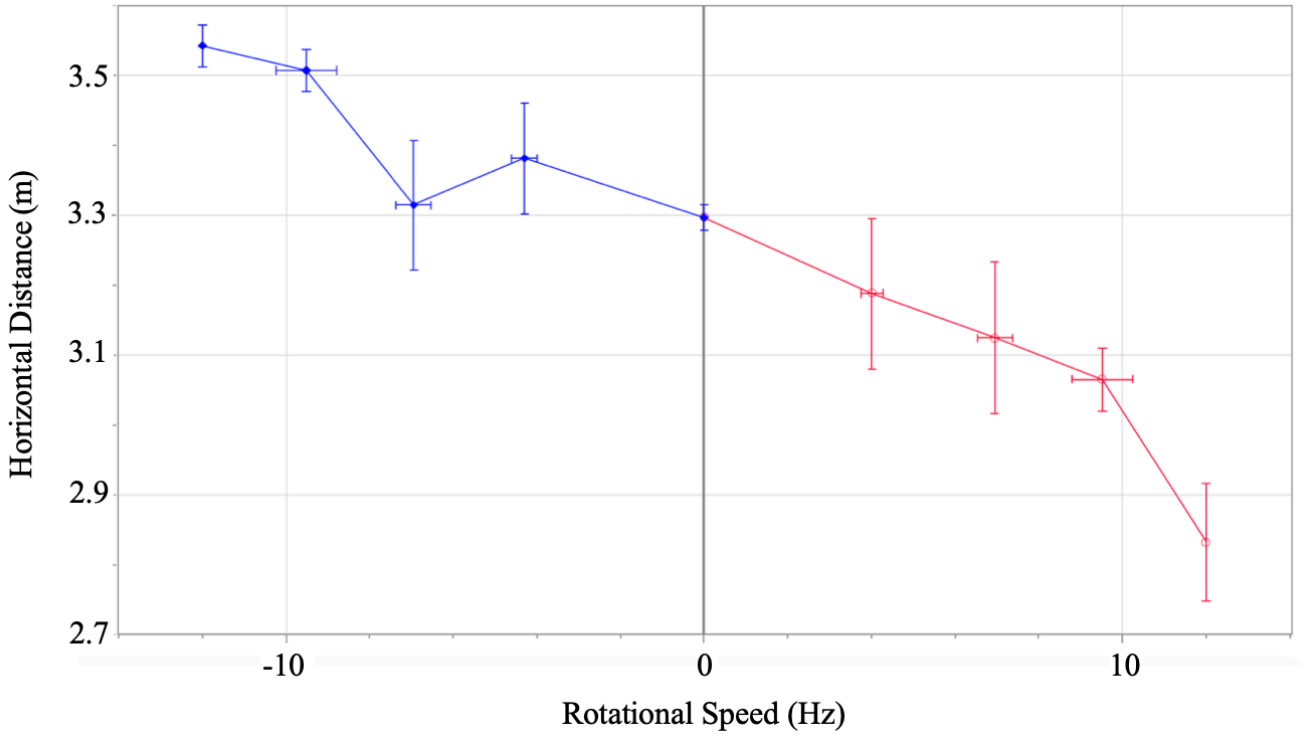
$$\frac{(-12.00)Hz-(-12.00)Hz}{2} = \pm 0.00Hz \qquad \frac{3.569m-3.509m}{2} = \pm 0.030m$$

Table 4: The Average Distances Corresponding to Their Average Rotational Speeds with Their Uncertainties

Average Rotational Speed (Hz)	Rotational Speed Uncertainty (Hz)	Average Distance (m)	Distance Uncertainty (m)
-12.00	0.00	3.542	0.030
-9.52	0.72	3.507	0.031
-6.95	0.42	3.314	0.093
-4.30	0.31	3.381	0.079
0.00	0.00	3.296	0.019
4.01	0.27	3.188	0.108
6.95	0.42	3.124	0.109
9.52	0.72	3.065	0.046
12.00	0.00	2.833	0.084

The data from Table 4 is graphed in Graph 3.

