ISSN 2664-4150 (Print) & ISSN 2664-794X (Online)
South Asian Research Journal of Engineering and Technology
Abbreviated Key Title: South Asian Res J Eng Tech

| Volume-7 | Issue-4 | Jul-Aug- 2025 | **DOI: https://doi.org/10.36346/sarjet.2025.v07i04.003**

Original Research Article

Modeling the Effect of Air Drag on Basketball Trajectories: An Analysis of Trajectory Deviation, Drag Coefficient, and Optimal Launch Angle

Haohang Li[1*], ZhenWei Tian[1], Haohan Pu[1], Aditya Vishwakarma Aditya Vishwakarma[1]
[1]Dipont Huayao Collegiate School Kunshan, Kunshan, Suzhou, China

***Corresponding Author:** Haohang Li
Dipont Huayao Collegiate School Kunshan, Kunshan, Suzhou, China

Article History
Received: 07.06.2025
Accepted: 15.07.2025
Published: 21.07.2025

Abstract: Projectile motion is a fundamental topic in physics, typically taught under idealized conditions such as the absence of air resistance. However, real-world scenarios—such as basketball shooting—show significant deviations from these idealized trajectories. In this study, we investigate the effects of air drag on basketball trajectories, focusing on how it alters the projectile's path, the effective drag coefficient, and the optimal launch angle for maximum range. We assume standard conditions: gravitational acceleration at 9.8 m/s², air density at 1.225 kg/m³, and negligible Magnus effect. Despite limitations in our model (discussed later), it successfully reveals the differences between ideal and real trajectories, estimates the average drag coefficient (~1.15), and identifies a realistic optimal launch angle (lower than the ideal 45°). This research encourages further refinement of the model and highlights the complexity of real-world physics beyond textbook assumptions.

Keywords: Projectile motion, Air resistance, Air drag, Basketball trajectories, Physics.

INTRODUCTION

Projectile motion is a cornerstone of classical mechanics because it exemplifies two-dimensional motion. A solid grasp of this concept enables students to explore more complex three-dimensional kinematics. This research paper delves into realistic projectile motion through the use of video analysis software and Java-based simulation, offering deeper insights into the dynamics of a basketball in flight.

Under ideal conditions, a projectile launched at a given velocity follows a parabolic trajectory governed by constant gravitational acceleration [6] and zero air resistance. This simplification allows for the derivation of clean mathematical equations to describe the projectile's position [5]. However, this model fails to capture the actual behavior of objects like basketballs traveling through air.

In real scenarios, air drag significantly affects the projectile's motion—especially for objects with relatively low mass and large surface area, such as basketballs. The drag force is proportional to the square of velocity [4] and depends on air density, cross-sectional area, and a drag coefficient determined by the object's shape and texture [3]. This force reduces the height and range of the trajectory, causing deviation from the ideal path.

Understanding these deviations has implications for athletes [1], engineers [2], and physics educators. For instance, optimizing launch angles is valuable for sports like javelin throwing or artillery simulations. Academically, modeling non-ideal projectile motion encourages students to apply Newtonian mechanics to real-world situations, enhancing computational and analytical thinking.

This study assumes :-
Gravity g = 9.8 m/s²

- Air density ρ = 1.225 kg/m³
- Negligible Magnus effect

Our investigation comprises four main stages:

1. Derivation of the ideal trajectory model based on real-world video data
2. Residual analysis comparing the ideal and actual trajectories.
3. Estimation of the drag coefficient using horizontal motion data.
4. A Java-based simulation to identify the optimal launch angle considering air resistance.

Despite observed flaws in our velocity estimation method, the ideal model retains value in verifying experimental data and estimating initial conditions. The study also emphasizes the importance of precision in data collection and encourages further refinements using more accurate tools.

$F_d = 0.5 * C_d * \rho * A * V^2$(Eqn 1)

RESULTS

To derive the ideal trajectory, we first used Logger Pro's internal functions to identify the ball's launch position, extract realistic trajectory equations, and generate a position-time data table (Figure 1), after establishing a 2D coordinate system on the video (Figure 2). The launch position was at (0.2345, 1.966), and a quadratic curve fit of the ball's real path produced the equation for the realistic trajectory (Eqn 2).

