



UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY

Escuela de Ciencias Matemáticas y Computacionales

TÍTULO: The Math Behind Basketball Free Throws

Trabajo de integración curricular presentado como requisito para
la obtención
del título de Matemático

Autor:

Chamorro Cupuerán Kevin Andrés

Tutores:

Ph.D López Ríos Juan Carlos
Ph.D Infante Quirpa Saba Rafael

Urcuquí, septiembre 2019

Urcuquí, 2 de septiembre de 2019

SECRETARÍA GENERAL
(Vicerrectorado Académico/Cancillería)
ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES
CARRERA DE MATEMÁTICA
ACTA DE DEFENSA No. UITEY-ITE-2019-00010-AD

En la ciudad de San Miguel de Urcuquí, Provincia de Imbabura, a los 2 días del mes de septiembre de 2019, a las 09:30 horas, en el Aula AI-101 de la Universidad de Investigación de Tecnología Experimental Yachay y ante el Tribunal Calificador, integrado por los docentes:

Presidente Tribunal de Defensa	Dr. BORGES VILORIA, NERIO ALBERTO , Ph.D.
Miembro No Tutor	Dr. AMARO MARTIN, ISIDRO RAFAEL , Ph.D.
Tutor	Dr. LOPEZ RIOS, JUAN CARLOS , Ph.D.

Se presenta el(la) señor(ita) estudiante **CHAMORRO CUPUERAN, KEVIN ANDRES**, con cédula de identidad No. **1003586623**, de la **ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES**, de la Carrera de **MATEMÁTICA**, aprobada por el Consejo de Educación Superior (CES), mediante Resolución RPC-SO-15-No.174-2015, con el objeto de rendir la sustentación de su trabajo de titulación denominado: **The Math behind Basketball Free Throws**, previa a la obtención del título de **MATEMÁTICO/A**.

El citado trabajo de titulación, fue debidamente aprobado por el(los) docente(s):

Tutor _____ Dr. LOPEZ RIOS, JUAN CARLOS , Ph.D.

Y recibió las observaciones de los otros miembros del Tribunal Calificador, las mismas que han sido incorporadas por el(la) estudiante.


Previamente cumplidos los requisitos legales y reglamentarios, el trabajo de titulación fue sustentado por el(la) estudiante y examinado por los miembros del Tribunal Calificador. Escuchada la sustentación del trabajo de titulación, que integró la exposición de el(la) estudiante sobre el contenido de la misma y las preguntas formuladas por los miembros del Tribunal, se califica la sustentación del trabajo de titulación con las siguientes calificaciones:


Tipo	Docente	Calificación
Miembro Tribunal De Defensa	Dr. AMARO MARTIN, ISIDRO RAFAEL , Ph.D.	10,0
Presidente Tribunal De Defensa	Dr. BORGES VILORIA, NERIO ALBERTO , Ph.D.	10,0
Tutor	Dr. LOPEZ RIOS, JUAN CARLOS , Ph.D.	10,0

Lo que da un promedio de: **10 (Diez punto Cero)**, sobre 10 (diez), equivalente a: **APROBADO**

Para constancia de lo actuado, firman los miembros del Tribunal Calificador, el/la estudiante y el/la secretario ad-hoc.


CHAMORRO CUPUERAN, KEVIN ANDRES
Estudiante


Dr. BORGES VILORIA, NERIO ALBERTO , Ph.D.
Presidente Tribunal de Defensa


Dr. LOPEZ RIOS, JUAN CARLOS , Ph.D.
Tutor

Rafael Amaro

Dr. AMARO MARTIN, ISIDRO RAFAEL , Ph.D.
Miembro No Tutor

Tatiana Beatriz

TORRES MONTALVÁN, TATIANA BEATRIZ
Secretario Ad-hoc

AUTORÍA

Yo, **KEVIN ANDRÉS CHAMORRO CUPUERÁN**, con cédula de identidad 1003586623, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así como, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad de el/la autora(a) del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, septiembre 2019.



Kevin Andrés Chamorro Cupuerán
CI: 1003586623

AUTORIZACIÓN DE PUBLICACIÓN

Yo, **KEVIN ANDRÉS CHAMORRO CUPUERÁN**, con cédula de identidad 1003586623, cedo a la Universidad de Tecnología Experimental Yachay, los derechos de publicación de la presente obra, sin que deba haber un reconocimiento económico por este concepto. Declaro además que el texto del presente trabajo de titulación no podrá ser cedido a ninguna empresa editorial para su publicación u otros fines, sin contar previamente con la autorización escrita de la Universidad.

Asimismo, autorizo a la Universidad que realice la digitalización y publicación de este trabajo de integración curricular en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior

Urcuquí, septiembre 2019.



Kevin Andrés Chamorro Cupuerán
CI: 1003586623

Acknowledgment

I would like to express my sincere gratitude to my advisor and friend, Ph.D. Juan Carlos López, for allowing me to undertake this work and his continuous guidance advice effort and invertible suggestion throughout the degree project giving me the main concepts to develop a mathematical model and finish successfully this work.

I am also grateful to my co-advisor PhD. Saba Infante for providing me the main concepts of probabilistic and statistical in a game of basketball, for his support, and his valuable suggestion to carry out my degree project successfully.

I would like to thank to Mary and Andrés in a very special way for helping me unconditionally during this degree project and being my unconditional support at every moment.

Thanks to all my friends, classmates, teachers who have helped me throughout my student career to acquire my current level of knowledge and capacity of researching, specially to Yachay Tech's professors.

Lastly I would like to express my sincere appreciation to God and family especially my Mom for encouraging and supporting me throughout the study.

Kevin Andrés Chamorro Cupuerán

Abstract

Nowadays, mathematics and statistics are everywhere and are very important to improve the performance of a company or industry. For example, in basketball industry, professional teams, especially in the NBA use mathematics and statistics to improve the player's performance and to win games through the study of free throws. Why study free throws? Statistical studies indicate that the number of points obtained in games through free throws is between 20-25% of the total points in a game. It means that the team that will win will be the team that has the highest probability in order to obtain a successful free throw. Therefore, we propose a mathematical model without interaction the board in 2D that which allows us to improve the probability to obtain a successful shot in a player through the analytical, numerical, probabilistic, and statistical study of the optimal release angle and release velocity. Since we are not robots to throw exactly with the angle and velocity that give us. We will find the maximum error allowed in release angle and velocity in order to continue having a successful shot. We use MATLAB software to solve the equations, and problems of univariate and multivariate optimization to obtain the solutions about the heights of Shaq 2.16 m and Kev 1.83 m (author). In addition, we will include Monte Carlo simulations to find the zone of success with the greatest probability of each throw and thus validate our model. Finally, we will our conclusions through a comparative analysis between simulated data and real data using a multiple regression model.

Key Words:

Free throws, release angle, release velocity, maximum error allowed, univariate optimization, multivariate optimization, Monte Carlo simulations, multiple regression model.

Resumen

Hoy en día, las matemáticas y las estadísticas están en todas partes y son muy importantes para mejorar el rendimiento de una empresa o industria. Por ejemplo, en la industria del baloncesto, los equipos profesionales, especialmente en la NBA, usan las matemáticas y las estadísticas para mejorar el rendimiento del jugador y ganar juegos a través del estudio de tiros libres. ¿Por qué estudiar tiros libres? Los estudios estadísticos indican que el número de puntos obtenidos en juegos a través de tiros libres es entre 20-25% del total de puntos en un juego. Significa que el equipo que ganará será el equipo que tenga la mayor probabilidad de obtener un tiro libre exitoso. Por lo tanto, proponemos un modelo matemático sin interacción del tablero en 2D que nos permita mejorar la probabilidad de obtener un disparo exitoso en un jugador a través del estudio analítico, numérico, probabilístico y estadístico del ángulo de liberación y la velocidad de liberación óptima. Ya que no somos robots para lanzar exactamente con el ángulo y la velocidad que nos dan. Encontraremos el error máximo permitido en el ángulo de liberación y la velocidad para continuar teniendo un lanzamiento exitoso. Utilizamos el software MATLAB para resolver las ecuaciones y los problemas de optimización univariada y multivariada para obtener soluciones sobre las alturas de Shaq 2.16 m y Kev 1.83 m (autor). Además, incluiremos simulaciones de Monte Carlo para encontrar la zona de éxito con la mayor probabilidad de cada lanzamiento y así validar nuestro modelo. Finalmente, sacaremos nuestras conclusiones a través de un análisis comparativo entre datos simulados y datos reales utilizando un modelo de regresión múltiple.

Palabras clave:

Tiros libres, ángulo de liberación, velocidad de liberación, error máximo permitido, optimización univariante, optimización multivariante, simulaciones de Monte Carlo, modelo de regresión múltiple.

Contents

1	Introduction	4
2	Preliminaries	5
2.1	Numerical Optimization with MATLAB	5
2.1.1	Minimization of a Function	6
2.1.2	Golden Ratio Search and Parabolic Interpolation	6
2.1.3	The function <i>fminbnd</i>	7
2.1.4	The function <i>fminsearch</i>	8
2.2	Monte Carlo simulations	8
2.2.1	Monte Carlo Integration	10
2.2.2	Monte Carlo method with importance sampling	11
2.2.3	Estimation of errors	12
2.2.4	Bivariate Normal Distribution	13
2.3	Regression analysis model	14
2.3.1	Simple linear regression	14
2.3.2	Multiple linear regression	16
2.4	Model of basketball free throw	18
2.4.1	General Objective	19
2.4.2	Specific Objectives	19
3	Mathematical and numerical analysis of the model	20
3.1	First Case	20
3.1.1	Problem definition	20
3.1.2	Mathematical Analysis.	20
3.1.3	Derivation of an equation for the new horizontal position of the ball as it comes back down to the basket height	23
3.1.4	Derivation of the conditions so that the ball does not hit the front or the back of the ring	24
3.1.5	Derivation of the error allowed function for a given initial angle θ_0 to still have a successful throw	27
3.1.6	Presentation of the results.	28
3.2	Second Case.	28
3.2.1	Problem definition.	29
3.2.2	Numerical Analysis and presentations of the results.	29
3.2.3	Constructing the error and percent error in release angle and release velocity.	32
3.3	Third case	34
3.3.1	Problem definition	35
3.3.2	Numerical Analysis and presentations of the results	35
3.4	Summary	37
4	Probabilistic Analysis	37
4.1	Problem definition	37
4.2	Numerical Analysis and presentations of the results	38
5	Statistical Analysis	39
5.1	Problem definition	39
5.2	Methods	40
5.3	Presentation of the results	40
6	Conclusions	44
	References	44
	Appendix	46

A	First case codes.	46
A.1	Initialize global variables (initialize.m)	46
A.2	Range of angles to the back of the ring	46
A.3	Program to plot errors about θ_0 (plot_angerror_vopt.m)	46
A.3.1	Program to calculate the maximum allowed error in the angle with respect to the back and front of the rim when the ball is thrown with optimal velocity (calcerrorvopt.m)	47
A.3.2	Program to calculate the optimal velocity to hit the center of the basket given the initial angle of θ_0 (calcOptv.m)	47
A.3.3	Program to calculate the maximum allowed error in the angle with respect to the back and front of the rim when the ball is thrown with given velocity (calcerror.m)	47
A.3.4	Program to calculate the minimum distance from the front of the rim for the ball in the relevant time interval (frontfzero.m)	48
A.3.5	Program to calculate the distance from the front of the rim for the ball at time t (distancefromrim.m)	48
A.3.6	Program to calculate the position of the ball at time t, thrown with given velocity and angle (position.m)	49
A.3.7	Program 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 (calcfronerror.m)	49
A.3.8	Program to calculate the maximum allowed error in the angle with respect to the back of the rim (calcbackerror.m)	49
A.3.9	Program to calculate the distance from the back of the rim for the ball when its center is level with the rim (distancefromback.m)	50
B	Second case codes.	50
B.1	Criterion back of the ring	50
B.1.1	Program to calculate the velocity to hit the back of the rim (vbackrim.m)	50
B.2	Criterion center of the ring	51
B.2.1	Program to calculate the velocity to hit the center of the rim (vcenterrim.m)	51
B.3	Criterion front of the ring	51
B.3.1	Program to calculate the minimum distance from the front of the rim for the ball in the relevant time interval (frontfzerov.m)	52
B.3.2	Program to calculate the velocity to hit the front of the rim (vfrontrim.m)	52
B.4	Feasible Region	53
B.5	Program to calculate the error allowed angle in the feasible region (error_allowed_angle.m)	53
B.6	Program to plot and calculate the solution for the second case (plot_second_case.m)	54
B.7	Program to calculate the error allowed velocity in the feasible region (error_allowed_velocity.m)	56
C	Third case codes.	56
C.1	Program to calculate the weighted error allowed angle and velocity (weighted_error.m)	56
D	Probabilistic analysis codes.	58
D.1	Monte Carlo procedure (montecarlo.m)	58
D.2	Contour plot of score probability distribution (mc.m)	59
D.2.1	Program to calculate simulated data with a Monte Carlo simulations (montecarlo1.m)	60
D.2.2	Probability in each point of the feasible region (success.m)	61
E	Statistical Analysis	62

1 Introduction

Basketball is one of the most famous sports in the world, it is better paid than many other sports and one of the most played. Basketball has millions of fans around the world. Apparently, basketball and mathematics seem to have little in common. While basketball is a hugely popular sport in the world, mathematics have a considerably smaller number of followers. Showing how math is used in basketball is a great way to make children and youth more passionate with angles and statistics while helping them to realize how important mathematics are in everyday life. If we analyze a little further these two concepts, there is much math in basketball than we think. For instance, in free throws, mathematics are used to improve player's performance, considering that, many basketball games are won or lost on the free throw line. Studies [1] indicate that the number of points obtained in matches through free throws is between 20-25% of the total scored in a game. It means that, if a match is constantly tied, the match will be defined by the team that has the most success in their free throws. Now, if we take as example the statistics of the famous basketball player Shaquille O'Neal, in the 2004-2005 season, he reached a free throw percentage of 53 % in the qualifying phase, and in the direct elimination (Playoffs) he reduced his percentage to 45% in free throws. For this reason, there is a famous saying (Hack-a-Shaq) that is referred to make an intentional foul to players bad at free throws. It is often seen in the last few minutes of the Professional Basketball League game that the team with lower scores may use Hack-a-Shaq against the player in the other team who are bad at free throws, so the team with lower scores will greatly increase the chance to win. Hack-a-Shaq is often used in NBA games to turn the match, but if every basketball player's free throw skill is good, then the Hack-a-Shaq will automatically collapse. Then, if every player can do both successful free throws, it will greatly increase the chance of the team to win. Free throw shootings are therefore very crucial in games, so it is necessary to apply math studies and hard training to improve basketball player's free throw skills as well as consistency and hit rate. Since throwing consistently could be an important part of the game, most players perform training with throwing exercises to develop a consistent shooting technique. Initially, the player must focus on understanding the best throw conditions, that is, at what angle and velocity should the ball be thrown and where should it be aimed? For example, the release height of the ball influences the optimal release angle having an effect also in where to aim. Due to a large number of factors involved, a great many free throws must be studied to obtain an entire understanding of the optimal release conditions.

Studying the optimal release conditions through systematic experimentation with players can result extremely time-consuming. Therefore, an important alternative is to investigate the optimal release conditions through computer simulations, in which many throws are investigated in a reduced time besides the analytical study of it. From a mathematical viewpoint, basketball is a game of trajectories. These trajectories are unique in the sense that the ball's motion does not differ much when it is flying through the air, but then rapidly changes when the ball collides with the hoop or the backboard. So, in this project we are going to carry out a study of several mathematical models in basketball free throws, through computational simulations to find, depending on the height of the player, which is the best release angle and release velocity for a successful throw.

Several studies about free throw shooting were developed by different researchers. Shibukawa [2] who analyzed the angle and velocity of release for free throw, suggested that the release angle of a successful free throw was $52-55^\circ$. Brancazio, [3], describes an analytical method and states that if a free throw shooting is made with the minimum energy required, the player will have the best control and, therefore, the best chance of success. In other words, he emphasizes the importance of throwing the ball with the minimum possible velocity, which allows a smoother throw. In addition, he argues that the angle of 49° is the one that gives the highest probability of a successful free throw based on the allowable margin of error for both the velocity and the release angle. The optimal trajectory of the free throw is developed by Hamilton and Reinschmidt [4] who studied free throws as a function of angle, velocity and, spin at release. They predict that a release angle of 59° with 7.26 m/s and a high backspin has an optimal trajectory. Tran and Silverberg [5] studied a three dimensional numerical method to determine the math behind the optimal free throw and extended their results to bank throws (reflected from backboard) [6]. Since a free throw takes about 1 second for a ball to reach the basket, they found that about 3 hertz of backspin from the instant the ball leaves the player's hands to when it reaches the basket, is the best amount. Moreover, they argued that a 52° release angle allowed a great possibility for a successful free throw. In that angle, the release velocity is the lowest, and the probability of the throw being successful is the greatest. On the other hand, Gablonsky and Lang [7] studied a two dimensional analytic and numerical model. They started by proposing a model with ideal conditions in which they assumed that the best release angle is the one that achieves enter right through the center of the ring. This model assumes the player has consistency in his release velocity, that is, there is no expected error in the release velocity; however, it is

not consistent in its release angle, that is, there is an expected error for each release angle. Then, to find the best angle, only the expected angle error is maximized and with that angle, its velocity is found so that the ball enters through the center of the ring. Then, they noticed that in order to define optimal release conditions, has to be maximized the expected error in angle and velocity. Therefore, they proposed another model, based on the best trajectory with the use of a multiobjective optimization in which the best trajectory is one that puts five times as much emphasis on the error in velocity as on the error in angle. Finally, they found that the best angles were from 52° to 56° and the best velocities from 6.64 m/s to 7.34 m/s for heights from 1.52 m to 2.21 m. Barzykina [8] suggested that the optimal throwing conditions are determined numerically by maximizing the expected number of successful throws given the error pattern inherent to the player. Moreover, his analysis is supported by Monte Carlo simulations and a series of free throws.

Therefore, the main objective of a basketball game is to win the game by getting as many baskets as possible, either free throw of 2 or 3-points throws. So, for the coaches it is very important to make their players take advantage of every opportunity they have to make a basket and over all, not missing the free throws that are one of the most important strategies to win, since these are the ones that most occur during a game allowing the team to beat matches and even championships. During free throws, points are scored by shooting the ball through a horizontal hoop, elevated 3.05 meters from the ground. The release conditions: velocity, angle, and height, have been considered the principal factors determining the outcome of a basketball throw. Therefore, the question of many basketball coaches is what is the best free throw shooting? That is, what is the best angle and release velocity that a player of a certain height must have to score a basket? The main aspects to solve these questions are the analytical, numerical, probabilistic and statistical methods involved in the mathematical study of free throws with interaction with and without the board in 2D and 3D. We will focus on a 2D model proposed by Gablonsky and Lang [7], in which we will follow their methods to obtain our optimal release angle and velocity. We use MATLAB software to solve the equations and obtain the solutions, as well as for developing our own codes to find optimal release angles and velocities. In addition, we will use the methodology by Barzykina [8] to include Monte Carlo simulations to simulate the normal bi-variate random variable that will allow us to simulate millions of free throws given a target point (optimal release angle and release velocity), in which we will verify the number of successful throws obtained for each simulation of different targets which allows to find the zone of success with the greatest probability of each target and thus validate that our throw is in the greatest zone of success. Finally, we will apply this last methodology to validate a mathematical model with real data from a semiprofessional basketball team of Ecuador, combining the methods and analysis from Gablonsky and Lang [7] through a comparative analysis between simulated data and real data using a multiple regression model with the help of R software.

The rest of the work has been organized into 6 sections. Section 2 introduces classic mathematical concepts that are necessary for the resolution of the model as, uniobjective and multiobjective Optimization using a computer algebra system's routine, Markov chain, Monte Carlo simulation to reproduce a million shot free throw, and multiple regression model to analyze real data obtained from a semi-professional basketball team. In section 3, the mathematical and numerical analysis of the model is performed using the methodology of Gablonsky and Lang [7] divided into three cases, which consist mainly of studying the projectile motion equations for a free throw, finding a feasible throwing range depending on the height of the player and find the allowed errors for both the angle and velocity of release. In section 4 a probabilistic analysis is performed by using the Barzykina methodology [8] to validate our model of best free-throw. In section 5 we will study a statistical analysis where we make a comparative analysis between simulated data from section 3 vs real data using a multiple regression model for basketball free throws. Finally, in section 6 we present some conclusions and recommendations.

2 Preliminaries

In this section, we will introduce classical mathematical concepts that will be necessary for the resolution of the model.

2.1 Numerical Optimization with MATLAB

In this subsection we will study the Optimization problem with the Toolbox of MATLAB, apply the different methods of optimization functions without constrains for one and several variables.

From a mathematical perspective, optimization tries to find the maximum and minimum of a function that depends on one or more variables.

2.1.1 Minimization of a Function

We are going to enunciate some definitions and theorems of Mathews's and Fink's [9] book for the optimization problem.

Definition 2.1.1 (Local Extremum). The function f is said to have a **local minimum value** at $x = p$, if there exists an open interval I containing p so that $f(p) \leq f(x)$ for all $x \in I$. Similarly, f is said to have a local maximum value at $x = p$ if $f(x) \leq f(p)$ for all $x \in I$. If f has either a local minimum or maximum value at $x = p$, it is said to have a **local extremum** at $x = p$.

Definition 2.1.2 (Increasing and Decreasing). Assume that $f(x)$ is defined on the interval I .

- (i) If $x_1 < x_2$ implies that $f(x_1) < f(x_2)$ for all $x_1, x_2 \in I$, then f is said to be **increasing** on I .
- (ii) If $x_1 < x_2$ implies that $f(x_1) > f(x_2)$ for all $x_1, x_2 \in I$, then f is said to be **decreasing** on I .

Theorem 2.1.1. Suppose that $f(x)$ is continuous on $I = [a, b]$ and is differentiable on (a, b) .

- (i) If $f'(x) > 0$ for all $x \in (a, b)$, then $f(x)$ is increasing on I .
- (ii) If $f'(x) < 0$ for all $x \in (a, b)$, then $f(x)$ is decreasing on I .

Theorem 2.1.2. Assume that $f(x)$ is defined on $I = [a, b]$ and has a local extremum at an interior point $p \in (a, b)$. If $f(x)$ is differentiable at $x = p$, then $f'(p) = 0$.

Theorem 2.1.3 (First Derivative Test). Assume that $f(x)$ is continuous on $I = [a, b]$. Furthermore, suppose that $f'(x)$ is defined for all $x \in (a, b)$, except possibly at $x = p$.

- (i) If $f'(x) < 0$ on (a, p) and $f'(x) > 0$ on (p, b) , then $f(p)$ is a local minimum.
- (ii) If $f'(x) > 0$ on (a, p) and $f'(x) < 0$ on (p, b) , then $f(p)$ is a local maximum.

Theorem 2.1.4 (Second Derivative Test). Assume that f is continuous on $[a, b]$ and f' and f'' are defined on (a, b) . Also, suppose that $p \in (a, b)$ is a critical point where $f'(p) = 0$.

- (i) If $f''(p) > 0$, then $f(p)$ is a local minimum of f .
- (ii) If $f''(p) < 0$, then $f(p)$ is a local maximum of f .
- (iii) If $f''(p) = 0$, then nothing can be affirmed.

2.1.2 Golden Ratio Search and Parabolic Interpolation

Another method for finding the minimum of $f(x)$ is to evaluate the function many times and search for a local minimum. To reduce the number of function evaluations, it is important to have a good strategy for determining where $f(x)$ is evaluated. The most efficient methods are called the **golden ratio search** and **parabolic interpolation**. See [10] for more details about the algorithms.

Golden-Section Search

- Pick two initial guesses, x_l and x_u , that bracket one local extremum of $f(x)$.
- Choose two interior points x_1 and x_2 according to the golden ratio $\phi = \frac{1+\sqrt{5}}{2}$.

$$\begin{aligned}d &= (\phi - 1)(x_u - x_l), \\x_1 &= x_l + d, \\x_2 &= x_u - d.\end{aligned}$$

- Evaluate the function at x_1 and x_2 .
- If $f(x_1) < f(x_2)$, x_2 becomes the new lower limit and x_1 becomes the new x_2 (see Figure 1).
- If $f(x_2) < f(x_1)$, x_1 becomes the new upper limit and x_2 becomes the new x_1 . (see Figure 1).

- The benefit of using golden ratio is that we do not need to recalculate all the function values in the next iteration.
- In any case, only one new interior point is needed and the function is only evaluated one more time.

