Problema da trajetória da bola em rotação (Magnus Effect)

December 15, 2020

Este projeto possui como principal finalidade a consolidação de fundamentos aprendidos na disciplina de Introdução à Mecânica dos Fluidos, ministrada pelo professor José Miguel Aroztegui, de forma computacional, ou seja, uma prática utilizando códigos para o entendimento da problemática principal: trajetória de bola em rotação e o Magnus Effect. Serão apresentados as fundamentações teóricas, a resolução via código destes e os resultados obtidos durante a execução prática.

Autor: Epitácio Pessoa de Brito Neto

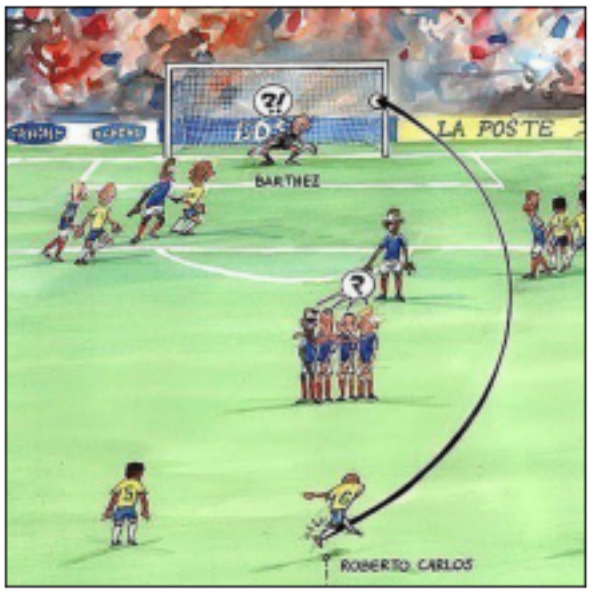
Matrícula: 11506856

0.1 1. Introdução

Uma das características mais fascinantes dos esportes que possuem uma bola como princípio, além da dinâmica instigante entre os times e o jogo, são as capacidades físicas dos atletas e a capacidade dos mesmos de aplicar conceitos físicos sem o conhecimento teórico que está por trás destas técnicas. Em especial o futebol, esporte tradicional brasileiro (que será o esporte-tema principal deste projeto), achamos impressionante quando jogadores profissionais aplicam efeitos diferentes em seus chutes, à exemplo do lateral esquerdo da Seleção Brasileira de 2002, Roberto Carlos e, também da Seleção Brasileira de 2002, Ronaldinho Gaúcho, utilizavam técnicas de efeito em seus chutes durante cobranças de falta impressionantes.

Para quem os conhece e os viram jogar, sabem de que efeito estamos falando. Mas como boa parte das coisas que nos rodeiam, este efeito que os dois jogadores foram reconhecidos dentro de seus repertórios, existem explicação científica, e é disso que se trata este projeto. Para explicar estes fenômenos, estudamos as forças aerodinâmicas que atuam sobre uma bola de futebol.

[14]:
```
imagem_path = 'DkQTtI3W0AAE0o6.jpg'

imagem = abrir_imagem(imagem_path)
imagem = np.array(imagem, dtype='int')
imagem  = np.clip(imagem, a_min = 0, a_max = 255)

fig = imshow(imagem)
fig.axes.get_xaxis().set_visible(False)
fig.axes.get_yaxis().set_visible(False)
```



Temos, a seguir, uma breve explicação de alguns conceitos que aparecem durante a situação:

I. Crise do arrasto: Ocorre quando a camada limite torna-se turbulenta. Esta turbulência permite que a camada resista melhor à tendência de separação, com isso, o ponto de descolamento move-se mais para trás da esfera, diminuindo a área da esteira. Uma esfera rugosa irá sofrer crise em um número de Reynolds (que será dito posteriormente) inferior ao de uma esfera lisa. A rugosidade precipita a turbulência na camada limite e, consequentemente, diminui a resistência do ar em altas velocidades;

II. Força de arrasto:

$$F_{arrasto} = \frac{1}{2} C A \rho A (nV)^2 (-V_{uni}),$$

onde nV é o módulo do vetor velocidade, ρ é a densidade do ar $(1,224 \frac{kg}{m^3})$, V_{uni} o vetor unitário e A a área da bola;

