PROBLEM B - OPTIMAL DIAMETER OF TABLE TENNIS BALLS FOR SPECTATOR ENJOYMENT

TEAM 463

Abstract. In this paper we analyze the effect of increasing the diameter of a table tennis ball on its trajectory while rallying, and how it relates to the enjoyment of spectators. We define spectator enjoyment based on the speed of the shot, as this relates to how easily the ball may be seen, and the spin of the shots, which contributes to the variety of the shots and thus variety of gameplay which is essential to the enjoyment of spectators. As the ball diameter increases, the speed of the shots decrease, as does the total spin. We find that a ball of diameter 44 mm, maintains a balance of reduced speed of ease of observation by spectators, and sufficient spin to maintain shot complexity and variety.

1. Introduction

Table tennis, since its inception into the 1988 Olympics, has been well known as a fast paced sport. In the past, table tennis has been played with a hard smooth paddle, making the use of spin in shots extremely difficult. Since the 1970s, the use of sponged rubber on table tennis paddles gained popularity. With these new paddles, the use of spin in conjunction with speed in table tennis became extremely prevalent, making the sport ever more exciting to watch [3].

Recently however, more elastic paddles have made the game excessively fast and difficult to watch on television [3]. Hence it is necessary to consider methods to slow down the game for general audiences. In this paper, we determine the effect of increasing radius as a method to slow down the game.

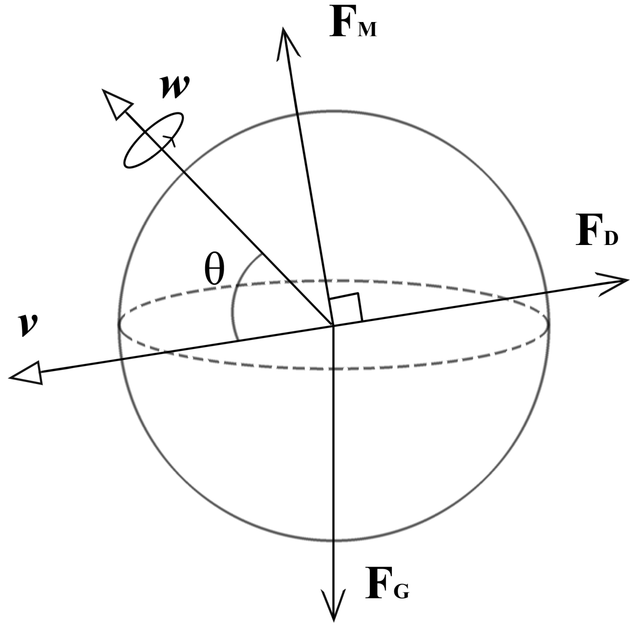
To define what a 'fun' game to watch would be, Team 463 visited a local table tennis tournament to gather the opinion of resident table tennis players. All players agreed that for a general audience, a slower game would be more fun to watch. However, many of them felt that a larger ball was more difficult to spin and thus resulted in a poorer game to watch. Hence, to make the game as fun to watch as possible, we analyze the effect of radius on both the speed and spin of the ball, reaching a compromise between the two.

2. Theory and Assumptions

In order to simplify the question, the speed of the game was quantified by the duration of one rally (i.e. the length of time a ball takes to travel from one edge of the table to the other given some initial velocity, $\mathbf{v_0}$, and spin $\boldsymbol{\omega_0}$). The rules of table tennis require that the ball travels over the net and bounce only once on the other side of the table. It is then natural to break the problem into two distinct parts, the in flight trajectory of the ball, and the rebound of the ball off the table. Following this, we may easily evaluate the time for a single shot by piecing the two solutions together. We present theoretical considerations for both these problems below.

2.1. **In Flight Trajectory.** To derive the in flight trajectory of the ping pong ball, we subject it to three forces: gravity, drag by air resistance and Magnus force. The free body diagram of the ping pong ball is shown in 2.1, and we can use it to derive the differential equation governing the trajectory. We can immediately consider the following equations,

Figure 2.1. Free Body Diagram of Table Tennis Ball in flight. Here $\mathbf{F_G}$ represents the force of gravity, $\mathbf{F_M}$ the magnus force, $\mathbf{F_D}$ the drag, $\boldsymbol{\omega}$ is the angular velocity and $\mathbf{v}$ the velocity.



$$\mathbf{F} = \mathbf{F}_G + \mathbf{F}_M + \mathbf{F}_D \tag{2.1}$$

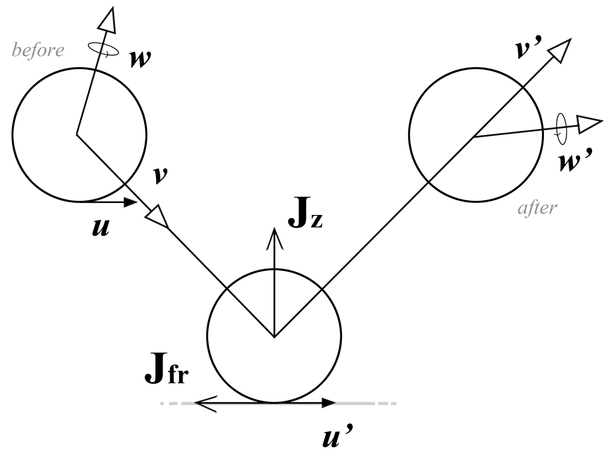
$$\mathbf{F}_G = -m \cdot g\hat{\mathbf{k}} \tag{2.2}$$

$$\mathbf{F}_D = -\frac{1}{2}\rho C_D(\pi R)^2||\mathbf{v}||\mathbf{v}, \tag{2.3}$$

where $\mathbf{F}_G$ accounts for gravity, and $\mathbf{F}_D$ accounts for drag. We denote m as the mass of the ball, g as the acceleration due to gravity, ρ as the air density, C_D as the drag coefficient, and R as the radius of the ball. To derive the Magnus force takes a little more effort. Using the Kutta-Joukowski Lift Theorem [7] for a cylinder, we find that

$$\mathbf{F}_M = \rho(2\pi R)^2(\boldsymbol{\omega} \times \mathbf{v}). \tag{2.4}$$

FIGURE 2.2. Impluse during and velocities before and after the collision with table. $\boldsymbol{\omega}$ and $\boldsymbol{\omega'}$ represent the angular velocity before and after respectively. $\mathbf{v}$ and $\mathbf{v}'$ are likewise the velocities of the ball, and $\mathbf{u}$, $\mathbf{u}'$ the contact velocities of the ball with the table. $\mathbf{J_f r}$ refers to the impulse due to friction with the table, and $\mathbf{J_z}$ is the z-component of impluse.



