Contents

The Best Angle of 3-point shoots Based on Stephen Curry

Kelvin, Chan Ka Wai

May 20, 2024

Abstract

We will analyze the shooting angle, height, and ball parabola of Stephen Curry, the most representative player in the NBA, and provide a comparison with the optimal shooting angle as the model prediction. We will also learn how to apply differential equation to model an intriguing real-world problem, from problem identification to interpretation and verification.

1 Introduction

1.1 Background

The NBA, short for the **National Basketball Association**, is a professional men's basketball league based in North America. It comprises 30 teams, with 29 located in the United States and one in Canada.

In the NBA, there exists a group of extraordinary scorers whose points per game(PPG) significantly surpass the league's average. This implies that their shooting techniques are more precise than other players, and they play crucial roles in their teams. For example, Curry, Bird, Lillard, Anthony etc. Among them, Stephen Curry stands out as a representation. His shooting form is not only aesthetically pleasing but also deadly effective, particularly his free throws and three-pointers, which are among the most lethal weapons in the league. His shooting technique has garnered the attention of the majority of basketball fans and has been the subject of in-depth analysis.

Figure 1: Curry

Figure 2: Lillard

Figure 3: Bird

Figure 4: Anthony

1.2 Model Background

1.2.1 Importance of Solving

[7]

1. Efficiency Improvement: The study hypothesizes that Curry's already efficient shooting form can be further optimized to enhance his shooting consistency
2. Kinematic Analysis: It applies principles of kinematics, inverse kinematics, and non-linear dynamics to model Curry's shot
3. Energy Reduction: The proposed optimization method suggests adjustments that could reduce Curry's energy expenditure by 23%

1.2.2 Applications

[1]

- Defensive Blocking: By calculating the angle of the ball and the trajectory of the parabola, it is possible to figure out how high a defender would have to jump to successfully complete a block
- Hawk-eye Technology: It's a sophisticated system used in sports to track the trajectory of balls and is mentioned in the context of basketball performance analysis

You may see the images in A.1 as references.

1.3 Mathematical Model

1.3.1 Problem Defining

When we watching the players shoot in game, we know that sometimes they make small error but still make the basket. It seems reasonable for them to shoot go in depends on the initial angle that the ball was thrown. We'll find the best shooting angle for Curry in this model and later we will define some parameters as following. [6]

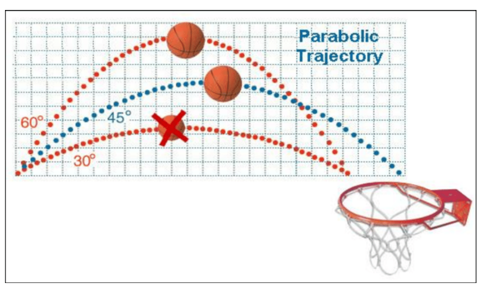


Figure 5: Trajectory of ball shooting

1.3.2 Assumptions

1. Only allowed trajectories are those that enter the net directly, or strike the back of the rim before going straight in
2. Ignore air resistance and any spin the ball may have
3. No sideways error in the trajectory
4. No error in the initial shooting velocity
5. The best shot is one that goes through the center of the hoop
6. Rim height = 10 feet

1.3.3 Constraints

The model uses fixed values for the rim diameter, ball diameter, horizontal and vertical distances traversed by the ball, and the acceleration due to gravity.We thus take the horizontal distance traversed, l, to be 23'9" (23 feet 9 inches). The model focuses on the release angle as the primary variable, with the assumption that the shooter is 6'22" tall. It has also been observed [3] that the ball will be released from a height of approximately 1.25 times the shooter's own height. For Curry, he will shoot at a height of approximately 7'8.5", we would take the net vertical distance traversed(Rim height - 7'8.5"), h, to be 2'3.6". And the restriction of the initial angles :

$$0 < \theta_0 < 90^\circ$$

1.3.4 Terms of Using

Physical constant	Symbol	Value(feet,inches)
Rim diameter	D_r	1.5 ft = 0.46m
Ball diameter	D_b	0.8 f = 0.24m
Horizontal distance traversed	l	23 ft 9 in = 7.24m
Vertical distance traversed	h	2 ft 3.6 in =0.7m
Acceleration due to gravity	g	-32ft/s^2 = -9.8m/s^2

Table 1: Terms of Using

2 Model Solution

2.1 Data Collection

We will show you the field-gold percentage of Curry's 3-point shot in 2022-2023 regular season, trajectory of the ball as well as the shooting angle.