Graph 3: The Horizontal Distance Versus the Rotational Speed of the Tennis Ball

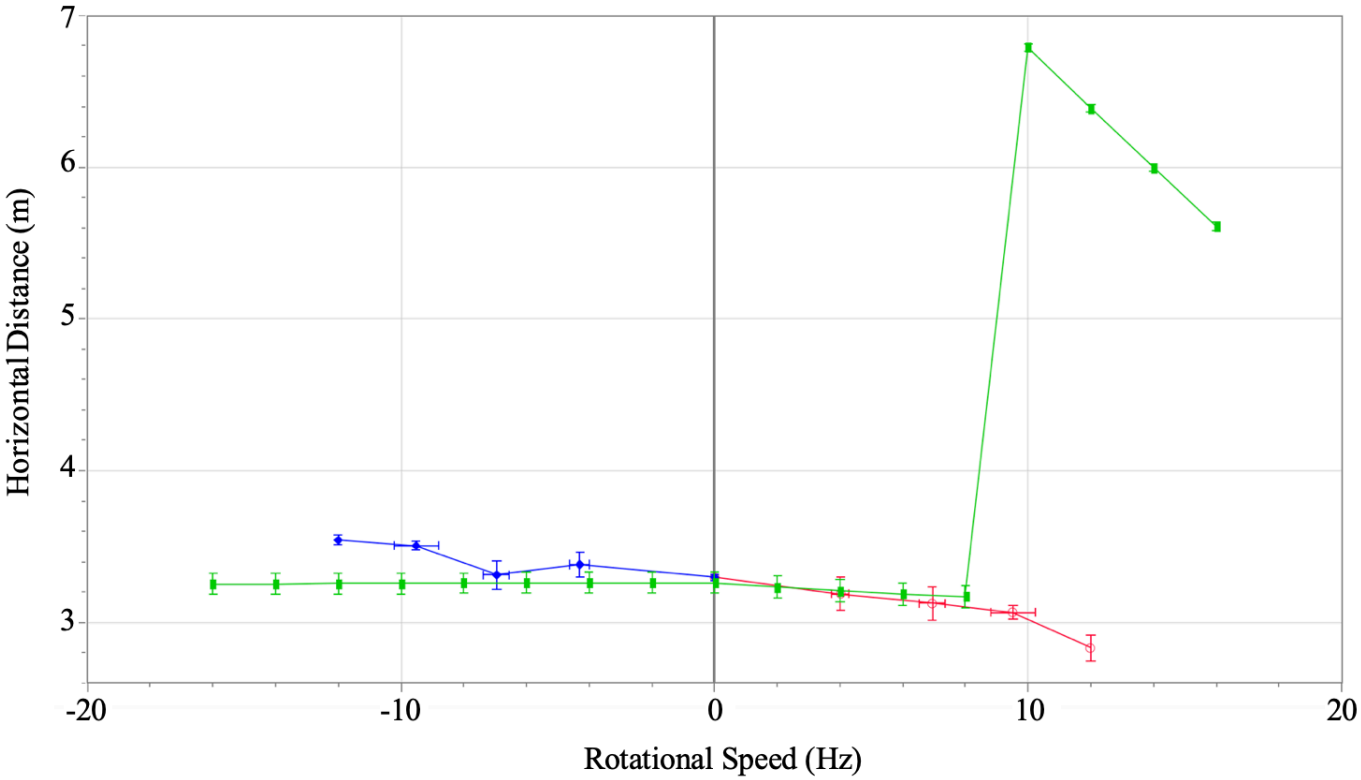


For the backspin region, the data points between -6.95Hz and 0Hz fit the predicted trend with the ball covering a greater distance than a ball with no spin and the linear decrease in distance covered as the rotational speed increases. However, for the balls with rotational speeds higher than -6.95Hz, the distance starts increasing.

For the topspin region, the data points between 0Hz and 9.52Hz fit the predicted trend with a linear decrease. However, there is no sharp rise in distance above 10Hz but instead a sharp drop.

The data from the Graph 2 and Graph 3 were superimposed in Graph 4 for better comparison and explanations of the deviations. Green is the theoretical data, red is topspin, and blue is backspin.

Graph 4: The Horizontal Distance Versus the Rotational Speed of the Tennis Ball, Theoretical and Experimental Data Superimposed

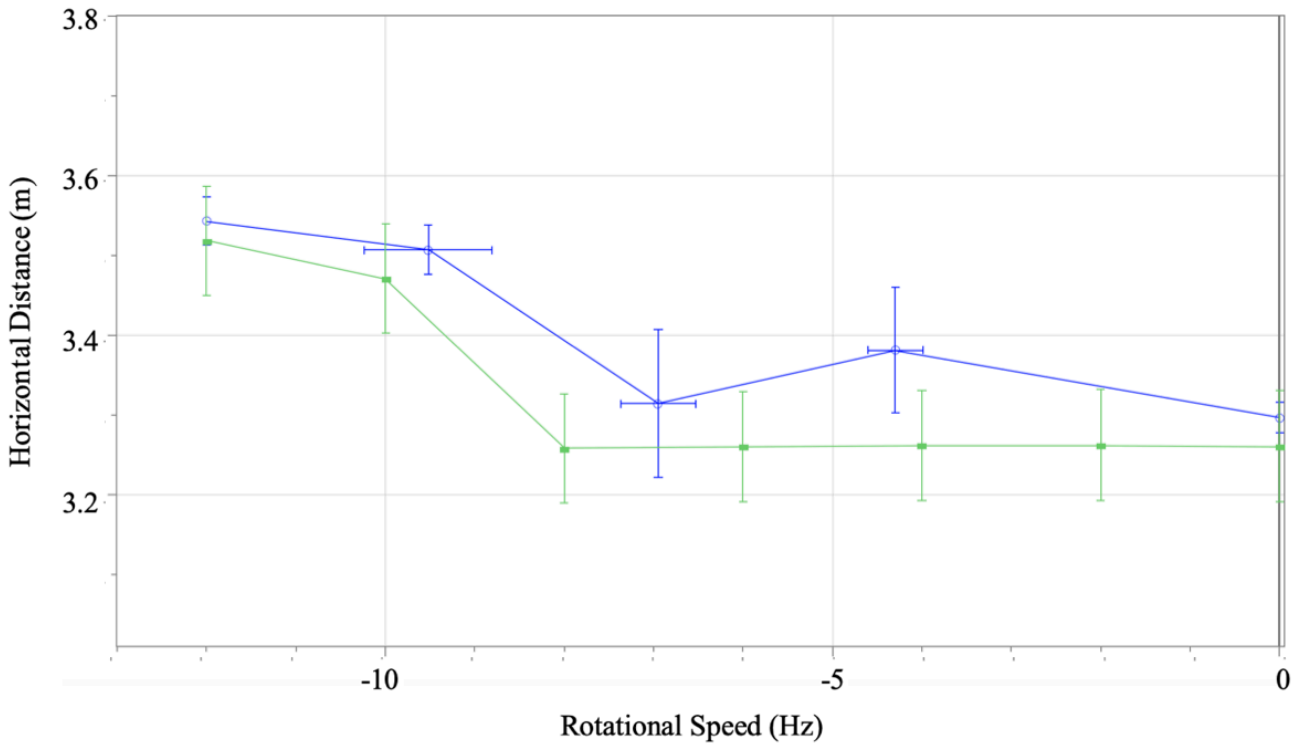


As it can be seen from the Graph 4, the experimental and theoretical values agree between the range of -6.95Hz and 6.95Hz within the error bounds.

The deviations at -9.52Hz and -12Hz is caused by the increasing reaction force between the pneumatic cylinder and the tennis ball.

When the ball is rotating with backspin, the side of the ball that hits the pneumatic cylinder has a tangential velocity pointing downwards. During the impact, that side pushes the pneumatic cylinder down. Due to the Newton's third law of motion, the pneumatic cylinder applies an equal force to the tennis ball in the opposite direction. This effect increases as the rotational speed of the tennis ball increases. When this additional vertical component of velocity is taken into account, the theoretical estimate fits with the experimental results (Graph 5).

Graph 5: The Horizontal Distance Versus the Rotational Speed of the Tennis Ball for the Backspin Region with -10Hz and -12Hz Estimates Corrected, -14Hz and -16Hz Removed