Integrating along the axis of rotation, we find

$$\mathbf{F}_M = \int_{-R}^{R} \rho \left(2\pi\sqrt{R^2 - x^2}\right)^2 (\boldsymbol{\omega} \times \mathbf{v})\,\mathrm{d}x = \frac{16}{3}\pi^2 \rho R^3 (\boldsymbol{\omega} \times \mathbf{v}), \tag{2.5}$$

which describes the Magnus force acting on a sphere. We then obtain the net force,

$$\mathbf{F} = -mg\hat{\mathbf{k}} + \frac{16}{3}\pi^2 \rho R^3 (\boldsymbol{\omega} \times \mathbf{v}) - \frac{1}{2}\rho C_D (\pi R)^2 ||\mathbf{v}||\mathbf{v}. \tag{2.6}$$

by taking the sum of the individual forces. We further assume that the angular velocity is non-constant, and thus decreases over time due to drag torque. We may use an analogous formula for the drag torque found in [5] where

$$\boldsymbol{\tau}_D = I\frac{d\boldsymbol{\omega}}{dt} = -\frac{1}{2}C_\omega \rho R^5 ||\boldsymbol{\omega}||\boldsymbol{\omega}. \tag{2.7}$$

When numerically solving these differential equations, we can also adjust the rotational velocity in parallel to the main calculation, to account for the decay in rotational velocity, and how it affects the Magnus force.

2.2. Rebound of Ball on Table.

Due to friction as well as the deformation of the ball during play, we cannot expect a simple elastic collision when the ball hits the table. Instead we follow the model of, which has been verified experimentally to good accuracy [1]. For clarity, we give a quick derivation of the model below.

Let $\mathbf{v}$, $\mathbf{v}'$, $\boldsymbol{\omega}$ and $\boldsymbol{\omega'}$ be the velocity and angular velocity of the ball respectively, before and after collision. Let u and u' be the velocity of the contact point of the ball with the table 2.2. It follows that:[1]

$$\mathbf{u} = \mathbf{v_{xy}} + (\boldsymbol{\omega} \times \mathbf{r}) \tag{2.8}$$

Since a table tennis ball is relatively rigid, we treat the ball as a rigid object. This assumption has been shown to be accurate experimentally by [1]. In an inelastic collision, the vertical component of the ball is given by:

$$v_z = -\epsilon v_z', \tag{2.9}$$

where ϵ is the coefficient of restitution of the ball.

Let $\mathbf{J}$ be the impulse of imparted on the ball by the table and μ be the coefficient of sliding friction between the ball and the table. Since the impulse in the z-direction is directly proportional to the normal force, the frictional force (or proportionally the impulse in the xy direction) is governed by:

$$\mathbf{J_{xy}} = -\mu J_z \frac{\mathbf{u}}{||\mathbf{u}||}. \tag{2.10}$$

Here we assume that J_z is always positive since the ball will always rebound into the positive z direction.

Let m, r, and $I = \frac{2}{3}mr^2$ be the mass, radius, and moment of inertia of the ball respectively. From the Impulse-Momentum theorem,

$$m\mathbf{v}' - m\mathbf{v} = \mathbf{J}, \tag{2.11}$$

$$I\boldsymbol{\omega'} - I\boldsymbol{\omega} = \mathbf{r} \times \mathbf{J}. \tag{2.12}$$

Considering 2.11 and 2.12 component-wise, we obtain two new equations:

$$J_z = -m(1+\epsilon)v_z \tag{2.13}$$

$$\mathbf{u}' = -\mu J_z \left(\frac{1}{m} + \frac{r^2}{I}\right)\frac{\mathbf{u}}{||\mathbf{u}||} + \mathbf{u}, \tag{2.14}$$

where $\mathbf{u}'$ is the velocity of the contact point right after the bounce.

[1]All subscripted vectors in this paper refer to the subscripted components only.

Since the contact time of the ball with the table is small, we may make the assumption that the direction of the xy component of $\mathbf{v}'$ does not change until the ball leaves the table. Thus we have the proportionality relation:

$$\mathbf{u}' = c\mathbf{v}' \tag{2.15}$$

while the ball is on the table.

From 2.9, 2.14, and 2.15, we may determine c:

$$c = 1 - \frac{5}{2}\mu\left(1+\epsilon\right)\frac{|v_z|}{||\mathbf{u}||} = 1 - \frac{5}{2}\alpha, \tag{2.16}$$

where

$$\alpha = \mu\left(1+\epsilon\right)\frac{|v_z|}{||\mathbf{u}||}.$$

By our assumption above (reference the direction assumption), we must have $c \geq 0$.