$Y(x)_r = -0.2768x\text{^}2 + 1.326x + 1.663$(Eqn 2)

Figure 2: Video analysis set-up. The mutually perpendicular yellow lines represent the 2-D coordinate of the scene, and the green lines represents the calibration for the scene, which represent this amount of length corresponds 1 meter in the real life. The uninterrupted blue dots represent the basketball's track, and the realistic track equations that logger pro created is based on this track

To determine the ideal trajectory (i.e., without drag), we averaged the first three frames of velocity data (Figure 1) to calculate the initial velocity components: vyi = 4.284 m/s, vxi = 3.9016 m/s. With these and assuming gravitational acceleration of 9.8 m/s², we derived a system of parametric equations (Eqn 3), and then eliminated time t to convert the equations into standard y-x form (Eqn 4).

$x(t)_i = 0.2345 + 3.9016t$ (Eqn 3)

$y(t)_i = 1.966 + 4.284t - 4.9t\text{^}2$(Eqn 3)

$y(x)_i = 1.966 + 1.098(x - 0.2345) - 0.3223(x - 0.2345)^2$(Eqn 4)

video time (s)	X (m)	Y (m)	X Velocity (m/s)	Y Velocity (m/s)	t' (s)
2.836166667	0.234452292	1.965907806	3.797782344	4.384439012	0.000166667
2.869533333	0.334643573	2.107208017	3.926390632	4.277890333	0.033533333
2.9029	0.477562799	2.246138363	4.113712763	4.189773212	0.0669
2.936266667	0.623766986	2.379648525	4.109091066	3.837707985	0.100266667
2.969633333	0.75054298	2.506987655	4.101941695	3.493698236	0.133633333
3.003	0.893626454	2.609102984	4.162066731	3.227667416	0.167
3.036366667	1.036522216	2.722434106	4.027029573	2.962105407	0.200366667
3.069733333	1.156282474	2.809860971	4.032967848	2.637082512	0.233733333
3.1031	1.305677764	2.893557061	4.0253692	2.440924094	0.2671
3.136466667	1.431726374	2.971199438	3.858609151	2.277641061	0.300466667
3.169833333	1.558244264	3.048255215	3.821768404	2.044485626	0.333833333
3.2032	1.68473869	3.11163148	3.832961271	1.682582916	0.3672
3.236566667	1.81301638	3.160741633	3.875994232	1.306752608	0.400566667
3.269933333	1.94786399	3.200255009	3.804559126	0.904414932	0.433933333
3.3033	2.068468952	3.215131186	3.695267516	0.675322528	0.4673
3.336666667	2.188768882	3.242630994	3.72304458	0.521044577	0.500666667
3.370033333	2.317116964	3.252133914	3.749864487	0.311232033	0.534033333
3.4034	2.443048254	3.267268194	3.675147702	-0.005078788	0.5674
3.436766667	2.561752632	3.252556266	3.619593576	-0.354675204	0.600766667
3.470133333	2.683835825	3.240096882	3.587499543	-0.603203739	0.634133333
3.5035	2.796134531	3.212831714	3.673311525	-0.849114745	0.6675
3.536866667	2.932483837	3.179043553	3.677355022	-0.994563419	0.700866667
3.570233333	3.048677567	3.149760481	3.514189192	-1.234184541	0.734233333
3.6036	3.156823145	3.10517888	3.596797631	-1.68805238	0.7676
3.636966667	3.287142203	3.030281791	3.712496328	-1.97613686	0.800966667
3.670333333	3.413003101	2.970026238	3.626235068	-2.165380313	0.834333333
3.7037	3.528868334	2.887503349	3.528878609	-2.378962881	0.8677
3.737066667	3.64574252	2.808453131	3.492385564	-2.506780346	0.901066667
3.770433333	3.762217818	2.723888874	3.457277466	-2.728940205	0.934433333
3.8038	3.875830508	2.620905376	3.429471102	-2.923897294	0.9678

Figure 1: Data table. As shown on the table, the first three frames' velocity for y and x are 4.384439012, 4.277890333, 4.189773212 and 3.797782344, 3.926390632, 4.113712763 respectively. In addition, the column t' locate at the right most of the table means "time since released"

A residual analysis was performed by subtracting the ideal model from the realistic trajectory (Eqn 5). The resulting graph (Figure 3) shows that for most values of x, the realistic trajectory lies above the ideal one, with the residual increasing along the x-axis.