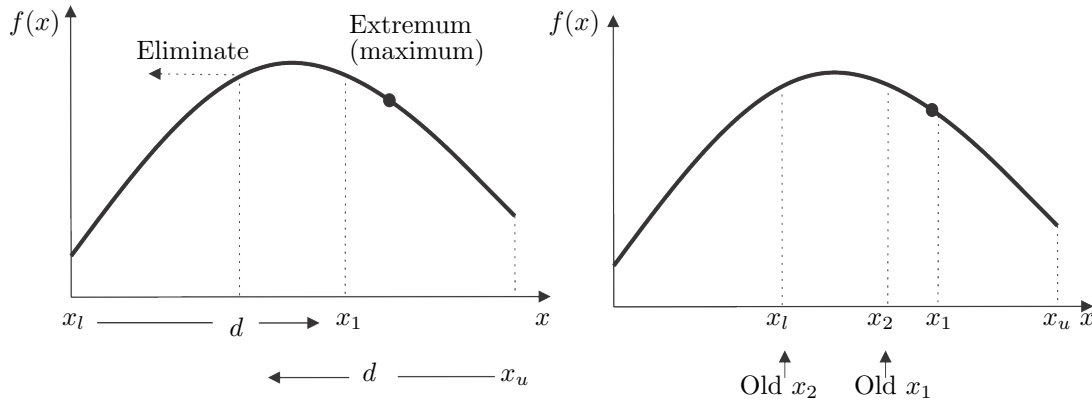


Figure 1: Golden-Section search.

Parabolic Interpolation. Another algorithm uses parabolic interpolation of three points (see Figure 2) to estimate optimum location. The location of the maximum/minimum of a parabola defined as the interpolation of three points (x_1, x_2 and x_3) is

$$x_4 = x_2 - \frac{1}{2} \frac{(x_2 - x_1)^2[f(x_2) - f(x_3)] - (x_2 - x_3)^2[f(x_2) - f(x_1)]}{(x_2 - x_1)[f(x_2) - f(x_3)] - (x_2 - x_3)[f(x_2) - f(x_1)]}$$

The new point x_4 and the two surrounding it (either x_1 and x_2 or x_2 and x_3) are used for the next iteration of the algorithm. See [10] for more details about the algorithm.

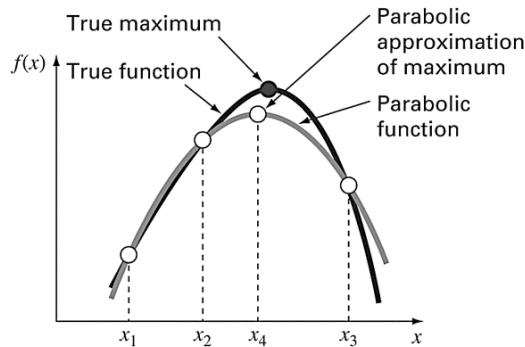


Figure 2: Parabolic Interpolation

2.1.3 The function f_{minbnd}

Problems of optimization function without constrains for one variable, are formulated mathematically in the following way:

$$\min_x f(x) \quad \text{such that} \quad x_1 < x < x_2,$$

where, x, x_1 and x_2 are finite scalars, and $f(x)$ is a function that returns a scalar [11].

This function looks to find the variable x , in the real domain, that minimizes the function f . That is, that the value of the function evaluated in x is the minimum value that can reach f in the region defined by the “limits” imposed on x .

MATLAB has a built-in function, ***fminbnd***, which combines the golden-section search and the parabolic interpolation. It finds the minimum of a function of one variable on a fixed interval, and have the following syntax:

$$x = \text{fminbnd}(@fun, x_1, x_2, options).$$

This function returns the solution to the variable x , which is defined by fun . In the optimization functions, the name of the function is preceded by @.

Note that the solution is the scalar that minimizes the value of the function f , and the parameters x_1 and x_2 define the search region of the solution. In the defined parameter $options$. The parameters can be specified both of the algorithm and of execution of the function. If it is not used, since these parameters have default values, you can put $[\cdot]$ in its place, or simply skip it.

$$[x, fval] = \text{fminbnd}(@fun, x_1, x_2, options).$$

In this case, also in $fval$, the value of the function f is returned evaluated in the solution x .

2.1.4 The function *fminsearch*

Another method for finding the minimum of $f(x)$ but in this case with several variables is the **Nelder-Mead Method**.

A simplex method for finding a local minimum of a function of several variables has been devised by Nelder and Mead (1965). For two variables, a simplex is a triangle, and the method is a pattern search that compares function values at the three vertices of a triangle. The worst vertex, where $f(x, y)$ is largest, is rejected and replaced with a new vertex. A new triangle is formed and the search is continued. The process generates a sequence of triangles (which might have different shapes), for which the function values at the vertices get smaller and smaller. The size of the triangles is reduced and the coordinates of the minimum point are found. The algorithm is stated using the term simplex (a generalized triangle in N dimensions) and will find the minimum of a function of N variables. It is effective and computationally compact [9].

So, the problem of optimization function without constrains for several variables, is formulated mathematically in the following way:

$$\min_x f(x),$$

where $f(x)$ is a function that returns a scalar, and x is a vector or a matrix [11].

This function seeks to find the vector x , whose components belong to the real domain, which minimizes the function f . That is, the value of the function evaluated in x is the minimum value f can reach. This type of problem is usually called Nonlinear Optimization without restrictions and can be solved using the Nelder-Mead method [11].

MATLAB has a built-in function, ***fminsearch***, which uses the Nelder-Mead simplex algorithm as described in Lagarias [12]. This is a direct search method that does not use numerical or analytic gradients. Therefore, this function ***fminsearch*** can be used to determine the minimum or maximum of a multidimensional function and have the following syntax:

$$x = \text{fminsearch}(@fun, x_0, options).$$

The function *fminsearch* finds the value of the variables x that minimize the function described in fun , starting with the initial value specified in x_0 .

2.2 Monte Carlo simulations

Monte Carlo simulation is a technique used to understand the impact and uncertainty in probability models. The precision of the simulation depends on the precision of the model. We can say that, simulating aims to duplicate characteristics and behaviors own of a real system [13]. As an example we can estimate the value of $\pi = 3.141592\dots$, using a Monte Carlo method. This methods consists of drawing a square with an inner circle. We then generate a large number of random points within the square and count how many fall in the enclosed circle.

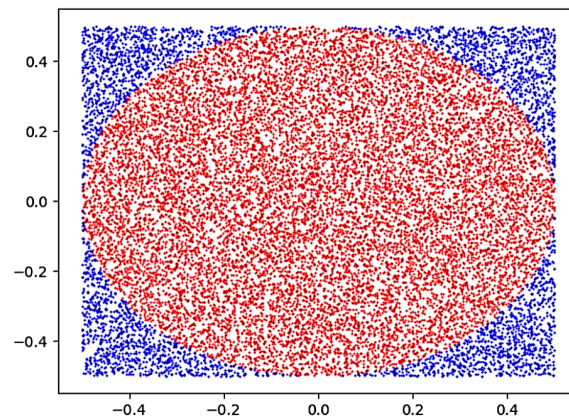


Figure 3: illustration of the Monte Carlo method used to calculate ϕ . Red points are bounded by a circle of radius $\frac{1}{2}$ and blue points are outside the circle and within the unit square.

- The area of the circle is πr^2 .
- The area of the square is $width^2 = (2r)^2 = 4r^2$.
- If we divide the area of the circle, by the area of the square we get $\frac{\pi}{4}$.
- The same ratio can be used between the number of points within the square and the number of points within the circle.
- Hence we can use the following formula to estimate π ,

$$\pi = 4 \times \frac{\text{number of points in the circle}}{\text{total numbers of points}}.$$

As another example, if we assume that the probability to see the number of faces in a coin toss experiment is unknown, we can perform the experiment of throwing the coin n times and to approximate the probability of faces $p(C)$ that appear as

$$p(C) = \frac{\text{Number of faces observed}}{\text{Number of times the experiment was executed}}.$$

Nevertheless, for many practical problems, it is not possible to determine probabilities by running experiments a large number of times. With today's computer processing capabilities, it just needs a high-level language, which can generate random numbers, to deal with these problems.

Monte Carlo methods are a class of computational algorithms that can be applied to a wide range of problems, and are based on random repeated sampling and provide generally approximate solutions, they are used in cases where analytical or numerical solutions do not exist or are too difficult to implement.

In this work, ball launches will be simulated considering the angles and velocities to validate a mathematical model that represents in a reliable way the reality of a basketball game. The methodology allows to introduce new variables, to vary their values, to analyze the consequences of these modifications, with the objective to make optimal decisions.

- Advantages
 - It is a direct and flexible method.
 - There are many programs and languages (Matlab, R, Python, etc) designed to simulate.
 - When the mathematical model is very complicated, the simulation allows obtaining an approximation.

- The simulation allows formulating extreme conditions with the launches.
- Simulation does not intervene in the real world, it allows experimentation.
- Simulation allows solving problems that have no analytical solution.
- Disadvantages
 - A good simulation can be complicated when there are a large number of variables.
 - Simulation does not generate optimal global solutions.
 - It does not provide the decision to be made if it does not solve the problem by means of an approximation for initial conditions.
 - Each simulation is unique, and chance intervenes.

Monte Carlo methods generally follow the following steps:

- Determine the statistical properties of the possible entries.
- Generate many observations of possible entries that follow the properties noted above.
- Perform a deterministic calculation with these data sets.
- Statistically analyze the results.
- The error in the results generally decreases as $\frac{1}{\sqrt{N}}$.

2.2.1 Monte Carlo Integration

The basic idea of the Monte Carlo (MC) method is to write the integral desired as an expected value with respect to some distribution of probability. Suppose that we want to calculate the integral of some smoothed function in a known range (a, b) , that is:

$$I = \int_a^b g(\theta)d\theta. \quad (1)$$

The integral given in (1), can be written as:

$$I = \int_a^b (b-a)g(\theta) \frac{1}{(b-a)} d\theta = \mathbb{E}_{U(a,b)} [(b-a)g(\theta)],$$

where $U(a, b)$ is a random variable whose distribution is uniform at (a, b) .

The method of the moments is an estimator of this quantity, that is,

$$\hat{I} = \frac{1}{n} \sum_{i=1}^n (b-a)g(\theta_i),$$

where $\theta_1, \dots, \theta_n$ is a random sampling selected from a uniform distribution over (a, b) .

Algorithm: Monte Carlo.

1. Are generated $\theta_1, \dots, \theta_n \sim U(a, b)$.
2. It is calculated $g(\theta_1), \dots, g(\theta_n)$.
3. It is estimated $\bar{g} = \frac{1}{n} \sum_{i=1}^n g(\theta_i)$.
4. It is approximated $\hat{I} = (b-a)\bar{g}$.

A generalization can be obtained clearly. Let

$$I = E_p[g(\theta)],$$

be the expected value of $g(\theta)$ with respect to a probability density function $p(\theta)$. The algorithm is similar to the previous one, only sampling modifications are made in step [1]; that is, there are generated $\theta_1, \dots, \theta_n \sim p(\cdot)$ instead of a uniform and the rest is the same. The multivariate extension is based on the following:

$$I = \int_{a_1}^{b_1} \dots \int_{a_p}^{b_p} g(\theta) d\theta,$$

and the Monte Carlo estimator is:

$$\hat{I} = \frac{1}{n} \sum_{i=1}^n g(\theta_i),$$

where $\theta_1, \dots, \theta_n$ is a random sample selected from the uniform distribution over $(a_1, b_1) \times \dots \times (a_p, b_p)$.

2.2.2 Monte Carlo method with importance sampling

Monte Carlo method with importance sampling is a technique developed to reduce the variance estimator. Consider explicitly that the integral I of interest is the expected value of a given function g with respect to a p.d.f. $p(\cdot)$:

$$I = \int g(x)p(x)dx = \int g(x)\frac{p(x)}{h(x)}h(x)dx,$$

where $h(x)$ is a positive function for all x , $p(x) > 0$ and $\int h(x)dx = 1$. An alternative method to the moments one is the following, let:

$$\bar{I} = \frac{1}{n} \sum_{i=1}^n g(x_i)w(x_i),$$

where

$$w(x_i) = p(x_i)/h(x_i) \quad \text{and} \quad x_i \sim h(x), \quad i = 1, \dots, n,$$

h is the importance function, and

$$\text{Var}(\bar{I}) = \frac{1}{n} \int [g(x)w(x) - I]^2 h(x)dx.$$

These methods can be used to solve some Bayesian inference problems, such as when you want to evaluate $\mathbb{E}[g(\theta)|x]$.

Algorithm: Monte Carlo with importance sampling.

1. It is generated $\theta_1, \dots, \theta_n$ from $p(\theta|x)$ or from $h(\theta)$.
2. It is calculated:

$$g_i = g(\theta_i) \quad \text{or} \quad g_i = \frac{g(\theta_i)P(\theta_i|x)}{h(\theta_i)}, \quad i = 1, \dots, n.$$

3. It is obtained the estimator:

$$\mathbb{E}[g(\theta)] = \frac{1}{n} \sum_{i=1}^n g_i.$$

2.2.3 Estimation of errors

Given any arbitrary probability distribution and provided that a random variable can be sampled appropriately from the distribution (i.e., $x \sim f(x)$) Monte Carlo simulations can be used:

- To determine the properties of the distribution (such as media, fashion, variance, quantiles).
- To determine the confidence intervals, that is:

$$p(x > \alpha) = \int_{\alpha}^{\infty} f(x)dx.$$

- To determine the composition of the functions of the random variables, that is,

$$p(x), \quad p(h(x)), \quad h(x) = x^2, \quad p(x) = \cos(x) - \sin(x), \dots$$

For example, assume you have N that follow a normal probability model $x_i \sim N(0, \sigma)$, $i = 1, \dots, N$, and you want to determine the uncertainty about the mean. The estimator of the mean is

$$\hat{\mu} = \bar{x} = \frac{1}{N} \sum_{i=1}^N x_i,$$

and its uncertainty is

$$\hat{\sigma}_{\bar{x}} = \frac{\sigma}{\sqrt{N}}.$$

Algorithm: the Monte Carlo algorithm for this problem will be:

Step 1: to generate a set of N random variables

$$y_i \sim N(0, \sigma), \quad i = 1, \dots, N.$$

Step 2: to calculate the mean of the sample

$$\bar{y} = \frac{1}{N} \sum_{i=1}^N y_i.$$

Step 3: to repeat M times Step 1 and Step 2.

Step 4: to calculate the uncertainty of the mean $\hat{\sigma}_{\bar{x}}$, such that

$$\hat{\sigma}_{\bar{y}}^2 = \frac{1}{M-1} \sum_{j=1}^M (\bar{y}_j - \hat{y})^2,$$

where

$$\hat{y} = \frac{1}{M} \sum_{j=1}^M \bar{y}_j.$$

2.2.4 Bivariate Normal Distribution

In order to perform the Monte Carlo simulation we need to simulate a random variable, in our case it is the Bivariate Normal. Let $z_1, z_2 \sim N(0, 1)$, which we will use to build a general Bivariate Normal Distribution

$$f(z_1, z_2) = \frac{1}{2\pi} \exp \left[-\frac{1}{2}(z_1^2 + z_2^2) \right].$$

We want to transform these unit normal distribution to have the following arbitrary parameters : $\mu_{\theta_0}, \mu_{V_0}, \sigma_{\theta_0}, \sigma_{V_0}, \rho$. Let,

$$\theta_0 = \sigma_{\theta_0} Z_1 + \mu_{\theta_0}, \quad (2)$$

$$V_0 = \sigma_{V_0} \left[\rho Z_1 + \sqrt{1 - \rho^2} Z_2 \right] + \mu_{V_0}. \quad (3)$$

Marginals: first, lets examine the marginal distributions of θ_0 and V_0 .

$$\begin{aligned} \theta_0 &= \sigma_{\theta_0} Z_1 + \mu_{\theta_0} \\ &= \sigma_{\theta_0} \mathcal{N}(0, 1) + \mu_{\theta_0} \\ &= \mathcal{N}(\mu_{\theta_0}, \sigma_{\theta_0}^2). \end{aligned} \quad (4)$$

$$\begin{aligned} V_0 &= \sigma_{V_0} \left[\rho Z_1 + \sqrt{1 - \rho^2} Z_2 \right] + \mu_{V_0} \\ &= \sigma_{V_0} \left[\rho \mathcal{N}(0, 1) + \sqrt{1 - \rho^2} \mathcal{N}(0, 1) \right] + \mu_{V_0} \\ &= \sigma_{V_0} \left[\mathcal{N}(0, \rho^2) + \mathcal{N}(0, 1 - \rho^2) \right] + \mu_{V_0} \\ &= \sigma_{V_0} \mathcal{N}(0, 1) + \mu_{V_0} \\ &= \mathcal{N}(\mu_{V_0}, \sigma_{V_0}^2). \end{aligned} \quad (5)$$

Covariance and correlation: second, we can find $Cov(\theta_0, V_0)$ and $\rho(\theta_0, V_0)$.

$$\begin{aligned} Cov(\theta_0, V_0) &= E \left[(\theta_0 - E(\theta_0)) (V_0 - E(V_0)) \right] \\ &= E \left[(\sigma_{\theta_0} Z_1 + \mu_{\theta_0} - \mu_{\theta_0}) \left(\sigma_{V_0} \left[\rho Z_1 + \sqrt{1 - \rho^2} Z_2 \right] + \mu_{V_0} - \mu_{V_0} \right) \right] \\ &= E \left[(\sigma_{\theta_0} Z_1) \left(\sigma_{V_0} \left[\rho Z_1 + \sqrt{1 - \rho^2} Z_2 \right] + \mu_{V_0} \right) \right] \\ &= \sigma_{\theta_0} \sigma_{V_0} E \left[\rho Z_1^2 + \sqrt{1 - \rho^2} Z_1 Z_2 \right] \\ &= \sigma_{\theta_0} \sigma_{V_0} \rho E \left[Z_1^2 \right] \\ &= \sigma_{\theta_0} \sigma_{V_0} \rho, \end{aligned}$$

and

$$\rho(\theta_0, V_0) = \frac{Cov(\theta_0, V_0)}{\sigma_{\theta_0} \sigma_{V_0}} = \rho.$$

Consequently, if we want to generate a Bi-variate Normal random variable with $\theta_0 \sim \mathcal{N}(\mu_{\theta_0})$ and $V_0 \sim \mathcal{N}(\mu_{V_0}, \sigma_{V_0}^2)$ where the correlation of θ_0 and V_0 is ρ we can generate two independent unit normals Z_1 and Z_2 and use the transformation:

$$\theta_0 = \sigma_{\theta_0} Z_1 + \mu_{\theta_0}, \quad (6)$$

$$V_0 = \sigma_{V_0} \left[\rho Z_1 + \sqrt{1 - \rho^2} \right] + \mu_{V_0}. \quad (7)$$

General Bivariate Normal- Density (Matrix Notation): we can find the density from using matrix notation,

$$f(z) = \frac{1}{2\pi} (\det \Sigma)^{-1/2} \exp \left[-\frac{1}{2} (z - \mu)^T \Sigma^{-1} (z - \mu) \right],$$

where

$$z = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}, \quad \mu = \begin{pmatrix} \mu_{\theta_0} \\ \mu_{V_0} \end{pmatrix}, \quad \Sigma = \begin{pmatrix} \sigma_{\theta_0}^2 & \rho \sigma_{\theta_0} \sigma_{V_0} \\ \rho \sigma_{\theta_0} \sigma_{V_0} & \sigma_{V_0}^2 \end{pmatrix}.$$

We can confirm our results by checking the value of $(\det \Sigma)^{-1/2}$ and $(z - \mu)^T \Sigma^{-1} (z - \mu)$ for the bivariate case:

$$\begin{aligned} (\det \Sigma)^{-1/2} &= (\sigma_{\theta_0}^2 \sigma_{V_0}^2 - \rho^2 \sigma_{\theta_0}^2 \sigma_{V_0}^2)^{-1/2} \\ &= \frac{1}{\sigma_{\theta_0} \sigma_{V_0} (1 - \rho^2)^{1/2}}. \end{aligned}$$

Recall for a 2×2 matrix,

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \quad A^{-1} = \frac{1}{\det A} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}.$$

Then,

$$\begin{aligned} (z - \mu)^T \Sigma^{-1} (z - \mu) &= \frac{1}{\sigma_{\theta_0}^2 \sigma_{V_0}^2 (1 - \rho^2)} \begin{pmatrix} z_1 - \mu_{\theta_0} \\ z_2 - \mu_{V_0} \end{pmatrix}^T \begin{pmatrix} \sigma_{\theta_0}^2 & -\rho \sigma_{\theta_0} \sigma_{V_0} \\ -\rho \sigma_{\theta_0} \sigma_{V_0} & \sigma_{V_0}^2 \end{pmatrix} \begin{pmatrix} z_1 - \mu_{\theta_0} \\ z_2 - \mu_{V_0} \end{pmatrix} \\ &= \frac{1}{\sigma_{\theta_0}^2 \sigma_{V_0}^2 (1 - \rho^2)} \begin{pmatrix} \sigma_{V_0}^2 (z_1 - \mu_{\theta_0}) - \rho \sigma_{\theta_0} \sigma_{V_0} (z_2 - \mu_{V_0}) \\ -\rho \sigma_{\theta_0} \sigma_{V_0} (z_1 - \mu_{\theta_0}) + \sigma_{\theta_0}^2 (z_2 - \mu_{V_0}) \end{pmatrix}^T \begin{pmatrix} z_1 - \mu_{\theta_0} \\ z_2 - \mu_{V_0} \end{pmatrix} \\ &= \frac{1}{\sigma_{\theta_0}^2 \sigma_{V_0}^2 (1 - \rho^2)} (\sigma_{V_0}^2 (z_1 - \mu_{\theta_0})^2 - 2\rho \sigma_{\theta_0} \sigma_{V_0} (z_1 - \mu_{\theta_0})(z_2 - \mu_{V_0}) + \sigma_{\theta_0}^2 (z_2 - \mu_{V_0})^2) \\ &= \frac{1}{1 - \rho^2} \left(\frac{(z_1 - \mu_{\theta_0})^2}{\sigma_{\theta_0}^2} - 2\rho \frac{(z_1 - \mu_{\theta_0})(z_2 - \mu_{V_0})}{\sigma_{\theta_0} \sigma_{V_0}} + \frac{(z_2 - \mu_{V_0})^2}{\sigma_{V_0}^2} \right). \end{aligned}$$

2.3 Regression analysis model

The regression aims to determine a simple mathematical function that describes the behavior of a variable given the values of one variable or several variables, we will focus on the book of Seber and Lee [14] to define each of these.

2.3.1 Simple linear regression

Let's assume a model in the form:

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i \quad ; \quad i = 1, \dots, n,$$

where

- y_i : r.v. that represents the i-th observation of the response variable, of the predictive variable X .
- $\beta = (\beta_0, \beta_1)$ is a vectors of parameters.
- x_i : represents a independent variable or covariable.
- ϵ_i : unobservable random error associated with y_i .

Example 2.3.1. The budget of a university whose variation can be predicted by the explanatory variable number of students. In the term of the random error it can be included, the effect of the number of professors, the number of laboratories, the available surface of facilities, the number of administrative personnel, etc.

Least Squares Estimation : let $L = \sum_{i=1}^n \sigma_i^n [y_i - (\beta_0 + \beta_1 x_i)]^2$, minimizing L we have:

$$\hat{\beta}_0 = \bar{y} - b_1 \bar{x}, \quad \text{and} \quad \hat{\beta}_1 = \frac{S(x, y)}{S_x^2},$$

where :

1. $\bar{x} = \frac{\sum_{i=1}^n x_i}{n}$.
2. $\bar{y} = \frac{\sum_{i=1}^n y_i}{n}$.
3. $S_x^2 = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n}$.
4. $S_y^2 = \frac{\sum_{i=1}^n (y_i - \bar{y})^2}{n}$.
5. $S_{xy} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{n}$.

Estimated regression line

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i \quad \text{or} \quad \hat{y}_i = \bar{y} + \hat{\beta}_1 (x_i - \bar{x}).$$

where

- $\hat{\beta}_1$: the variation that occurs in \hat{y} for each unit of increase in x .

Linear correlation coefficient (is a measure of the linear association of the variables x and y)

$$r = \frac{S(x, y)}{S_x S_y}, \quad -1 \leq r \leq 1.$$

- If $r = -1$ then there is a negative relationship between x and y .
- If $r = 1$ then there is a positive relationship between x and y .
- If $r = 0$ then there is no linear relationship between x and y .

