The Science of the Free Kick: Analysis, Modeling and Optimization of Accuracy in Football

HALIMI Akram

2023-2024

1 Introduction

A direct or indirect free kick is awarded to a team when a player is fouled by an opponent. The most serious fouls result in a direct free kick, thus representing a major opportunity to score if the shot is well executed. However, even world-class players experience significant difficulty in scoring regularly from free kicks, as illustrated by the notorious example of Kylian Mbappé. Thus, this study focuses on improving individual player performance through an in-depth analysis and modeling of free kicks.

2 Problematic

How can the modeling and in-depth analysis of the free kick in football help improve the individual skills of players?

3 Study Plan

Our approach will proceed in several stages:

1. **Study and analysis of the physical phenomena** involved during the shot.
2. **Establishment of equations** governing the shot and their computer implementation.
3. **Experimentation and validation** of the computer model.
4. **Conclusion and perspectives**, highlighting the contributions and limits of our model.

4 Study and Analysis of Physical Phenomena

During a free kick, several forces influence the ball:

- **Weight**:
$$\vec{P} = m\vec{g}$$
where m is the mass of the ball and g the acceleration due to gravity.
- **Drag force**: When the ball moves through the air at speed v, the airflow around the ball becomes turbulent, characterized by the Reynolds number (Re):
$$Re = \frac{\rho d v}{\eta}$$
with d the diameter of the ball, ρ the density of air, and η the dynamic viscosity of air.

For a football with the following parameters: $d = 0.11\,m$, $\eta = 1.8 \times 10^{-5} Pa \cdot s$, $\rho = 1.2\,kg \cdot m^{-3}$ and $1 < v < 50\,m \cdot s^{-1}$, we find:

$$7.3 \times 10^3 < Re < 3.7 \times 10^5$$

In this domain, the drag force is given by:

$$\vec{F_d} = -\frac{1}{2}\rho A C_x v \vec{v}$$

where A is the frontal surface and C_x the drag coefficient, assumed constant in this study.

- **Magnus force**: When a ball spins while moving through a fluid, a pressure difference is created in the boundary layer around the ball, causing a deviation in trajectory. This force is described by:

$$\vec{F_{mag}} = \alpha \vec{\omega} \wedge \vec{v}$$

where $\vec{\omega}$ is the vector of rotation of the ball and α a coefficient related to the physical and aerodynamic characteristics of the ball.

In our modeling, the following expressions are used:

$$F_{MC} = \frac{1}{2}\rho A C_{MC} v^2 \quad \text{and} \quad F_{MB} = \frac{1}{2}\rho A C_{MB} v^2$$

with

$$C_{MC} = \frac{1}{2 + \frac{v}{Re_z}} \quad \text{and} \quad C_{MB} = \frac{1}{2 + \frac{v}{Re_y}}$$

where Re_z and Re_y are Reynolds numbers relative to the respective axes.

5 Detailed Equation Setup

We consider the football as the system studied in a terrestrial frame assumed to be Galilean. We clearly define the balance of external mechanical actions (BAME) acting on the ball:

$$\vec{P}, \quad \vec{f}, \quad \vec{F}_{MC}, \quad \vec{F}_{MB}$$

where:

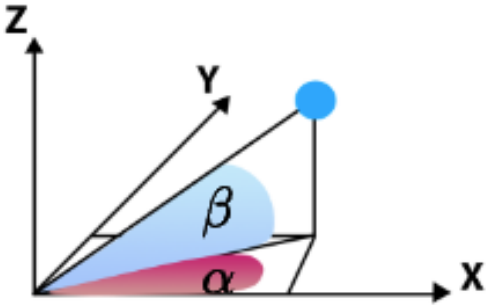
- $\vec{P}$ is the weight,
- $\vec{f}$ the drag force,
- $\vec{F}_{MC}$ and $\vec{F}_{MB}$ the horizontal and vertical components of the Magnus force.

To simplify the notations, we define the constant:

$$k = \frac{1}{2m}\rho A$$

where m is the mass of the ball, ρ the air density, and A the frontal surface of the ball.

The initial velocity of the ball is projected along the reference frame axes :



$$\begin{cases} v_x = v\cos(\beta)\cos(\alpha) \\ v_y = v\cos(\beta)\sin(\alpha) \\ v_z = v\sin(\beta) \end{cases}$$

where α and β are the angles defining the shot direction.