$Residual = R = y(x)_r - y(x)_i$(Eqn 5)

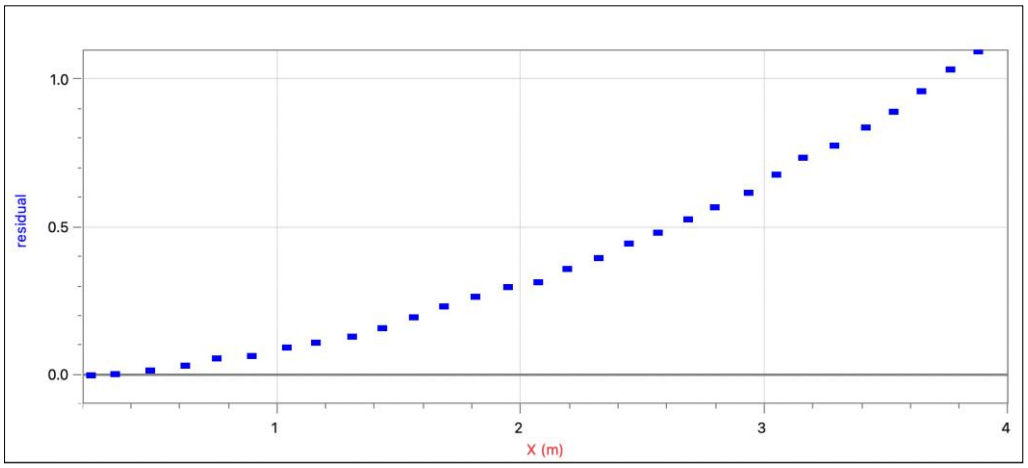

Figure 3: Residual Graph. As shown in the graph, the differences between the height of realistic track and height for ideal track are mostly positive, and the value of the residual increase as the x (displacement from origin) increase. The equation we used to perform the residual analysis is shown at (Eqn 5)

To quantify the drag force, we focused on x-direction motion, where the only force acting is air resistance. Using x-velocity data from Logger Pro, we estimated acceleration by computing the rate of change between consecutive velocity values. Applying Newton's Second Law and rearranging the drag equation, we calculated drag coefficients for each time interval (Eqn 6). Only positive values were used to mitigate the effect of noise. The average drag coefficient obtained was approximately 1.15.

$C_d = \frac{-2*m*a_x}{\rho*A*v^2}$(Eqn 6)

Finally, we developed a Java simulation using Euler's method to evaluate how drag impacts the projectile's range. Assuming a constant speed of 5.78 m/s and launch point (0.2345, 1.966), we tested launch angles from 20° to 60° in 1° increments. The simulation revealed that the maximum horizontal range (~4.476 m) occurred at a launch angle of approximately 33°.

DISCUSSION

The residual analysis (Figure 3) yielded a surprising result: the ideal trajectory lay below the realistic one, contradicting expectations that drag should reduce projectile height. Upon reviewing our methodology, we identified that the initial vertical velocity was underestimated due to missing early-frame data. This highlights the importance of capturing the precise moment of launch in video analysis. Next, in the part of drag coefficient estimation, we also find some model's limitation. Our calculation produced an average drag coefficient about 1.15 which substantially higher than published drag coefficient values for a basketball which about 0.47-0.54 [7,8]. This discrepancy likely results from a combination of factors: Frame-to-frame velocity noise inflated acceleration estimates; air drag was assumed to be the only horizontal force; and data resolution limited the ability to capture smooth velocity decay. Rather than being dismissed as erroneous, this inflated value served a useful role in our Java-based simulation. By assuming a stronger drag force, the model remained conservative, ensuring that any overestimation of range was minimized.

In the last part of our investigation, optimal launch angle simulation, our simulation revealed a launch angle of 33° as the best angle for maximize the horizontal range. This degree is significantly below the ideal 45°, and thus validate that the actual optimal angle to acquire maximize range is deviate from the ideal condition. Although our incorrect established model may cause error in the simulation, but the result still reveals the trend that the optimal angle should be lower than 45º. The detail that worth to noticed is that we have used a fixed value of drag coefficient in our simulation of optimal angle for simplicity, to acquire more accurate angles, future investigations can try to incorporate more dynamic factors, such as velocity-dependent changes in drag or the effect of ball spin.