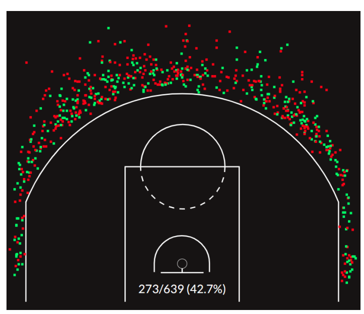


Figure 6: Regular Season 3-Point Shoots in 2022-2023

According to Stats LLC[5], Stephen Curry's average 3-point shot reaches a maximum height of **16.23 feet**, which is higher than the NBA average of 15.77 feet. This added height gives Curry an advantageous angle at the basket. The higher the arc, the better angle the ball has to travel through the rim, effectively making the rim "bigger." Curry typically shoots the ball between **50-55 degrees**.

2.2 Methodology

[4] [1]

2.2.1 Approaches

We'll do this by taking standard projectile motion equations that are derived from Newton's second law of motion. We start by resolving the initial velocity v_0 into horizontal and vertical line:

$$v_H = v_0 cos(\theta_0)$$

$$u_V = v_0 sin(\theta_0)$$

and then we find the horizontal distance($x(t)$ = vt),we have:

$$l = x(t) = v_0 cos(\theta_0) t$$

and the vertical distance of motion is given by:

$$h = y(t) = vt + \frac{1}{2} g t^2 = v_0 sin(\theta_0) + \frac{1}{2} g t^2$$

, where h is the vertical distance to the center of basket.

we now need to find the t :

$$t = \frac{l}{v_0 cos(\theta_0)}$$

after substitution, we find the initial velocity v_0 that we needed,so that the ball goes though the middle of the hoop:

$$v_0 = \frac{l}{cos(\theta_0)} \sqrt{\frac{-g}{2(l tan(\theta_0) - h)}}$$

Note:

$$tan^{-1}(\frac{h}{l}) < \theta_0 < 90°$$

2.2.2 Error Analysis I

We assume the ball will not hit the front of the rim on trajectories where the ball passes through the center of the hoop(may not be true). We now derive the equations for the amount of error that can be made in the initial angle θ_0 and still have the ball go directly into basket. we fix the initial velocity and allow the initial release angle to vary, after some substitution, we can obtain the new horizontal position as it come back down to the basket height:

$$x = \frac{v_0 cos(\theta_0^{oops})}{-g} (v_0 sin(\theta_0^{oops}) + \sqrt{v_0{}^2 sin^2(\theta_0^{oops}) + 2gh})$$

, where θ_0^{oops} corresponds to a larger/smaller release angle(consider the player error) than the ideal initial angle θ_0 where the ball passes through the center of the hoop. You may refer to the image below.

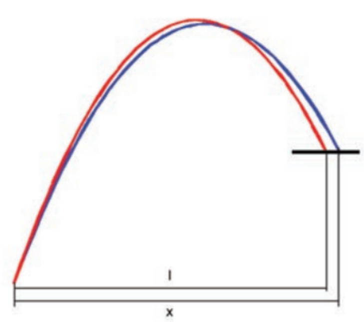


Figure 7: The Ideal Trajectory Through The Middle of The Hoop(v_0,θ_0)(Red) and with An Error in The Release Angle(v_0,θ_0^{opps}) (Blue)

2.2.3 Solving the Equations

After that, we have 2 criteria for the ball still goes in the net(refer to the fig.8):