III. Efeito de Magnus (Magnus Effect): O efeito pode ser descrito na situação de quando a bola de futebol gira em torno de seu centro, uma força de sustentação (perpendicular à velocidade) passa a agir sobre ela, então, chamamo-as de força de Magnus. Esta força é escrita como:

$$\vec{F}_M = \frac{1}{2} C s A V^2 \frac{\vec{w} x \vec{V}}{|\vec{w} x \vec{V}|}$$

IV. Número de Reynolds: Mede a razão entre as forças inerciais e viscosas que atuam na bola, e indica o tipo de escoamento do fluido. É utilizado como única métrica na situação onde a bola possui velocidade muito menor que a do som. O número de Reynolds pode ser calculado por:

$$R_E = \frac{\rho D V}{n},$$

onde D é o diâmetro da bola, V é a velocidade em módulo da bola, ρ é a densidade do ar $(1,224\frac{kg}{m^3})$ e n é a viscosidade do ar $(1,83x10^{-5}kgm^{-1}s^{-1})$;

V. Coeficiente de arrasto de Morrison:

$$CA_{Morrison} = \frac{24}{R_E} + \frac{2,6(\frac{R_E}{5})}{1+(\frac{R_E}{5})^{1,52}} + \frac{0,411(\frac{R_E}{263000})^{-7,94}}{1+(\frac{R_E}{263000})^{-8}} + \frac{0,25(\frac{R_E}{10^6})}{1+(\frac{R_E}{10^6})},$$

onde $CA_{Morrison}$ é o coeficiente de arrasto e R_E = número de Reynolds, descrito anteriormente;

VI. Velocidade:

$$V = \frac{R_E n}{(\rho D)}$$

VII. Força de Sustentação:

$$F_{sustentacao} = \frac{1}{2} CM\rho Arcross(\Omega, V),$$

onde Ω é a velocidade angular, V a velocidade, A é a área da bola, ρ a densidade e CM o coeficiente de Magnus. Quanto ao $cross()$, que será utilizado durante a parte de Código, vem da biblioteca do Python chamada Numpy e retorna o produto entre dois vetores.

0.2 2. Esquema numérico

Para a resolução computacional, foi escolhida a linguagem Python 3, dado que suas funcionalidades de sintaxe, bibliotecas disponíveis e complexidade trazem a problemática para uma situação mais simples. Como foram citadas anteriormente, as equações anteriores estarão descritas em sintaxe de programação para serem aplicadas. Dentro do programa, disponibilizado pelo professor Miguel Aroztegui como material complementar do trabalho, podemos alterar as circunstâncias que serão submetidas na simulação do caso real.

Dito isso, podemos simular situações mudando valores de algumas variáveis trabalhadas e declaradas durante o código. Dentre elas, poderemos alterar: Diâmetro da bola, coeficiente de Magnus, velocidade inicial (V_0), tempos inicial e final, altura, densidade do ar (ρ). A alteração destes deve ser feita de forma que a lógica física se mantenha, quanto à situação que você quer mostrar, como as características do esporte escolhido estejam mantidas (por exemplo, não mudar diâmetro ou massa da bola de futebol para valores que fujam da realidade) e, vale dizer, que estas alterações influenciarão, obviamente, no cálculo de boa parte das equações mencionadas anteriormente. Assim, temos o nosso código.

0.3 3. Código

Os imports das bibliotecas do código original disponibilizado pelo professor foram transferidos para essa cell a parte, dado que foi preciso definir uma função (abrir_imagem) para abrir as imagens ao longo do documento, tanto para expor a situação in loco quanto os gráficos de comportamento conseguidos pelo mesmo código via Spyder.

[8]:
```
%matplotlib inline
from numpy import *
from numpy.linalg import norm
```

```
import matplotlib.pyplot as plt
from PIL import Image
import os
import numpy as np
from matplotlib.pyplot import imshow

def abrir_imagem(path):
    filename, file_extension = os.path.splitext(path)

    imagem = Image.open(path)

    return imagem
```