For these students aiming to replicate or extend this study we suggest they should consider examining the effect of varying ball size, mass, or surface texture, all of the factors which may influence the drag. Furthermore, we sincerely suggest these students that have ability to use other experimental instruments other than logger pro to use other way to analyze this investigation, because it is hard to ensure the video shooting angle is hundred percent horizontal to the plane of projectile, and any deviation of the degree will result errors in the software analysis.

Overall, this investigation ultimately revealed that while simple physics models offer a foundational understanding of projectile motion, they can still fall short when applied to real-world conditions without careful calibration. The ideal model failed to predict the trajectory accurately, as seen in our residual analysis (Figure 3), but this failure provided valuable insight into the limitations of early-frame velocity extraction. The drag coefficient, although higher than expected, helped us construct a more conservative simulation that remained directionally accurate. Most notably, the simulation identified 33° as the optimal launch angle for maximum horizontal range which far from the theoretical 45°, which reveals the simulation clearly shaped by the presence of air resistance. Collectively, these findings show that even with imperfect data and assumptions, a thoughtfully structured approach using video analysis, mathematical modeling, and numerical simulation still can yield meaningful physical conclusions.

MATERIALS AND METHODS

Video Recording and Calibration: The motion of a size 7 basketball (mass: 600g, diameter: 22.5 cm) was recorded at 60 frames per second from a fixed side view. A meter stick placed in the plane of motion provided scale for calibration. Efforts were made to ensure minimal camera tilt.

Position and Velocity Extraction: The ball's center was manually tracked frame by frame. Logger Pro generated x- and y-position versus time data (Figure 4), which were used to derive velocity components. Only data captured after complete release from the hand were used.