Analysis of Variance

If \hat{y}_i are estimators of y_i then,

$$y_i - \bar{y} = (y_i - \hat{y}_i) + (\hat{y}_i - \bar{y}).$$

Basic equation of the analysis of variance

$$\sum (y_i - \bar{y})^2 = \sum (y_i - \hat{y}_i)^2 + \sum (\hat{y}_i - \bar{y})^2.$$

This equation may also be written as:

$$SST = SSE + SSM,$$

where SS is notation for **sum of squares** and T, M and E are notations for **total**, **model** and **error**, respectively.

ANOVA TABLE				
Source of Variation	Sum of Squares	Degrees of Freedom	Mean Square	F
Regression	$SSM = \sum(\hat{y}_i - \bar{y})^2$	1	$MSM = \frac{SSM}{1}$	$\frac{MSM}{MSE}$
Error	$SSE = \sum(y_i - \hat{y}_i)^2$	n-2	$MSE = \frac{SSE}{n-2}$	
Total	$SST = \sum(y_i - \bar{y})^2$	n-1	$MST = \frac{SST}{n-1}$	

Table 1: ANOVA for simple regression.

Coefficient of determination

$$R^2 = \frac{SSM}{SST} = 1 - \frac{SSE}{SST}; \quad 0 \leq R^2 \leq 1.$$

It is the statistic that represents the proportion of variation explained by the regression.

- If $R^2 = 0$, then $SSM = 0$. So the model does not explain anything about y since x .
- If $R^2 = 1$, then $SSM = SST$. Then y depends on x .
- A value of R^2 close to 0, lowers the explanatory capacity of the line.
- A value of R^2 next to 1 increases the explanatory capacity of the line.

The hypothesis regression test

$$\begin{cases} H_0 : \beta_1 = 0, \\ H_1 : \beta_1 \neq 0. \end{cases}$$

Setting a level of significance α , it is rejected H_0 if $F_{exp} > F_{\alpha,1,n-2}$.

2.3.2 Multiple linear regression

The r.v. y is related with k explanatory variables x_1, \dots, x_k ,

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_k x_k + \epsilon.$$

The parameters $\beta_0, \beta_1, \dots, \beta_k$ are estimated for least squares. For n observations it can be written:

$$\begin{aligned} y_1 &= \beta_0 + \beta_1 x_{11} + \beta_2 x_{12} + \dots + \beta_k x_{1k} + \epsilon_1, \\ &\vdots \\ y_n &= \beta_0 + \beta_1 x_{n1} + \beta_2 x_{n2} + \dots + \beta_k x_{nk} + \epsilon_n. \end{aligned}$$

In matrix notation

$$Y = X\beta + \epsilon,$$

where,

$$X = \begin{pmatrix} 1 & x_{11} & \dots & x_{1k} \\ 1 & x_{21} & \dots & x_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \dots & x_{nk} \end{pmatrix}; \quad \beta = \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_k \end{pmatrix}; \quad \epsilon = \begin{pmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \end{pmatrix}; \quad Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}.$$

The coefficients vector β is estimated by least squares by:

$$\hat{\beta} = (X^t X)^{-1} X^t Y.$$

The resulting regression adjusted equation is:

$$\hat{Y} = X\hat{\beta}.$$

Analysis of Variance

$$y_i - \bar{y} = (y_i - \hat{y}_i) + (\hat{y}_i - \bar{y}).$$

Basic equation of the analysis of variance

$$\sum (y_i - \bar{y})^2 = \sum (y_i - \hat{y}_i)^2 + \sum (\hat{y}_i - \bar{y})^2.$$

This equation may also be written as:

$$SST = SSE + SSM,$$

where *SS* is notation for **sum of squares** and T, M and E are notations for **total**, **model** and **error**, respectively.

ANOVA TABLE				
Source of Variation	Sum of Squares	Degrees of Freedom	Mean Square	F
Regression	$SSM = B^t X^t Y^t - \frac{1}{n} (\sum y_i)^2$	k	$MSM = \frac{SSM}{k}$	$\frac{MSM}{MSE}$
Error	$SSE = Y^t Y - B^t X^t Y$	n-k-1	$MSE = \frac{SSE}{n-k-1}$	
Total	$SST = Y^t Y - \frac{1}{n} (\sum y_i)^2$	n-1		

Table 2: ANOVA for multiple regression.

Multiple Determination Coefficient

$$R^2 = \frac{SSM}{SST} = 1 - \frac{SSE}{SST}; \quad 0 \leq R^2 \leq 1.$$

Represents the proportion of variation of *y* explained by the regression.

- If $R^2 = 0$ then, $SSM=0$. Then, the model does not explain anything about the variation of *y* from its linear relationship with x_1, \dots, x_k .
- If $R^2 = 1$, then, $SSM= SST$. Then, all the variation of *y* is explained by the terms present in the model.
- A value of R^2 close to 1. Then, a greater amount of total variation is explained by the regression model.

Adjusted coefficient of determination

$$\bar{R}^2 = 1 - \frac{\frac{\sum e_i^2}{n-k-1}}{\frac{\sum (y_i - \bar{y})^2}{n-1}},$$

$$e_i = y_i - \hat{y}_i.$$

The hypothesis regression test

$$\begin{cases} H_0 : \beta_1 = \beta_2 = \dots = \beta_k = 0, \\ H_1 : \beta_j \neq 0 \text{ for some } j = 1, \dots, k. \end{cases}$$

Setting a level of significance α , it is rejected H_0 if $F_{exp} > F_{\alpha,1,n-2}$.

2.4 Model of basketball free throw

The problem of a free throw is considered to be a special case of two-dimensional motion also known as projectile motion due to the position from which the projectile leaves is not at the same level that the one has it to reach (ring). Since we ignore the air resistance, the projectile follows a curved trajectory shaping to a parabola. The fate of a free throw depends on the casting conditions, the height of the player, the throwing velocity the throwing angle and the condition of 3 hertz of backspin equivalent to three revolutions in the air, from the instant the ball leaves the player's hands to when it reaches the ring, studied by Tran and Silverberg [6]. So, our question is, what is the optimal angle θ_0 and velocity V_0 for throwing a ball? Since the player never throws the ball in the same way, because we are not robots, the answer to the question that will be to find the trajectory (V_0, θ_0) with the highest error allowed. That is, we will find the angle and velocity which allows greater error with respect to V_0 and θ_0 and continue having a successful release.

To address the problem of finding an optimum release angle and velocity for the ball to enter the basket under ideal conditions, we are required to take several steps and make several assumptions in each of the cases of our model.

Assumptions:

1. Allow only trajectories that go straight to the ring.
2. Ignore air resistance.
3. Ignore any spin that the ball may have.
4. Assume the player always throws in a straight line, this makes the model bi-dimensional.
5. There is no error in velocity release.
6. The best trajectory is when the ball enter through the center of the ring.
7. The player is 2,16 to Shaq and 1.83 to Kev (Author) meters high.

Together with well established assumptions, there are several constants (see Figure 4) given to help finding and relating the release angle and release velocity:

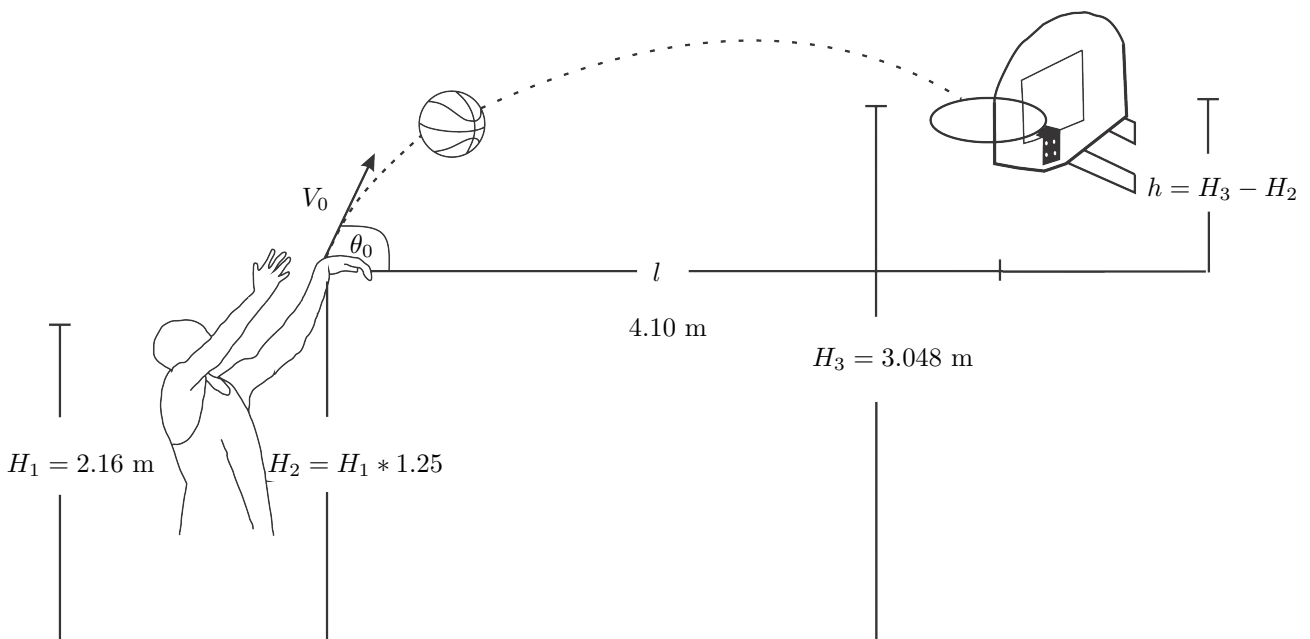


Figure 4: the mathematical variables and constants related in the free throw.

1. Rim diameter (Dr): 0.457 m.
2. Ball diameter (Db): 0.244 m.
3. The horizontal distance from the player position to the center of the ring (l): 4.10 m.
4. Height of player (H_1): 2.16 m.
5. Vertical position of center of the ball at release (H_2): $H_1 * 1.25$ m.
6. Vertical position of the ring (H_3): 3.048 m.
7. Vertical distance traversed (h): $H_3 - H_2$.
8. Acceleration due to gravity (g): $-9.81 \frac{m}{s^2}$.

The trajectory of the ball can be defined through the study of the following set of equations: let V_0 be the initial release velocity and θ_0 be the release angle.

Horizontal motion: given a time $t > 0$, the horizontal position of the ball at time t is

$$x(t) = V_0 \cos(\theta_0)t. \quad (8)$$

Vertical motion: assuming the origin is located at the release point of the ball (see Figure 4), the vertical position of the ball at time t is

$$y(t) = V_0 \sin(\theta_0)t + \frac{1}{2}gt^2. \quad (9)$$

We will develop the mathematical model, reviewing articles mainly from Gablonsky and Lang [7], Silverberg and his contributors [5, 6], and Barzykina [8]; verifying in detail their calculations and developing numerical simulations. Through the reading process, we will find out other criteria to improve the current results. In addition, we want to optimize some calculations and propose some other methodologies to accurate the optimal interval of free throw shootings. We will perform a numerical analysis in the MATLAB software to reproduce the results obtained in the literature and to compare among them. During the first part, we will work on a 2D model proposed by Gablonsky and Lang [7] in which we will study and provide more details to the used methodology. Then, we will find a proper range of angles for the ball reaching the ring without exceeding it and satisfying the model conditions. Also, we will propose new criteria to obtain the range that θ_0 can take for the “angles of the front of the ring” as they are named in the article. In addition, we will use the Barzykina [8] methodology to validate the model by Monte Carlo simulations with the multivariate normal distribution and obtain simulated data. After completing the 2D model, we will make a pilot experiment in a semiprofessional basketball team in Ecuador to obtain real data, to finally make a comparative analysis between simulated data and real data using a multiple regression model for basketball free throws. For future work we would like, to adapt the 2D strategy to a 3D model by removing some restrictions such as the ball going straight to the center of the ring, and allowing the ball hitting the board.

2.4.1 General Objective

To determine the optimal angle and velocity in free throw shooting in a pilot experiment on a semiprofessional basketball team in Ecuador.

2.4.2 Specific Objectives

1. To understand the mathematical aspects of a free throw shooting in basketball in two dimensions.
2. To find the optimal region of angles and velocities where the throws are successful.
3. To validate the theoretical model using Monte Carlo simulations.
4. To use MATLAB to determine the optimal solutions and to compare by means of graphics.
5. To use R software to make a comparative analysis between simulated data and real data using a multiple regression model for basketball free throw.

3 Mathematical and numerical analysis of the model

The modeling process typically begins with the construction of very simple cases which are easy to solve. Then, the cases are refined to make them more realistic, which in turn requires the introduction of more mathematics in order to solve them. Finally, the model should be refined enough to describe reality as closely as possible while still being solvable. We will see this refinement process in action as we go through the modeling procedure. In this section, we are going to discuss three cases for the development of our model of basketball free throws. For all three cases, we must take into account the assumptions and constants that were defined in section 2.4. Our model has excluded air resistance due to free throws are characterized by low velocity and short-time travel. Hamilton and Reinschmidt [4] performed a qualitative study of the inclusion of air resistance and proposed that any derived optimal release angles would be lower by approximately 2 degrees. For the first case, we will assume that the players keep consistent only their release velocity but not their release angle. This will be enough to find an angle θ_0 that allows throwing with higher or lower angles to this and still have a successful throw, that is to find the angle with the highest error allowed. For the second case, we divided it into two parts. In the first part, we will stop considering assumption 6, which now will allow the ball entering by any position of the ring, then we will find the optimal release angle in a similar way to the first case. In the second part, we will stop considering assumptions 6 and also 5, which means that we must find the error allowed for the release velocity since this will no longer be consistent. Finally, in the third case, we will find the best trajectory by making a relationship between the percentages of error allowed between velocity and the release angle through a multiobjective optimization with heuristic criterion.

3.1 First Case

When we see basketball players throw free throws, we notice that sometimes they make small mistakes in the throws and they are still successful, this is due to each player has an amount of error allowed to get successful throws. This amount of error will depend on the initial angle at which the ball was thrown and its release velocity.

3.1.1 Problem definition

For the first case of our model, we will consider that the player is not consistent in his throwing angle but is consistent with its release velocity; this means that the player will only have an error in his release angle. Therefore, we will begin by defining the problem in the following way: given the height of a certain basketball player, what is the best angle for the player to throw the free throw by having a fixed release velocity?

3.1.2 Mathematical Analysis.

Now, we consider a special case of two-dimensional motion also known as projectile motion. Since we ignore the air resistance, projectiles follow a curved trajectory shaping a parabola. So, our question is, what is the optimal angle θ_0 for shooting a ball? Moreover, we know that the optimal angle θ_0 for reaching the maximum horizontal distance, with a minimum velocity, is 45° . This is due to ground-to-ground projectile motion i.e. a projectile launched from the origin returns to the same horizontal level. In basketball, the shooting problem is that the beginning level is different from the final level. Therefore, the optimal angle in basketball shooting problem is different from the optimal angle in ground-to-ground projectile motion.

The trajectory of the ball can be defined by equations (8) and (9): let V_0 be the initial release velocity and θ_0 be the release angle. Then the horizontal V_H and vertical V_V components of the velocity are

$$\begin{aligned} V_H &= V_0 \cos(\theta_0), \\ V_V &= V_0 \sin(\theta_0). \end{aligned}$$

Horizontal motion: given a time $t > 0$, the horizontal position of the ball at time t is

$$\begin{aligned} x(t) &= V_H t, \\ x(t) &= V_0 \cos(\theta_0) t. \end{aligned} \tag{10}$$

Vertical motion: assuming the origin is located at the release point of the ball (see Figure 4), the vertical position of the ball at time t is

$$\begin{aligned} y(t) &= V_V t + \frac{1}{2}gt^2, \\ y(t) &= V_0 \sin(\theta_0)t + \frac{1}{2}gt^2, \end{aligned} \quad (11)$$

where t is such that $0 \leq t \leq T$ and T is the time for the ball reaching the center of the ring. Note that we are considering the gravity, g , to be negative. Evaluating at $t = T$

$$x(T) = l = V_0 \cos(\theta_0)T, \quad (12)$$

$$y(T) = h = V_0 \sin(\theta_0)T + \frac{1}{2}gT^2. \quad (13)$$

By, (12) we have

$$T = \frac{l}{V_0 \cos(\theta_0)}. \quad (14)$$

In order to find the initial velocity V_0 for the ball goes through the center of the ring, given an initial angle, and an explicit relation between V_0 and θ_0 , we replace (14) in (13) to get

$$\begin{aligned} h &= \frac{V_0 \sin(\theta_0)l}{V_0 \cos(\theta_0)} + \frac{1}{2} \frac{gl^2}{V_0^2 \cos^2(\theta_0)} \\ &= l \cdot \tan(\theta_0) + \frac{1}{2} \frac{gl^2}{V_0^2 \cos^2(\theta_0)}. \end{aligned}$$

We solve for V_0 ,

$$\begin{aligned} h - \tan(\theta) \cdot l &= \frac{1}{2} \frac{gl^2}{V_0^2 \cos^2(\theta_0)}, \\ \frac{g \cdot l^2}{2(h - l \cdot \tan(\theta_0))} &= V_0^2 \cos^2(\theta_0), \\ V_0 &= \frac{l}{\cos(\theta_0)} \sqrt{\frac{-g}{2(l \cdot \tan(\theta_0) - h)}}. \end{aligned} \quad (15)$$

We note that (15) is real valued for $l \cdot \tan(\theta_0) - h > 0$. Therefore, the release angle, θ_0 , is such that $\tan^{-1}(\frac{h}{l}) < \theta_0 < 90^\circ$.

We stress the right choice of θ_0 so that the ball at least reaches and also does not pass the ring. This will be important for the numerical methods used to find solutions later in this work. Then, to determine the adequate range of angles, we need to study the methods to obtain the maximum height and maximum horizontal distance as follow.

For the maximum height of the ball, first let us find the time when the ball reaches its maximum height. Let V_0 be a given velocity such that there exists θ_α with vertical motion (11) at time t . Then,

$$y(t) = V_0 \sin(\theta_\alpha)t + \frac{1}{2}gt^2. \quad (16)$$

Since, the velocity is the first derivative of the position,

$$y'(t) = V_0 \sin(\theta_\alpha) + gt.$$

If $t = t_{max}$, the time when the ball reaches its maximum height before descending, then $y'(t) = 0$. Namely,

$$t_{max} = -\frac{V_0 \sin(\theta_\alpha)}{g}. \quad (17)$$

Since $y(t_{max})$ is the maximum height position of the ball, by replacing (17) into (16)

$$\begin{aligned} y(t_{max}) &= V_0 \sin(\theta_\alpha) \left(-\frac{V_0 \sin(\theta_\alpha)}{g} \right) + \frac{g}{2} \left(\frac{V_0^2 \sin^2(\theta_\alpha)}{g^2} \right), \\ &= -\frac{2V_0^2 \sin^2(\theta_\alpha)}{2g} + \frac{V_0^2 \sin^2(\theta_\alpha)}{2g} \\ &= \frac{-V_0^2 \sin^2(\theta_\alpha)}{2g}. \end{aligned} \quad (18)$$

Finally, we will discard the angles such that the maximum height (18) is below h , because for those angles the ring will not be reached. That is,

$$-\frac{V_0^2 \sin^2(\theta_\alpha)}{2g} < h.$$

Since $g < 0$:

$$\begin{aligned} \sin(\theta_\alpha) &< \sqrt{-\frac{2gh}{V_0^2}} \\ \theta_\alpha &< \arcsin \sqrt{-\frac{2gh}{V_0^2}}, \\ \theta_\alpha &< \arcsin \sqrt{-\frac{2gh}{V_0^2}}. \end{aligned} \quad (19)$$

In the same way, to find the maximum horizontal distance, we consider the corresponding time t_{max} , by making $y = 0$ in (11). Let V_0 be a given velocity such that there exists θ_β with vertical motion (11) at time t . Then,

$$\begin{aligned} 0 &= V_0 \sin(\theta_\beta)t + \frac{1}{2}gt^2 \\ &= t(V_0 \sin(\theta_\beta) + \frac{1}{2}gt), \end{aligned}$$

where we discard the trivial case $t=0$. Then $t = -\frac{2V_0 \sin(\theta_\beta)}{g}$, and we get

$$t_{max} = -\frac{2V_0 \sin(\theta_\beta)}{g}. \quad (20)$$

Since $x(t_{max})$ is the maximum horizontal distance of the ball, by replacing (20) into (10)

$$\begin{aligned} x_{max} &= V_0 \cos(\theta_\beta) \left(-\frac{2V_0 \sin(\theta_\beta)}{g} \right) \\ &= -\frac{2V_0^2 \cos(\theta_\beta) \sin(\theta_\beta)}{g}. \end{aligned} \quad (21)$$

Now, we are going to discard the angles such that the maximum horizontal distance (21) is below l , because with those angles the position of the ring will not be reached.

$$-\frac{V_0^2 \sin(2\theta_\beta)}{g} < l.$$

Since $g < 0$:

$$\begin{aligned}\sin(2\theta_\beta) &< \frac{l \cdot g}{V_0^2} \\ \theta_\beta &< \frac{\arcsin\left[-\frac{l \cdot g}{V_0^2}\right]}{2} \\ \theta_\beta &< \frac{\arcsin\left[-\frac{l \cdot g}{V_0^2}\right]}{2}.\end{aligned}\quad (22)$$

In this case we have two solutions due to the arcsin function:

$$\theta_\beta < 90^\circ - \frac{\arcsin\left[-\frac{l \cdot g}{V_0^2}\right]}{2} \quad \text{or} \quad \theta_\beta < \frac{\arcsin\left[-\frac{l \cdot g}{V_0^2}\right]}{2}.$$

We consider the solution $\theta_\beta < 90^\circ - \frac{\arcsin\left[-\frac{l \cdot g}{V_0^2}\right]}{2}$, because it takes values greater or equal than 0. Therefore, θ_0 is such that

$$\theta_0 \in \left(\arcsin\sqrt{\frac{-2gh}{V_0^2}}, 90^\circ - \frac{\arcsin\left[-\frac{l \cdot g}{V_0^2}\right]}{2} \right). \quad (23)$$

3.1.3 Derivation of an equation for the new horizontal position of the ball as it comes back down to the basket height

From the modeled equations of motion, we will now find the equations that allow us to find the amount of error allowed for an initial angle θ_0 (where the ball passes through the center of the ring) and still have a successful shoot. We know that the release angle could have an error and still the ball enter the ring.

Keeping the initial velocity fixed (the player has a consistent release velocity V_0), allowing the release angle to vary (the player has error in his release angle θ_0^*) and replacing l by x in (14), we have that

$$\begin{aligned}h &= V_0 \sin(\theta_0)T + \frac{1}{2}gT^2 \\ &= \frac{V_0 \sin(\theta_0)x}{V_0 \cos(\theta_0)} + \frac{1}{2} \frac{gx^2}{V_0^2 \cos^2(\theta_0)} \\ &= \tan(\theta_0)x + \frac{1}{2} \frac{gx^2}{V_0^2 \cos^2(\theta_0)},\end{aligned}$$

which implies

$$\frac{g}{V_0^2 \cos^2(\theta_0)}x^2 + x \tan(\theta_0) - h = 0. \quad (24)$$

Solvin for x ,

$$\begin{aligned}x &= \frac{-\tan(\theta_0) \pm \sqrt{\tan^2(\theta_0) - \frac{4g(-h)}{2V_0^2 \cos^2(\theta_0)}}}{\frac{2g}{2V_0^2 \cos^2(\theta_0)}}, \\ &= \frac{-V_0^2 \tan(\theta_0) \cos^2(\theta_0)}{g} \pm \frac{V_0^2 \cos^2(\theta_0) \sqrt{\tan^2(\theta_0) + \frac{2gh}{V_0^2 \cos^2(\theta_0)}}}{g} \\ &= \frac{-V_0^2 \sin(\theta_0) \cos(\theta_0)}{g} \pm \frac{V_0 \sqrt{V_0^2 \sin^2(\theta_0) \cos^2(\theta_0) + 2gh \cos^2(\theta_0)}}{g} \\ &= \frac{-V_0^2 \sin(\theta_0) \cos(\theta_0)}{g} \pm \frac{V_0 \cos(\theta_0) \sqrt{V_0^2 \sin^2(\theta_0) + 2gh}}{g}\end{aligned}$$

we take the sign $-$, because the gravity g is negative, and we want the greater distance x ,

$$x = \frac{V_0 \cos(\theta_0^*)}{-g} \left(V_0 \sin(\theta_0^*) + \sqrt{V_0^2 \sin^2(\theta_0^*) + 2gh} \right). \tag{25}$$

In the equation (25), θ_0^* corresponds to a higher or lower release angle due to player error than the ideal initial angle θ_0 where the ball passes through the center of the ring (see Figure 5).