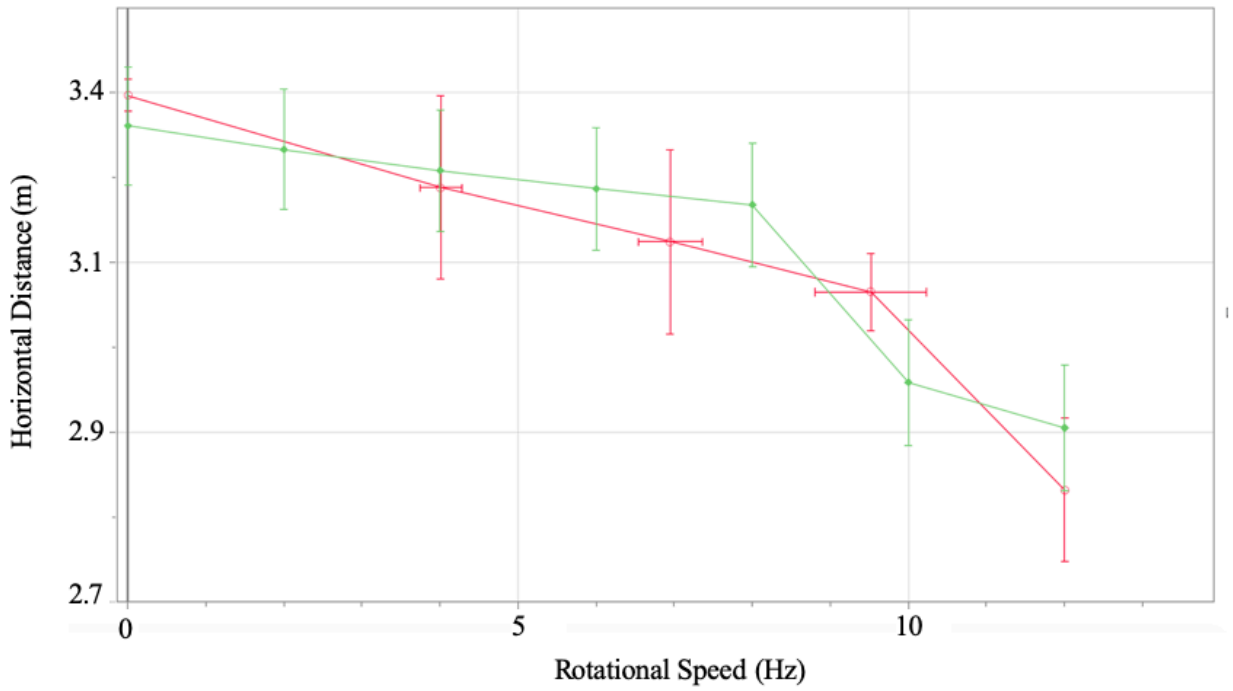


In Graph 5, the data points of -10Hz and -12Hz were given an additional $0.45ms^{-1}$ and $0.55ms^{-1}$ vertical velocity components respectively. This addition makes the angle between the tennis ball and the ground for the -10Hz ball approximately 5.14° and 6.28° for the -12Hz ball. The reason why all the experimental data points are above the theoretical estimates can be explained by the same phenomenon. In lower rotational speeds, however, the reaction force has less effect and doesn't create enough vertical velocity component to make the data points deviate significantly from the expected results. It causes a systematic error nevertheless.

When the ball is rotating with topspin, the side of the ball that hits with the pneumatic cylinder has a tangential velocity pointing upwards. During the impact, that side pushes the pneumatic cylinder up. Due to the Newton's third law of motion, the pneumatic cylinder applies an equal force to the tennis ball in the opposite direction. This effect increases as the rotational speed of the tennis ball increases. However, this experimental error alone is not enough to correct the datapoints. The

theoretical estimate only fits the experimental result if the ball is assumed to not start rolling. When the additional vertical component of velocity is taken into account and the ball is assumed to only slide, theoretical estimate fits with the experimental results (Graph 6).

Graph 6: The Horizontal Distance Versus the Rotational Speed of the Tennis Ball for the Topspin Region with 10Hz and 12Hz Estimates Corrected, 14Hz and 16Hz Removed



In Graph 6, the data points of 10Hz and 12Hz were given an additional $-0.45ms^{-1}$ and $-0.55ms^{-1}$ vertical velocity components respectively. This addition makes the angle between the tennis ball and the ground for the 10Hz ball approximately $-5.14°$ and $-6.28°$ for the 12Hz ball. The reason why all the experimental data points are below the theoretical estimates can be explained by the reaction force. In lower rotational speeds, however, the reaction force has less effect and doesn't create enough vertical velocity component to make the data points deviate significantly from the expected results. It causes a systematic error nevertheless. The reason why 9.52Hz data point is above the theoretical estimate might indicate that it is an outlier and needs further investigation.

Conclusion and Evaluation

The aim of this investigation was to form a mathematical model to describe how the rotational speed of a tennis ball affects the distance that it travels when it is launched parallel to the ground from a height of 1 meter with a speed of $5ms^{-1}$ and compare the predictions coming from the model with the experimental results.

According to the mathematical model, the maximum distance was achieved with the topspin of 12Hz. This estimate contradicted with the experimental findings. The contradiction could not be resolved by only factoring in the error caused by the reaction force between the pneumatic cylinder and the ball. After changing the code to disable rolling during the bounce, the contradiction was resolved. This result leads to a significant conclusion that the tennis ball doesn't roll during the bounce. This finding is supported by Rod Cross suggesting the tennis ball grips the surface, stretches in horizontal direction and starts vibrating instead of commencing rolling (Cross, "Grip-slip behavior" 1093). Therefore, the rolling behavior of a tennis ball deserves further investigation with higher speed footage.

Removing the option of rolling makes the maximum distance to be achieved with the backspin of around (-4.30±0.31)Hz according to the experimental results which also agrees with the mathematical model.

The distance increases up to (-4.30±0.31)Hz and decreases linearly after with increasing rotational speed for backspin while the distance decreases linearly after 0Hz for topspin.

Limitations

The reaction force between the pneumatic cylinder and the ball at higher rotational speeds causes a systematic error. This error can be reduced by working in small rotational speeds or using a faster pneumatic cylinder which reduces the impact time (and therefore the impulse).

After hitting balls with high rotational speeds, the pneumatic cylinder recoils due to loosened pipe clamps. This causes a random error and can be minimized by tightening the pipe clamps after every trial.

The assumption of no viscous torque acting on the tennis ball was validified through measuring the rotational speed of the tennis ball right after it gets thrown and right before it hits the ground. No difference in rotational speed was observed.

The assumptions of the ball not deforming significantly during the bounce and normal force acting through the center of the ball were validified through qualitative observation during video analysis (Figure 22).

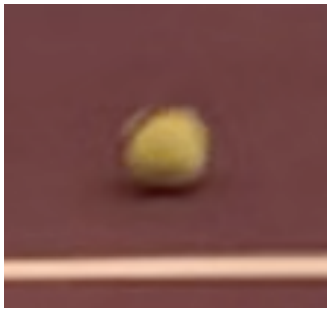


Figure 22: The Tennis Ball During the Bounce

The fastest rotational speed that the tennis ball could acquire was 12Hz and fastest horizontal speed was $5ms^{-1}$ due to the limitations of the apparatus. Therefore, the scope of this investigation was between -12Hz to 12Hz for a height of 1m and speed of $5ms^{-1}$ and the mathematical model was only tested for this region. In this region, $v_x > wr$ was always true and therefore the effect of friction force that acts on the direction of motion described in Figure 10 couldn't be tested. The

accuracy of the mathematical model may also vary depending on the height, speed and the rotational speed range.