Using equations 2.11, 2.12, and 2.16, we may write the relationship between $\mathbf{v}$ and $\mathbf{v}'$ compactly as:

$$\mathbf{v}' = \mathbf{A_v}\mathbf{v} + \mathbf{B_v}\boldsymbol{\omega}, \tag{2.17}$$

$$\boldsymbol{\omega}' = \mathbf{A}_\omega\mathbf{v} + \mathbf{B}_\omega\boldsymbol{\omega}. \tag{2.18}$$

Physically, the $c > 0$ case is the case of sliding friction, where there is no physical bound on the magnitude of c.

$$\mathbf{A_v} = \begin{pmatrix} 1-\alpha & 0 & 0 \\ 0 & 1-\alpha & 0 \\ 0 & 0 & -\epsilon \end{pmatrix}, \mathbf{B_v} = \begin{pmatrix} 0 & \alpha r & 0 \\ -\alpha r & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \tag{2.19}$$

$$\mathbf{A}_\omega = \begin{pmatrix} 0 & -\frac{3\alpha}{2r} & 0 \\ \frac{3\alpha}{2r} & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \mathbf{B}_\omega = \begin{pmatrix} 1-\frac{3\alpha}{2} & 0 & 0 \\ 0 & 1-\frac{3\alpha}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{2.20}$$

Physically, the $c \leq 0$ case corresponds to rolling friction. By assumption, $\mathbf{u}'$ must be parallel to $\mathbf{u}$. Hence, the minimum physical c corresponds to $\mathbf{u}' = \mathbf{0}$ (when the ball is purely rolling). Hence, $\alpha = \frac{2}{5}$.

Since the rolling case is a limiting case of zero slide, we have:

$$\mathbf{A_v} = \begin{pmatrix} \frac{3}{5} & 0 & 0 \\ 0 & \frac{3}{5} & 0 \\ 0 & 0 & -\epsilon \end{pmatrix} \mathbf{B_v} = \begin{pmatrix} 0 & \frac{2}{5}r & 0 \\ -\frac{2}{5}r & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{2.21}$$

$$\mathbf{A}_\omega = \begin{pmatrix} 0 & -\frac{3}{5r} & 0 \\ \frac{3}{5r} & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \mathbf{B}_\omega = \begin{pmatrix} \frac{2}{5} & 0 & 0 \\ 0 & \frac{2}{5} & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{2.22}$$

2.3. **Estimation of spin and shot time.** First, let us define a valid shot as a shot that goes over the net and bounces once on the other side. From our results above, we may obtain the complete motion of a single valid table tennis shot. We may determine both the total time, t over the entire motion, as well as the total path length, ℓ. Furthermore, we reason that ℓ is a good indicator of spin, as a greater spin leads to a greater effect from the Magnus force, and hence a larger in flight distance.

By averaging ℓ and t over all valid shots due to initial conditions for a particular radius, we can measure both spin and shot time efficiently. These computations are simple to do numerically and are described in the follow section.

3. Numerical Implementation

The motion of the ball was solved numerically using Matlab's ODE suite, by breaking the problem into motion before and after the bounce 3.3. If the motion of the ball did not constitute a valid shot as defined previously, then the test was discarded. The simulation for a valid shot was stopped when

- the ball either hit the table a second time or
- went out of the table.

To determine t and ℓ for a given radius, we may use a Monte Carlo method to simulate randomly over all possible shots using several initial speeds and spins. However, due to time constraints and the speed of the Monte Carlo simulation, we assumed that the ball was always hit from a fixed location across all simulations. This removes 3 dimensions of the simulation. For a given radius of the ball, the Monte Carlo assumed a constant energy and calculated the initial speed and angular speed accordingly. The estimate for energy was obtained from **??**, which gives the speed of one of the fastest table tennis shots ever recorded. We assume that this record shot had purely translational energy and use this energy as the maximum energy a table tennis ball can attain. Additionally, it was assumed that the typical shot would have approximately half of the total energy going into angular velocity and the other half going into translational velocity. Hence $\mathbf{v}$ and $\boldsymbol{\omega}$ follows:

$$||\mathbf{v}|| = \sqrt{\frac{E_{max}}{m}} \tag{3.1}$$

$$||\boldsymbol{\omega}|| = \sqrt{\frac{E_{max}}{I}} \tag{3.2}$$

By keeping E_{max} constant, we may determine the typical speed and angular velocity for any single radius. We can then do a Monte Carlo simulation over all possible directions of the initial velocity and angular velocity. In

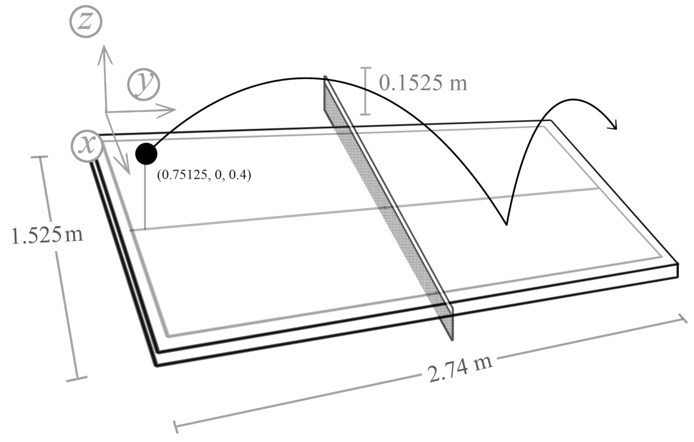


FIGURE 3.1. Table coordinate system used and dimensions.

polar coordinates, this means we take uniform samples over ϕ and θ, where ϕ is the zenith angle and θ is the azimuthal angle on the xy-plane. Hence, the initial velocity and angular velocity is:

$$\mathbf{v} = ||\mathbf{v}||(\sin\phi_1\cos\theta_1, \sin\phi_1\sin\theta_1, \cos\phi_1), \tag{3.3}$$

$$\boldsymbol{\omega} = ||\boldsymbol{\omega}||(\sin\phi_2\cos\theta_2, \sin\phi_2\sin\theta_2, \cos\phi_2). \tag{3.4}$$

From this point we evaluate the effect of changing the radius by performing several simulations over various radii.