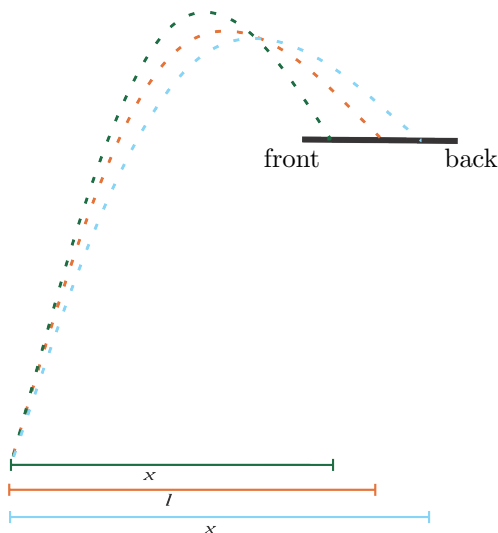


Figure 5: comparison of the trajectory l (center of the ring), with an x trajectory with error in the release angle (V_0, θ_0^*) .

3.1.4 Derivation of the conditions so that the ball does not hit the front or the back of the ring

We now want to give conditions in order to the ball does not hit the front or the back of the ring:

- From (25), we can conclude that $x + Db/2$ is the horizontal distance to the rightmost part of the ball when the center of the ball is at the level of the basket. Also, $l + Dr/2$ is the horizontal distance to the back of the rim. Therefore the criterion for the ball hit the back of the rim and enter as the center of the ball passes through the basket is:

$$x \leq l + Dr/2 - Db/2. \tag{26}$$

- We are going to study the release angle in order to the ball enter just through the front of the ring.

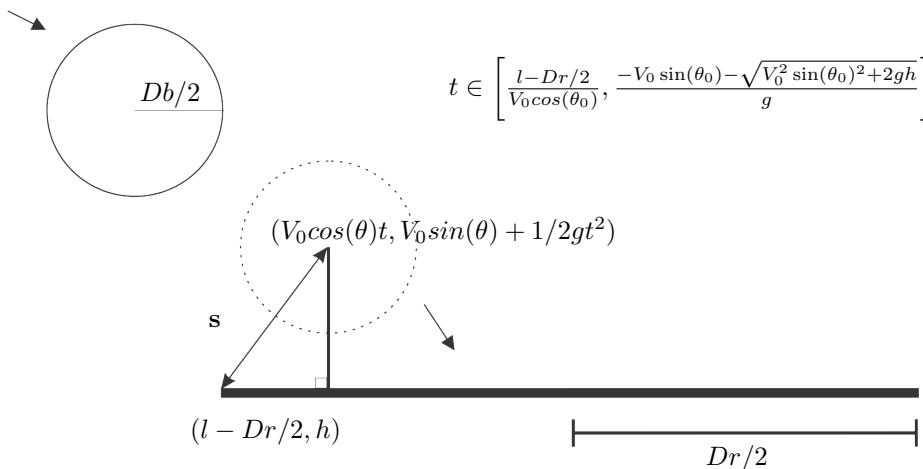


Figure 6: the distance s between the front of the ring and the center of the ball.

To avoid contact with the front of the rim, the distance s between the rim and the center of the ball must remain greater than the radius of the ball throughout its trajectory, i.e., for all times t such that $0 < t < T$ where T is the time when the center of the ball passes through the ring; (see Figure 6). We are looking for a t , such that

$$t \in \left[\frac{l - Dr/2}{V_0 \cos(\theta_0)}, \frac{-V_0 \sin(\theta_0) - \sqrt{V_0^2 \sin^2(\theta_0) + 2gh}}{g} \right]. \tag{27}$$

In Gablonsky and Lang [7] there is an error about the time suggested to find the angles for the front of the ring. Using for convenience the square distance to find the distance s , we have the following criterion for the ball not hitting the front of the ring:

$$s^2 = \left(x(t) - \left(l - \frac{Dr}{2} \right) \right)^2 + (y(t) - h)^2 > \left(\frac{Db}{2} \right)^2, \tag{28}$$

where $x(t)$ and $y(t)$ are given by (10) and (11) respectively.

$$\tag{29}$$

- The following is our own proposal to find the lowest release angle. Let m be a fixed real number. Then, it is possible to find two points on the circumference such that the slope of the tangent line at those two points is equal to m . Let P_0 one of those two points and D the corresponding direction of the tangent line. Since P_0 and D are perpendicular (see Figure 8), the inner product is 0,

$$\begin{pmatrix} 1 \\ m \end{pmatrix} \cdot \begin{pmatrix} r \cos(\theta) \\ r \sin(\theta) \end{pmatrix} = 0.$$

Namely

$$\cos(\theta) + m \sin(\theta) = 0.$$

Solving for θ

$$\theta_p = \operatorname{arccot}(-m) + \pi.$$

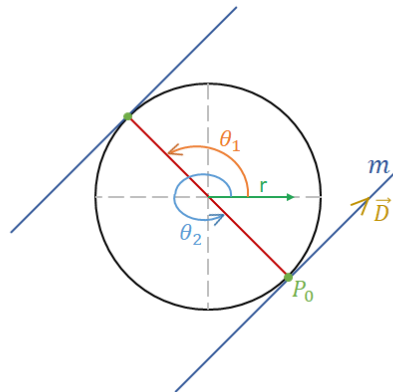


Figure 7: two tangent lines on the circumference having a given slope m .

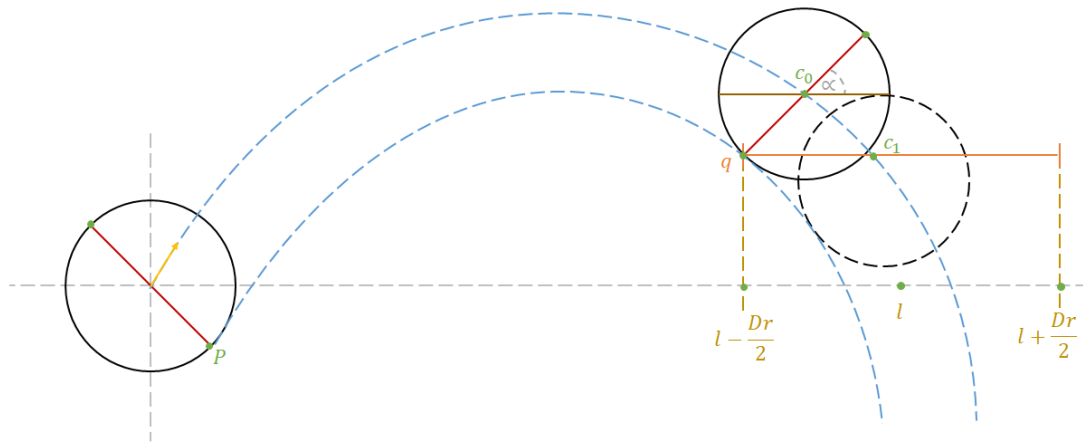


Figure 8: parabolas describing the ball approaching into the ring.

In order to find the slope, we find the velocity through the first derivative of the distance $(x(t), y(t))$:

$$\begin{aligned} \begin{pmatrix} x'(t) \\ y'(t) \end{pmatrix} &= \begin{pmatrix} V_0 \cos(\theta_0 t) \\ V_0 \sin(\theta_0 t) + \frac{1}{2}gt^2 \end{pmatrix}, \\ \begin{pmatrix} x'(t) \\ y'(t) \end{pmatrix} &= \begin{pmatrix} V_0 \cos(\theta_0) \\ V_0 \sin(\theta_0) + gt \end{pmatrix}, \\ m(t) &= \frac{y'(t)}{x'(t)} = \frac{V_0 \sin(\theta_0) + gt}{V_0 \cos(\theta_0)}. \end{aligned}$$

Evaluating at $t=0$

$$m(0) = \tan(\theta_0),$$

so the release angle is such that

$$-\frac{\pi}{2} < \theta_P = \text{arccot}(-\tan(\theta_0)) < 0.$$

Then P is given by

$$P = \begin{pmatrix} \frac{D_b}{2} \cos(\theta_P) \\ -\frac{D_b}{2} \sin(\theta_P) \end{pmatrix},$$

and if q_0 is the left extreme point of the ring (which is known), we have

$$q_0 = \begin{pmatrix} q_{0_x} \\ q_{0_y} \end{pmatrix} = \begin{pmatrix} l - \frac{D_r}{2} - P_{0_x} \\ h \end{pmatrix}.$$

Then the time to reach to q is

$$t_1 = \frac{l - \frac{D_r}{2} - \frac{D_b}{2} \cos(\theta_0)}{V_0 \cos(\theta_0)}.$$

On the other hand, by (11), the time to get the ring is given by

$$t_2 = \frac{-V_0 \sin(\theta_0) - \sqrt{V_0^2 \sin^2(\theta_0) + 2gh}}{g}.$$

Now, the coordinates of the frontal part of the ring are (see Figure 7)

$$q = \left(l - \frac{Dr}{2}, h \right),$$

which implies (see Figure 7)

$$\alpha = \arctan\left(\frac{h}{l - \frac{Dr}{2}}\right).$$

Since we know t_1, t_2 , we can compute

$$\begin{aligned} x(t_1) &= V_0 \cos(\theta_0) t_1, \\ x(t_2) &= V_0 \cos(\theta_0) t_2, \end{aligned}$$

and then

$$x(t_2 - t_1) = V_0 \cos(\theta_0)(t_2 - t_1).$$

Thus, we obtain a second criterion to ensure the ball does not hit the front of the rim:

$$l - \frac{Dr}{2} + \frac{Db}{2} \cos(\alpha) + V_0 \cos(\theta_0)(t_2 - t_1) < x(t_2). \quad (30)$$

3.1.5 Derivation of the error allowed function for a given initial angle θ_0 to still have a successful throw

In order to find the error allowed for a release angle to center of the ring θ_0 , we set V_0 and find the angles θ through the equation (15), which allows the trajectory (V_0, θ_0) to pass through the center of the ring. Then, we calculate release angles higher $\theta_{high} > \theta_0$ and release angles lower $\theta_{low} < \theta_0$ with the following equations respectively:

$$x - l + \frac{Db - Dr}{2} = 0. \quad (31)$$

$$s^2 - \left(\frac{Db}{2}\right)^2 = 0. \quad (32)$$

Now, every (V_0, θ_{high}) or (V_0, θ_{low}) must comply with the conditions (26) and (28) to satisfy with a successful throw. After, we find the minimum distance from θ_0 which will denote as the error allowed in release angle

$$e(\theta_0) = \min\{\theta_{high} - \theta_0, \theta_0 - \theta_{low}\}. \quad (33)$$

Finally, to obtain the best release angle, we need to maximize (33). In order to find the maximum of a function, we know that the tangent in any maximum will have slope 0, so we look for points where $f'(x) = 0$. To check that this is really a maximum, and not a minimum or an inflection point, we could take the second derivative and confirm that it is negative at the point x . Moreover, the requirement for f to be continuous and differentiable is important. In this case, we have a non-differential function at the maximum; due to the left-hand slope is not equal to the right-hand slope, (see Figure 9). This can be explained by recognizing that (33) contains the min function, which implies nondifferentiability. Then, to find the maximum we use a computer algebra system optimization routine with the help of MATLAB.

3.1.6 Presentation of the results.

To show the results obtained in each of the proposed cases, we will consider a comparison study between Shaq (2.16 m) and myself Kev (1.83 m). The MATLAB codes will employ function **initialize** see appendix A.1, to declare all the constants and variables used in the model. Now, to get the best angle θ_0 for our first case, we design an algorithm to find the maximum of the function (33) see Appendix A.3:

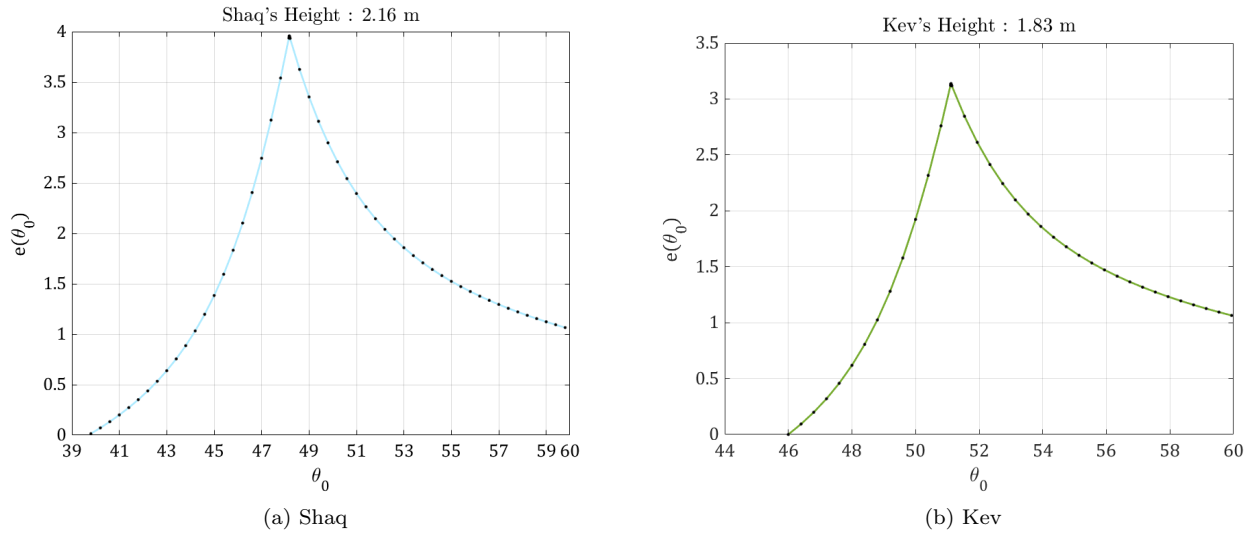


Figure 9: the error about θ_0 for which the ball still goes in.

NAME	Shaq	Kev
Height	2.16 m	1.83 m
θ_{center}	48.18°	51.11°
V_{center}	6.62 m/s	6.96 m/s

Table 3: the Best Angle for Shaq and Kev.

Interpreting the model, we determine the best shooting angle for Shaq and Kev during their free throws. That is, assuming that Shaq and Kev are able to consistently control their release velocity, and bearing in mind that the best free throw is when the ball passes right through the center of the ring. If we analyze these assumptions, they don't seem to be realistic. The player is likely to make mistakes not only in his release angle but also in his velocity, and not necessarily the ball must enter the center of the ring to have the best throw. We made the assumptions in this model in order to make calculations easier. Therefore, would these be the best angles for Shaq and Kev? Probably not, especially if they have trouble to release the ball with a consistent velocity. Furthermore, with these results, it can be seen in Figure 9 that the error allowed of the angle to keep having a successful throw for Shaq is greater than Kev so the release angle θ_0 may be related to the height of the player.

3.2 Second Case.

When modeling, it is normal to make some assumptions, as we did in the first case, that make solutions easier to find. The more assumptions the model has, the less accurate it becomes. Since we found a solution with the first case, it is usual to try to remove as many conditions as possible, usually one at a time, to obtain solutions that better simulate reality. Following the interpretation of the solution of our first case, the first assumption that we will remove is the condition that the ball must pass through the center of the hoop, assumption (6) in Section 2.4. By doing so we will have to rederive a more accurate case. The cyclic process of refining and rederiving is standard in real-life modeling [7].

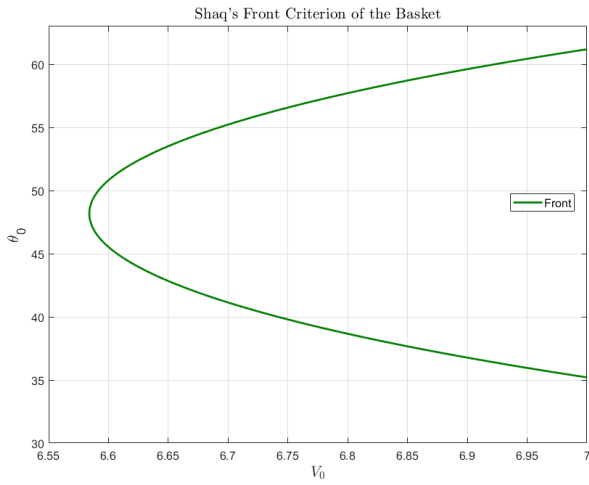
3.2.1 Problem definition.

In this second case, we improve our model by removing the assumption that the best throw is one where the ball goes through the center of the basket. Still keeping the same equations of two-dimensional motion, assuming that the players have a consistent release velocity and the height of our players Shaq and Kev. Now, we are going to vary independently the initial velocity V_0 and the initial angle θ_0 at the same time. Each pair (V_0, θ_0) will give the ball a trajectory that results in a successful throw or failed throw. By independently varying V_0 and θ_0 , we will construct a feasible region of the trajectories, which is the set of all possible pairs (V_0, θ_0) that result in a successful free throw using assumptions on allowable trajectories (26) and (28). Then it will be maximized the error function allowed at the thrown angle (33) in the entire feasible region, not only for the θ_0 angles that go to the center of the ring.

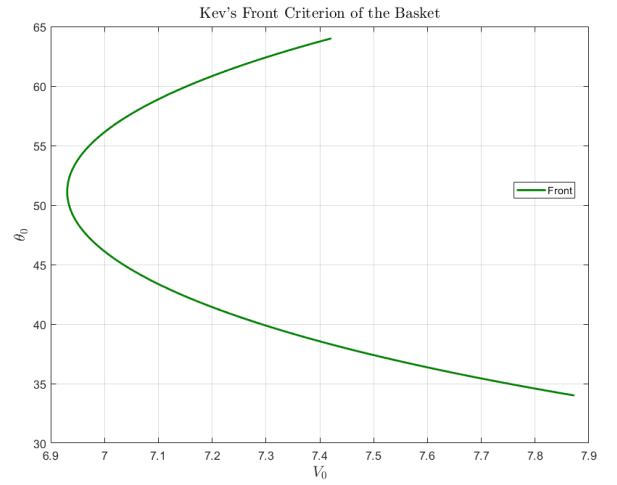
3.2.2 Numerical Analysis and presentations of the results.

In order to construct a feasible region of our second model, we need to consider the solutions for the ball enter into the ring following the boundary conditions (31), (32) and the center of the ring condition (15). This boundaries can be found by writing a program in MATLAB that find the minimum of a uni-variate function. Our program is *feasible_region*, see Appendix B.4, which is a program to find the success region of throws from Shaq or Kev. For this program we need to find the angles and velocities when the ball just skims the front of the rim, center of the ring and back of the rim. For the front of the rim condition (32), its corresponding MATLAB code can be seen in Appendix B.3. For the back and for the center conditions we wrote a MATLAB program solving (31), see Appendix B.1, and (15), see Appendix B.2.

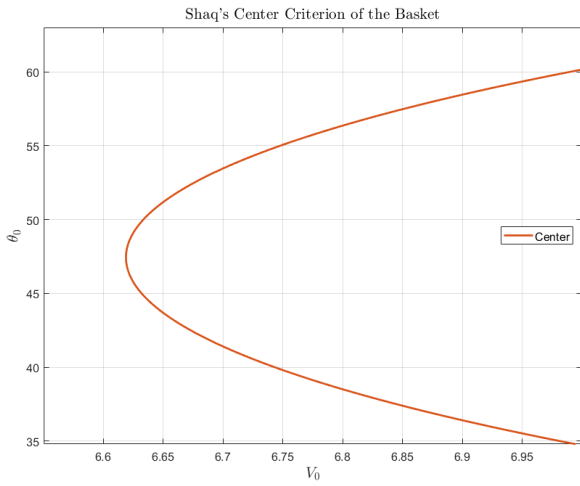
The green boundary corresponds to solution of (32), the ball skims the front of the rim as it goes in. The orange boundary corresponds to solutions of (15), the ball passing through the center of the ring. The light blue boundary corresponds to solution of (31), the ball hitting the back of the rim as it goes in, see Figure 10.



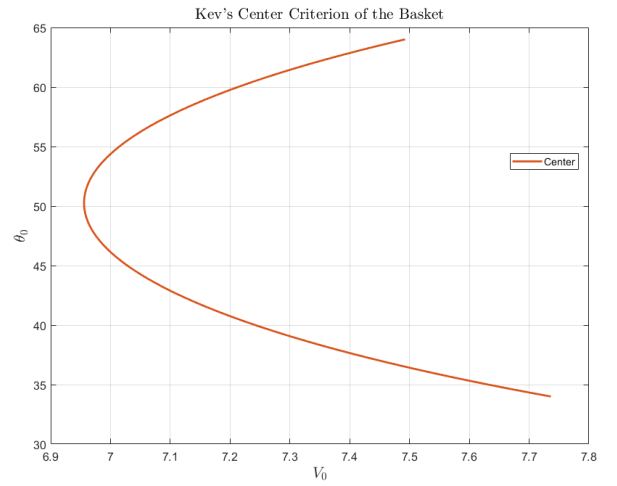
(a) Shaq



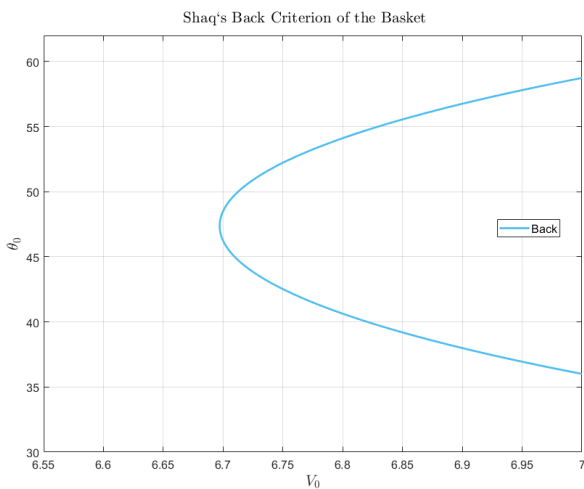
(b) Kev



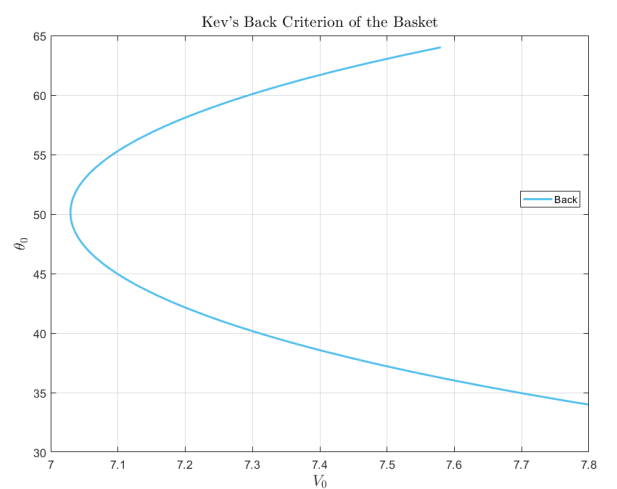
(c) Shaq



(d) Kev



(e) Shaq



(f) Kev

Figure 10: boundaries of the front, center, and back when the ball enters skimming these parts of the ring.

Now, with the defined ring boundaries, we can find the feasible region for Shaq and Kev.

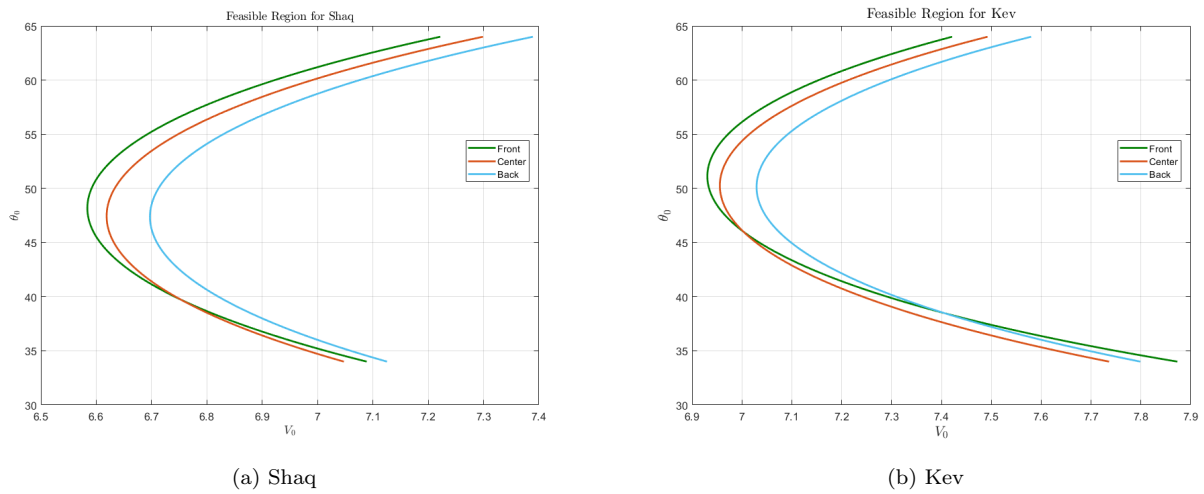


Figure 11: feasible range of angles and velocities that result in a successful free throw for a ball release height of Shaq and Kev.