5.1 Fundamental Principle of Dynamics (FPD)

Applying the fundamental principle of dynamics, we obtain:

$$m\vec{a} = \vec{P} + \vec{f} + \vec{F}_{MC} + \vec{F}_{MB}$$

where $\vec{a}$ is the acceleration of the ball.

5.2 Projections on Axes

By projecting the forces on the reference frame axes, we obtain the following equations:

On the x-axis:

$$a_x = -kv(C_x v_x + C_{MC} v\sin(\alpha) - C_{MB} v_z \cos(\alpha))$$

On the y-axis:

$$a_y = -kv(C_x v_y - C_{MC} v\cos(\alpha) - C_{MB} v_z \sin(\alpha))$$

On the z-axis:

$$a_z = -kv(C_x v_z + C_{MB} v\cos(\beta)) - g$$

5.3 System of Differential Equations

Finally, the motion of the ball is described by a system of coupled second-order differential equations:

$$\begin{cases} \ddot{x} = -kv(C_x \dot{x} + C_{MC} v\sin(\alpha) - C_{MB} \dot{z}\cos(\alpha)) \\ \ddot{y} = -kv(C_x \dot{y} - C_{MC} v\cos(\alpha) - C_{MB} \dot{z}\sin(\alpha)) \\ \ddot{z} = -kv(C_x \dot{z} + C_{MB} v\cos(\beta)) - g \end{cases}$$

These equations will be solved numerically using the Runge-Kutta method during the computer implementation of the model.

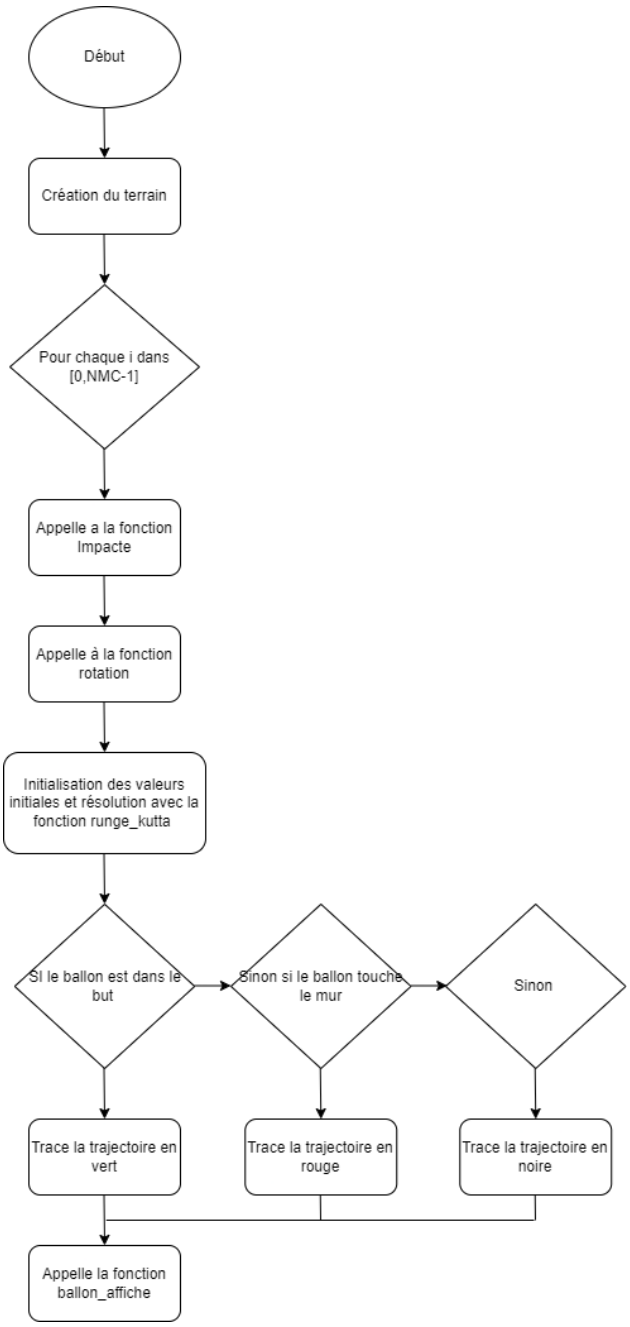
6 Detailed Computer Implementation

6.1 Assumptions and Details

We assume that the ball is a smooth sphere with a drag coefficient $C_x = 0.47$. The contact between the foot and the ball is assumed to be point-like, and the trajectories are computed under standard weather conditions with negligible wind.