1. The basketball does not touch the front of rim, so the distance s between the rim and the center of ball must greater than the radius of the ball which through the trajectory(square both side for convenience):
$$s^2 = (x(t) - (l - D_r/2)^2 + ((y(t) - h)^2 > (D_b/2)^2)$$
$$,where \;\; \frac{l - D_r}{v_0 cos(\theta_0^{oops})} \leq t \leq \frac{1}{-g(v_0 sin(\theta_0^{oops}) + \sqrt{v_0{}^2 sin^2(\theta_0^{oops}) + 2gh)}}$$

2. The basketball hit the back of the rim as the center of the ball passes through the basket, obviously we can obtain the relation:
$$x + D_b/2 = l + D_r/2$$

The front

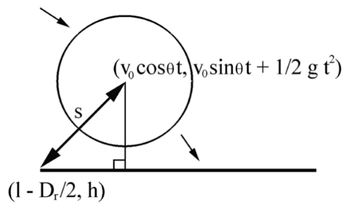


Figure 8: Graphically Analyze 2 Criteria

Next, we keep the initial velocity unchanged and solve numerically for the unique release angles
$$\theta_{low} < \theta_0 \;\; \& \;\; \theta_{high} > \theta_0$$

Thus,we have:
$$s^2 - (D_b/2)^2 = 0 \;\; and \;\; x - l + \frac{D_b - D_r}{2} = 0$$

For solving for θ_{low} and θ_{high} we find the minimum deviation from θ_0, we obtain:
$$e(\theta_0) = min(\theta_{high} - \theta_0, \theta_0 - \theta_{low})$$

The solution has been shown by the graph.The best angle and you can refer the completed code in A.2 if you like. We first show you the error without discretization.

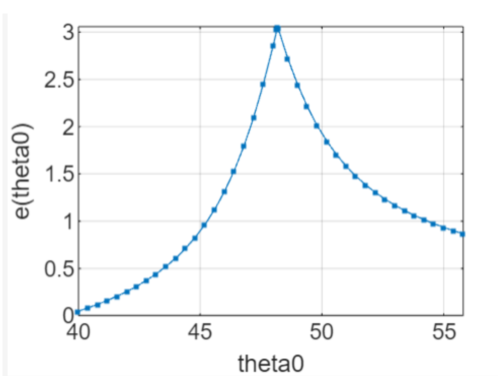
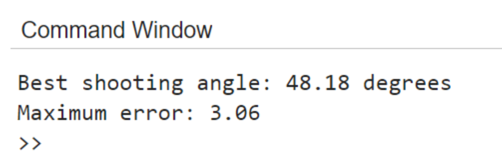


Figure 9: The Error About θ_0 For Which The Basketball Still Goes In (Without Discretization



Command Window

```
Best shooting angle: 48.18 degrees
Maximum error: 3.06
>>
```

Figure 10: Result Without Discretization

2.2.4 Error Analysis II with Discretization

Now, We make some changes in the "function of frontfzero" and we verify that is a little bit differences compared with the previous results.

The purpose of discretization and Euler's method is to calculate the position of the ball and determine if it hits the front rim. By discretizing the time interval and approximating the ball's motion at each time step, the function iterates through the time steps to determine if the ball reaches the front rim ($y_{temp} \leq D_b$). If it does, the function returns the position at that time. Otherwise, it keeps track of the minimum distance (y) from the rim.

We use 2 screenshots of code to show the differences:

```
global Dr Db l h g
sintheta = sin(x*pi/180);
% Calculate the relevant time interval.
timeinterval = [(l-Dr/2)/(v*cos(x*pi/180)),-1/g*(v*sintheta+...
                sqrt(v*v*sintheta*sintheta+2*g*h))];
% If this time interval is not a interval, return a negative
% value. This signals the calling program that the ball either
% goes through the front rim, or never even reaches the front
% rim.
if (timeinterval(1) > timeinterval(2))
    y = -1;
    return;
end

% Calculate the minium distance from the front rim in the
% relevant time interval.
[~, y] = fminbnd(@distancefromrim, timeinterval(1), ...
                 timeinterval(2), [], v, x);
end
```