To obtain an accurate model, several parameters in the simulation were estimated:

The dimensions of the table were obtained from the International Table Tennis Federation [2]. We always assume that the ball is shot at 0.4 m above the surface of the table, as this height is the average height at which a table tennis ball is typically hit during play [10]. A diagram of the model setup is shown in 3.1

For the in flight portion of the shot, the important forces to consider were gravity, drag and the Magnus force which cases the lift when the ball spins. We estimate the mass of the ball by considering a fixed mass per area of the ball. From [2], we use 2.7 g for the 0.40 mm diameter. Gravity can be estimated to be 9.8 ms^{-2}. For the drag, we required an estimation of the drag coefficient for a hollow sphere, which we took to be 0.5. This value falls within the estimated range in literature [7]. Since the drag coefficient is dependent on the Reynolds number, taking it to be constant is a simplification. However, literature suggests that at higher Reynolds numbers, the drag constant stays roughly the same [8].

FIGURE 3.2. Backspin shot moving left, with $\boldsymbol{\omega}$ = (100,0,0), $\boldsymbol{v} = (0, 15, 0)$, and no spin decay. This is clearly non-physical, but would make for a very exciting spectator sport.

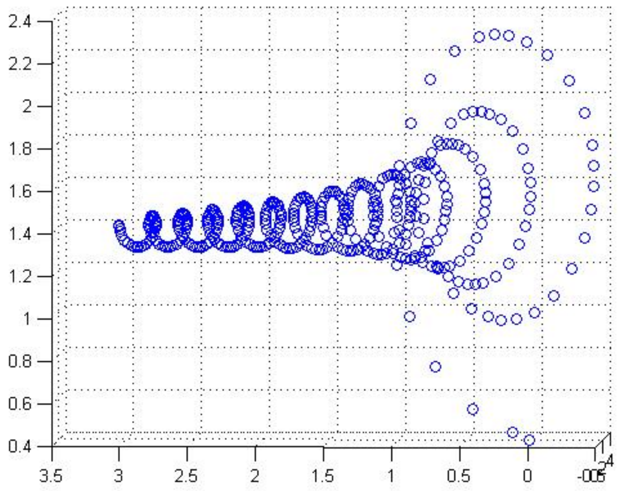
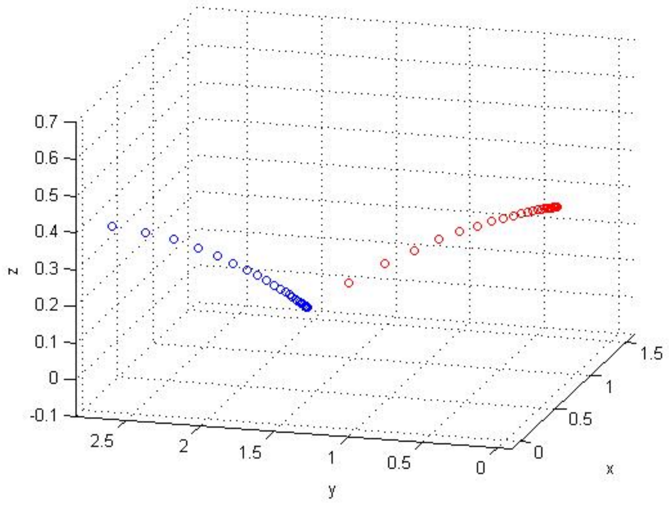


FIGURE 3.3. Simulation of average top spin shot. The red spheres represent the trajectory of the ball before the bounce.



Without considering a drag torque in the spin of the ball, the Magnus force dominates the in flight motion. Since the ball is light, and substantial spin can be delivered to the ball by a well trained player. Without considering decay, the model is highly unrealistic 3.2.

FIGURE 4.1. Average path length of shot for varying radii with input E = 0.39J.

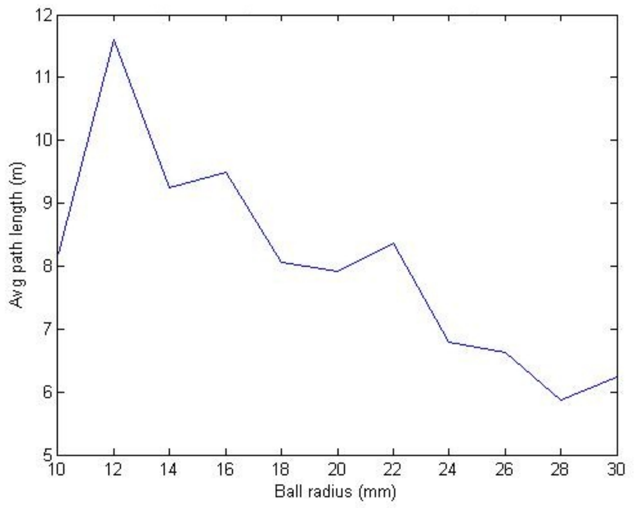
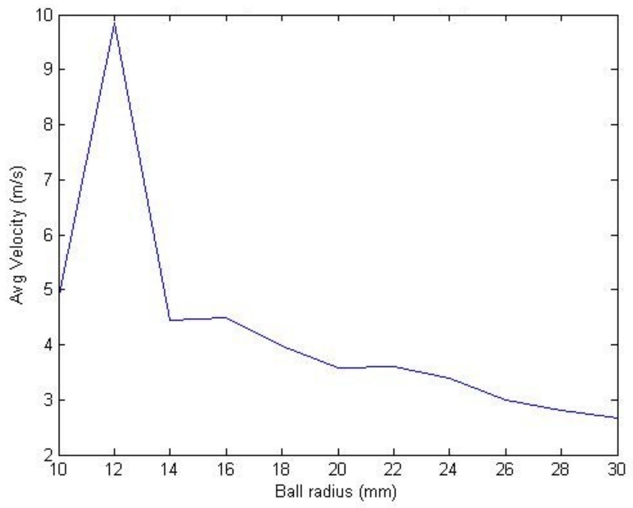


FIGURE 4.2. Average velocity of shot for varying radii with input E = 0.39J.