Extensions

The effect of friction force that acts on the direction of motion could be investigated by either decreasing the launch speed or increasing the rotational speed.

How launching speed affects horizontal distance travelled may be investigated by keeping the rotational speed constant.

Acknowledgements

I am very grateful to my Extended Essay Supervisor for his continuous support, my dad for recording the videos, and a local company for lending the compressor.

Works Cited

Alam, F, et al. "An Experimental and Computational Study of Tennis Ball Aerodynamics." *The Impact of Technology on Sport II*, 7 Dec. 2007, pp. 324–327., doi:10.1201/9781439828427.ch63.

Brody, Howard. "That's How the Ball Bounces." *The Physics Teacher*, vol. 22, no. 8, 1984, pp. 494–497., doi:10.1119/1.2341635.

Cross, Rod. "Ball Trajectories ." *The Physics and Technology of Tennis*, by Howard Brody et al., Racquet Tech Publishing, 2007, pp. 367–374.

---. "Bounce of a Spinning Ball near Normal Incidence." *American Journal of Physics*, vol. 73, no. 10, 4 July 2005, pp. 914–920., doi:10.1119/1.2008299.

---. “Grip-Slip Behavior of a Bouncing Ball.” *American Journal of Physics*, vol. 70, no. 11, 2002, pp. 1093–1102., doi:10.1119/1.1507792.

---. “Measurements of the Horizontal Coefficient of Restitution for a Superball and a Tennis Ball.” *American Journal of Physics*, vol. 70, no. 5, 2002, pp. 482–489., doi:10.1119/1.1450571.

Engineering ToolBox .“Air - Density, Specific Weight and Thermal Expansion Coefficient at Varying Temperature and Constant Pressures.” 2003, www.engineeringtoolbox.com/air-density-specific-weight-d_600.html.

Ferreira Da Silva, M F. “Meaning and Usefulness of the Coefficient of Restitution.” *European Journal of Physics*, vol. 28, no. 6, 2007, pp. 1219–1232., doi:10.1088/0143-0807/28/6/019.

Garwin, Richard L. “Kinematics of an Ultraelastic Rough Ball.” *American Journal of Physics*, vol. 37, no. 1, 1969, pp. 88–92., doi:10.1119/1.1975420.

Hall, Nancy. “What Is Drag?” *National Aeronautics and Space Administration*, NASA, 5 May 2015, www.grc.nasa.gov/WWW/K-12/airplane/drag1.html.

Lei, U., et al. “Viscous Torque on a Sphere under Arbitrary Rotation.” *Applied Physics Letters*, vol. 89, no. 18, 2006, p. 181908., doi:10.1063/1.2372704.

Mehta, R. D., and J. M. Pallis. “The Aerodynamics of a Tennis Ball.” *Sports Engineering*, vol. 4, no. 4, 13 Jan. 2002, pp. 177–189., doi:10.1046/j.1460-2687.2001.00083.x.

Press, William H., et al. *Numerical Recipes in C: the Art of Scientific Computing*. Cambridge University Press, 1992.

Rdurkacz. “Sketch of Magnus Effect with Streamlines and Turbulent Wake.” *Wikimedia Commons, the Free Media Repository*, 4 Sept. 2017, commons.wikimedia.org/wiki/File:Sketch_of_Magnus_effect_with_streamlines_and_turbulent_wake.svg.

Štěpánek, Antonín. “The Aerodynamics of Tennis Balls—The Topspin Lob.” *American Journal of Physics*, vol. 56, no. 2, 30 Apr. 1987, pp. 138–142., doi:10.1119/1.15692.

Swinburne University of Technology. “Gravitational Constant.” *COSMOS The SAO Encyclopedia of Astronomy*, astronomy.swin.edu.au/cosmos/G/Gravitational+Constant.

Wazir, Ibrahim, et al. *Mathematics 2012 Edition - Higher Level: Developed Specifically for the IB Diploma*. Pearson, 2012.

Weisstein, Eric W. “Moment of Inertia--Spherical Shell .” *ScienceWorld Wolfram*, Wolfram Research, 2007, scienceworld.wolfram.com/physics/MomentofInertiaSphericalShell.html.

Appendix 1 – Constants Used in the Mathematical Model

Rationale for Assuming the Tennis Ball as a Thin Spherical Shell – k and Radius

The moment of inertia of the tennis ball is calculated by assuming that it is a thin spherical shell. The radius of the tennis ball was found by wrapping its circumference by a soft tape measure (Figure A). The circumference was measured to be $(20.40 \pm 0.05)cm$. Dividing the measured value by 2π yields the radius of the tennis ball due to $2\pi r =$ circumference. The thickness of the walls was measured to be $(0.60 \pm 0.05)cm$ by using a ruler after cutting the tennis ball in half (Figure B). The thickness of the walls only being %3 of the total radius validates the assumption of a thin spherical shell. Therefore, the value of k in $I = kmr^2$ is accepted to be $\frac{2}{3}$ (Weisstein).

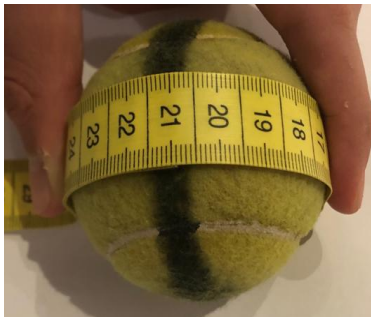


Figure A: Measurement of the Circumference

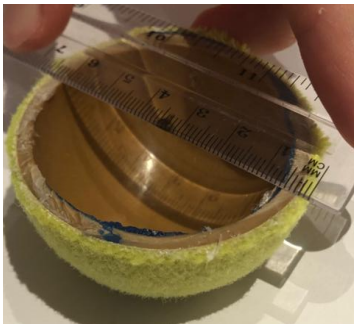


Figure B: Measurement of the Thickness of the Walls

$$Radius = \frac{(20.40 \pm 0.05)cm}{2\pi \cdot 100} = (0.0325 \pm 0.0001)m$$

$$\text{Uncertainty} = \frac{0.05}{2\pi \cdot 100} = \pm 0.0001$$

The Mass of the Tennis Ball

The mass of the tennis ball was measured to be $(57.0 \pm 0.1)g$ by using an electronic balance (Figure C).

$$Mass = \frac{(57.0 \pm 0.1)g}{1000} = (0.0570 \pm 0.0001)kg$$

$$\text{Uncertainty} = \frac{0.1}{1000} = \pm 0.0001$$

Figure C: Measurement of the Mass of the Tennis Ball

Air Density (p)

The value for air density was taken from Engineering Toolbox at 35℃ (as it was the temperature in the experiment day) as $1.146kgm^{-3}$

Gravitational Field Strength (g)

The value for gravitational field strength was taken from Swinburne University of Technology COSMOS - The SAO Encyclopedia of Astronomy as $9.81ms^{-2}$

Bounce Duration (t)

The value for bounce duration was taken from (Cross, "Measurements of the horizontal coefficient of restitution for a superball and a tennis ball" 486) as $0.004s$.