Figure 11: Without Discretization

```
    % Discretize the time interval
    num_steps = 100; % Number of time steps
    dt = diff(timeinterval) / num_steps; % Time step size

    % Initialize variables for position and velocity
    y = inf; % Initialize minimum distance to infinity
    vy = v * sintheta;
    y_temp = 0; % Temporary variable to store position

    % Perform Euler's method to calculate the position at each time step
    for i = 1:num_steps
        % Update position and velocity
        y_temp = y_temp + vy * dt;
        vy = vy - g * dt;

        % Check if the ball has hit the front rim
        if y_temp <= Db
            y = y_temp;
            return;
        else
            y = min(y, y_temp); % Update the minimum distance
        end
    end
end
```

Figure 12: Discretization

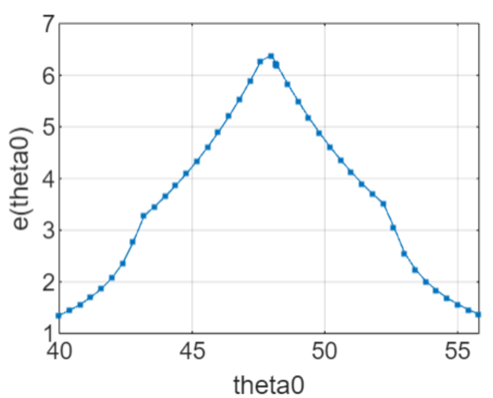


Figure 13: The Error About θ_0 For Which The Basketball Still Goes In (Discretization)

```
Command Window

Best shooting angle: 48.00 degrees
Maximum error: 6.38
>>
```

Figure 14: Result with Discretiztion

We first can easily to know that the difference of 2 predictions = 48.18 - 48 = 0.18 degrees which is a really small difference. And second our predictions close to the Curry's typical shoots(50-55°) in real games.

2.2.5 Error Analysis III with Stability

To analyze the error introduced by Euler's method by the given equations above, we need to consider the Lipschitz condition for the function f(t, y) in the context of the problem.

1. Local truncation error:Using the Lipschitz condition, we can write:

$$|f(t, y) - f(t, z)| \leq L\,|y - z|$$

$$|-g - (-g)| \leq L\,|y - z|$$
$$0 \leq L\,|y - z|$$
$$y_{n+1} = y_n + dt(t_n, y_n), y(t_{n+1}) = y_n + dt\dot{(} - g)$$
$$|y(t_{n+1}) - y_{n+1}| = \left|-g\dot{d}t\right| = g\dot{d}t$$

which is bounded by $g \cdot dt$ is bounded by $g \cdot dt$

2. Global error: The global error is the accumulation of local truncation errors over all time steps. Since the local truncation error is bounded by $g \cdot dt$ at each step, the global error will depend on the number of time steps (num_steps) and the time step size (dt). The total time interval is divided into num_steps segments of size dt. Therefore, the total time (T) is given by $T = num_steps \cdot dt$. The global error (GE) can be estimated as:

$$GlobalError = num\_step\dot{s}(g\dot{d}t) = g\dot{T}$$

where the local truncation error is bounded by $g \cdot dt$ at each step, the global error will depend on the number of time steps (num_steps) and the time step size (dt). The total time interval is divided into num_steps segments of size dt. Therefore, the total time (T) is given by $T = num\_steps \cdot dt$.

We also conclude [2]that Eular's method is convergent and thus stable.

3 Discussion and Interpretation of Model Solution

3.1 Strengths

Firstly, this model can be used by everyone, you just need to change the parameter depends on your self. For example, the 3-point line distance between NBA and WNBA are difference, the height of basket may difference. Secondly, the model has more than 1 analysis about how the ball can be shoot in the net. You can back to 2.2.3 again to feel this interesting model. Thirdly, I think this title is more attractive especially for who enthusiastic about playing basketball and even a fan of Stephen Curry.