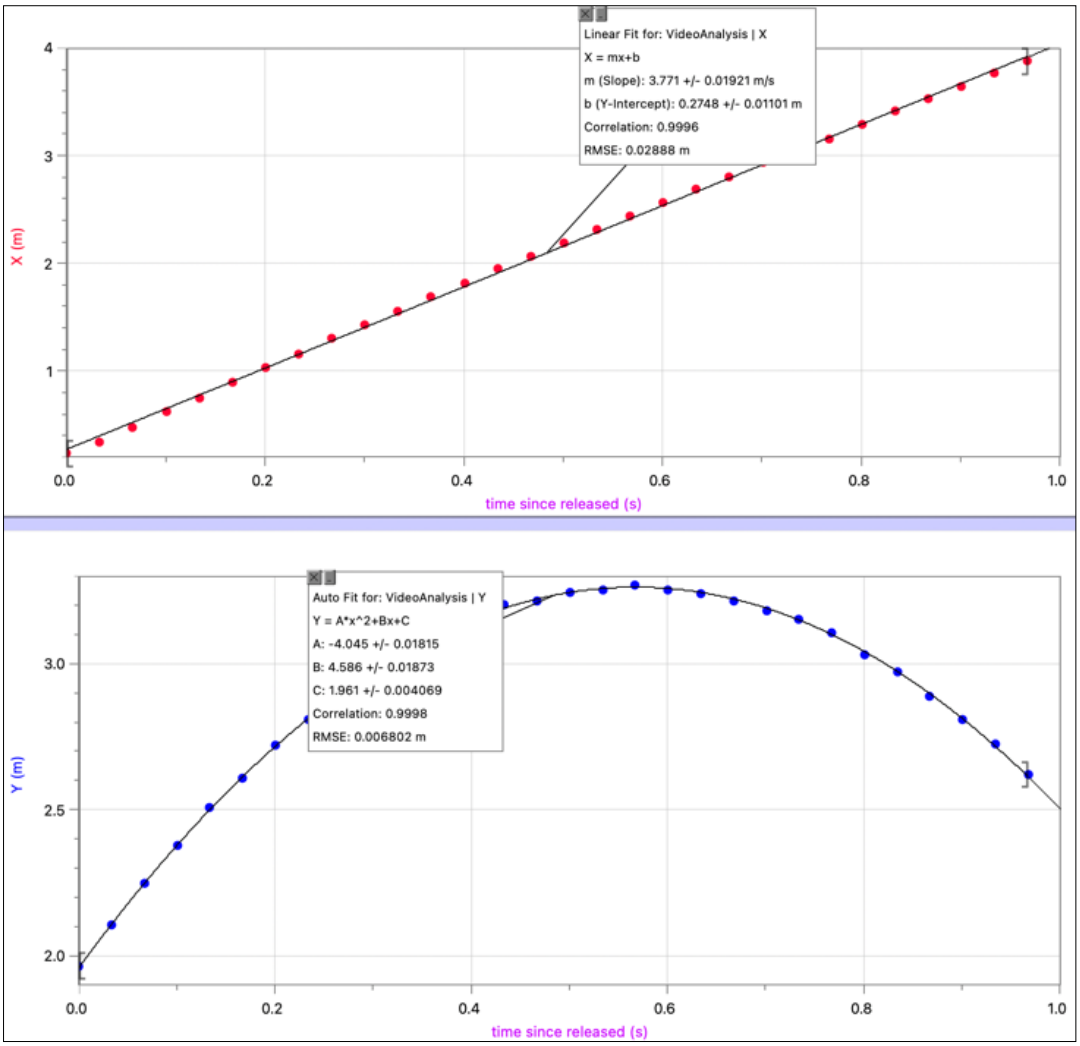

Figure 4: x and y position-time graph generated by logger pro. The two graphs show above correspond to the x-t and y-t graph respectively, and the best fit-line information could automatically create by the logger pro since you choose the type of function that you like to fit

Ideal Trajectory Derivation:

The ideal model assumed constant gravity (9.8 m/s²) and no drag. Initial velocity components were averaged over the first three frames. The parametric equations (Eqn 3) were converted to y(x) form (Eqn 4) by eliminating time.:

$x(t)_i = 0.2345 + 3.9016t$ (Eqn 3)

$y(t)_i = 1.966 + 4.284t - 4.9t\text{^}2$(Eqn 3)

$t = \frac{x-0.2345}{3.9016}$ (derive from x(t))

$y(x)_i = 1.966 + 4.284(\frac{x-0.2345}{3.9016}) - 4.9\left(\frac{x-0.2345}{3.9016}\right)^2$ (plug t in to y(t) equation)

$y(x)_i = 1.966 + 1.098(x - 0.2345) - 0.3223(x - 0.2345)^2$(simplify, Eqn 4)

Residual Analysis:

A second-degree polynomial fit the realistic y(x) data. Residuals were computed as the difference between realistic and ideal y-values across x-values (Eqn 5).

Drag Coefficient estimation:

Acceleration was calculated from changes in consecutive x-velocities. Newton's Second Law in the x-direction was applied to estimate the drag coefficient (Eqn 6):

$$F_{xTotal} = -F_d = -0.5 * C_d * \rho * A * V^2 = ma_x$$

$$C_d = \frac{-2*m*a_x}{\rho*A*v^2} \text{(Eqn 6)}$$

Simulation-Based Optimal Angle Model:

Using a Java program, we simulated trajectories for angles between 20° and 60°. Constants used were: mass (0.6 kg), air density (1.225 kg/m³), cross-sectional area (0.0398 m²), and drag coefficient (1.15). Each simulation ran until the ball returned to the ground. Maximum range was identified.

REFERENCES

1. Sports, Just Fly. "Optimal Javelin Flight: Physics and Fixes." *Just Fly Sports LLC*, 12 Aug. 2024, www.just-fly-sports.com/optimal-javelin-flight-physics-and-fixes.
2. Wonder Why a 45 Degree Angle Gives Longest Projectile..." *Trad Talk Forums*, 23 Oct. 2010, www.tradtalk.com/threads/wonder-why-a-45-degree-angle-gives-longest-projectile-flight.24158.
3. Editor Engineering toolbox. *Drag Coefficient*. 13 May 2025, www.engineeringtoolbox.com/drag-coefficient-d_627.html.
4. *The Drag Equation*. www.grc.nasa.gov/www/k-12/VirtualAero/BottleRocket/airplane/drageq.html.
5. Libretexts. "10.7: Parametric Equations- Graphs." *Mathematics LibreTexts*, 27 Apr. 2025, math.libretexts.org/Bookshelves/Algebra/Algebra_and_Trigonometry_1e_(OpenStax)/10%3A_Further_Applications_of_Trigonometry/10.07%3A_Parametric_Equations-_Graphs.
6. *The Value of G*. www.physicsclassroom.com/class/circles/lesson-3/the-value-of-g.
7. "Drag Coefficient." *Wikipedia*, 9 June 2025, en.wikipedia.org/wiki/Drag_coefficient?utm.
8. Hubbard, Mont, and Hiroki Okubo. "Identification of Basketball Parameters for a Simulation Model." *ResearchGate*, June 4AD, www.researchgate.net/publication/245481335_Identification_of_basketball_parameters_for_a_simulation_model. Accessed 24 June 2025.
9. Libretexts. "3.1: Euler's Method." *Mathematics LibreTexts*, 27 July 2020, math.libretexts.org/Courses/Monroe_Community_College/MTH_225_Differential_Equations/03%3A_Numerical_Methods/3.01%3A_Euler's_Method.