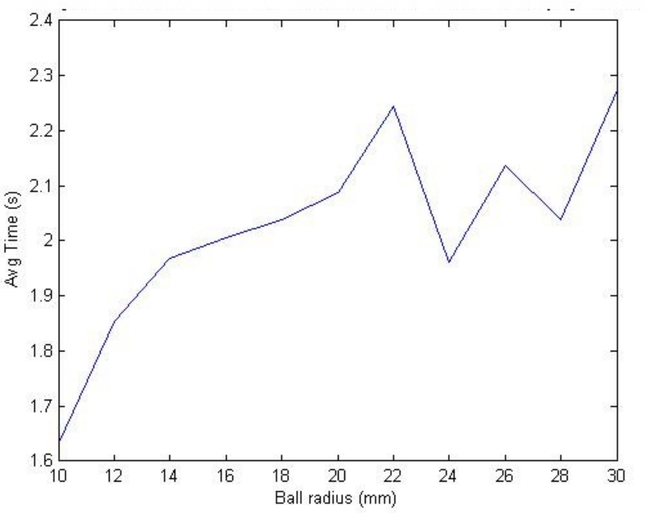


Using [5], we amended the angular velocity at every iteration of the DE solver, whose differential equation was analogous to that of translational drag. The value of C_ω was considered by [9], which demonstrated a formula for C_ω with an error of $o(Re)$. Since the work of [9] was based on a small Reynolds number approximation, the assumption is less valid for a ball with substantial spin. Hence, the $o(Re)$ error term begins to dominate, which was the main factor in our equation when estimating C_ω. The rotational drag coefficient was assumed to be a linear function of the rotational Reynolds number, for which [8] gave an approximate formula. The resulting simulation is much more realistic, though not as visually appealing, as seen in 3.3.

4. Results

Let the average time, and path length for a given radius be t_{av} and $\ell + av$ respectively. Our model shows that as the radius of the ball increases, t_{av}, increases and ℓ_{av} decreases. Moreover, the average speed, calculated by averaging $\frac{\ell}{t}$ over all tested radii decreases as well. For the 38 mm ball, our data shows that the average time for a shot falls within the uncertainty of the experiment conducted by [6]. As expected, the game seems to continually slow down as the radius increases. The larger radius contributes to a larger cross-sectional area which increases drag quadratically, slows down the translational velocity. A larger radius also means that the mass of the ball is greater, as is its moment of inertia, which means that for the same input energy, less spin can be generated, as shown by the decrease in the path length since the balls were curving less during flight.

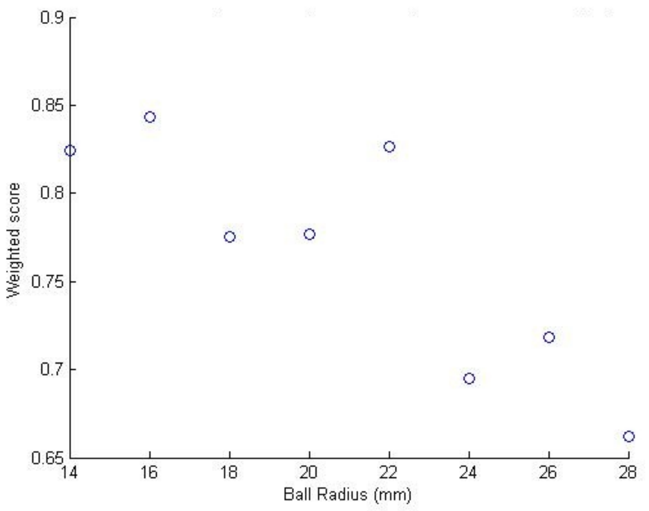
FIGURE 4.3. Average time for shot to make it to the other end of the table, for varying radii with input E = 0.39J.



Even though the Magnus force would increase for an increasing radius, the increase in mass and moment of inertia prevent the ball from achieving the same angular velocity. We see indeed that the decision to change the diameter of the ball to 40 mm did make the game slower. Thus if the aim to slow the ball down, one can increase the size to no end, but this of course is nonsensical. To make the game 'fun' to watch, it must retain a high level of athleticism; the ball cannot be so slow that there is no difficulty for the players to return the ball.

In the end, it remains a subjective opinion on what the optimal speed of the game should be. However, we

FIGURE 4.4. Weighted sum of path length and average time at $\beta = 0.6$.



can do a toy analysis below. At this point, the analysis is largely subjective, as the users of this model can simply select the optimal speed they want according to their specifications. We attempt to maximize the enjoyment of the game for both the players and the spectators. As stated, Team 463 visited a local table tennis tournament to gauge the opinion local table tennis players. When asked to weigh the importance of spin and speed against each other, it was determined that spin was "slightly more" important than speed. This is in part due to the small size of the table tennis table. A player hitting the ball with large speed will not have as much of an advantage as a player familar with spin, since it is quite easy to hit the ball out of bounds. Hence, we weigh the factors of spin and time with a factor β. We intend to normalize the weighted sum of the normalized time and path length, as shown below:

$$f(r) = \beta\frac{\ell_{av}(r)}{\ell_{max}} + (1-\beta)\frac{t_{av}(r)}{t_{max}} \tag{4.1}$$

where $0 \leq \beta \leq 1$, ℓ_{max} is the maximum path length obtained our simulation range, and t_{max} is the maximum time obtained our simulation range. The parameter of β can be manipulated to give a desired weighing of the two factors. We use $\beta = 0.6$, to weigh spin more than time. The function f is plotted in 4.4.

We see in 4.4 that f seems to decrease at increasing radius. The model seems to show that we should pick a ball around either 14 mm or 22 mm in radius. Since smaller balls are harder to see and thus may not be as suitable for television broadcast, we determine 44 mm to be the optimal radius. There are some obvious weaknesses here, which will be further discussed in the next section.

4.1. **Weaknesses.**

There are some weaknesses to our simplified model, mostly owing to the estimation of the parameters which govern the strength of the forces at play. We go into detail in the three sections below.