6.2 Description of the Computer Program

Our computer program takes into account the force exerted by the foot on the ball to compute the coordinates of the point of impact and the optimal angles for an accurate shot, as illustrated in the general flowchart provided. The goal is to determine the optimal trajectory for a specific free kick, such as a top corner shot.



6.3 Impact Function: Monte-Carlo Simulation

The **impact** function forms the core of the statistical model. It generates random numbers following a normal distribution to simulate uncertainty in the point of impact on the ball. This function returns a percentage for the Y and Z coordinates of the point of impact, as well as the angles α and β needed to optimize the ball's trajectory.

```python
def Impact():
    '''
    Randomly generates impact parameters:
        Yc = y-coordinate of impact point as a percentage of radius
        Zc = z-coordinate of impact point as a percentage of radius
        alphaHit = shot angle in degrees (x-y plane)
        betaHit = shot angle in degrees (x-y-r plane)
    '''
    randomY = 0.1 * np.random.randn(1, 1)
    randomZ = 0.1 * np.random.randn(1, 1)
    randomalpha = 0.1 * np.random.randn(1, 1)
    randombeta = 0.1 * np.random.randn(1, 1)

    Yc = randomY[0][0]
    Zc = randomZ[0][0]
    alphaHit = randomalpha[0][0]
    betaHit = randombeta[0][0]

    return Yc, Zc, alphaHit, betaHit
```

6.4 Rotation Function: Calculation of Rotation Components

The **rotation** function determines the ball's rotation components using the concept of angular momentum:

$$\vec{J\omega} = \overrightarrow{OM} \wedge m\vec{v} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ a & b & c \\ mv_x & mv_y & mv_z \end{vmatrix} = \begin{pmatrix} bv_z - cv_y \\ cv_x - av_z \\ av_y - bv_x \end{pmatrix}$$

where (a, b, c) are the coordinates of the point of impact relative to the center of the ball.

We use the impulse formula to deduce the components of initial speed:

$$\left\{ v_x = \frac{F_x}{m}, \quad v_y = \frac{F_y}{m}, \quad v_z = \frac{F_z}{m} \right.$$

For a sphere, the moment of inertia J is:

$$J = \frac{2}{5}mr^2$$

Thus, the rotation components are written explicitly as:

$$\begin{cases} \omega_x = \frac{5}{2mr^2}(bF_z - cF_y) \\ \omega_y = \frac{5}{2mr^2}(cF_x - aF_z) \\ \omega_z = \frac{5}{2mr^2}(aF_y - bF_x) \end{cases}$$

6.5 Numerical Resolution by the Runge-Kutta Method

The previously obtained differential system is second order. To solve it numerically with the Runge-Kutta method, we first transform it into a first-order system. To do this, we introduce the intermediate variables:

$$\begin{cases} x_1 = \dot{x}, \quad y_1 = \dot{y}, \quad z_1 = \dot{z} \\ \dot{x}_1 = \ddot{x}, \quad \dot{y}_1 = \ddot{y}, \quad \dot{z}_1 = \ddot{z} \end{cases}$$

The initial system is then rewritten as:

$$\begin{cases} \dot{x}_1 = -kv(C_x x_1 + C_{MC} v \sin(\alpha) - C_{MB} z_1 \cos(\alpha)) \\ \dot{y}_1 = -kv(C_x y_1 - C_{MC} v \cos(\alpha) - C_{MB} z_1 \sin(\alpha)) \\ \dot{z}_1 = -kv(C_x z_1 + C_{MB} v \cos(\beta)) - g \end{cases}$$