Coefficient of Restitution (e_y)

The tennis ball was dropped from a height and the rebound height was measured with LoggerPro video analysis (Figure D).

$mgh_i = \frac{1}{2}mv_i{}^2$, $\sqrt{2gh_i} = v_i$ where h_i is the dropping height and v_i is the incident velocity

$\frac{1}{2}mv_f{}^2 = mgh_f$, $\sqrt{2gh_f} = v_f$ where v_f is the magnitude of the velocity right after the bounce and h_f is the rebound height

Since $e_y = \frac{v_f}{v_i}$ and $\frac{\sqrt{2gh_f}}{\sqrt{2gh_i}} = \frac{v_f}{v_i}$, $e_y = \sqrt{\frac{h_f}{h_i}}$

$e_y = \sqrt{\frac{(0.642 \pm 0.001)m}{(1.757 \pm 0.001)m}} = 0.604 \pm 0.001$

Uncertainty= $(\frac{0.001}{0.642} + \frac{0.001}{1.757}) \cdot 0.604 = \pm 0.0001$

Figure D: Measurement of the Coefficient of Restitution

Coefficient of Sliding Friction (μ)

Using a similar arrangement in (Cross, "Measurements of the horizontal coefficient of restitution for a superball and a tennis ball" 486) 2 identical tennis balls (which were identical with the ball

used in the experiment) were cut in half and glued to a wooden surface. Additional masses were glued on top of the wooden surface to increase the Friction Force which decreases the relative uncertainty and makes reading the force measurement easier. A Newton Meter was connected to the apparatus and the apparatus was dragged at constant speed on the Tennis court (Figure E). At that moment $F_{sliding\ friction} = F_{on\ the\ Newton\ Meter}$

Inserting $F_{sliding\ friction} = m \cdot g \cdot \mu$ gives $\frac{F_{on\ the\ Newton\ Meter}}{m \cdot g} = \mu$

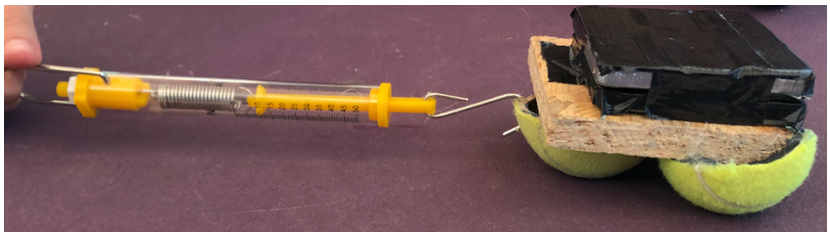


Figure E: Measurement of the Friction Force

The force reading on the Newton Meter was $(11.0 \pm 0.5)N$ and the total mass of the system measured by an electronic balance was $(1715.0 \pm 0.1)g$ which is equal to $(1.7150 \pm 0.0001)kg$. Inserting these values and $9.81ms^{-2}$ for g:

$$\mu = \frac{(11.0 \pm 0.5)N}{(1.7150 \pm 0.0001)kg \cdot 9.81ms^{-2}} = 0.654 \pm 0.030$$

Uncertainty= $(\frac{0.5}{11.0} + \frac{0.0001}{1.7150}) \cdot 0.654 = \pm 0.030$

Height

The height of the ball was measured from its geometric center to the ground with a soft tape and was found to be $(100.00 \pm 0.05)cm$ which equals $(1.000 \pm 0.001)m$.