4.1.1. *The Trajectory Simulation.*

As indicated by 3.2, the Magnus force in itself was too powerful, even though the derivation matches literature. Hence, spin decay of the ball induced by a drag torque was considered. However, owing to the complexity of fluid dynamics, and a highly variable Reynolds number when transitioning between slow spin and fast spin, the estimated dependence of drag on the Reynolds number was likely somewhat inaccurate. We observed that at high initial angular velocities, the limit of how much the ball could change directions after the bounce, and 'curve' (by Magnus) during flight after the bounce, was less than what could be observed during a match, especially on a particularly well-delivered side-spin shot.

4.1.2. *The Monte Carlo Simulation.*

Due to a time and resource constraint, we could not perform the Monte Carlo simulation for as many steps as we hoped for, and were limited to only 2000 iterations per radius. Since the simulation is four dimensional We assumed all shots were made from the edge of the table and a fixed position. Although the position itself was reasonable (roughly 40cm above the table, where most shots are taken when on the offensive in a game), it would not be representative of the whole game. But considering that the game is far slower when a player is on the defensive far behind the court, as they are often lobbing the ball back, this was not our primary concern for when to slow the game down. It as more interesting to see how the game is affected by ball size for a player on the offensive, who must hit the ball faster, (and with a wider angle) than his opponent can reach, and so this is where slowing down the game by changing the radius would have greatest effect. This would also be where most spectators would have trouble tracking the ball, considering the increase in speed during an attack rather than a defensive lob.

Due to the time constraints, we considered only one input energy as our radius for the Monte Carlo simulation. Several table tennis players noted that larger balls spin less [4], but a ball given the same initial angular velocity would experience more lift due to Magnus (for the same perpendicular velocity), not less, since it grows with the radius. Thus the difference is in the input the players can impart to the ball, which means that their ability to impart the same speed and spin must be compromised by an increase to the size of the ball. We place a limit on the kinetic energy that can be put into the ball to account for this, so that even though a larger ball may have spin imparted to it, the amount of spin decreases due to the increase of the moment of inertia, and the decrease in the perpendicular component of the translational velocity.

From 4.4, we see that the simulations made were not enough to give us a detailed profile of f. Furthermore, due to the random simulation, the noise of the randomness can be easily seen in all of the simulations generated. The solution to these problems would be to run a longer simulation with larger iterations. However, we were unable to do this within the prescribed time limit, making this model potentially inaccurate.

5. Conclusions

From Monte Carlo simulations, we see that the game will slow down as the radius increases. Hence the 40 mm ball results in both a slower spin and a slower game than the 38 mm ball. Taking into account both the spin and the speed of the game, we determine the optimal ball size to be 44 mm.

References

1. Y. Ogawa A. Nakashima, Y. Kobayashi and Y. Hayakawa, *Modeling of rebound phenomenon between ball and racket rubber with spinning effect*, ICCAS-SICE (2009), 2295 –2300.
2. International Table Tennis Federation, *The laws of table tennis*, handbook, 2012/2013.
3. The International Table Tennis Federation, *A comprehensive history of table tennis*, handbook, 2010.
4. S. Toyoshima H. Tang, M. Mizoguchi, *Speed and spin characteristics of the 40mm table tennis ball*, ITTF (2000).
5. Y. Kvurt J. Miles N. Lukerchenko, I. Keita, *Experimental evaluation of the drag torque, drag force and magnus force acting on a rotating prolate spheroid*, Colloquium Fluid Dynamics (2010).
6. et al. N. Yuza, *Game analysis of table tennis in top japanese player of different playing styles*, ITTF (1992).
7. T. Benson (NASA), *Lift of a rotating cylindar - http://www.grc.nasa.gov/www/k-12/airplane/cyl.html*, Web Article, 2010.
8. M. Rhodes, *Introduction to particle technology*, 2008.
9. S. I. Rubinow and Joseph B. Keller, *The transverse force on a spinning sphere moving in a viscous fluid*, J. Fluid Mech. **2011** (1961), 447–459.
10. Z. F. Qin W. Xie, K.C. Teh, *Speed and spin of the 40mm table tennis ball and the effects on elite players*, ISBS (2002).

6. Appendix

6.1. Table of Values.

All constants used in the paper are defined below for convenience.

r		radius of the ball in meters
m_r		mass of ball of radius r
$\mathbf{v}$		translational velocity of ball
$\boldsymbol{\omega}$		translational velocity of ball
$\mathbf{F_G}$		force by gravity
$\mathbf{F_D}$		force by drag
$\mathbf{F_M}$		force by Magnus
g	$9.8\,\mathrm{ms}^{-2}$	acceleration by gravity
C_D	0.5	drag coefficient of hollow sphere
ρ	$1.225\,\mathrm{kgm}^{-3}$	density of air
μ	0.25	coefficient of kinetic friction between ball and table
table length	$2.74\,\mathrm{m}$	length of table tennis table
table width	$1.525\,\mathrm{m}$	width of table tennis table
net height	$0.1525\,\mathrm{m}$	table tennis table net height
E_{max}	$\frac{1}{2}m_{0.020}\|\|\mathbf{v}_{avg}\|\|^2$	Input energy used, equal to kinetic energy of average speed shot with no spin.
ϵ	0.93	coefficient of restitution