3.2 Limitations

Obviously, it has lots of limitations. The model does not consider the air resistance, that's a big factor to affect the angle and trajectory of shooting. Next, we do not consider about the optimal trajectory of the ball. What a pity. Moreover, we also do not consider the sideways error in the trajectory and the field-goal percentage for Curry.

3.3 Improvement

We can define two parts about finding the best trajectory and adding the air resistance. Also we can do the classification for each entity and implemented new and more difficult algorithms to do such an analysis. Such as finding the recall, precision, f-1 score and accuracy in a vector. These can be used to verify the accuracy rate of the model.

4 Conclusion

From my first point of view, this is a good opportunity to demonstrate what I have learnt in this semester between mathematical modelling and Natural Language Processing course.

By applying Euler's method, we were able to approximate the ball's trajectory and determine the minimum distance from the front rim for various shooting angles. The Lipschitz condition allowed us to analyze the error introduced by Euler's method, considering the local truncation error (LTE) and the global error (GE). Some of them are new knowledge for me and you need to try to understand and use them in the model.I am so happy that I can handle with different problems and have a great improvement for studying the differential equations. At the end, thanks for the Professor Hu teach this course and look forward to meeting in the future.

A Appendix

A.1 First Appendix:

Figure 15: Defensive Blocking

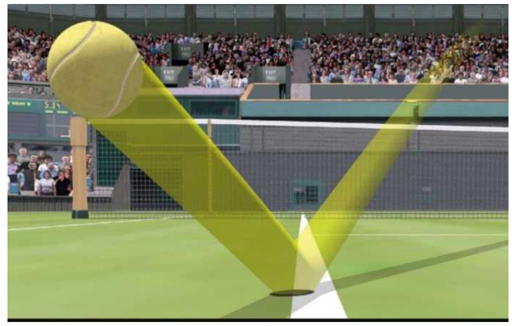


Figure 16: Hawk-eye Technology

A.2 Second Appendix:

```
%% Sample Matlab code
Dr = 0.46;                % Rim diameter in meters (1.5 ft)
Db = 0.24;               % Ball diameter in meters (0.8 ft)
l  = 7.24;                % Horizontal distance in meters
                            % (23 ft., 9 in)
playerh = 1.88;              % Height of player (in meters)
releaseh = 1.25*playerh; % Vertical position of center of
                            % ball at release
basketh  = 3.05;           % Vertical position of the ring
                            % (in meters), 10ft.
h  = basketh - releaseh; % Vertical distance traversed in
                            % meters
h  = 0.7;               % Vertical distance traversed in
                            % meters (2 ft 3.6 in)
g  = -9.8;                 % Gravity constant in meters per
                            % second squared (-32 ft (s)^(-2))
convfactor = 1/0.3048;     % Factor by which measurements in
                            % m have to be multiplied to get feet.

function [v0, T0] = calcOptv(theta0)
    global l h g
    % Function to calculate the optimal velocity to hit the center
    % of the basket, given an initial angle of theta0
    v0 = l / cos(theta0 * pi / 180) * sqrt(-g / (2 * (l * tan(theta0 *
        pi / 180) - h)));
    T0 = l / (cos(theta0 * pi / 180) * v0);
end


function [x,y] = position(v,theta,t);
% Position of the ball at time t, with initial velocity v,
% and angle theta.
global g
x = v*cos(theta*pi/180)*t;
```

```
y = v*sin(theta*pi/180)*t + .5*g*t.*t;
end


function s = distancefromrim(t,v0,theta)
% Calculate the distance from the front of the rim for the ball
% at time t, with initial velocity v, and angle theta. Use this
% distance to calculate how much space is left between the ball
% and the rim. If this function is negative, the ball hits the
% rim. If it is zero, the ball scims the rim for this velocity
% and this release angle at time t.

global Dr Db l h
%global v pos

[x,y] = position(v0,theta,t);
% Calculate the position of the center of the ball at time t,
% with initial velocity v and release angle theta.

part1 = x -(l - Dr/2);
part2 = y - h;

% Calculate the distance from the front of the rim.
s = sqrt(part1.*part1 + part2.*part2)-Db/2;
end


function dist = distancefromback(v,theta)
% Calculate the distance from the back of the rim for the ball
% when its center is level with the rim. The ball is assumed to
% have initial velocity v, and initial angle theta.
global Dr Db l h g

sintheta = sin(theta*pi/180);
sqrtval  = v*v*sintheta*sintheta+2*g*h;

if (sqrtval < 0)
    dist = -inf;
    return
end
x = v * cos(theta*pi/180)/(-g);
x = x*(v*sintheta + sqrt(sqrtval));
dist = x-l+(Db-Dr)/2;
end


function y = frontfzero(x, v)
    global Dr Db l h g

    sintheta = sin(x * pi / 180);
    % Calculate the relevant time interval.
    timeinterval = [(l - Dr / 2) / (v * cos(x * pi / 180)), -1 / g * (
       v * sintheta + sqrt(v * v * sintheta * sintheta + 2 * g * h))];
    % If this time interval is not a valid interval, return a negative
        value.
```

```