This transformation facilitates the use of the Runge-Kutta numerical method, known for its stability and accuracy in solving differential equations.

```python
def runge_kutta_4th_order(f, y0, t_0, t_end, h, v, C_MC, C_MB, sinA, cosA, cosB, distance):
    '''
    Solves the system using 4th-order Runge-Kutta method.
    '''
    num_steps = int((t_end - t_0) / h) + 1
    t_values = np.linspace(t_0, t_end, num_steps)
    y_values = np.zeros((num_steps, len(y0)))
    y_values[0] = y0

    for i in range(1, num_steps):
        k1 = h * f(t_values[i - 1], y_values[i - 1], v, C_MC, C_MB, sinA, cosA, cosB)
        k2 = h * f(t_values[i - 1] + 0.5 * h, y_values[i - 1] + 0.5 * k1, v, C_MC, C_MB, sinA, cosA, cosB)
        k3 = h * f(t_values[i - 1] + 0.5 * h, y_values[i - 1] + 0.5 * k2, v, C_MC, C_MB, sinA, cosA, cosB)
        k4 = h * f(t_values[i - 1] + h, y_values[i - 1] + k3, v, C_MC, C_MB, sinA, cosA, cosB)

        y_values[i] = y_values[i - 1] + (k1 + 2 * k2 + 2 * k3 + k4) / 6

        # Check if the ball is out of bounds
        if (y_values[i, 0] < 0) or (y_values[i, 2] < -10) or (y_values[i, 2] > 10) or y_values[i, 0] > distance:
            return y_values[:i+1]

        # Check if the ball hits the ground
        if y_values[i, 4] < 0:
            return y_values[:i+1]

        # Check if the ball scores a goal
        if y_values[i, 0] == distance and (y_values[i, 4] > 2.44) and (-3.66 < y_values[i, 2] < 3.66):
            return y_values[:i+1]

        # Check if it hits the wall
        if y_values[i, 0] == 9.1 and ((y_values[i, 4] < 2.00) and (-1.5 < y_values[i, 2] < 0)):
            return y_values[:i+1]

    return y_values
```

7 Experiments and Model Validation

7.1 Experiment 1: Reproduction of Roberto Carlos' Shot

We reproduce the famous shot by Roberto Carlos taken from 35 meters with an estimated impulse of $16.72\,N.s$. For this, we use a Monte Carlo method by initially generating 1000 random shots following a normal distribution of impact parameters (coordinates and angles).

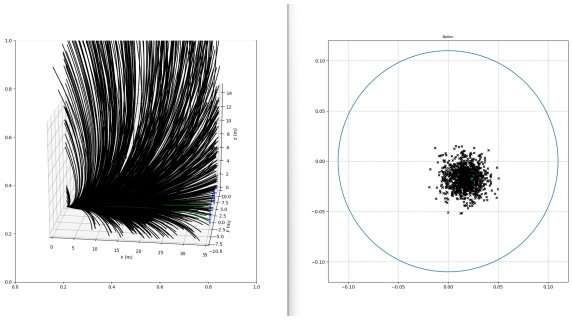
7.1.1 First Series of Shots

In the first series, we randomly generate:

$$
\begin{aligned}
Y &= 0.1 \times np.random.randn(1,1) + 0.16 \\
Z &= 0.1 \times np.random.randn(1,1) - 0.150 \\
\alpha &= 0.1 \times np.random.randn(1,1) - 14 \\
\beta &= 0.1 \times np.random.randn(1,1) + 7.5
\end{aligned}
$$

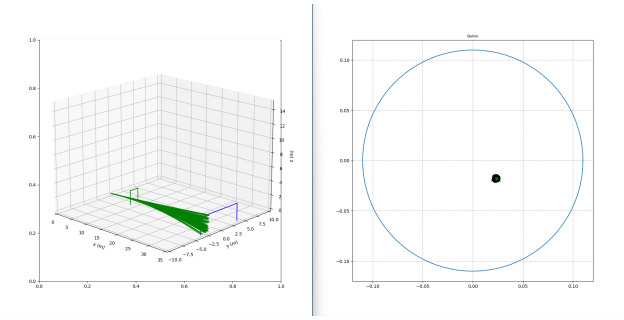
We obtain a wide dispersion of impact points, but with an initial identification of maximum values:

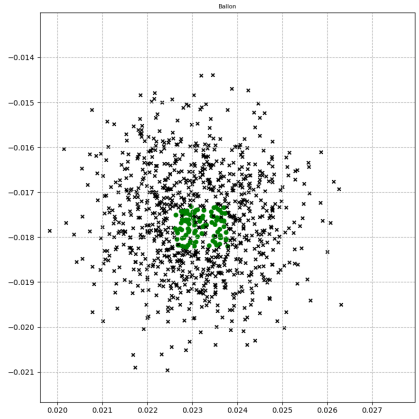
$$
\begin{aligned}
Y_{max} &= 0.21305, \quad Z_{max} = -0.16098 \\
\alpha_{max} &= -13.92386, \quad \beta_{max} = 7.65979
\end{aligned}
$$