6.2. Code.

```
function [L, V, T, SL, SV, ST, N] = final_plot(R)
% tries a range of r's, plots avg path length and avg time.
s = size(R);
L = zeros(s);
V = zeros(s);
T = zeros(s);
SL = zeros(s);
SV = zeros(s);
ST = zeros(s);
N = zeros(s);


c = 1;
for r = R
     [l, v, t, sl, sv, st, n] = all_shots(r/1000);
     L(c) = l;
     V(c) = v;
     T(c) = t;
     SL(c) = sl;
     SV(c) = sv;
     ST(c) = st;
     N(c) = n;

     c = c+1;
end

figure
plot(R, L, 'r');
figure
plot(R, T, 'b');

end
```

```
function [l, v, t, sl, sv, st, n] = all_shots( r )
% average path length and velocity over all shots
% for a particular radius r
```

```
    changeR(r);

    pause(1);
    clear all;

    E = getConstant('energy');
    m = getConstant('m');
    r = getConstant('R');

    I = (2/3) * m * r^2;
    v0 = sqrt(2*(1/2)*E/m);
    w0 = sqrt(2*(1/2)*E/I);
    r0 = [1.525/2, 0, 0.4];
    disp(v0);
    disp(w0);
    disp(m);
    disp(r);
    [l, v, t, sl, sv, st, n] = ...
        monte_carlo(r0, v0, w0, 2000);

end
```

```
function [avg_path, avg_vel, avg_time, stdev_path, stdev_vel, stdev_t, num_valid] = monte_carlo(r0, ↩
    v0_norm, w0_norm, step_size)
% Monte carlo simulation of various shots made
% Shot input is anywhere going forward with norm >= v_min, <= v_max

table_l = getConstant('table_l');
table_w = getConstant('table_w');
R = getConstant('R');

count = 0;
table_bounce_count = 0;

paths = zeros(step_size,1);
vels = zeros(step_size,1);
times = zeros(step_size,1);

for i = 1:step_size
    v_theta = rand * pi;
    v_phi = rand * 100/180 * pi;
    w_theta = rand * 2*pi;
    w_phi = rand * pi;

    [ traj, t, ~, ~, valid ] = ...
        simulate_shot( r0, ...
        v0_norm*[sin(v_phi)*cos(v_theta), sin(v_phi)*sin(v_theta), cos(v_phi)], ...
        w0_norm*[sin(w_phi)*cos(w_theta), sin(w_phi)*sin(w_theta), cos(w_phi)], 0.05 );

    if (~valid)
        continue;
    end

    % Determine when the shot goes out of bounds
    for j = 1:min(size(t,1),size(traj,1))
        if (traj(j,3) < R)
            table_bounce_count = table_bounce_count + 1;
        end

        if ((~(0 <= traj(j,1) && traj(j,1) <= table_w &&...
              0 <= traj(j,2) && traj(j,2) <= table_l)) || ...
              (j == min(size(t,1),size(traj, 1))) || ...
              table_bounce_count == 2)
            tot_time = t(j);
            break;
        end
    end
    count = count+1;

    traj_diff = diff(traj);
    traj_diff = traj_diff .* traj_diff;
    traj_diff = sum(traj_diff,2);
    traj_diff = sqrt(traj_diff);
    path_length = sum(traj_diff);

    paths(count) = path_length;
    vels(count) = path_length/tot_time;
    times(count) = tot_time;

    table_bounce_count = 0;

end
%count is the number of valid tests

avg_path = mean(paths(1:count));
avg_vel = mean(vels(1:count));
avg_time = mean(times(1:count));

stdev_path = std(paths(1:count));
stdev_vel = std(vels(1:count));
stdev_t = std(times(1:count));

num_valid = count;
end
```

```
function [ traj, t, v, w, valid ] = simulate_shot( r0, v0, w0, step )
% simulates a single shot with bounce.
% valid is false if shot is out of bounds/into the net
v = zeros(6,1);
v(1:3) = v0;
v(4:6) = w0;

[r2, traj1, vel, tot_time1, over_net_flag, in_table_flag] = ...
    get_position_on_table(r0, v, step);

t = [];

%did the ball go out of bounds?
if (~over_net_flag || ~in_table_flag)
    v = 0;
    w = 0;
    traj = 0;
    valid = false;
    return;
end

%time for first part of shot
t = tot_time1;

%new spin and velocity before bounce.
v1 = vel(1:3);
w1 = vel(4:6);

%new spin and velocity after bounce.
[v2, w2] = bounce(v1, w1);
v(1:3) = v2;
v(4:6) = w2;

%colour of first path before bounce
c = repmat([1 0 0], size(traj1,1), 1);
```

```
[pos, traj2, vel, tot_time2, over_net_flag, in_table_flag] = ...
    get_position_on_table(r2, v, step);

%total time for shot
t = [t ; t(end) + tot_time2];

%colour of second path after bounce
d = repmat([0 0 1], size(traj2,1), 1);

warning('off', 'MATLAB:hg:patch:RGBColorDataNotSupported');
C = [c; d];

v = vel(1:3);
w = vel(4:6);
traj = [traj1; traj2];
valid = 1;

scatter3(traj(:,1), traj(:,2), traj(:,3), 30, C );

xlabel('x');
ylabel('y');
zlabel('z');

table_l = getConstant('table_l');
table_w = getConstant('table_w');
xlim([-0.1 table_w+0.1]);
ylim([-0.1 table_l+0.1]);
zlim([-0.1, 0.7]);

end
```

```
function [pos, traj, vel , tot_time, over_net_flag, in_table_flag] = get_position_on_table(r0, v0, ↩
    stepSize)
% Returns (x,y) position of ball after it hits the table for a given
% initial position (r0) and initial velocity (v0).
% Requires r0(3)>R (z coordinate of bottom of ball must be above the
% table)
% table height is assumed to be 0

table_l = getConstant('table_l');
table_w = getConstant('table_w');
net_h = getConstant('net_h');
R = getConstant('R');

if (r0(1,3) <= R)
    return
end

%% Solve for trajectory
options = odeset('MaxStep', stepSize);
[t_vel, v] = ode23(@ball_flight_vel, [0 5], v0, options);

[t_pos, r] = ball_flight_pos(r0, t_vel, v(:,1:3));

%% Binary search to check if the ball went over the net

left = 1;
right = size(r,1);
mid = floor((left+right)/2);
R = getConstant('R');