From Figure 11 it is noticed that the success area of Shaq is bigger than that of Kev’s. Mathematically, this shows us that the higher the player is, there is a greater area of success, which would mean that Shaq should have a larger percentage of successful throws. In real life, this is not happening since as we mentioned before, Shaq has a 53% of success which is a very low percentage. So, what is happening with Shaq throws to not been successfully? How to suggest what angle and velocity should throw to increase its success rate? These questions will be answered when we study the error allowed for the angle and velocity of throwing throughout the feasible region.

Now, locating in the feasible region the previous solution of the first case, see Table 3, the \times in the Figure 12, we notice that the requirement to reach the center of the ring would not be necessarily the best for Shaq and Kev. Since there is much more room to overshoot than to undershoot which would imply that this trajectory (θ_0, V_0) would be useful for the player when he missed, missed by overshooting, but for someone who missed most undershoots is definitely not the best.

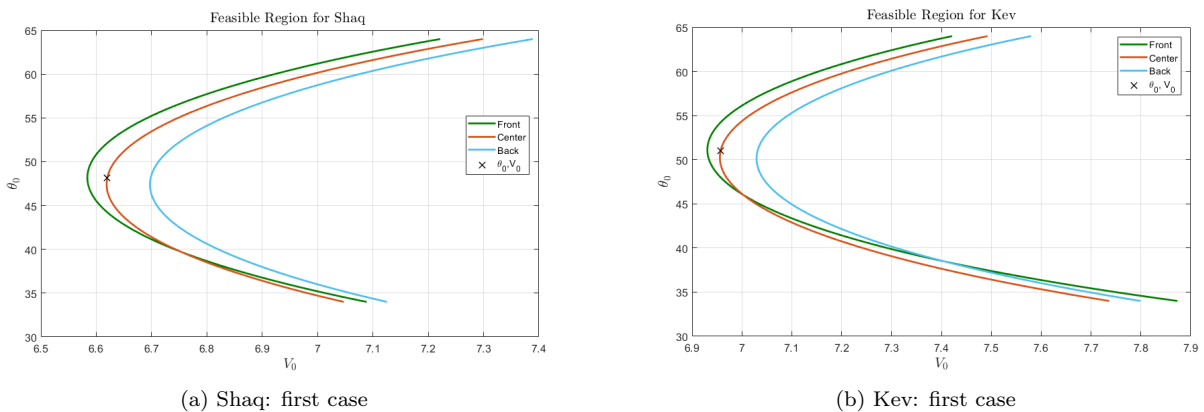


Figure 12: feasible range of angles and velocities that result in a successful free throw for a ball release height of Shaq and Kev.

3.2.3 Constructing the error and percent error in release angle and release velocity.

As we have already noted that the shoot by the center of the ring is not the best option, we let the ball go through the hoop at any position and maximize again to find the new allowed error in the initial angle. To create the contour figures see Figure 13, we need to calculate the allowed error in the release angle and release velocity for a grid 400×400 points. So, to find the error allowed in the release angle, we need to fix a vector of velocities and then find the respective angle from the boundary to the boundary of the feasible region which satisfies each velocity to obtain a successful shoot. Finally, we find the minimum deviation from each angle in the same way that in the first case in the function (33), see Appendix B.5.

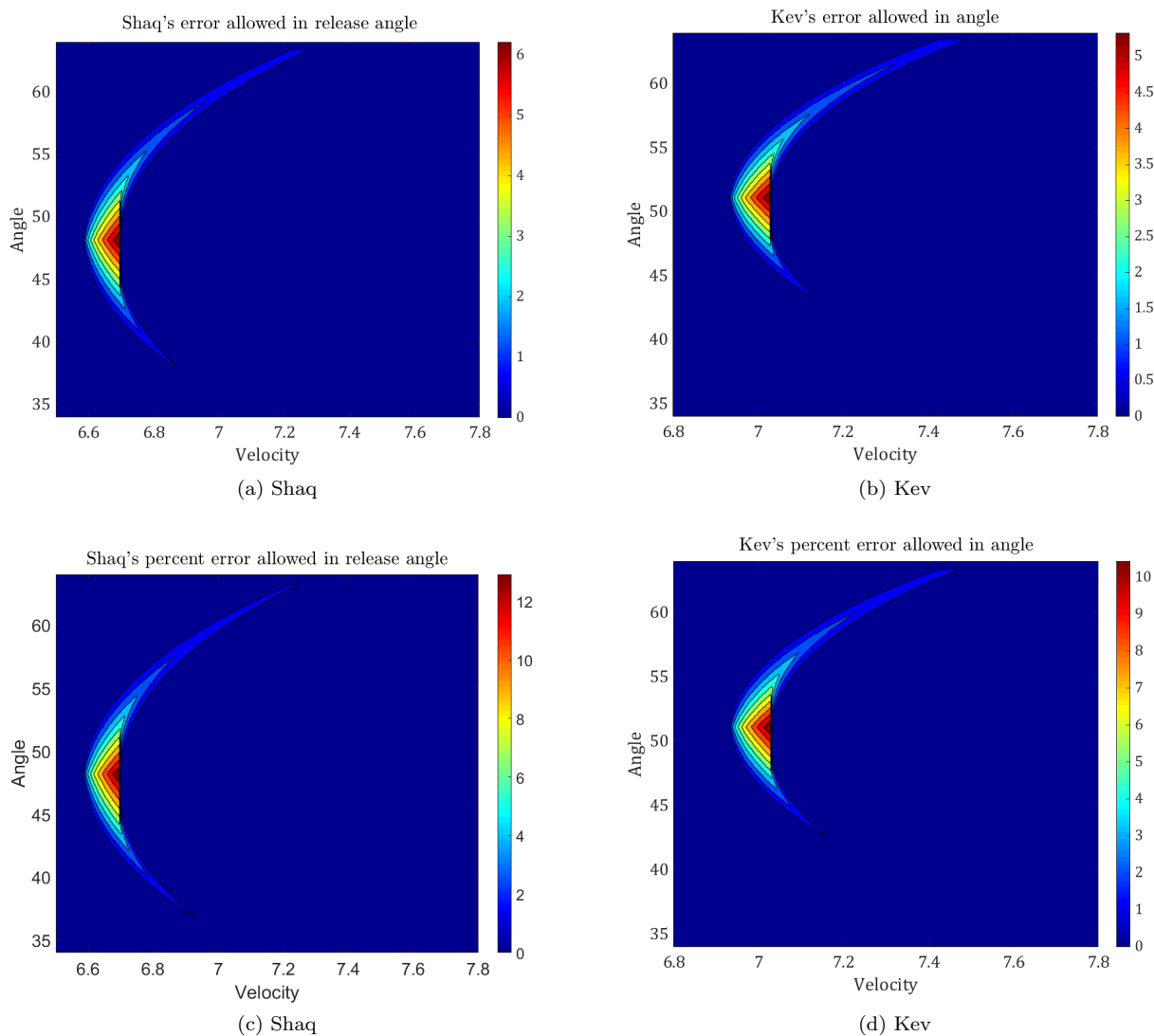


Figure 13: feasible range of angles and velocities that result in a successful free throw for a ball release height of Shaq and Kev.

From Figure 13 we can notice that the error allowed for a successful release angle for Shaq 13a is greater than the error allowed for Kev 13b. This means that the higher the player is, the more mistakes in his release angle can make and still has a high probability that the release being successful. This allows us to affirm what was said in the first case “the higher the player, the higher the percentage of a successful throw”.

Once the allowed errors for each release angle have been calculated for it to be successful, we need to maximize again the feasible region, which is a univariate not differentiable function in order to see what is the

best angle and the optimal release velocity for the second case. For this, we are going to design a program using algebra system’s optimization routine (see Appendix B.6) to obtain the point marked with a dot in Figure 14.

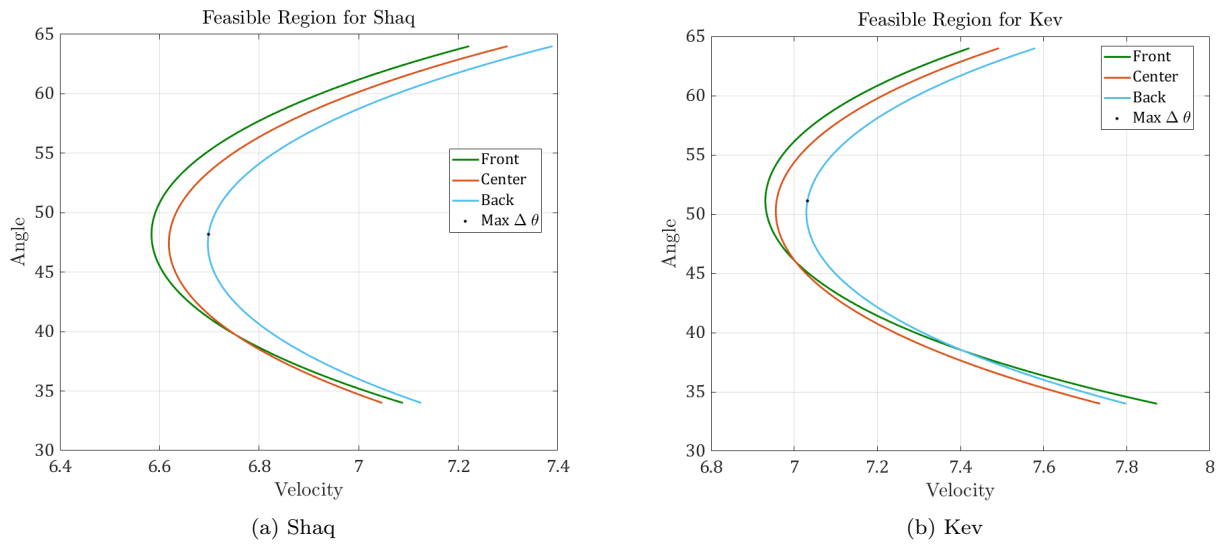


Figure 14: the feasible region for our second case with location of suggested optimal trajectory for Shaq and Kev.

NAME	Shaq	Kev
Height	2.16 m	1.83 m
θ_{center}	48.19°	51.12°
V_{center}	6.69 m/s	7.03 m/s

Table 4: optimal release angle and velocity from Shaq’s and Kev’s second case.

From Figure 13 and Figure 14, the solution of Shaq and Kev is closely symmetric around the optimal release angle of the first case, located in the farthest red area. So if we do not require the ball to go through the center of the ring, any trajectories that is made with a velocity between about 6.60 m/s and 6.70 m/s for Shaq and between about 6.93 m/s and 7.1 m/s will lie in the farthest red region that allows the maximum error in the release angle.

Now, to find the allowed error in the velocity, we create a program (see Appendix B.7) where we need to fix an interval of angles and then find the respective velocity that satisfies each angle from boundary to boundary which allows us to obtain a success throw, then we find the minimum deviation from each velocity which satisfies the condition to be a success throw.

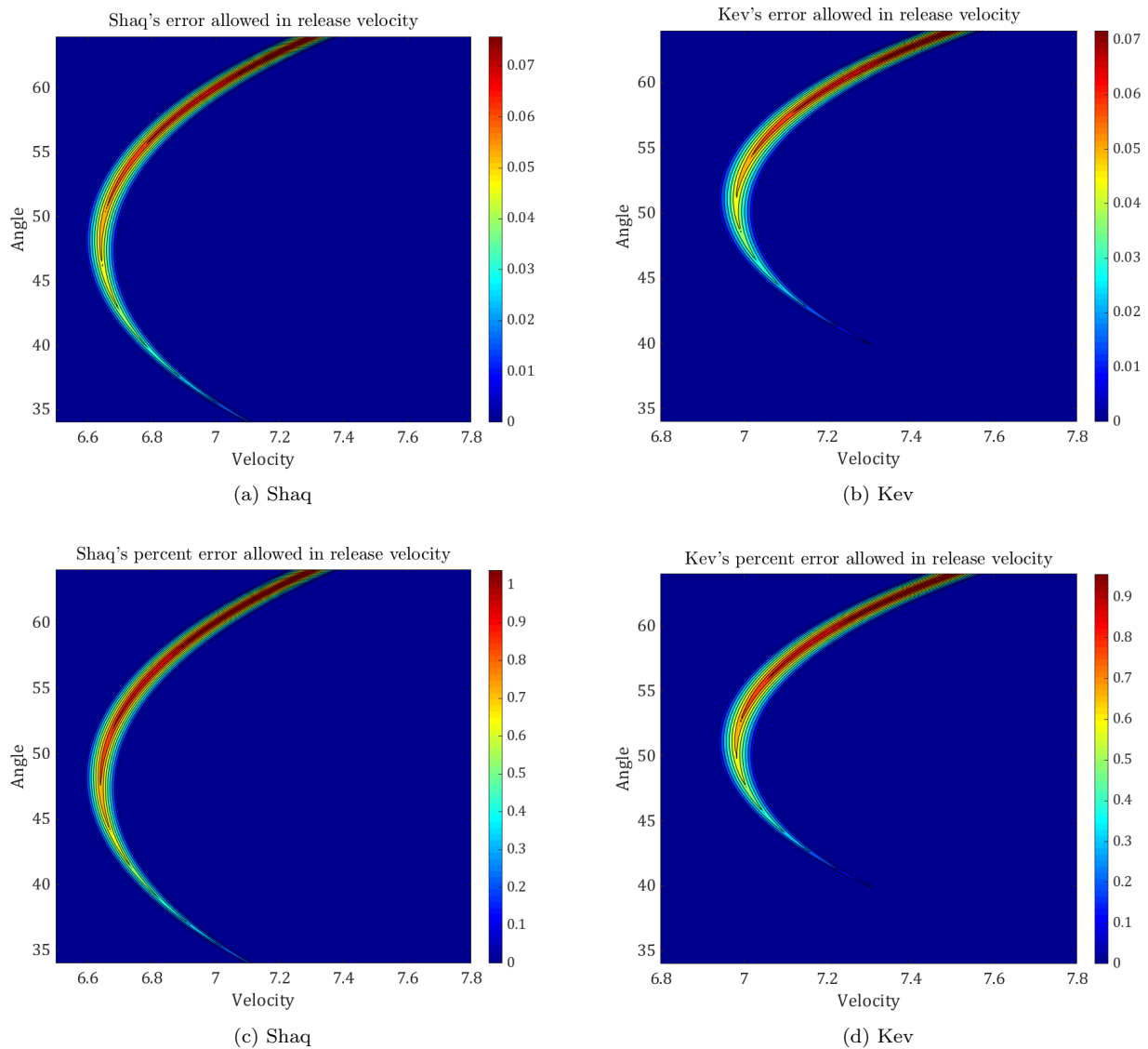


Figure 15: the feasible region of angles and velocities that result in a successful free throw for a ball release height of Shaq and Kev.

Figure 15 shows the maximum allowable error and the percentage of error allowed in the release velocity. It is clear that these graphics are not symmetrical and seems to favor larger launch angles, and we note again that Shaq has a higher percentage of error allowed in the release velocity than Kev’s but this time their differences are very similar. In addition, we can observe that the error allowed in the release angle are greater than the error in the release velocity. Therefore, we conclude that it is more important to use the right velocity when compare to the right angle [7].

3.3 Third case

Summarizing so far, in our model we have studied the release angle error that goes toward the center of the ring and at any position of the ring in the feasible region. Up to this point, the two cases have excluded the error in release velocity to facilitate computations. Besides, it is known that we are going to make errors during the release for both the velocity and the angle, so it was also found the error and percentage of error allowed for the release velocity.

To answer the question of what is the best free throw, we really need to consider both the release angle and

release velocity simultaneously to make it more realistic. This is because the trajectory that maximizes the allowed error in the release angle is also the trajectory that allows no error in the release velocity. Similarly, the trajectory that maximizes the allowed error in the release velocity is also the trajectory that allows no error in the release angle. So we were right to be unsatisfied with the optimal angle θ_0 and velocity V_0 from the first case, especially if our player makes errors in his initial velocity as well as in his initial angle when throwing. Therefore, to be more accurate in the analysis, we use numerical methods to construct regions of percent error in both angle and velocity.

3.3.1 Problem definition

In this third case, we will study how to optimize the allowed error of release angle and release velocity simultaneously in order to find our best release angle and velocity while obtaining a large percentage of success in each free throw. How do we find the optimal solution when we have two different measures that we want to minimize, and the two of them oppose each other? Problems of this type are called multiobjective optimization problems (see section 2.1). To solve the multiobjective problem we consider the weights for each objective and optimize a weighted combination of objectives using the methodology of Gablonsky and Lang [7].

3.3.2 Numerical Analysis and presentations of the results

To solve the weighted error between the angle and the release velocity we will combine the two weighted objectives with a heuristic argument that will be made by taking the minimum of the percent error in angle plus five times the percent error in velocity. This means that we are going to focus five times more in the release velocity than the release angle, this is because there is less error allowed in the release velocity than the angle, as we saw in Figures 13 and 15. Also, this argument may solve why Shaq has not a better chance of having successful free throws. We can see the code in Appendix C.1.

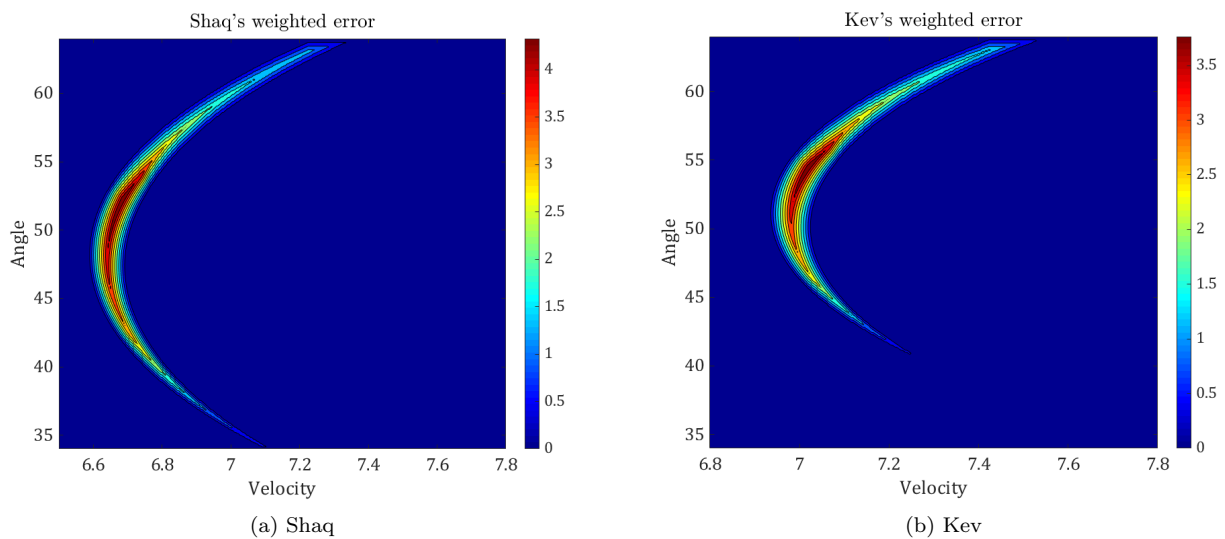


Figure 16: the weighted error for Shaq and Kev.

To find the optimum of the weighted error function for Shaq and Kev (see Figure 16) we note that this combined function is not differentiable, what leads us to apply algebra system's optimization routine again for the multivariate case.

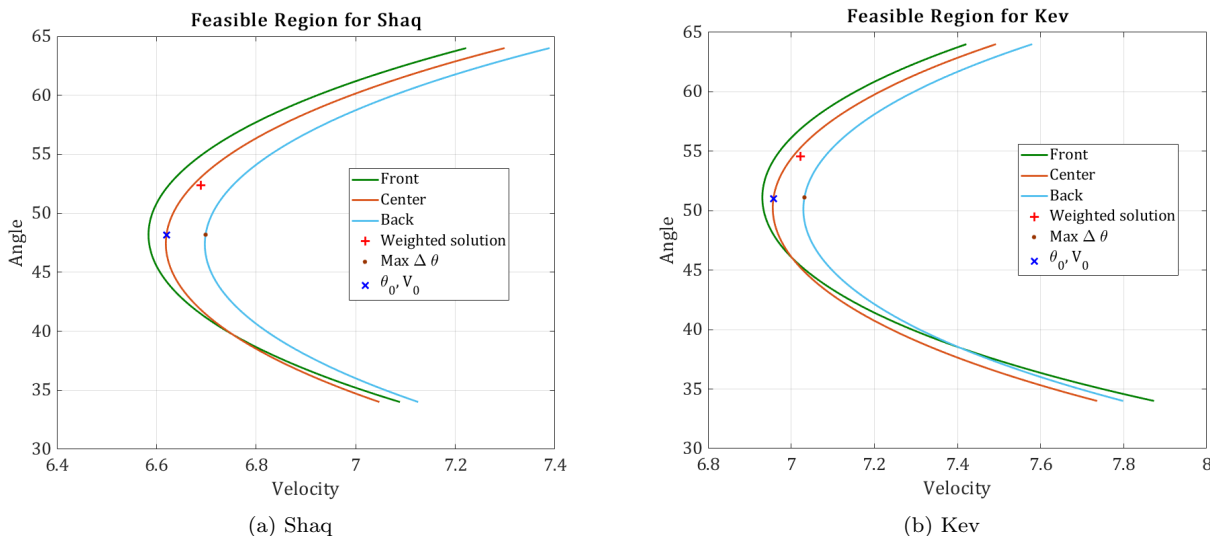


Figure 17: the weighted error for Shaq and Kev.

The optimum angle and velocity for this case are marked in Figure 17 with a red +. Formulating the combined objective function for a multiobjective optimization allows us to control its behavior. For example, if a player has variation in their velocity in each throw, a greater weight can be placed in the velocity than in the angle. By minimizing the percentage of error in the angle and five times the percentage of error in the velocity, we find an optimum angle and velocity of release for Shaq and Kev (see Table 5)

NAME	Shaq	Kev
Height	2.16 m	1.83 m
θ_{center}	52.37°	54.59°
V_{center}	6.69 m/s	7.02 m/s

Table 5: optimal release angle and velocity from Shaq’s and Kev’s to obtain a higher probability of successful throws.

We notice that both the release angle and the release velocity are important to accurately control the success of our free throw. Moreover, it is clear that the optimal trajectory should be decided player by player according to whether the player consistently has more trouble controlling his initial velocity or his initial angle. For example, if the player is worried about the consistency of his release angle than the consistency of his release velocity, then less weight should be placed on error in velocity as compared to error in angle. Therefore, we emphasize that the previous result of the third case is for us the best of the previous cases for Shaq, Kev and for any player of whom it is known, his height, release angle consistency and release velocity consistency.

To finish the model, we will make a table with the optimal angles, velocities and their respective allowed errors for both the angle and velocity for players of different heights:

Height	Release angle	Release velocity	Error allowed θ_0	Error allowed V_0
1.52 m	56.54°	7.34 m/s	2.08°	0.0538 m/s
1.55 m	56.47°	7.32 m/s	2.09°	0.0542 m/s
1.57 m	56.34°	7.29 m/s	2.11°	0.0547 m/s
1.60 m	56.14°	7.26 m/s	2.13°	0.0551 m/s
1.70 m	55.45°	7.16 m/s	2.20°	0.0568 m/s
1.73 m	55.28°	7.13 m/s	2.22°	0.0573 m/s
1.75 m	55.11°	7.10 m/s	2.24°	0.0577 m/s
1.83 m	54.60°	7.02 m/s	2.29°	0.0590 m/s
2.13 m	52.54°	6.71 m/s	2.50°	0.0640 m/s
2.16 m	52.37°	6.69 m/s	2.52°	0.0644 m/s
2.18 m	52.20°	6.66 m/s	2.54°	0.0648 m/s
2.21 m	52.02°	6.64 m/s	2.55°	0.0652 m/s

Table 6: the optimum trajectories for players of various heights.