7.1.2 Refinement and Second Series

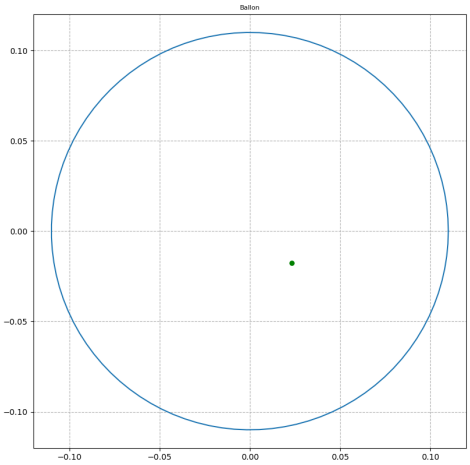
We refine by using the maximum values found as the mean of the new random distribution. This step allows us to concentrate results more toward a zone of optimal solutions, clearly visible on the graphs.

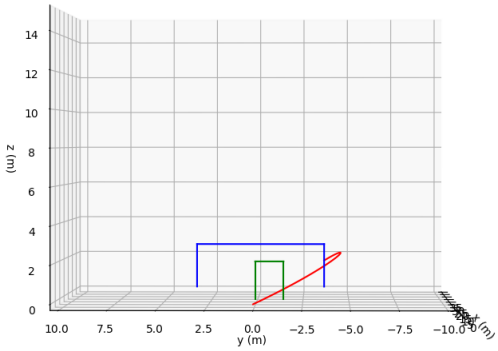




7.1.3 Final Series of Shots

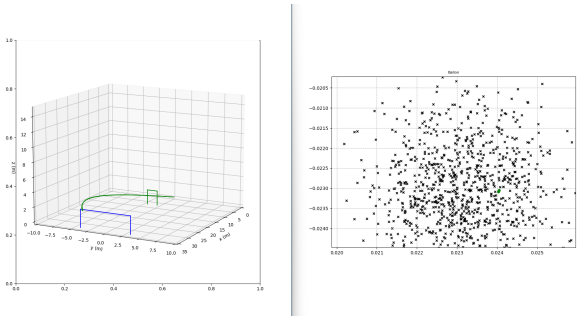
The final series shows a clear convergence of trajectories toward a stable solution. Final validation with the identified optimal parameters ($Y = 0.21$, $Z = -0.16$, $\alpha = -14$, $\beta = 7.66$) gives an initial speed of $38.7\, m \cdot s^{-1}$ and an initial angle very close to those reported in the literature ($12°$ and $38.1\, m \cdot s^{-1}$ respectively).

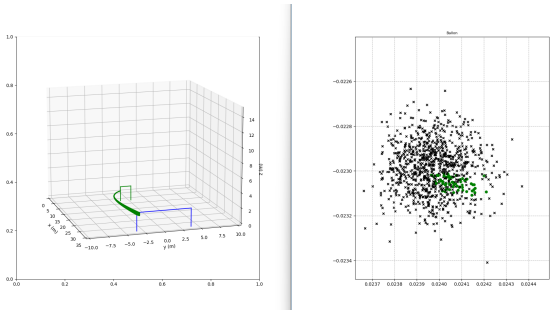




7.2 Experiment 2: Personal Optimization of a Free Kick

To optimize a shot toward the top right corner, we use Regressi software to precisely measure the initial speed of a real shot, estimated at $21.5\,m \cdot s^{-1}$. Three additional series of Monte Carlo simulations allow us to find an optimal solution after successive refinements of the random distributions of impact parameters.





8 Conclusion

Our implementation allows us to reproduce free kicks with a certain variability in the results. However, although it provides significant help in understanding and improving shots, its results remain limited. For example, among 20 shots performed following the program's predictions, only 3 were fully satisfactory. Thus, this model does not replace practical training but rather offers a support tool for improvement.

Many important factors remain excluded from our model: weather conditions, the actual texture of the ball, and the mental state of the player. These factors can significantly influence the ball's trajectory, as was the case during the 2010 World Cup with the famous Jabulani ball, whose unpredictable trajectories sparked much controversy.

Thus, our work provides a solid theoretical framework but highlights the importance of gradually integrating these additional variables to make the model even more realistic and effective.