    % This signals that the ball either goes through the front rim or never reaches it.
    if (timeinterval(1) > timeinterval(2))
        y = -1;
        return;
    end

    % Discretize the time interval
    num_steps = 100; % Number of time steps
    dt = diff(timeinterval) / num_steps; % Time step size

    % Initialize variables for position and velocity
    y = inf; % Initialize minimum distance to infinity
    vy = v * sintheta;
    y_temp = 0; % Temporary variable to store position

    % Perform Euler's method to calculate the position at each time step
    for i = 1:num_steps
        % Update position and velocity
        y_temp = y_temp + vy * dt;
        vy = vy - g * dt;

        % Check if the ball has hit the front rim
        if y_temp <= Db
            y = y_temp;
            return;
        else
            y = min(y, y_temp); % Update the minimum distance
        end
    end
end


function error = calcfronterror(theta0, v0);
% Routine to calculate the maximum allowed error in the angle
% with respect to the front of the rim when the ball is thrown
% with given velocity.

% Calculate the angle when the ball just skims the front rim,
% and still goes in.
ang  = fzero(@(x) frontfzero(x,v0), theta0);
ang2 = fzero(@(x) frontfzero(x,v0), theta0+1);
% Calculate the error allowed.
error = min(abs(theta0-ang),abs(theta0-ang2));
end


function [error, ang] = calcbackerror(theta0, v0);
% Routine to calculate the maximum allowed error in the angle
% with respect to the back of the rim when the ball is thrown
% with given velocity.
% Joerg Gablonsky, 06/07/2005
global Dr Db l h g
% Find the maximum distance from the back the ball thrown
% with this velocity, and varying angles can have. Note that
```

```
% we have to multiply the distance with (-1) to use the Matlab
% fminsearch function to find a maximum.
[ang, val] = fminsearch(@(x) -distancefromback(v0,x), theta0);

% Negate the value to reverse the multiplication with (-1) that
% was necessary to do a maximization.
val = -val
% If this maximum is small enough, the ball cannot reach the
% back of the rim. Therefore the error can be infinite.
if (val < 0)
    error = NaN;
else
    if (val > Dr-Db/2)
        error = NaN;
    else
        % The ball can reach the back of the rim. Find the
        % angle when the ball just skims the back of the rim
        % by minimizing the negative distance from the rim.
        ang = fzero(@(x) -distancefromback(v0,x), theta0);
        % Calculate the error in angle allowed.
        error = abs(theta0-ang);
    end
end
end


function error = calcerror(y)
% Set all variables.
theta0 = y(1);
v0     = y(2);

% Check if the angle is in the valid range, that is, between
% 10 and 85 degrees.
if ((theta0 < 10) || (theta0 > 85))
    sprintf(...
    'Angle theta0 = %5.2f is either too small or too large.'
    ,theta0)
    error = NaN;  % The ball is too far from the back of the
                  % rim to still be in the rim.
    return
end

% Calculate this value since it is used several times below.
sintheta = sin(theta0*pi/180);
global g h l Dr Db
% Calculate the horizontal position of the ball as it comes
% back down to the basket height.
x = v0 * cos(theta0*pi/180)/(-g);
x = x*(v0*sintheta + sqrt(v0*v0*sintheta*sintheta+2*g*h));

if (x < l - Dr/2+Db/2)
    sprintf('The ball does not reach the basket.')
    error = NaN;  % The ball is too far from the back of the
                  % rim to still be in the rim.
    return
end
```

```
if (x > l + Dr/2 - Db/2)
    sprintf('The ball goes too far.')
    error = NaN;  % The ball is too close to the back of the
                  % rim or behind the back of the rim.
    return
end

% Check to see if the ball hits the front of the rim. This is
% done by calculating the minimum distance the ball has from
% the front of the rim.
% If this distance is below 0, the ball hits the front rim.
interm = frontfzero(theta0, v0);
if (interm < 0)  % The ball hits the front rim.
    sprintf('The ball hits the front of the rim.')
    error = NaN;
    return
end
fronterror = calcfronterror(theta0, v0);
backerror  = calcbackerror(theta0, v0);
error = -min(fronterror, backerror);
end


function error = calcerrorvopt(theta0)
    % First calculate the optimal velocity for this angle.
    v0 = calcOptv(theta0);
    % Use the general routine to calculate the maximum allowable
    % error given an initial velocity and angle.
    error = -calcerror([theta0, v0]);
end

thetas = [40:0.4:48, 48.15:0.001:48.2, 48.6:0.4:56];
errors = thetas;
for i = 1:size(thetas, 2)
    errors(i) = calcerrorvopt(thetas(i));
end


plot(thetas, errors, '.-')
grid on
ylabel('e(theta0)');
xlabel('theta0');

[maxError, maxIndex] = max(errors);
bestTheta_deg = thetas(maxIndex);

fprintf('Best shooting angle: %.2f degrees\n', bestTheta_deg);
fprintf('Maximum error: %.2f\n', maxError);
```