while (not(r(mid,2) < (table_l/2) && r(mid+1,2) >= (table_l/2)))
    if (r(mid,2) > (table_l/2))
        right = mid;
    else
        left = mid;
    end
    mid = floor((left+right)/2);
    if (mid == 1 || mid == size(r,1)-1)
        break;
    end
end

if (r(mid,3) > net_h + R)
    over_net_flag = 1;
else
    over_net_flag = 0;
end

%% Binary search to find point where ball hits table
left = 1;
right = size(r,1);
mid = floor((left+right)/2);

while (not(r(mid,3) > R && r(mid+1,3) <= R))
    if (r(mid+1,3) > R)
        left = mid;
    else
        right = mid;
    end

    mid = floor((left+right)/2);
    if (mid == 1 || mid == size(r,1)-1)
        break;
    end
end

pos = r(mid, :);
traj = r(1:mid, :);
vel = v(mid, :);
tot_time = t_pos;

%% Check if the ball landed in the table
if (table_l/2 <= pos(2) && pos(2) <= table_l && 0 <= pos(1) && pos(1) <= table_w)
    in_table_flag = 1;
else
    in_table_flag = 0;
end

end
```

```
function [v2, w2] = bounce( v1, w1 )
% obtains the new velocity (v2) and new spin (w2) of the ball after one
% bounce on the table with initial velocity (v1) and spin (w2)
mu = getConstant('mu');
cor = getConstant('cor');
r = getConstant('R');

%u = v_xy + (w x r);
u = v1(1:2) + [ -r * w1(2), r * w1(1) ];

%alpha = mu * (1+coeff of restitution) * |v_z|/||u||
a = mu * (1+cor) * abs(v1(3))/norm(u, 2);
a = min(a, 2/5);

Av = [1-a  0    0;
     0     1-a  0;
     0     0    -cor];
Bv = [0     a*r  0;
      -a*r 0    0;
      0     0    0];
```

```
Aw = [0             -3*a/(2*r)    0;
      3*a/(2*r)     0             0;
      0             0             0];
Bw = [1-3*a/2    0          0;
      0          1-3*a/2    0;
      0          0          0];

v2 = Av*v1' + Bv*w1';
w2 = Aw*v1' + Bw*w1';

end
```

```
function vp = ball_flight_vel(t,v)
% Velocity of ping pong ball during flight influenced by gravity, drag and
% magnus force
% v(1:3) is translational velocity
% v(4:6) is rotational velocity

    R = getConstant('R');
    rho = getConstant('rho');
    kin_vis = getConstant('kin_vis');
    m = getConstant('m');
    mag_const = 16/3*pi*pi*R^3;
    drag_const = 0.5*pi*R^2*getConstant('C_d')*rho;
    v_norm = norm(v(1:3),2);
    vp(1:3,1) = 1/m * ...
              [-drag_const*v_norm,    -mag_const*v(6),        mag_const*v(5);

                mag_const*v(6),       -drag_const*v_norm,    -mag_const*v(4);

               -mag_const*v(5),        mag_const*v(4),       -drag_const*v_norm] * v(1:3);

    vp = vp - [0,0,getConstant('g')]';

    %Update angular rotation
    I = 2/3 * m * R^2;
    w_norm = norm(v(4:6),2);
    Re = w_norm*R*2*R/kin_vis;

    rot_drag_cost = 0.5 * R^5 * rho * Re/I;
    vp(4:6) = -rot_drag_cost * diag([w_norm,w_norm,w_norm]) * v(4:6);
end
```

```
function [t_new, r] = ball_flight_pos(init, t, v)
% Given the velocity at given times (via ode**), returns the position of
% the ball

    del_t = diff(t);
    del_t = repmat(del_t,1,3);
    v_times_del_t = del_t .* v(1:end-1,:);

    r = cumsum(v_times_del_t,1);
    init = repmat(init,size(r,1),1);
    r = r + init;
    t_new = t(1:end-1,:);

end
```

```
function changeR( r )
% changes the radius defined in the constants file

    str = fileread('getConstant.m');
    rep = horzcat('R = ', num2str(r), ';');
    str = regexprep(str, 'R = .*;', rep, 'dotexceptnewline');

    fid = fopen('getConstant.m', 'w');
    fwrite(fid, str, '*char');              %# write characters (bytes)
    fclose(fid);

end
```

```
function res = getConstant(name)
% Constants. A convoluted way to share global constants across different
% m-files.
    R = 0.03;
    rho = 1.225;
    C_d = 0.5;

    g = 9.8;
    abs_vis = 1.81*10^-5;
    kin_vis = abs_vis / rho;
    mu = 0.25;
    cor = 0.93;
    table_l = 2.74;
    table_w = 1.525;
    net_h = 0.1525;

    m_40 = 0.0027; % mass of the 40mm ball
    mass_p_area = m_40/(4*pi*(0.02)^2);
    m = mass_p_area * 4 * pi * R * R;

    v_avg = 17; %typical speed of a fast ping pong ball during play;
    E = 0.5 * m_40 * v_avg * v_avg; %an estimate for the maximum energy of the ball

    if (strcmp(name, 'C_d'))
        res =  C_d;
        return
    end
    if (strcmp(name, 'R'))
        res =  R;
        return
    end
    if (strcmp(name, 'rho'))
        res =  rho;
        return
    end
    if (strcmp(name, 'm'))
        res =  mass_p_area * 4 * pi * R * R;
        return
    end
    if (strcmp(name, 'g'))
        res =  g;
        return
    end
    if (strcmp(name, 'kin_vis'))
        res = kin_vis;
        return
    end
    if (strcmp(name, 'abs_vis'))
        res = abs_vis;
        return
    end
    if (strcmp(name, 'mu'))
        res = mu;
        return
    end
    if (strcmp(name, 'cor'))
        res = cor;
        return
    end
    if (strcmp(name, 'table_l'))
        res =  table_l;
        return
    end
    if (strcmp(name, 'table_w'))
        res =  table_w;
        return
    end
    if (strcmp(name, 'net_h'))
        res =  net_h;
        return
    end
    if (strcmp(name, 'energy'))
        res =  E;
        return
    end
    if (strcmp(name, 'v_avg'))
        res =  v_avg;
        return
    end


end
```