Appendix 2 – Python Code for Theoretical Distances

The occasional vertical lines on the upper left of the code is the cursor icon seen in the screenshot and doesn't have something to do with the code.

```
import math
from array import *
                                                    # Movement of the Tennis Ball in the Air

# Defining Constants (The uncertainties for the measured values are added as comments next to the values)
# To get the maximum distance insert maximum Mass and VerticalDistance,minimum CoefficientofRestitution, Radius and CoefficientofFriction that the uncertainty allows
# Insert the opposites for the minimum distance
AirDensity = 1.146
Mass = 0.057 # ±0.001kg
Radius = 0.0325 # ±0.0001m
HorizontalVelocity = 5
VerticalVelocity = 0 # To account for the error caused by the reaction force, add a vertical component of velocity here
VerticalDistance = 1.000 #±0.001m (In other words Height)
HorizontalDistance = 0
Velocity = math.sqrt(VerticalVelocity * VerticalVelocity) + math.sqrt(HorizontalVelocity * HorizontalVelocity)
g = 9.81
Constant = (AirDensity * math.pi * Radius * Radius)/(2 * Mass)
t = 0 # time just before tennis ball is launched
k = (2/3) # the tennis ball was assumed a thin spherical shell, the assumption is justified in Appendix 1
StepLength = 0.00001 # as StepLength gets smaller, the prediction becomes more accurate. However, the calculations take longer
# Asking for Spin Value
print("Enter Spin:")
Spin = float(input())
# Using the Spin Value to Calculate Drag and Lift Coefficients
AngularVelocity = 2 * math.pi * Spin
TangentialSpeed = abs(AngularVelocity) * Radius
DragCoefficient = 0.55 + 1/(pow((22.5+(4.2*(pow((Velocity/TangentialSpeed),2.5)))),0.4))
LiftCoefficient = 1/(2+(Velocity/TangentialSpeed))
# Defining Acceleration Functions
def AccelerationFunctionTopSpinY (HorizontalVelocity,VerticalVelocity,LiftCoefficient,DragCoefficient):
    AccelerationTopSpinY = (-1 * Constant * Velocity * ((LiftCoefficient * HorizontalVelocity)+(DragCoefficient * VerticalVelocity)))-g
    return AccelerationTopSpinY;
def AccelerationFunctionTopSpinX (HorizontalVelocity,VerticalVelocity,LiftCoefficient,DragCoefficient):
    AccelerationTopSpinX = Constant * Velocity * ((LiftCoefficient * VerticalVelocity)-(DragCoefficient * HorizontalVelocity))
    return AccelerationTopSpinX;
def AccelerationFunctionBackSpinY (HorizontalVelocity,VerticalVelocity,LiftCoefficient,DragCoefficient):
    AccelerationBackSpinY = (Constant * Velocity * ((LiftCoefficient * HorizontalVelocity)-(DragCoefficient * VerticalVelocity)))-g
    return AccelerationBackSpinY;
def AccelerationFunctionBackSpinX (HorizontalVelocity,VerticalVelocity,LiftCoefficient,DragCoefficient):
    AccelerationBackSpinX = -1 * Constant * Velocity * ((LiftCoefficient * VerticalVelocity)+(DragCoefficient * HorizontalVelocity))
    return AccelerationBackSpinX;

# Runge Kutta 4th Order

def RungeKuttaTopSpinVertical(HorizontalVelocity,VerticalVelocity,LiftCoefficient,DragCoefficient):
    h = StepLength
    k1 = AccelerationFunctionTopSpinY(HorizontalVelocity,VerticalVelocity,LiftCoefficient,DragCoefficient)
    k2 = AccelerationFunctionTopSpinY((HorizontalVelocity+(h/2)),(VerticalVelocity+((h/2)*k1)),LiftCoefficient,DragCoefficient)
    k3 = AccelerationFunctionTopSpinY((HorizontalVelocity+(h/2)),(VerticalVelocity+((h/2)*k2)),LiftCoefficient,DragCoefficient)
    k4 = AccelerationFunctionTopSpinY((HorizontalVelocity+h),(VerticalVelocity+(h*k3)),LiftCoefficient,DragCoefficient)
    return VerticalVelocity + ((1/6)*( k1 + (2 * k2) + (2 * k3) + k4 ) * h);

def RungeKuttaTopSpinHorizontal(HorizontalVelocity,VerticalVelocity,LiftCoefficient,DragCoefficient):
```

```
    h = StepLength
    k1 = AccelerationFunctionTopSpinX(HorizontalVelocity,VerticalVelocity,LiftCoefficient,DragCoefficient)
    k2 = AccelerationFunctionTopSpinX((HorizontalVelocity+(h/2)),(VerticalVelocity+((h/2)*k1)),LiftCoefficient,DragCoefficient)
    k3 = AccelerationFunctionTopSpinX((HorizontalVelocity+(h/2)),(VerticalVelocity+((h/2)*k2)),LiftCoefficient,DragCoefficient)
    k4 = AccelerationFunctionTopSpinX((HorizontalVelocity+h),(VerticalVelocity+(h*k3)),LiftCoefficient,DragCoefficient)
    return HorizontalVelocity + ((1/6)*( k1 + (2 * k2) + (2 * k3) + k4 ) * h);

def RungeKuttaBackSpinVertical(HorizontalVelocity,VerticalVelocity,LiftCoefficient,DragCoefficient):
    h = StepLength
    k1 = AccelerationFunctionBackSpinY(HorizontalVelocity,VerticalVelocity,LiftCoefficient,DragCoefficient)
    k2 = AccelerationFunctionBackSpinY((HorizontalVelocity+(h/2)),(VerticalVelocity+((h/2)*k1)),LiftCoefficient,DragCoefficient)
    k3 = AccelerationFunctionBackSpinY((HorizontalVelocity+(h/2)),(VerticalVelocity+((h/2)*k2)),LiftCoefficient,DragCoefficient)
    k4 = AccelerationFunctionBackSpinY((HorizontalVelocity+h),(VerticalVelocity+(h*k3)),LiftCoefficient,DragCoefficient)
    return VerticalVelocity + ((1/6)*( k1 + (2 * k2) + (2 * k3) + k4 ) * h);

def RungeKuttaBackSpinHorizontal(HorizontalVelocity,VerticalVelocity,LiftCoefficient,DragCoefficient):
    h = StepLength
    k1 = AccelerationFunctionBackSpinX(HorizontalVelocity,VerticalVelocity,LiftCoefficient,DragCoefficient)
    k2 = AccelerationFunctionBackSpinX((HorizontalVelocity+(h/2)),(VerticalVelocity+((h/2)*k1)),LiftCoefficient,DragCoefficient)
    k3 = AccelerationFunctionBackSpinX((HorizontalVelocity+(h/2)),(VerticalVelocity+((h/2)*k2)),LiftCoefficient,DragCoefficient)
    k4 = AccelerationFunctionBackSpinX((HorizontalVelocity+h),(VerticalVelocity+(h*k3)),LiftCoefficient,DragCoefficient)
    return HorizontalVelocity + ((1/6)*( k1 + (2 * k2) + (2 * k3) + k4 ) * h);

#Creating Lists to record values
HorizontalVelocityList = array('d',[])
VerticalVelocityList = array('d',[])
VerticalDistanceList = array('d',[])
HorizontalDistanceList = array('d',[])
AngularVelocityList = array('d',[AngularVelocity])
# Determining the direction of spin
print("\nDoes the ball have a topspin or backspin?")
SpinQuestion = input()
# Using the acceleration functions with the iteration methods to record the trajectory
if SpinQuestion == "top" or SpinQuestion == "topspin":
    while VerticalDistance > 0:
        u= StepLength
        VerticalDistanceList.append(VerticalDistance)
        HorizontalDistanceList.append(HorizontalDistance)
        VerticalVelocityList.append(VerticalVelocity)