References

[1] S. Angeles, "Analyzing Basketball Free Throw Trajectory Using Vector Kinematics — IB Mathematics Applications Interpretations HL Sample Internal Assessment," 09 2022. [Online]. Available: https://www.researchgate.net/

publication/363413269_Analyzing_Basketball_Free_Throw_Trajectory_Using_Vector_Kinematics_IB_Mathematics_Applications_Interpretations_HL_Sample_Internal_Assessment

[2] R. L. Burden and J. D. Faires, *Numerical analysis*. Cengage Learning, 8 2010.

[3] M. R. Eddings, "Effect of Manipulating Angle of Projection on Height of Release and Accuracy in the Basketball Free Throw: A Biomechanical Study."

[4] J. M. Gablonsky and A. S. I. D. Lang, "Modeling basketball free throws," *SIAM review*, vol. 47, no. 4, pp. 775–798, 1 2005. [Online]. Available: https://doi.org/10.1137/s0036144598339555

[5] D. Johnson, "Quick and Deadly: Analyzing Stephen Curry's shot," 1 2015. [Online]. Available: https://bluemanhoop.com/2015/01/20/quick-and-deadly-analyzing-stephen-currys-shot/

[6] P. Mačura, "Physical Factors in Basketball Shooting," Tech. Rep., 2013. [Online]. Available: https://fsport.uniba.sk/fileadmin/ftvs/kniznica/Basketbal/PHYSICAL_FACTORS_IN_BASKETBALL_SHOOTING_Peter_Macura

[7] R. Raj-Prasad, J. Joseph, B. Jin, and Y. S. Chen, "Dynamic Analysis and Optimization of Steph Curry's 3-point shot," Tech. Rep., 1 2016. [Online]. Available: https://search.datacite.org/works/10.13140/RG.2.2.35452.13445