3.4 Summary

Summing up our model, we have worked with the angles and release velocities with different initial conditions in order to answer the question “What is the best angle to shoot a free throw?” We have seen that defining what we mean by best, really depends upon the player and both in its height and in its consistent velocity and release angle. In general, we have reached the following conclusions:

1. The taller you are, the better free throw shooter you should be. This is because taller players have more room to make errors in both release angle and release velocity and still have the ball go in the basket as we saw in the feasible region, see Figure 12, and Table 6. Therefore, tall players who are poor free throw shooters either are shooting at the wrong angle or more likely are inconsistent in their release angle, release velocity, or both.
2. The shorter you are, the larger the release angle should be. This makes sense physically, as shorter players have more vertical distance to cover when shooting. It is good to see that our model confirms this (see Table 6).
3. The shorter you are, the closer to the back of the rim you should aim. That is for the trajectories that allow for maximum error pass somewhere between the center of the basket and the back ring. This can be noticed through the location of the optimal solution in the feasible region.
4. It is more important to use the right velocity as compared to the right angle, as we can saw in the error allowed for the release angle and the release velocity (see Figures 13, and 15)

4 Probabilistic Analysis

With the optimal free throw solution for Shaq and Kev, how can we verify that this solution is ensuring a high probability of success in their throws? Since we do not have the presence of Shaq to test throws, the most convenient thing is to perform computational simulations of free throws which allow us to observe each trajectory (V_0, θ_0) in the feasible region to verify if it is a successful throw or not.

4.1 Problem definition

For this analysis, we will perform Monte Carlo simulations to reproduce the free-throw throws for Shaq and Kev. Then, we will start from the probability distribution of angles and velocities, that is, when the ball is released with certain target values θ_0 and V_0 . In our case, these will be the angles and optimal velocities in our third case (see Table 5). It must be emphasized that the obtained throw values may be different but they will always be dominated by the probability distribution centered around (θ_0, V_0) .

4.2 Numerical Analysis and presentations of the results

In order to perform the Monte Carlo simulation we need to simulate a random variable, in our case it is the Normal Bivariate in which we will consider that the release angle and velocity are dependent, that is, they will have a correlation $\rho = 1$ and will be characterized by standard deviations σ_θ and σ_V respectively.

Consequently, if we want to generate a Bi-variate Normal random variable with $\theta_0 \sim \mathcal{N}(\mu_{\theta_0})$ and $V_0 \sim \mathcal{N}(\mu_{V_0}, \sigma_{V_0}^2)$ where the correlation of θ_0 and V_0 is ρ we can generate two independent unit normals Z_1 and Z_2 and use the Bi-dimensional transformation (see section 2.2.4) :

$$\begin{aligned} \theta_0 &= \sigma_{\theta_0} Z_1 + \mu_{\theta_0}, \\ V_0 &= \sigma_{V_0} \left[\rho Z_1 + \sqrt{1 - \rho^2} Z_2 \right] + \mu_{V_0}. \end{aligned}$$

Then in a Monte Carlo simulation, a million of different release conditions are generated for a given target pair (θ_0, V_0) , using MATLAB built-in normal random number generator. It is checked whether generated (θ, V) values would result in successful throws (would hit feasible region) and the total probability of success counted as a fraction of successful realizations. The code in MATLAB can be seen in Appendix D.1

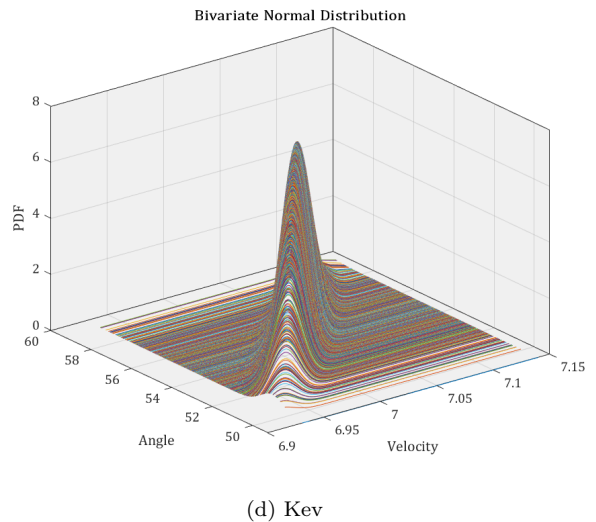
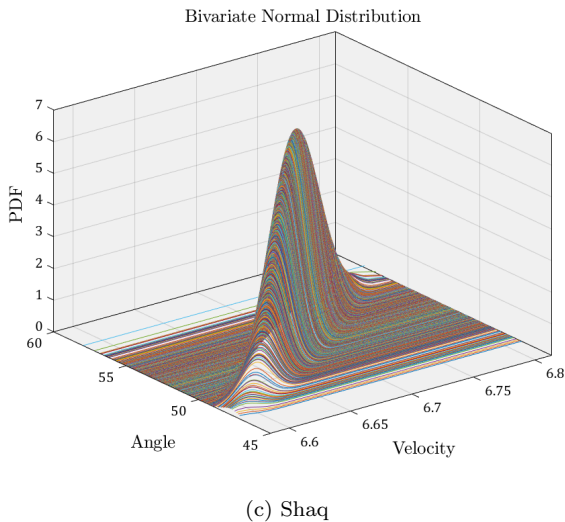
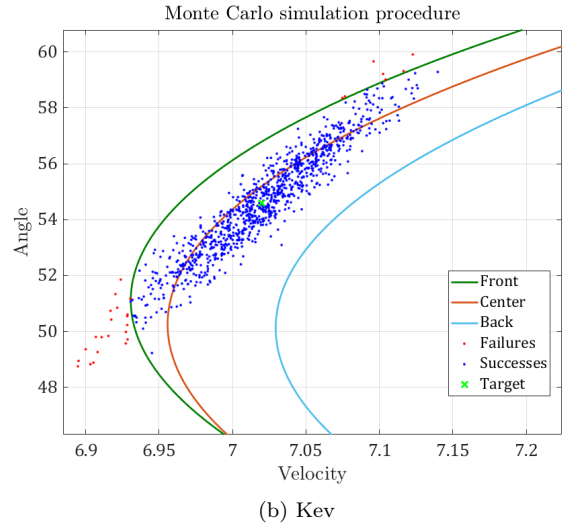
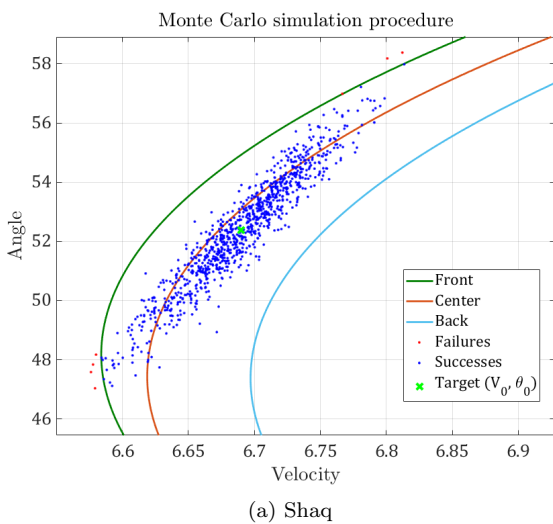


Figure 18: Monte Carlo simulation procedure in a feasible range of velocities and angles.

From the Figure 18 we see that our trajectory (V_0, θ_0) obtained in the third case (see Table 5) for both Shaq and Kev is satisfactory and verifies our model to get the higher percentage of success in free throws in each thrown. It can be observed that Shaq obtains fewer failures than Kev, this is due to the height of Shaq is greater than Kev's, this implies that he will span a larger area of the feasible region and a higher percentage of error allowed in the angle and velocity (see Figure 16). In addition, it is observed that the bell of the bivariate normal distribution is flattened in the direction to the optimal solution due to the correlation of almost 1 between the angle and velocity.

Now, to verify with Monte Carlo that, in fact, our best angle and velocity are the one with the highest probability of success we designed a program (see Appendix D.2). There, we centered our target (V_0, θ_0) in each part of the feasible region to obtain the probabilities of success in each one and thus check if it is the one that contains the highest probability of success for Shaq and Kev (see Table 5).

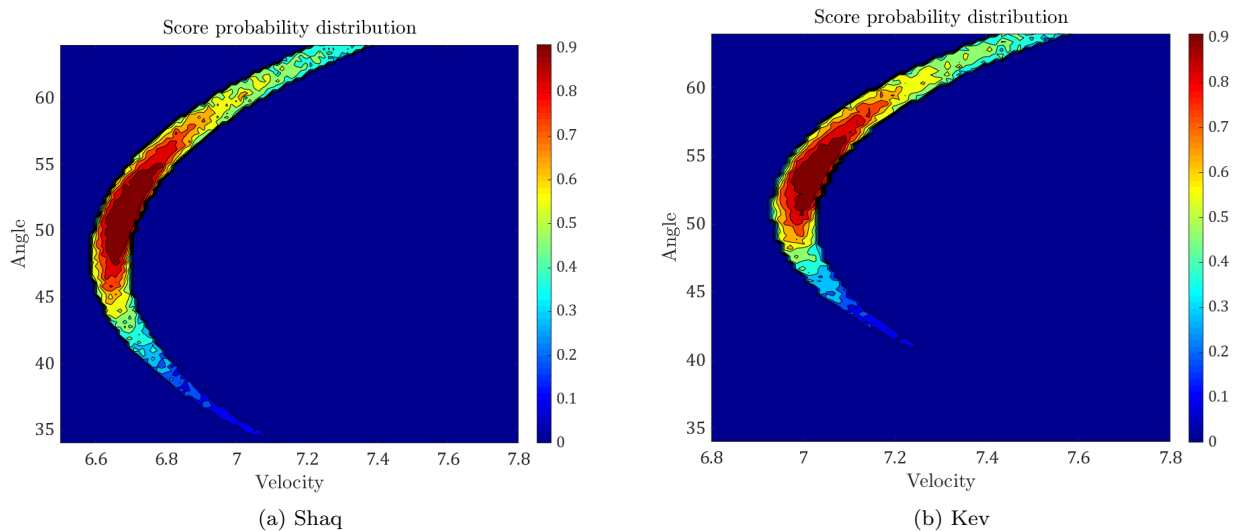


Figure 19: contour plot of score probability distribution as a function of target release angle θ_0 and target release velocity V_0 .

From Figure 19 we notice that our optimum result coincides with the area of high probability of successful throw for both Shaq and Kev. Which allows us to show that our mathematical model is good.

5 Statistical Analysis

This analysis aims to study a multiple regression model (see subsection 2.3) for successful and failed basketball free throws to validate the following conclusions of previous cases with simulated data. First, the taller you are, the better free throw shooter you should be, this is due to taller players have more room to make errors in both release angle and velocity. Second, the shorter you are, the larger the release angle should be. Third, the shorter you are, the closer to the back of the rim you should aim. Besides, it is much more important to use the right velocity as compared to the right angle. Finally, another objective is to perform an actual and adequate statistical study using the R software in order to improve and obtain an optimal mathematical model.

5.1 Problem definition

The variables analyzed for this model are the release angle θ_0 , release velocity V_0 , the height of the player H and the horizontal distance x when the center of the ball goes through the ring. Different data were collected by recording a video in slow-motion from an experiment performed on 9 semi-professional basketball players with a repetition of 10 free-throws per player in order to be analyzed later in R software.

5.2 Methods

To obtain the real data, 9 men served as subjects for this study, all of whom had been a member of a basketball college team. The skills of each player were similar to the others, due to the sample of players was taken from the two teams that always reach the finals in the basketball championships organized by Yachay Tech University. All subjects were right-handed. Their heights were between 1.58 - 1.90 meters. Each player threw 10 times.

Films records were taken by 1080 pixels to 120 frames per second in an iPhone's slow-motion camera which was located 5 m from the left side of the subject in the same position as the free throw line. We collected variables like the release velocity V_0 , the release angle θ_0 , the horizontal distance of the trajectory x given by (25) and by separating into two groups the variables that result in a successful throw or failure.

To analyze the trajectory characteristics of the basketball it was necessary to know the location of the ball center. The velocity was found by using the displacement of the ball center, the elapsed time between frames, and the equation of motion (10). The angle of the trajectory was the angle formed by the horizontal $x(t)$ and the position of the release (see Figure 4). These data were supplied to an R program to obtain the appropriate mathematical results. It was made an analysis of multiple regression techniques and analysis of variance with its respective verification of the assumptions.

5.3 Presentation of the results

To begin our comparative analysis between simulated data and real data using a multiple regression model for basketball free throws, we conducted a separate analysis from the players who had success and failure in their throws. We can see in the Appendix E in Figure 28a and Figure 28b the means of each player about their release angle and release velocity in addition to the number of successes that the player had in his throws.

We used the multiple regression analysis in the data of successes and failures since it allowed us to describe the behavior of the response variable given the values of the explanatory variables

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k + \varepsilon. \quad (34)$$

The parameters $\beta_0, \beta_1, \dots, \beta_k$ are estimated by least square. In matrix notation we have :

$$Y = X\beta + \varepsilon. \quad (35)$$

Then, for the selection of the best model, we consider as a response variable to the horizontal distance that the ball traveled until reaching the basket, and the coefficient of determination. The coefficient of determination is the statistic that represents the proportion of variation explained by the regression. A value of R near 0 implies the model does not explain anything about the variation of y with the relation with x_1, \dots, x_k , a value close to 1 means the greater amount of total variation is explained by the regression model. In the case of successful throws of all players we have the following model:

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{11} x_1^2 + \varepsilon, \quad (36)$$

where,

- Y = horizontal distance.
- x_1 = height of the player.
- x_2 = velocity.
- x_1^2 = square of the height of the player.

To obtain a better model we use a quadratic effect in the height of the player. Next, we have the output of the summary (36) and the ANOVA of the model in Figures 20 and 21 respectively, where we obtained a $R^2 = 0.9151$; that is, approximately 92% of variation in horizontal distance can be explained by our model (36). Moreover, all of explanatory variables were significant.


```

Call:
lm(formula = ds$hd ~ ds$player + ds$velocity + ds$player2, data = ds,
    x = TRUE)

Residuals:
    Min       1Q   Median       3Q      Max
-0.054903 -0.014945  0.003838  0.016682  0.040043

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -8.1523     1.1878  -6.863 1.19e-08 ***
ds$player      6.7835     1.2362   5.488 1.51e-06 ***
ds$velocity    0.8451     0.1093   7.733 5.59e-10 ***
ds$player2    -1.8252     0.3649  -5.001 8.03e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.02231 on 48 degrees of freedom
Multiple R-squared:  0.9151,    Adjusted R-squared:  0.9097
F-statistic: 172.4 on 3 and 48 DF,  p-value: < 2.2e-16
    
```

Figure 20: output about success free throw model

By (36) and Figure 20 we obtained the following equation of regression :

$$Y = -8.1523 + 6.7835x_1 + 0.8451x_2 + -1.8252x_1^2. \tag{37}$$

The interpretation of this equation is the following

- The estimated mean horizontal distance for someone with height, velocity and height square of 0.
- The slope for height player is the effect of height player on horizontal distance adjusting or controlling for velocity and height square . We associate an increase of 1 unit in height of player with an increase of 6.7835 in horizontal distance adjusting or controlling for velocity and height player .
- The slope for velocity is the effect of velocity on horizontal distance adjusting for height and height square. We associate an increase of 1 unit in velocity with an increase of 0.8451 in horizontal distance adjusting or controlling for height and height square.
- The slope for height square is the quadratic effect of height square on horizontal distance adjusting for height and velocity. We associate an increase of 1 unit in height square with a decrease of 1.8252 in horizontal distance adjusting or controlling for height and velocity.

As we can notice the explanatory variable of angle is not considered, because it was not significant, but it does not necessarily mean that it is not affecting the horizontal distance of the trajectory of the ball, maybe it could be that the velocity is also taking that behavior of the angle. So far, velocity is more important than the angle for explaining the horizontal distance of the trajectory of the ball. Moreover, we can notice that the quadratic effect explain in a better way the model due to the behavior is similar to the arguments of simulated data. That is, if the height of the player increase the horizontal distance decrease, this means that tall players make their throws near the front of the ring and that the shorter ones make their throws close to the back of the ring.

```

              Df Sum Sq Mean Sq F value    Pr(>F)
ds$player    1  0.19101  0.19101  383.83 < 2e-16 ***
ds$velocity  1  0.05385  0.05385  108.22 6.84e-14 ***
ds$player2   1  0.01245  0.01245   25.01 8.03e-06 ***
Residuals   48  0.02389  0.00050
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
    
```

Figure 21: output about the ANOVA

Now, we can check the assumption of multiple linear regression.

1. Linearity : we check from a scatterplot of the data in figure 22. Since we had negative correlation in height and positive correlation in velocity and angle then we use this data for a multiple linear regression.

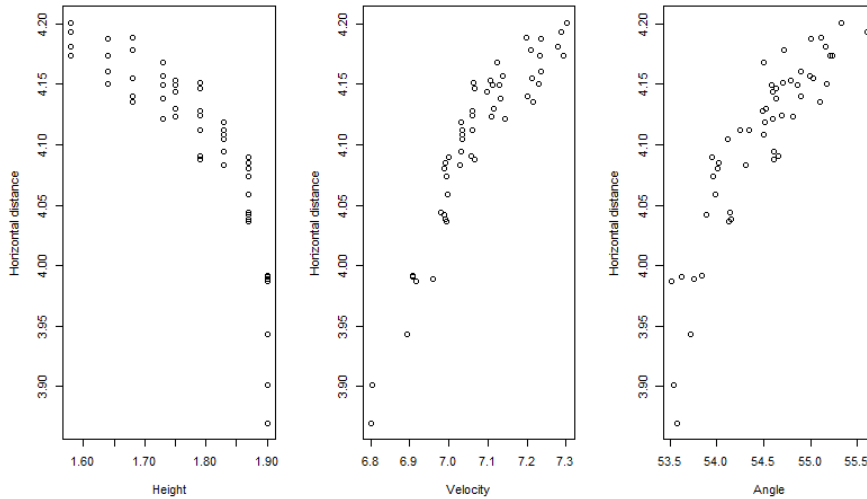


Figure 22: scatterplot of the data

2. Equal Residual Variance : we check from Figure 23 if the residuals form a pattern. Since, there is no pattern (so random) we validate this assumption.

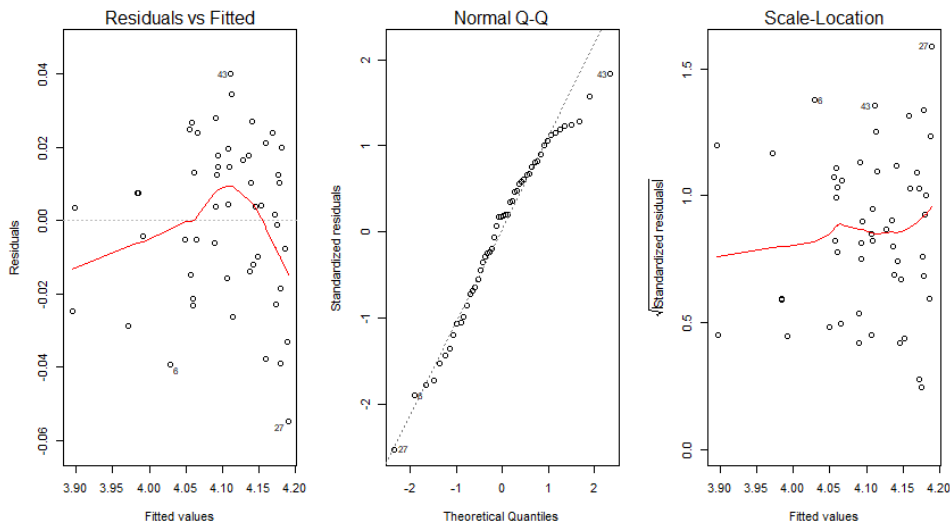


Figure 23: residuals vs fitted, normal Q-Q , scale -location

3. Normality of residuals: since the closer the residuals are to the fit line, the more normal they are.

We note for this model that the player who throw his balls between 3,990 and 4,210 meters were successful. The consistency of release velocity allow players to obtain greater success than the consistency of release angle (see Figure 24). The height had a quadratic effect that allowed improve the model to realistic argument due to while the height of the player decreases the horizontal distance of the trajectory of the ball increases. Also, it was appreciated that the shorter the player is, the higher the angle he must throw and finally, the more release velocity, the greater the horizontal distance the trajectory travels.

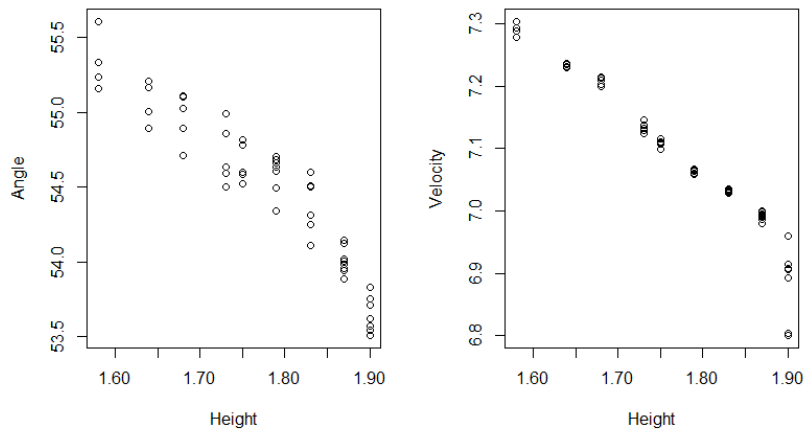


Figure 24: height vs angle and height vs velocity

Velocity	Angle
0.015 m/s	0.268°

Table 7: variance of the model of success throws.

In the case of the failed throws of all players, we will perform a descriptive statistics analysis due to the data of the response variable are not continuous in relation to the angle and velocity of thrown as we observed in the Figure 25. Knowing that we are not going to have observation for the response variable between 3,990 and 4,210 meters. Therefore, to obtain a better model to failures throws, we need to analyze two models between the intervals 3,870 -3.989 and 4,215-4.330 meters.

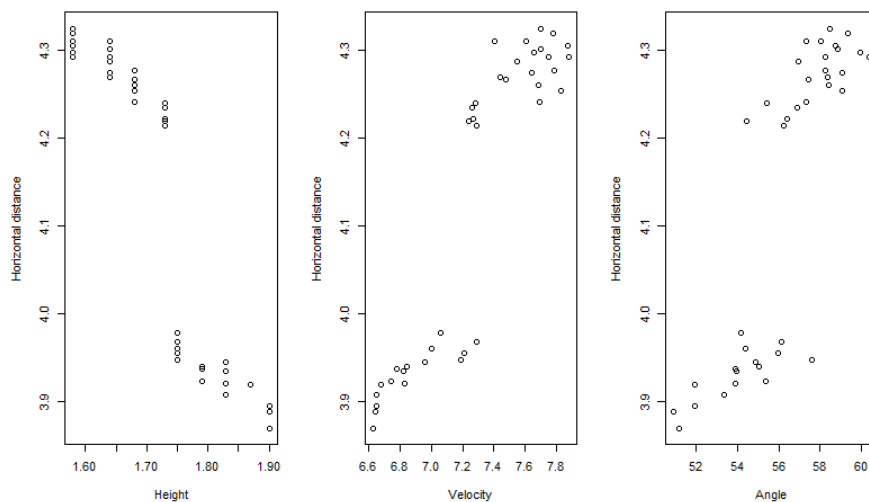


Figure 25: scatterplot of the data

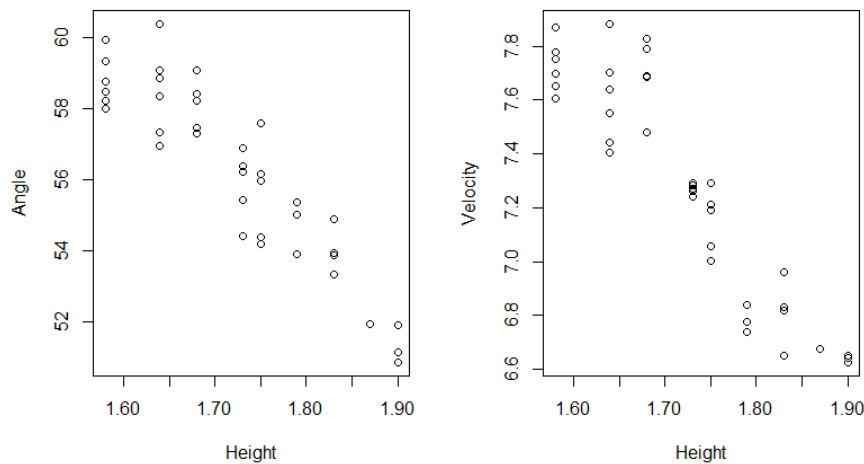


Figure 26: height vs angle and height vs velocity

From Figure 25 we check the linearity. Since we have negative correlation in height and positive correlation in velocity and angle but without information of the horizontal distance between 3,990 and 4,210 meters because we are in the model of failure throws. We note for this failures case that the player who throw his balls between 3,870 -3.989 and 4,215-4.330 meters make failures. Moreover, the non consistency of release velocity allow players to obtain greater failures than the non consistency of release angle (see Figure 26). Also, it was appreciated that the shorter the player is, the higher the angle he must throw and finally, the more release velocity, the greater the horizontal distance the trajectory travels.