Appendix:

```java
package src;

public class Optimal_Angle {


    public static final double GRAVITY = 9.8;                    // gravitational acceleration (m/s^2)
    public static final double MASS = 0.6;                       // mass of basketball (kg)
    public static final double DIAMETER = 0.225;                 // diameter (m)
    public static final double RADIUS = DIAMETER / 2;
    public static final double AREA = Math.PI * RADIUS * RADIUS;  // cross-sectional area (m^2)
    public static final double AIR_DENSITY = 1.225;              // air density (kg/m^3)
    public static final double DRAG_COEFFICIENT = 1.15;          // drag coefficient from data

    //Initial velocity derived from first three frames
    public static final double INITIAL_VX = 3.9016;
    public static final double INITIAL_VY = 4.284;


    public static final double INITIAL_SPEED = Math.sqrt(INITIAL_VX * INITIAL_VX + INITIAL_VY * INITIAL_VY);

    // Initial position
    public static final double INITIAL_X = 0.2345;
    public static final double INITIAL_Y = 1.966;

    // Time step for simulation
    public static final double TIME_STEP = 0.01;

    public static void main(String[] args) {
        double maximumRange = 0.0;
        double optimalAngle = 0.0;

        System.out.println("Angle (degrees)\t    Range (meters)");

        // Try angles from 20 to 60 degrees
        for (double angleDegrees = 20.0; angleDegrees <= 60.0; angleDegrees += 1.0) {

```

```
        // Try angles from 20 to 60 degrees
        for (double angleDegrees = 20.0; angleDegrees <= 60.0; angleDegrees += 1.0) {

            // Convert angle to radians
            double angleRadians = Math.toRadians(angleDegrees);

            // Set initial conditions for this trial
            double x = INITIAL_X;
            double y = INITIAL_Y;

            double velocityX = INITIAL_SPEED * Math.cos(angleRadians);
            double velocityY = INITIAL_SPEED * Math.sin(angleRadians);

            // Simulate flight until the ball hits the ground (y <= 0)
            while (y > 0) {
                double speed = Math.sqrt(velocityX * velocityX + velocityY * velocityY);
                double dragForce = 0.5 * DRAG_COEFFICIENT * AIR_DENSITY * AREA * speed * speed;

                // Resolve drag force into components
                double accelerationX = -dragForce * velocityX / (speed * MASS);
                double accelerationY = -GRAVITY - (dragForce * velocityY / (speed * MASS));

                // Update velocity using acceleration
                velocityX = velocityX + accelerationX * TIME_STEP;
                velocityY = velocityY + accelerationY * TIME_STEP;

                // Update position using velocity
                x = x + velocityX * TIME_STEP;
                y = y + velocityY * TIME_STEP;
            }

                // Update position using velocity
                x = x + velocityX * TIME_STEP;
                y = y + velocityY * TIME_STEP;
            }

            double range = x - INITIAL_X;
            System.out.printf("%.1f\t\t\t%.3f\n", angleDegrees, range);

            // Check if this is the best range so far
            if (range > maximumRange) {
                maximumRange = range;
                optimalAngle = angleDegrees;
            }
        }

        // Output the optimal angle and range
        System.out.println("\nBest angle for maximum range: " + optimalAngle + " degrees");
        System.out.printf("Maximum range achieved: %.3f meters\n", maximumRange);
    }
}
```

Figure 1: code content