        HorizontalVelocityList.append(HorizontalVelocity)
        VerticalDistance = VerticalDistance + (VerticalVelocity * u)
        HorizontalDistance = HorizontalDistance + (HorizontalVelocity * u)
        t= t+u
        Velocity = math.sqrt(VerticalVelocity * VerticalVelocity) + math.sqrt(HorizontalVelocity * HorizontalVelocity)
        DragCoefficient = 0.55 + 1/(pow((22.5+(4.2*(pow((Velocity/TangentialSpeed),2.5)))),0.4))
        LiftCoefficient = 1/(2+(Velocity/TangentialSpeed))
        VerticalVelocity = RungeKuttaTopSpinVertical(HorizontalVelocity,VerticalVelocity,LiftCoefficient,DragCoefficient)
        HorizontalVelocity = RungeKuttaTopSpinHorizontal(HorizontalVelocity,VerticalVelocity,LiftCoefficient,DragCoefficient)

if SpinQuestion == "back" or SpinQuestion == "backspin":
    while VerticalDistance > 0:
        u= StepLength
        VerticalDistanceList.append(VerticalDistance)
        HorizontalDistanceList.append(HorizontalDistance)
```

```
        VerticalVelocityList.append(VerticalVelocity)
        HorizontalVelocityList.append(HorizontalVelocity)
        VerticalDistance = VerticalDistance + (VerticalVelocity * u)
        HorizontalDistance = HorizontalDistance + (HorizontalVelocity * u)
        t= t+u
        Velocity = math.sqrt(VerticalVelocity * VerticalVelocity) + math.sqrt(HorizontalVelocity * HorizontalVelocity)
        DragCoefficient = 0.55 + 1/(pow((22.5+(4.2*(pow((Velocity/TangentialSpeed),2.5)))),0.4))
        LiftCoefficient = 1/(2+(Velocity/TangentialSpeed))
        VerticalVelocity = RungeKuttaBackSpinVertical(HorizontalVelocity,VerticalVelocity,LiftCoefficient,DragCoefficient)
        HorizontalVelocity = RungeKuttaBackSpinHorizontal(HorizontalVelocity,VerticalVelocity,LiftCoefficient,DragCoefficient)

# The print fuctions below can be used to check if some calculation is wrong

#print('Vertical Distance Before the Bounce: ',VerticalDistanceList[-1],"\n")
#print('Horizontal Distance Before the Bounce: ',HorizontalDistanceList[-1],"\n")
#print('Vertical Velocity Before the Bounce: ',VerticalVelocityList[-1],"\n")
#print('Horizontal Velocity Before the Bounce: ',HorizontalVelocityList[-1],"\n")
#print('Angular Velocity Before the Bounce: ',AngularVelocity,"\n")

VerticalVelocity = VerticalVelocityList[-1]
HorizontalVelocity = HorizontalVelocityList[-1]
VerticalDistance = VerticalDistanceList[-1]
HorizontalDistance = HorizontalDistanceList[-1]
t = t - u # Accounting for the extra time that the while loop added
# Defining Constants
BounceDuration = 0.004
CoefficientofRestitution = 0.604 #±0.001
CoefficientofFriction = 0.654 #±0.030
# Defining the bounce function
def BounceFunctionBackSpin(t,VerticalVelocity,HorizontalVelocity,AngularVelocity,k):
    t = t+BounceDuration
    VerticalVelocity = -CoefficientofRestitution * VerticalVelocity
    HorizontalVelocity = HorizontalVelocity + (CoefficientofFriction*((-VerticalVelocity*(CoefficientofRestitution+1))+(g*BounceDuration)))
    AngularVelocity = AngularVelocity + (((CoefficientofFriction/(k*Radius))*(-VerticalVelocity*(CoefficientofRestitution+1)))+(g*BounceDuration))
    Values= array('d',[t,VerticalVelocity,HorizontalVelocity,AngularVelocity])
    return Values;

AngularVelocity = 2 * math.pi * Spin
TangentialSpeed = abs(AngularVelocity) * Radius
# To disable rolling add # before the lines marked as "make comment to disable roll" and remove one indent from the lines marked as "delete indent to disable roll"
def BounceFunctionTopSpin(t,VerticalVelocity,HorizontalVelocity,AngularVelocity,k):
    VerticalVelocity = -CoefficientofRestitution * VerticalVelocity
    t = t+BounceDuration

    if HorizontalVelocity > TangentialSpeed:
        HorizontalVelocityNext = HorizontalVelocity + (CoefficientofFriction*((-VerticalVelocity*(CoefficientofRestitution+1))+(g*BounceDuration)))
        AngularVelocityNext = AngularVelocity + (((CoefficientofFriction/(k*Radius))*(-VerticalVelocity*(CoefficientofRestitution+1)))+(g*BounceDuration))

        if HorizontalVelocityNext < TangentialSpeed: # make comment to disable roll
            HorizontalVelocity = ((HorizontalVelocity-(k*Radius*AngularVelocity))/(1-k)) # make comment to disable roll
            AngularVelocity = Radius * ((HorizontalVelocity-(k*Radius*AngularVelocity))/(1-k)) # make comment to disable roll
        else: # make comment to disable roll
            HorizontalVelocity = HorizontalVelocityNext # delete indent to disable roll
            AngularVelocity = AngularVelocityNext # delete indent to disable roll
```

```
    elif HorizontalVelocity < TangentialSpeed:
        HorizontalVelocityNext = HorizontalVelocity - (CoefficientofFriction*((-VerticalVelocity*(CoefficientofRestitution+1))+(g*BounceDuration)))
        AngularVelocityNext = AngularVelocity - (((CoefficientofFriction/(k*Radius))*(-VerticalVelocity*(CoefficientofRestitution+1)))+(g*BounceDuration))

        if HorizontalVelocityNext > TangentialSpeed: # make comment to disable roll
            HorizontalVelocity = ((HorizontalVelocity-(k*Radius*AngularVelocity))/(1-k)) # make comment to disable roll
            AngularVelocity = Radius * ((HorizontalVelocity-(k*Radius*AngularVelocity))/(1-k)) # make comment to disable roll
        else: # make comment to disable roll
            HorizontalVelocity = HorizontalVelocityNext # delete indent to disable roll
            AngularVelocity = AngularVelocityNext # delete indent to disable roll

    elif HorizontalVelocity == TangentialSpeed: # make comment to disable roll
       HorizontalVelocity = HorizontalVelocity # make comment to disable roll
       AngularVelocity = AngularVelocity # make comment to disable roll


    Values = array('d',[t,VerticalVelocity,HorizontalVelocity,AngularVelocity])
    return Values;
TangentialSpeed = abs(AngularVelocity) * Radius
if SpinQuestion == "back" or SpinQuestion == "backspin":
    Values = BounceFunctionBackSpin(t,VerticalVelocity,HorizontalVelocity,AngularVelocity,k)
if SpinQuestion == "top" or SpinQuestion == "topspin":
    Values = BounceFunctionTopSpin(t,VerticalVelocity,HorizontalVelocity,AngularVelocity,k)

t = Values[0]
VerticalVelocity = Values[1]
HorizontalVelocity = Values[2]
AngularVelocity = Values[3]

# The print fuctions below can be used to check if some calculation is wrong

#print('Vertical Velocity After the Bounce: ',VerticalVelocity,"\n")
#print('Horizontal Velocity After the Bounce: ',HorizontalVelocity,"\n")