6 Conclusions

This work aimed to find the best release angle and velocity in free throw shooting to obtain a successful throw. Based on an analytic, numerical, probabilistic and statistical way of studying, mathematically, it can be concluded that the best trajectory depends upon the player and both in its height and in its consistent in release velocity and release angle. As a result we have a feasible area of throws where we can obtain the angle and release velocity depending on the height of a certain player. In addition, during the study we found a new geometric and numerical criteria to find the angles and velocities for the ball to enter through the front of the ring. On the other hand, by using Monte Carlo simulations we were able to validate our mathematical model, simulating thousands of throws with an optimal target of release angles and release velocities that allow a successful throw probability of almost 100%. Finally, we make a pilot experiment with a semiprofessional basketball team in Ecuador in order to make a comparative analysis between simulated data en real data using a multiple regression model for basketball free throw which allowed to affirm the conclusions obtained in our model with simulated data. For a future work we would like, to adapt the 2D strategy to a 3D model by removing some assumptions such as the ball going straight to the ring, and allowing the ball hitting the ring and the board before the ball enters.

References

- [1] D Hays and JV Krause. Score on the throw. *The Basketball Bulletin, Winter*, pages 4–9, 1987.
- [2] K. Shibukawa. Speed conditions of basketball shooting. *Bulletin of the Institute of Sport Science (The Faculty of Physical Education, Tokyo University of Education)*, 13:59–64, 1975.
- [3] Peter J Brancazio. Physics of basketball. *American Journal of Physics*, 49(4):356–365, 1981.

-
- [4] Gordon R Hamilton and Christoph Reinschmidt. Optimal trajectory for the basketball free throw. *Journal of Sports Sciences*, 15(5):491–504, 1997.
- [5] Chau M Tran and Larry M Silverberg. Optimal release conditions for the free throw in men’s basketball. *Journal of sports sciences*, 26(11):1147–1155, 2008.
- [6] Larry M Silverberg, Chau M Tran, and Taylor M Adams. Optimal targets for the bank shot in men’s basketball. *Journal of Quantitative Analysis in Sports*, 7(1), 2011.
- [7] Joerg M Gablonsky and Andrew SID Lang. Modeling basketball free throws. *Siam Review*, 47(4):775–798, 2005.
- [8] Irina Barzykina. The physics of an optimal basketball free throw. *arXiv preprint arXiv:1702.07234*, 2017.
- [9] John H Mathews, Kurtis D Fink, et al. *Numerical methods using MATLAB*, volume 3. Pearson Prentice Hall Upper Saddle River, NJ, 2004.
- [10] Steven C Chapra and Raymond P Canale. *Numerical methods for engineers*, volume 7. Mcgraw-hill New York, 2015.
- [11] Matlab Documentation. The mathworks inc, 2019.
- [12] Jeffrey C Lagarias, James A Reeds, Margaret H Wright, and Paul E Wright. Convergence properties of the nelder–mead simplex method in low dimensions. *SIAM Journal on optimization*, 9(1):112–147, 1998.
- [13] Helio S Migon, Dani Gamerman, and Francisco Louzada. *Statistical inference: an integrated approach*. CRC press, 2014.
- [14] George AF Seber and Alan J Lee. *Linear regression analysis*, volume 329. John Wiley & Sons, 2012.

Appendix

A First case codes.

A.1 Initialize global variables (initialize.m)

```

1 % Kevin Chamorro.
2 % 07/03/2019.
3 % Yachay Tech University.
4
5 % Routine to set the values of all constants.
6 % All values are metric.
7 global Dr Db l h g ballstyle rimstyle centerstyle
8 Dr = 0.4572; % Rim diameter in meters (1.5 ft)
9 Db = 0.24384; % Ball diameter in meters (0.8 ft)
10 l = 4.1016; % Horizontal distance in meters
11 % (13 ft., 4 in)
12 playerh = 2.159; % Height of player (in meters)
13 releaseh = 1.25*playerh; % Vertical position of center of
14 % ball at release
15 basketh = 3.048; % Vertical position of the ring
16 % (in meters), 10ft.
17 h = basketh - releaseh; % Vertical distance traversed in
18 % meters
19 % Vertical distance traversed in
20 % meters (1 ft 1.75 in)
21 g = -9.81; % Gravity constant in meters per
22 % second squared (-32 ft (s)^(-2))

```

A.2 Range of angles to the back of the ring

```

1 % Kevin Chamorro.
2 % 07/03/2019.
3 % Yachay Tech University.
4
5 clear
6 clc
7 initialize
8 v0=7;
9 ep = Dr/2 - Db/2;
10 x=l+ep;
11
12 ang=asin(sqrt((-2*g*h)/(v0^2))):pi*1/180:(pi*90/180)-asin((-l*g)/(v0^2))*0.5;
13 angles = [];
14 result=[];
15 results=[];
16 for i=1:length(ang)
17     result =[result,((-v0*cos(ang(i))/g)*(v0*sin(ang(i))+sqrt(v0^2*sin(ang(i))^2+(2*g*h)))]);
18     if result(i) <l+ep && result(i)>l-ep && result(i)>0
19         angles = [angles, ang(i)];
20         results=[results, result(i)];
21     end
22 end
23 radtodeg(ang) % Proper Range of Angles
24 radtodeg(angles)% Angles that satisfies the conditions
25 results % x new horizontal position due to error in release angle.

```

A.3 Program to plot errors about θ_0 (plot_angerror_vopt.m)

```

1 % Kevin Chamorro.
2 % 07/03/2019.
3 % Yachay Tech University.
4
5 % This routine calculates the maximum error to make in the
6 % angle to still make the basket. For each angle theta, we
7 % calculate how much of an error the shooter is allowed to
8 % make to still make the basket, assuming he or she shoots
9 % the freethrow with the optimal velocity for the given
10 % angle theta.
11 initDraw;
12 thetas = [[39:.4:48] [48.15:.001:48.2] [48.6:.4:60]]; %Shaq
13 %thetas = [[44:.4:51.08] [51.10:.001:51.13] [51.14:.4:60]]; %Kev
14 errors = thetas;
15 for i = 1:length(thetas)
16     errors(i) = calcerrorvopt(thetas(i));
17 end

```

```

18 maxError = max(errors);
19 for i=1:length(errors)
20     if (errors(i) == maxError)
21         ang = thetas(i);
22     end
23 end
24 theta0=ang;
25 v0=calcOptv(theta0);
26 plot(thetas, errors, '-');
27 grid on
28 ylabel('e(\theta_0)');
29 xlabel('\theta_0');

```

A.3.1 Program to calculate the maximum allowed error in the angle with respect to the back and front of the rim when the ball is thrown with optimal velocity (calcerrorvopt.m)

```

1 % Kevin Chamorro.
2 % 07/03/2019.
3 % Yachay Tech University.
4
5 function error = calcerrorvopt(theta0)
6 % Routine to calculate the maximum allowed error in the angle
7 % with respect to the back of the rim when the ball is thrown
8 % with optimal velocity.
9
10 % First calculate the optimal velocity for this angle.
11 v0 = calcOptv(theta0);
12 % Use the general routine to calculate the maximum allowable
13 % error given an initial velocity, and angle.
14 y = [theta0, v0];
15 error = -calcerror(y);

```

A.3.2 Program to calculate the optimal velocity to hit the center of the basket given the initial angle of θ_0 (calcOptv.m)

```

1 % Kevin Chamorro.
2 % 07/03/2019.
3 % Yachay Tech University.
4
5 function [ v0 , T0 ] = calcOptv ( theta0 ) ;
6 % Function to calculate the optimal velocity to hit the center
7 % of the basket , given an initial angle of theta0.
8 global Dr Db l h g ballstyle rimstyle centerstyle
9 v0 = l /cos ( theta0 *pi /180)* sqrt(-g /(2*( l *tan( theta0 *pi/180)-h ) ) ) ;
10 T0 = l /( cos ( theta0 *pi /180)* v0 ) ;

```

A.3.3 Program to calculate the maximum allowed error in the angle with respect to the back and front of the rim when the ball is thrown with given velocity (calcerror.m)

```

1 % Kevin Chamorro.
2 % 07/03/2019.
3 % Yachay Tech University.
4
5 function error = calcerror(y)
6 % Routine to calculate the maximum allowed error in the angle
7 % with respect to the back and front of the rim when the ball
8 % is thrown with given velocity.
9 % Note that we return the error multiplied by (-1) since we
10 % want to maximize the error, but the optimization methods
11 % wants to minimize.
12 global Dr Db l h g ballstyle rimstyle centerstyle
13
14 % Set all variables.
15 theta0 = y(1);
16 v0     = y(2);
17
18 % Check if the angle is in the valid range, that is, between
19 % 10 and 85 degrees.
20 if ((theta0 < 10) || (theta0 > 85))
21     sprintf(...
22         'Angle theta0 = %5.2f is either too small or too large.'...
23         ,theta0)
24     error = NaN; % The ball is too far from the back of the
25                 % rim to still be in the rim.
26     return
27 end
28
29

```

```

30 % Calculate this value since it is used several times below.
31 sintheta = sin(theta0*pi/180);
32
33 % Calculate the horizontal position of the ball as it comes
34 % back down to the basket height.
35 x = v0 * cos(theta0*pi/180)/(-g);
36 x = x*(v0*sintheta + sqrt(v0*v0*sintheta*sintheta+2*g*h));
37
38 if (x < 1 - Dr/2+Db/2)
39     sprintf('The ball does not reach the basket.')
40     error = NaN; % The ball is too far from the back of the
41                 % rim to still be in the rim.
42     return
43 end
44 if (x > 1 + Dr/2 - Db/2)
45     sprintf('The ball goes too far.')
46     error = NaN; % The ball is too close to the back of the
47                 % rim or behind the back of the rim.
48     return
49 end
50
51 % Check to see if the ball hits the front of the rim. This is
52 % done by calculating the minimum distance the ball has from
53 % the front of the rim.
54 % If this distance is below 0, the ball hits the front rim.
55 interm = frontfzero(theta0, v0);
56 if (interm < 0) % The ball hits the front rim.
57     sprintf('The ball hits the front of the rim.')
58     error = NaN;
59     return
60 end
61 fronterror = calcfronterror(theta0, v0);
62 backerror = calcbackerror(theta0, v0);
63 error = -min(fronterror, backerror);

```

A.3.4 Program to calculate the minimum distance from the front of the rim for the ball in the relevant time interval (frontfzero.m)

```

1 % Kevin Chamorro.
2 % 07/03/2019.
3 % Yachay Tech University.
4
5 function y = frontfzero(x,v);
6 % Function to calculate the minimum distance from the front
7 % rim in the relevant time interval.
8 global Dr Db l h g ballstyle rimstyle centerstyle
9 sintheta = sin(x*pi/180);
10 % Calculate the relevant time interval.
11 timeinterval = [(1-Dr/2)/(v*cos(x*pi/180)), -1/g*(v*sintheta+...
12             sqrt(v*v*sintheta*sintheta+2*g*h))];
13 % If this time interval is not a interval, return a negative
14 % value. This signals the calling program that the ball either
15 % goes through the front rim, or never even reaches the front
16 % rim.
17 if (timeinterval(1) > timeinterval(2))
18     y = -1;
19     return;
20 end
21
22 % Calculate the minium distance from the front rim in the
23 % relevant time interval.
24 [k, y] = fminbnd(@distancefromrim, timeinterval(1), ...
25             timeinterval(2), [], v, x);

```

A.3.5 Program to calculate the distance from the front of the rim for the ball at time t (distancefromrim.m)

```

1 % Kevin Chamorro.
2 % 07/03/2019.
3 % Yachay Tech University.
4
5 function s = distancefromrim(t,v0,theta)
6 % Calculate the distance from the front of the rim for the ball
7 % at time t, with initial velocity v, and angle theta. Use this
8 % distance to calculate how much space is left between the ball
9 % and the rim. If this function is negative, the ball hits the
10 % rim. If it is zero, the ball scims the rim for this velocity
11 % and this release angle at time t.
12 global Dr Db l h g ballstyle rimstyle centerstyle
13 %global v pos

```

```

14 [x,y] = position(v0,theta,t);
15 % Calculate the position of the center of the ball at time t,
16 % with initial velocity v and release angle theta.
17
18 part1 = x -(l - Dr/2);
19 part2 = y - h;
20
21 s = sqrt(part1.*part1 + part2.*part2)-Db/2;
22 % Calculate the distance from the front of the rim.
23

```

A.3.6 Program to calculate the position of the ball at time t, thrown with given velocity and angle (position.m)

```

1 % Kevin Chamorro.
2 % 07/03/2019.
3 % Yachay Tech University.
4
5 function [x,y] = position(v,theta,t);
6 % Position of the ball at time t, with initial velocity v,
7 % and angle theta.
8 global Dr Db l h g ballstyle rimstyle centerstyle
9
10 x = v*cos(theta*pi/180)*t;
11 y = v*sin(theta*pi/180)*t + .5*g*t.*t;

```

A.3.7 Program 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 (calcfronterror.m)

```

1 % Kevin Chamorro.
2 % 07/03/2019.
3 % Yachay Tech University.
4
5 function error = calcfronterror(theta0, v0);
6 % Routine to calculate the maximum allowed error in the angle
7 % with respect to the front of the rim when the ball is thrown
8 % with given velocity.
9 global Dr Db l h g ballstyle rimstyle centerstyle
10
11 % Calculate the angle when the ball just skims the front rim,
12 % and still goes in.
13 ang = fzero(@(x) frontfzero(x,v0), theta0);
14 ang2 = fzero(@(x) frontfzero(x,v0), theta0+1);
15 % Calculate the error allowed.
16 error = min(abs(theta0-ang),abs(theta0-ang2));

```

A.3.8 Program to calculate the maximum allowed error in the angle with respect to the back of the rim (calcbackerror.m)

```

1 % Kevin Chamorro.
2 % 07/03/2019.
3 % Yachay Tech University.
4
5 function [error, ang] = calcbackerror(theta0, v0);
6 % Routine to calculate the maximum allowed error in the angle
7 % with respect to the back of the rim when the ball is thrown
8 % with given velocity.
9 global Dr Db l h g ballstyle rimstyle centerstyle
10 % Find the maximum distance from the back the ball thrown
11 % with this velocity, and varying angles can have. Note that
12 % we have to multiply the distance with (-1) to use the Matlab
13 % fminsearch function to find a maximum.
14 [ang, val] = fminsearch(@(x) -distancefromback(v0,x), theta0);
15
16 % Negate the value to reverse the multiplication with (-1) that
17 % was necessary to do a maximization.
18 val = -val;
19 % If this maximum is small enough, the ball cannot reach the
20 % back of the rim. Therefore the error can be infinite.
21 if (val < 0)
22     error = NaN;
23 else
24     if (val > Dr-Db/2)
25         error = NaN;
26     else
27         % The ball can reach the back of the rim. Find the
28         % angle when the ball just skims the back of the rim
29         % by minimizing the negative distance from the rim.

```



```

30     ang = fzero(@(x) -distancefromback(v0,x), theta0);
31     % Calculate the error in angle allowed.
32     error = abs(ang-theta0);
33     end
34 end

```

A.3.9 Program to calculate the distance from the back of the rim for the ball when its center is level with the rim (distancefromback.m)

```

1 % Kevin Chamorro.
2 % 07/03/2019.
3 % Yachay Tech University.
4
5 function dist = distancefromback(v,theta)
6 % Calculate the distance from the back of the rim for the ball
7 % when its center is level with the rim. The ball is assumed to
8 % have initial velocity v, and initial angle theta.
9
10 global Dr Db l h g ballstyle rimstyle centerstyle
11
12 sintheta = sin(theta*pi/180);
13 sqrtval = v*v*sintheta*sintheta+2*g*h;
14
15 if (sqrtval < 0)
16     dist = -inf;
17     return
18 end
19 x = v * cos(theta*pi/180)/(-g);
20 x = x*(v*sintheta + sqrt(sqrtval));
21 dist = x-l+(Db-Dr)/2;

```

B Second case codes.

B.1 Criterion back of the ring

```

1 % Kevin Chamorro.
2 % 07/03/2019.
3 % Yachay Tech University.
4
5 clear all
6 clc
7 %Routine to calculate the angles and velocities that allow
8 %to obtain a successful shot in the back boundary of the ring.
9 global Dr Db l h g ballstyle rimstyle centerstyle
10
11 initialize
12 numx = 1;
13 numy = 1;
14 printing = 1;
15
16 drawgraphs = 0;
17 angstart = 34;
18 angend = 64;
19 angsteps = 200;
20
21 plotangstart = 40;
22 plotangend = 58;
23 vstart = 6.5;
24 vend = 7.8;
25 vsteps = 400;
26
27 mheightstart=playerh ; % Heigh of the player
28 releaseh=1.25*playerh;
29 h=basketh-releaseh;
30
31 thetarange = linspace(angstart , angend , angsteps);
32 vrange = linspace(vstart , vend , vsteps);
33
34
35 vback = thetarange;
36
37 opt = optimset;
38 for i = 1:length(thetarange)
39     vback(i) = vbackrim(thetarange(i));
40 end
41 plot(vback , thetarange)

```

B.1.1 Program to calculate the velocity to hit the back of the rim (vbackrim.m)

```

1 % Kevin Chamorro.
2 % 07/03/2019.
3 % Yachay Tech University.
4
5 function v0 = vbackrim(theta0);
6 % Function to calculate the optimal velocity to hit the back
7 % of the basket, given an initial angle of theta0.
8 global Dr Db l h g ballstyle rimstyle centerstyle
9
10 v0 = (l+1 +Dr - Db)/(2*cos(theta0*pi/180)) * ...
11      sqrt(g/(h + h - (l + l + Dr - Db)*tan(theta0*pi/180)));

```

B.2 Criterion center of the ring

```

1 % Kevin Chamorro.
2 % 07/03/2019.
3 % Yachay Tech University.
4
5 clear all
6 clc
7 %Routine to calculate the angles and velocities that allow
8 %to obtain a successful shot in the center boundary of the ring.
9 global Dr Db l h g ballstyle rimstyle centerstyle
10 initialize
11 numx = 1;
12 numy = 1;
13 printing = 1;
14
15 drawgraphs = 0;
16 angstart = 34;
17 angend = 64;
18 angsteps = 200;
19
20 plotangstart = 40;
21 plotangend = 58;
22 vstart = 6.5;
23 vend = 7.8;
24 vsteps = 400;
25
26 mheightstart=playerh ; % Heigh of the player
27 releaseh=1.25*playerh;
28 h=basketh-releaseh;
29
30 thetarange = linspace(angstart , angend , angsteps);
31 vrange = linspace(vstart , vend , vsteps);
32
33 vcenter = thetarange;
34
35 opt = optimset;
36 for i = 1:length(thetarange)
37     vcenter(i) = vcenterrim(thetarange(i));
38 end
39 plot(vcenter , thetarange)

```

B.2.1 Program to calculate the velocity to hit the center of the rim (vcenterrim.m)

```

1 % Kevin Chamorro.
2 % 07/03/2019.
3 % Yachay Tech University.
4
5 function v0 = vcenterrim(theta0);
6 % Function to calculate the optimal velocity to hit the center
7 % of the basket, given an initial angle of theta0.
8
9 global Dr Db l h g ballstyle rimstyle centerstyle
10
11 v0 = l/cos(theta0*pi/180)*...
12      sqrt(g/(2*(h - l*tan(theta0*pi/180))));

```

B.3 Criterion front of the ring

```

1 % Kevin Chamorro.
2 % 07/03/2019.
3 % Yachay Tech University.
4
5 clear all
6 clc
7 %Routine to calculate the angles and velocities that allow
8 %to obtain a successful shot in the front boundary of the ring.
9 global Dr Db l h g ballstyle rimstyle centerstyle

```

```

10 initialize
11 numx = 1;
12 numy = 1;
13 printing = 1;
14
15 drawgraphs = 0;
16 angstart = 34;
17 angend = 64;
18 angsteps = 200;
19
20 plotangstart = 40;
21 plotangend = 58;
22 vstart = 6.5;
23 vend = 7.8;
24 vsteps = 400;
25
26 mheightstart=playerh ; % Heigh of the player
27 releaseh=1.25*playerh;
28 h=basketh-releaseh;
29
30 thetarange = linspace(angstart , angend , angsteps);
31 vrange = linspace(vstart , vend , vsteps);
32
33 vfront = thetarange;
34
35 opt = optimset;
36 for i = 1:length(thetarange)
37     vfront(i) = fzero(@frontfzerov , vfrontrim(...
38         thetarange(i)), opt , thetarange(i));
39 end
40 plot(vcenter , thetarange)

```

B.3.1 Program to calculate the minimum distance from the front of the rim for the ball in the relevant time interval (frontfzerov.m)

```

1 % Kevin Chamorro.
2 % 07/03/2019.
3 % Yachay Tech University.
4
5 function y = frontfzerov(v0, theta);
6 % Routine to be a frontend to minimize the distance from the
7 % front. If the function returns 0, the ball just skim's the
8 % front of the rim.
9 global Dr Db l h g ballstyle rimstyle centerstyle
10 v = v0;
11 sintheta = sin(theta*pi/180);
12 % Calculate the relevant time interval.
13 timeinterval = [(1-Dr/2)/(v*cos(theta*pi/180)), -1/g*...
14     (v*sintheta+sqrt(v*v*sintheta*sintheta+2*g*h))];
15 % If this time interval is not a interval, return a negative
16 % value. This signals the calling program that the ball
17 % either goes through the front rim, or never even reaches
18 % the front rim.
19 if (timeinterval(1) > timeinterval(2))
20     y = -1;
21     return;
22 end
23 pos = theta;
24 % Calculate the minimum distance from the front rim in the
25 % relevant time interval.
26 [k, y] = fminbnd(@distancefromrim, timeinterval(1), ...
27     timeinterval(2), [], v, theta);

```

B.3.2 Program to calculate the velocity to hit the front of the rim (vfrontrim.m)

```

1 % Kevin Chamorro.
2 % 07/03/2019.
3 % Yachay Tech University.
4
5 function [vup, vdo] = vfrontrim(theta);
6 % Function to calculate the velocity when the ball hits
7 % the front of the rim with the center of the ball
8 % exactly on top of the front of the rim (vup), and when
9 % the center of the ball is exactly level with the rim (vdo).
10
11 global Dr Db l h g ballstyle rimstyle centerstyle
12 costheta = cos(theta*pi/180);
13 tantheta = tan(theta*pi/180);
14
15 vup = (1 + l - Dr)/(2*costheta) * ...