#print('Angular Velocity After the Bounce: ',AngularVelocity,"\n")

# The loops below are used to record the trajectory after the bounce

if AngularVelocity<0 and (SpinQuestion == "top" or SpinQuestion == "topspin") :
    AngularVelocity = abs(AngularVelocity)
    while VerticalDistance > 0:
        u= StepLength
        VerticalDistanceList.append(VerticalDistance)
        HorizontalDistanceList.append(HorizontalDistance)
        VerticalVelocityList.append(VerticalVelocity)
        HorizontalVelocityList.append(HorizontalVelocity)
        VerticalDistance = VerticalDistance + (VerticalVelocity * u)
        HorizontalDistance = HorizontalDistance + (HorizontalVelocity * u)
        t= t+u
        Velocity = math.sqrt(VerticalVelocity * VerticalVelocity) + math.sqrt(HorizontalVelocity * HorizontalVelocity)
        DragCoefficient = 0.55 + 1/(pow((22.5+(4.2*(pow((Velocity/TangentialSpeed),2.5)))),0.4))
        LiftCoefficient = 1/(2+(Velocity/TangentialSpeed))
        VerticalVelocity = RungeKuttaBackSpinVertical(HorizontalVelocity,VerticalVelocity,LiftCoefficient,DragCoefficient)
        HorizontalVelocity = RungeKuttaBackSpinHorizontal(HorizontalVelocity,VerticalVelocity,LiftCoefficient,DragCoefficient)

if AngularVelocity>0 and (SpinQuestion == "top" or SpinQuestion == "topspin") :
    while VerticalDistance > 0:
        u= StepLength
```

```
        VerticalDistanceList.append(VerticalDistance)
        HorizontalDistanceList.append(HorizontalDistance)
        VerticalVelocityList.append(VerticalVelocity)
        HorizontalVelocityList.append(HorizontalVelocity)
        VerticalDistance = VerticalDistance + (VerticalVelocity * u)
        HorizontalDistance = HorizontalDistance + (HorizontalVelocity * u)
        t= t+u
        Velocity = math.sqrt(VerticalVelocity * VerticalVelocity) + math.sqrt(HorizontalVelocity * HorizontalVelocity)
        DragCoefficient = 0.55 + 1/(pow((22.5+(4.2*(pow((Velocity/TangentialSpeed),2.5)))),0.4))
        LiftCoefficient = 1/(2+(Velocity/TangentialSpeed))
        VerticalVelocity = RungeKuttaTopSpinVertical(HorizontalVelocity,VerticalVelocity,LiftCoefficient,DragCoefficient)
        HorizontalVelocity = RungeKuttaTopSpinHorizontal(HorizontalVelocity,VerticalVelocity,LiftCoefficient,DragCoefficient)

if AngularVelocity>0 and (SpinQuestion == "back" or SpinQuestion == "backspin") :
    while VerticalDistance > 0:
        u= StepLength
        VerticalDistanceList.append(VerticalDistance)
        HorizontalDistanceList.append(HorizontalDistance)
        VerticalVelocityList.append(VerticalVelocity)
        HorizontalVelocityList.append(HorizontalVelocity)
        VerticalDistance = VerticalDistance + (VerticalVelocity * u)
        HorizontalDistance = HorizontalDistance + (HorizontalVelocity * u)
        t= t+u
        Velocity = math.sqrt(VerticalVelocity * VerticalVelocity) + math.sqrt(HorizontalVelocity * HorizontalVelocity)
        DragCoefficient = 0.55 + 1/(pow((22.5+(4.2*(pow((Velocity/TangentialSpeed),2.5)))),0.4))
        LiftCoefficient = 1/(2+(Velocity/TangentialSpeed))
        VerticalVelocity = RungeKuttaBackSpinVertical(HorizontalVelocity,VerticalVelocity,LiftCoefficient,DragCoefficient)
        HorizontalVelocity = RungeKuttaBackSpinHorizontal(HorizontalVelocity,VerticalVelocity,LiftCoefficient,DragCoefficient)

if AngularVelocity<0 and (SpinQuestion == "back" or SpinQuestion == "backspin") :
    AngularVelocity = abs(AngularVelocity)
    while VerticalDistance > 0:
        u= StepLength
        VerticalDistanceList.append(VerticalDistance)
        HorizontalDistanceList.append(HorizontalDistance)
        VerticalVelocityList.append(VerticalVelocity)
        HorizontalVelocityList.append(HorizontalVelocity)
        VerticalDistance = VerticalDistance + (VerticalVelocity * u)
        HorizontalDistance = HorizontalDistance + (HorizontalVelocity * u)
        t= t+u
        Velocity = math.sqrt(VerticalVelocity * VerticalVelocity) + math.sqrt(HorizontalVelocity * HorizontalVelocity)
        DragCoefficient = 0.55 + 1/(pow((22.5+(4.2*(pow((Velocity/TangentialSpeed),2.5)))),0.4))
        LiftCoefficient = 1/(2+(Velocity/TangentialSpeed))
        VerticalVelocity = RungeKuttaTopSpinVertical(HorizontalVelocity,VerticalVelocity,LiftCoefficient,DragCoefficient)
        HorizontalVelocity = RungeKuttaTopSpinHorizontal(HorizontalVelocity,VerticalVelocity,LiftCoefficient,DragCoefficient)
# Option to graph the predicted trajectory of the tennis ball is given
# If the answer given is no, the final distance the tennis ball reaches until its second bounce is printed.
print("\nAre you graphing the trajectory?")
Answer = input()
if Answer == "yes" or Answer == "y":
    print("\nThe Vertical Distances:\n")
    for x in VerticalDistanceList:
        print(x)
    print("\nThe Horizontal Distances:\n")
    for x in HorizontalDistanceList:
        print(x)
elif Answer == "no" or Answer == "n" or Answer == 1:
    print("\nThe Horizontal Distance:\n")
    print(HorizontalDistanceList[-1])
```

Appendix 3 – Arduino Uno Code for the Spinning Mechanism

Extended_Essay_Spinning_Mechanism §

```
//                                The Spinning Mechanism Code Extended Essay


// Defining the pins where jumper cables are connected to the Arduino Uno

//in1 and in2 are the cables leading to the terminals of the 1st DC motor

const int in1= 7;

const int in2= 6;

//in3 and in4 are the cables leading to the terminals of the 2nd DC motor

const int in3= 5;

const int in4= 4;

//enA is a pwm pin that enables changing the duty cycle to control the speed of the 1st DC motor

const int enA= 9;

//enB is a pwm pin that enables changing the duty cycle to control the speed of the 2nd DC motor

const int enB= 10;

void setup() {

// Setting all the pins as output

  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
  pinMode(enA, OUTPUT);
  pinMode(enB, OUTPUT);

}

void loop() {

// By inserting a value between 0 and 255 in the place of 0, the speed of the DC motors can be adjusted

  analogWrite(enA,0);
  analogWrite(enB,0);

// By giving in1 and in2 values of HIGH and LOW, the direction of the 1st DC motor is controlled

  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);

// By giving in3 and in4 values of HIGH and LOW, the direction of the 2nd DC motor is controlled

  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);

}
```