```

```

16     sqrt(g / ( h + h - (l + l - Dr) * tantheta));
17
18 vdo = (l+l - Dr + Db)/(2*costheta) * ...
19     sqrt(g / ( h + h - (l + l - Dr + Db) * tantheta));

```

B.4 Feasible Region

```

1  clear all
2  % Kevin Chamorro.
3  % 07/03/2019.
4  % Yachay Tech University.
5
6  clc
7  % Routine to calculate the feasible region with the front,
8  % center and back boundary
9  global Dr Db l h g ballstyle rimstyle centerstyle
10
11 initialize
12 numx = 1;
13 numy = 1;
14 printing = 1;
15
16 drawgraphs = 0;
17 angstart = 34;
18 angend = 64;
19 angsteps = 200;
20
21 plotangstart = 40;
22 plotangend = 58;
23 vstart = 6.5;
24 vend = 7.8;
25 vsteps = 400;
26
27 mheightstart=playerh ; % Heigh of the player
28 releaseh=1.25*playerh;
29 h=basketh-releaseh;
30
31 thetarange = linspace(angstart , angend , angsteps);
32 vrange = linspace(vstart , vend , vsteps);
33
34 vfront = thetarange;
35 vcenter = thetarange;
36 vback = thetarange;
37
38 opt = optimset;
39 for i = 1:length(thetarange)
40     vfront(i) = fzero(@frontzerov , vfrontrim (...
41         thetarange(i)), opt , thetarange(i));
42     vcenter(i) = vcenterrim(thetarange(i));
43     vback(i) = vbackrim(thetarange(i));
44 end
45 plot(vfront , thetarange , vcenter , thetarange , vback , ...
46     thetarange);

```

B.5 Program to calculate the error allowed angle in the feasible region (error_allowed_angle.m)

```

1  % Kevin Chamorro.
2  % 07/03/2019.
3  % Yachay Tech University.
4
5  clear all
6  clc
7  %Routine to calculate the error and the percentage of error
8  %allowed for the release angle in a free throw.
9  global Dr Db l h g ballstyle rimstyle centerstyle
10
11 initialize
12 numx = 1;
13 numy = 1;
14 printing = 1;
15
16 drawgraphs = 0;
17 angstart = 34;
18 angend = 64;
19 angsteps = 400;
20
21 plotangstart = 40;
22 plotangend = 58;
23 vstart = 6.5;
24 vend = 7.8;

```

```

25 vsteps = 400;
26
27 thetarange = linspace(angstart, angend, angsteps);
28 vrange = linspace(vstart, vend, vsteps);
29
30 ig = 1;
31
32 deltav = zeros(angsteps, vsteps);
33 deltatheta = deltav;
34 thetaper = deltav;
35 vper = deltav;
36 errortotal = deltav;
37
38 vfront = thetarange;
39 vcenter = thetarange;
40 vback = thetarange;
41
42 opt = optimset;
43 for i = 1:length(thetarange)
44     vfront(i) = fzero(@frontzerov, vfrontim(...
45         thetarange(i)), opt, thetarange(i));
46     vcenter(i) = vcenterim(thetarange(i));
47     vback(i) = vbackim(thetarange(i));
48 end
49     for j = 1:vsteps
50         for i = 1:angsteps
51             if ((vfront(i) <= vrange(j)) && ...
52                 (vrange(j) <= vback(i)))
53                 helpang1 = 0;
54                 helpang2 = 0;
55                 for il = i+1:angsteps
56                     if ((vfront(il) <= vrange(j)) && ...
57                         (vrange(j) <= vback(il)))
58
59                         helpang1 = thetarange(il) - ...
60                             thetarange(i);
61
62                     else
63                         break;
64                     end
65                 end
66                 for il = i-1:-1:1
67                     if ((vfront(il) <= vrange(j)) && ...
68                         (vrange(j) <= vback(il)))
69                         helpang2 = thetarange(i) - ...
70                             thetarange(il);
71
72                     else
73                         break;
74                     end
75                 end
76                 deltatheta(i,j) = min(helpang1, helpang2);
77                 thetaper(i,j) = deltatheta(i,j)/...
78                     thetarange(i)*100;
79             end
80         end
81     end
82     figure(60+(ig - (rem(ig-1,numx*numy)+1))/numx*numy+2)
83     subplot(numx, numy, rem(ig-1, numx*numy)+1)
84     contourf(vrange, thetarange, deltatheta, 10)
85     title('Percent error allowed in angle');
86     xlabel('Velocity');
87     ylabel('Angle');
88     hh = colorbar;

```

B.6 Program to plot and calculate the solution for the second case (plot_second_case.m)

```

1 % Kevin Chamorro.
2 % 07/03/2019.
3 % Yachay Tech University.
4
5 clear all
6 clc
7 %Routine to calculate the error and the percentage of error
8 %allowed for the release angle in a free throw.
9 global Dr Db l h g ballstyle rimstyle centerstyle
10
11 initialize
12 numx = 1;
13 numy = 1;
14 printing = 1;
15

```

```

16 drawgraphs = 0;
17 angstart = 34;
18 angend = 64;
19 angsteps = 400;
20
21 plotangstart = 40;
22 plotangend = 58;
23 vstart = 6.5;
24 vend = 7.8;
25 vsteps = 400;
26
27 thetarange = linspace(angstart, angend, angsteps);
28 vrange = linspace(vstart, vend, vsteps);
29
30 ig = 1;
31
32 deltav = zeros(angsteps, vsteps);
33 deltatheta = deltav;
34 thetaper = deltav;
35 vper = deltav;
36 errortotal = deltav;
37
38 vfront = thetarange;
39 vcenter = thetarange;
40 vback = thetarange;
41
42 opt = optimset;
43 for i = 1:length(thetarange)
44     vfront(i) = fzero(@frontzerov, vfrontrim(...
45         thetarange(i)), opt, thetarange(i));
46     vcenter(i) = vcenterim(thetarange(i));
47     vback(i) = vbackrim(thetarange(i));
48 end
49     for j = 1:vsteps
50         for i = 1:angsteps
51             if ((vfront(i) <= vrange(j)) && ...
52                 (vrange(j) <= vback(i)))
53                 helpang1 = 0;
54                 helpang2 = 0;
55                 for il = i+1:angsteps
56                     if ((vfront(il) <= vrange(j)) && ...
57                         (vrange(j) <= vback(il)))
58                         helpang1 = thetarange(il) - ...
59                             thetarange(i);
60                     else
61                         break;
62                     end
63                 end
64             for il = i-1:-1:1
65                 if ((vfront(il) <= vrange(j)) && ...
66                     (vrange(j) <= vback(il)))
67                     helpang2 = thetarange(i) - ...
68                         thetarange(il);
69                 else
70                     break;
71                 end
72             end
73         end
74         deltatheta(i, j) = min(helpang1, helpang2);
75         thetaper(i, j) = deltatheta(i, j)/...
76             thetarange(i)*100;
77     end
78 end
79 end
80
81
82
83 % Calculate the optimal velocity and angle with
84 % respect to errors in the release angle.
85
86 [val, I] = max(thetaper); %max I= indice row value teta
87 [vall, j] = max(val); % j= column v
88 pos = [vrange(j) thetarange(I(j))];
89 [x, val] = fminsearch(@calcdiff, pos, optimset, 0,1);
90 [diff, diffang, diffv] = calcdiff(x, 0, 1);
91 soltheta(ig, :) = [x -val -diffang -diffv 0 0]
92
93 plot(vfront, thetarange, vcenter, thetarange, vback, ...
94     thetarange, soltheta(ig,1), soltheta(ig,2), '.');
95 vdata = axis;
96 xlabel('Velocity');
97 ylabel('Angle');

```

B.7 Program to calculate the error allowed velocity in the feasible region (error_allowed_velocity.m)

```

1 % Kevin Chamorro.
2 % 07/03/2019.
3 % Yachay Tech University.
4
5 clear all
6 clc
7 %Routine to calculate the error and the percentage of error
8 %allowed for the release velocity in a free throw.
9 global Dr Db l h g ballstyle rimstyle centerstyle
10
11 initialize
12 numx = 1;
13 numy = 1;
14 printing = 1;
15
16 drawgraphs = 0;
17 angstart = 34;
18 angend = 64;
19 angsteps = 400;
20
21 plotangstart = 40;
22 plotangend = 58;
23 vstart = 6.5;
24 vend = 7.8;
25 vsteps = 400;
26
27 thetarange = linspace(angstart, angend, angsteps);
28 vrangle = linspace(vstart, vend, vsteps);
29
30 ig = 1;
31
32 deltav = zeros(angsteps, vsteps);
33 deltatheta = deltav;
34 thetaper = deltav;
35 vper = deltav;
36 errortotal = deltav;
37
38 vfront = thetarange;
39 vcenter = thetarange;
40 vback = thetarange;
41
42 opt = optimset;
43 for i = 1:length(thetarange)
44     vfront(i) = fzero(@frontzerov, vfrontrim(...
45                     thetarange(i)), opt, thetarange(i));
46     vcenter(i) = vcenterim(thetarange(i));
47     vback(i) = vbackrim(thetarange(i));
48
49     for j = 1:vsteps
50         deltav(i, j) = min(vrange(j) - vfront(i), ...
51                             vback(i) - vrangle(j));
52         vper(i, j) = deltav(i, j)/vrangle(j)*100;
53     end
54 end
55
56 deltav = max(deltav, 0);
57 vper = max(vper, 0);
58
59 figure(30+(ig - (rem(ig-1, numx*numy)+1))/numx*numy+2)
60 subplot(numx, numy, rem(ig-1, numx*numy)+1)
61 contourf(vrange, thetarange, vper, 10)
62 title('Percent error allowed in velocity');
63 xlabel('Velocity');
64 ylabel('Angle');
65 hh = colorbar;

```

C Third case codes.

C.1 Program to calculate the weighted error allowed angle and velocity (weighted_error.m)

```

1 % Kevin Chamorro.
2 % 07/03/2019.
3 % Yachay Tech University.
4
5 clear all

```

```

6   clc
7   %Routine to calculate the weighted error
8   %allowed for the release angle in a free throw.
9   global Dr Db l h g ballstyle rimstyle centerstyle
10
11  initialize
12  numx = 1;
13  numy = 1;
14  printing = 1;
15  %Grid of angles
16  drawgraphs = 0;
17  angstart = 34;
18  angend = 64;
19  angsteps = 400;
20  %Grid of velocities
21  plotangstart = 40;
22  plotangend = 58;
23  vstart = 6.5;
24  vend = 7.8;
25  vsteps = 400;
26
27  %weights
28  angw = 1;
29  vw = 5; % five times error in velocity
30  solution = '';
31
32
33
34  thetarange = linspace(angstart, angend, angsteps);
35  vrangle = linspace(vstart, vend, vsteps);
36
37  ig = 1;
38
39      deltav = zeros(angsteps, vsteps);
40      deltatheta = deltav;
41      thetaper = deltav;
42      vper = deltav;
43      errortotal = deltav;
44
45      vfront = thetarange;
46      vcenter = thetarange;
47      vback = thetarange;
48
49      opt = optimset;
50
51  %Conditions to obtain a successful throw.
52  for i = 1:length(thetarange)
53      vfront(i) = fzero(@frontzerov, vfrontrim(...
54          thetarange(i)), opt, thetarange(i));
55      vcenter(i) = vcenterim(thetarange(i));
56      vback(i) = vbackrim(thetarange(i));
57  %Deviation from given velocities
58
59      for j = 1:vsteps
60          deltav(i, j) = min(vrange(j) - vfront(i), ...
61              vback(i) - vrange(j));
62          vper(i, j) = deltav(i, j)/vrange(j)*100;
63      end
64  end
65
66
67
68  deltav = max(deltav, 0);
69  vper = max(vper, 0);
70
71  %Deviation from given angles
72  for j = 1:vsteps
73      for i = 1:angsteps
74          if ((vfront(i) <= vrange(j)) && ...
75              (vrange(j) <= vback(i)))
76              helpang1 = 0;
77              helpang2 = 0;
78              for i1 = i+1:angsteps
79                  if ((vfront(i1) <= vrange(j)) && ...
80                      (vrange(j) <= vback(i1)))
81
82                      helpang1 = thetarange(i1) - ...
83                          thetarange(i);
84
85                      else
86                          break;
87                      end
88              end
89          end

```



```

88         for i1 = i-1:-1:1
89             if ((vfront(i1) <= vrange(j)) && ...
90                 (vrange(j) <= vback(i1)))
91                 helpang2 = thetarange(i) - ...
92                     thetarange(i1);
93             else
94                 break;
95             end
96         end
97         deltatheta(i,j) = min(helpang1, helpang2);
98         thetaper(i,j) = deltatheta(i,j)/...
99                     thetarange(i)*100;
100     end
101     end
102     %Deviation with weighted error.
103     for i = 1:angsteps
104         for j = 1:vsteps
105             errortotal(i,j) = min(angw*thetaper(i,j),...
106                                 vw*vper(i,j));
107         end
108     end
109     end
110
111
112
113
114     figure(90+(ig - (rem(ig-1,numx*numy)+1))/numx*numy+2)
115     subplot(numx, numy, rem(ig-1, numx*numy)+1)
116     contourf(vrange, thetarange, errortotal, 10)
117     title('Shaq weighted error');
118     xlabel('Velocity');
119     ylabel('Angle');
120     hh=colorbar;

```

D Probabilistic analysis codes.

D.1 Monte Carlo procedure (montecarlo.m)

```

1 % Kevin Chamorro.
2 % 07/03/2019.
3 % Yachay Tech University.
4
5 clear
6 clc
7 initialize
8 % Program to calculate the simulations of a free throw basketball.
9 angstart = 34;
10 angend = 64;
11 angsteps = 200;
12 vstart = 6.5;
13 vend = 7.8;
14 vsteps = 400;
15
16 sol = zeros(1, 7);
17 soltheta = sol;
18 solthetavopt = sol;
19
20 thetarange = linspace(angstart, angend, angsteps);
21 vrange = linspace(vstart, vend, vsteps);
22
23 vfront = thetarange;
24 vcenter = thetarange;
25 vback = thetarange;
26 %Routine to create the feasible region
27 opt = optimset;
28 for i = 1:length(thetarange)
29     vfront(i) = fzero(@frontfzerov, vfrontrim(...
30                     thetarange(i)), opt, thetarange(i));
31     vcenter(i) = vcenterim(thetarange(i));
32     vback(i) = vbackrim(thetarange(i));
33 end
34 %Feasible region
35 plot(vfront, thetarange, vcenter, thetarange, vback, ...
36      thetarange)
37
38 %Simulations of the trajectories.
39 %z11=normrnd(50,20,1,100);
40 sigma1=sqrt(3.5); mu1=52.37; %Optimal angle
41 z1=normrnd(0,1,1,1000); % 1000 throws
42 %z22=normrnd(6.5,1,1,100);

```

```

43 sigma2=sqrt(0.001);mu2=6.69; %Optimal velocity
44 z2=normrnd(0,1,1,1000);
45 p=0.95;
46 v0=[];
47 v1=[];
48 successes = [];
49 failures = [];
50
51
52 for i=1:length(z1)
53     ang=sigma1*z1(i)+mu1;
54     ve0=sigma2*(p*z1(i)+z2(i)*sqrt(1-p^2))+mu2;
55     angv0=[ang, ve0];
56     %result =[result, ((ve0*cos(ang)/-g))*(ve0*sin(ang)+sqrt(ve0^2*sin(ang)^2+(2*g*h)))]];
57     throw = success(ang, ve0);
58     if(throw==0)
59         failures = [failures, ang];
60         v0=[v0, ve0];
61         % resultf=[resultf, result(i)];
62
63
64     else
65         successes = [successes, ang];
66         v1=[v1, ve0];
67         %results=[results, result(i)];
68
69     end
70 end
71
72 hold on
73 %Trajectories
74 plot(v0, failures, 'r');
75 plot(v1, successes, 'b');
76 %Target point
77 plot(mu2, mu1, 'g')
78
79
80 successes=sort(successes);
81 v1=sort(v1);
82
83 mu1=mean(successes);
84 sigma1=std(successes);
85 mu2=mean(v1);
86 sigma2=std(v1);
87 zv=[];
88 zthe=[];
89
90
91 for m=1:length(v1)
92     z1=(v1(m)-mu2)/(sigma2);
93     zv=[zv, z1];
94     z2= (successes(m)-mu1)/(sigma1);
95     zthe=[zthe, z2];
96 end
97
98
99 mu=[mu1 mu2];
100 sigma=[sigma1^2 p*sigma1*sigma2; p*sigma1*sigma2 sigma2^2];
101 [X1, X2] = meshgrid(successes(1,:), v1(1,:));
102 X=[X1(:) X2(:)];
103 % probability density function.
104 pdf=mvnpdf([X1(:) X2(:)], mu, sigma);
105 % cumulative distribution function.
106 cdf=mvncdf(X, mu, sigma);
107 y=size(X1);
108 Z=reshape(pdf, y);
109 figure;
110 subplot;
111 % Probability density function
112 plot3(X1, X2, Z)

```

D.2 Contour plot of score probability distribution (mc.m)

```

1 % Kevin Chamorro.
2 % 07/03/2019.
3 % Yachay Tech University.
4
5 clear all
6 clc
7
8 global Dr Db l h g ballstyle rimstyle centerstyle

```

```

9
10 initialize
11
12 angstart = 34;
13 angend   = 64;
14 angsteps = 100;
15
16 plotangstart = 40;
17 plotangend   = 58;
18 vstart      = 6.5;
19 vend        = 7.8;
20 vsteps      = 100;
21
22 thetarange = linspace(angstart, angend, angsteps);
23 vrangle    = linspace(vstart, vend, vsteps);
24
25 ig = 1;
26
27 deltav = zeros(angsteps, vsteps);
28 deltatheta = deltav;
29 thetaper  = deltav;
30 vper      = deltav;
31
32 vfront = thetarange;
33 vcenter = thetarange;
34 vback  = thetarange;
35 %Routine to create the feasible region
36 opt = optimset;
37 for i = 1:length(thetarange)
38     vfront(i) = fzero(@frontzerov, vfrontrim(...
39                     thetarange(i)), opt, thetarange(i));
40     vcenter(i) = vcenterrim(thetarange(i));
41     vback(i)   = vbackrim(thetarange(i));
42
43
44 end
45 %Monte Carlo siumlation procedure
46
47     for j = 1:vsteps
48         for i = 1:angsteps
49             if ((vfront(i) <= vrangle(j)) && ...
50                 (vrangle(j) <= vback(i)))
51
52                 helpang1 = thetarange(i);
53                 helpvel  = vrangle(j);
54                 p(i,j)=montecarlo1(helpang1, helpvel);
55
56             else
57                 p(i,j) = 0;
58             end
59         end
60     end
61
62 contourf(vrangle, thetarange, p, 10)
63 title('Contour plot of score probability distribution');
64 xlabel('Velocity');
65 ylabel('Angle');
66 hh = colorbar;

```

D.2.1 Program to calculate simulated data with a Monte Carlo simulations (montecarlo1.m)

```

1 % Kevin Chamorro.
2 % 07/03/2019.
3 % Yachay Tech University.
4
5 %Program to calculate simulated data with a Monte Carlo simulations with a
6 %target point fixed.
7 function probability= montecarlo1(mu1,mu2)
8 initialize
9 %z11=normrnd(50,20,1,100);
10 sigma1=sqrt(4) ;
11 z1=normrnd(0,1,1,100);
12 %z22=normrnd(6.5,1,1,100);
13 sigma2=sqrt(0.001);
14 z2=normrnd(0,1,1,100);
15 p=0.95;
16 ang0=[];
17 v0=[];
18 v1=[];
19 ST=[];
20 successes = [];
21 failures = [];

```

```

22 z=[];
23 for i=1:length(z1)
24     ang=sigma1*z1(i)+mul;
25     ve0=sigma2*(p*z1(i)+z2(i)*sqrt(1-p^2))+mu2;
26     angv0=[ang0,v0];
27     throw = success(ang,ve0);
28     if(throw==0)
29         failures = [failures ,ang];
30         v0=[v0,ve0];
31     else
32         successes = [successes ,ang];
33         v1=[v1,ve0];
34     end
35 end
36 successes=sort(successes);
37 v1=sort(v1);
38 probability= length(successes)/(100);
39
40 end

```

D.2.2 Probability in each point of the feasible region (success.m)

```

1 % Kevin Chamorro.
2 % 07/03/2019.
3 % Yachay Tech University.
4
5 function ring = success(theta0,v0)
6
7 global Dr Db l h g ballstyle rimstyle centerstyle
8
9
10 % Check if the angle is in the valid range, that is, between
11 % 10 and 85 degrees.
12 if ((theta0 < 10) || (theta0 > 85))
13     sprintf(...
14         'Angle theta0 = %5.2f is either too small or too large.'...
15         ,theta0);
16     ring = 0; % The ball is too far from the back of the
17               % rim to still be in the rim.
18     return
19 end
20
21
22 % Calculate this value since it is used several times below.
23 sintheta = sin(theta0*pi/180);
24
25 % Calculate the horizontal position of the ball as it comes
26 % back down to the basket height.
27 x = v0 * cos(theta0*pi/180)/(-g);
28 x = x*(v0*sintheta + sqrt(v0*v0*sintheta*sintheta+2*g*h));
29
30 if (x < l - Dr/2+Db/2)
31     sprintf('The ball does not reach the basket. ');
32     ring = 0; % The ball is too far from the back of the
33               % rim to still be in the rim.
34     return
35 end
36 if (x > l + Dr/2 - Db/2)
37     sprintf('The ball goes too far. ');
38     ring = 0; % The ball is too close to the back of the
39               % rim or behind the back of the rim.
40     return
41 end
42
43 % Check to see if the ball hits the front of the rim. This is
44 % done by calculating the minimum distance the ball has from
45 % the front of the rim.
46 % If this distance is below 0, the ball hits the front rim.
47 interm = frontfzero(theta0, v0);
48 if (interm < 0) % The ball hits the front rim.
49     sprintf('The ball hits the front of the rim. ');
50     ring = 0;
51     return
52 end
53 ring = 1;
54 end

```

E Statistical Analysis

Player	Release Angle	Release Velocity	Horizontal Distance
1.90	53.834	6.907	3.992
1.90	53.624	6.906	3.991
1.90	53.545	6.804	3.901
1.90	53.574	6.801	3.870
1.90	53.714	6.892	3.943
1.90	53.754	6.959	3.989
1.90	53.513	6.915	3.987
1.87	54.147	6.992	4.038
1.87	54.128	6.993	4.037
1.87	53.987	6.998	4.059
1.87	53.946	7.000	4.090
1.87	53.890	6.989	4.042
1.87	54.145	6.980	4.044
1.87	54.017	6.991	4.085
1.87	53.954	6.994	4.074
1.87	54.008	6.987	4.080
1.58	55.332	7.303	4.201
1.58	55.234	7.293	4.174
1.58	55.155	7.278	4.181
1.58	55.603	7.289	4.193
1.64	55.166	7.229	4.150
1.64	54.891	7.237	4.161
1.64	55.208	7.232	4.174
1.64	55.002	7.235	4.188
1.68	55.023	7.213	4.155
1.68	54.893	7.202	4.140
1.68	55.100	7.215	4.135
1.68	54.712	7.210	4.178
1.68	55.108	7.199	4.189
1.73	54.592	7.145	4.121
1.73	54.856	7.129	4.149
1.73	54.989	7.138	4.157
1.73	54.501	7.124	4.168
1.73	54.634	7.132	4.138
1.75	54.814	7.109	4.123
1.75	54.521	7.115	4.130
1.75	54.601	7.098	4.144
1.75	54.782	7.107	4.153
1.75	54.583	7.111	4.149
1.79	54.686	7.061	4.124
1.79	54.342	7.059	4.112
1.79	54.633	7.065	4.147
1.79	54.702	7.063	4.151
1.79	54.603	7.067	4.088
1.79	54.659	7.058	4.091
1.79	54.493	7.060	4.128
1.83	54.251	7.035	4.112
1.83	54.508	7.031	4.119
1.83	54.310	7.029	4.083
1.83	54.112	7.033	4.105
1.83	54.498	7.034	4.108
1.83	54.602	7.030	4.094

(a) Table of success throws

Player	Release Angle	Release Velocity	Horizontal Distance
1.90	51.148	6.627	3.870
1.90	50.874	6.641	3.889
1.90	51.924	6.650	3.895
1.87	51.947	6.674	3.920
1.58	59.332	7.778	4.320
1.58	58.480	7.698	4.325
1.58	59.934	7.653	4.298
1.58	58.765	7.871	4.305
1.58	58.021	7.606	4.311
1.58	58.239	7.751	4.293
1.64	56.954	7.549	4.287
1.64	58.342	7.443	4.269
1.64	60.383	7.881	4.292
1.64	59.096	7.638	4.274
1.64	58.871	7.701	4.301
1.64	57.325	7.403	4.310
1.68	57.324	7.691	4.241
1.68	59.095	7.829	4.254
1.68	58.430	7.687	4.260
1.68	57.459	7.478	4.267
1.68	58.239	7.788	4.277
1.73	56.398	7.270	4.222
1.73	55.436	7.281	4.240
1.73	54.411	7.240	4.219
1.73	56.897	7.262	4.235
1.73	56.231	7.292	4.214
1.75	55.987	7.213	3.955
1.75	56.154	7.290	3.968
1.75	57.583	7.189	3.948
1.75	54.183	7.055	3.979
1.75	54.397	7.001	3.960
1.79	55.032	6.841	3.940
1.79	53.908	6.778	3.938
1.79	55.377	6.739	3.924
1.83	53.893	6.829	3.921
1.83	53.952	6.818	3.935
1.83	53.328	6.650	3.908
1.83	54.898	6.961	3.945

(b) Table of failure throws

Figure 27: table of throws.

Height (m)	Release angle (°)	Velocity (m/s)	Horizontal distance (m)	Score
1.58	55.331	7.291	4.187	0.40
1.64	55.067	7.233	4.168	0.40
1.68	54.967	7.208	4.159	0.50
1.73	54.714	7.134	4.147	0.50
1.75	54.660	7.108	4.140	0.50
1.79	54.588	7.062	4.120	0.70
1.83	54.380	7.032	4.104	0.60
1.87	54.025	6.992	4.061	0.90
1.90	54.018	6.815	3.953	0.78

(a) Table of success throws mean

Height (m)	Release angle (°)	Velocity (m/s)	Horizontal distance (m)	Score
1.58	58.795	7.726	4.309	0.6
1.64	58.495	7.603	4.289	0.6
1.68	58.109	7.695	4.260	0.5
1.73	55.875	7.269	4.226	0.5
1.75	55.661	7.150	3.962	0.5
1.79	54.772	6.786	3.934	0.3
1.83	54.018	6.815	3.927	0.4
1.87	51.947	6.674	3.920	0.1
1.90	51.315	6.639	3.885	0.2

(b) Table of failure throws mean

Figure 28: table of throws means.