[2]:
```
# número de Reynolds para esfera lisa com
# diametro D
# velocidade (módulo) V
# viscosidade visco
Re = lambda rho,D,V,visco: rho*D*V/visco

# velocidade em função de Re, rho, D e visco
Velocidade = lambda Re,rho,D,visco: Re*visco/(rho*D)

# O Coeficiente de Arrasto (CA) para bola lisa, para cada número de Re é medido␣
 ↪experimentalmente no tunel do vento.
# Contudo Morrison escreveu uma função que calcula o CA aproximadamente para␣
 ↪cada Re (1e-1<=Re<=1e6).
# Referência: F A. Morrison, Data Correlation for Drag Coefficient for Sphere,␣
 ↪http://www.chem.mtu.edu/~fmorriso/ DataCorrelationForSphereDrag2013.pdf
#CAMorrison = lambda re: 24/re + 2.6*(re/5)/(1+(re/5)**(1.52)) + 0.411*(re/
 ↪263000)**(-7.94)/(1+(re/263000)**(-8))+re**0.8/461000
CAMorrison = lambda re: 24/re + 2.6*(re/5)/(1+(re/5)**(1.52)) + 0.411*(re/
 ↪263000)**(-7.94)/(1+(re/263000)**(-8)) + (0.25*(re/1e6))/(1+(re/1e6))

# nReynolds=linspace(1e-1,1e6,1000000)
# plt.loglog(nReynolds,CAMorrison(nReynolds))
# plt.xlabel('Re')
# plt.ylabel('CA')



# exemplo: bola de futebol
rho=1.224 # kg/m3, densidade do ar
D=22e-2 # 22cm, diâmetro
r=D/2 # raio
A=pi*r*r
visco=1.83e-5 # kg/(ms), viscosidade do ar
```

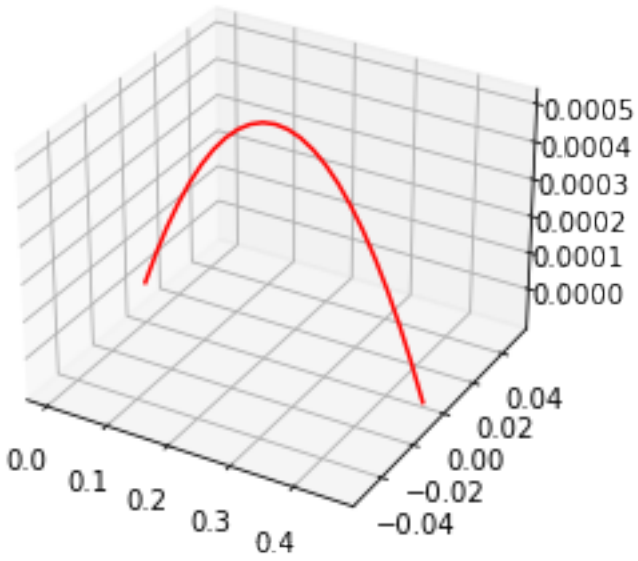
```
# Ren=0.5
# print('A velocidade para Re=%.1f é %.10f m/s.'␣
 ↪%(Ren,Velocidade(Ren,rho,D,visco)))
# Re1e3=1e3
# v1e3=Velocidade(Re1e3,rho,D,visco)
# Re1e5=1e5
# v1e5=Velocidade(Re1e5,rho,D,visco)
# print('A velocidade para Re=%.1f é %.10f m/s e para Re=%.1e é %.1f m/s.'␣
 ↪%(Re1e3,v1e3,Re1e5,v1e5))
# Ren=3e5
# v3e5=Velocidade(Ren,rho,D,visco)
# print('A velocidade para Re=%.1e (crise do arrasto) é %.1f m/s.' %(Ren,v3e5))



# bola de futebol, ar
massa=0.454 # kg, massa da bola
CM=1 # coeficiente de Magnus
g=9.81*array([0, 0, -1]) # m/s2, aceleração da gravidade
#v0=array([27.8, 0, 8.8]) # m/s, Pelé, velocidade inicial
#v0=1*[2 0 1]'
v0=array([20,0,.1])
rey=Re(rho,D,norm(v0),visco)
if rey<1e-1 or rey>1e6:
  print('A formula de Morrison pode nao ser aplicavel...\n')
p0=array([0, 0, 0]) # m, posição inicial
#omega=6.84*[0 -1 0]' # Pelé chuta de baixo, Hz, velocidade de rotação
#omega=2*v0 # se v0 é pra cima => quando desce folha seca?, Hz, velocidade de␣
 ↪rotação
#omega=[20 0 0]'
#omega=cross(v0,array([0, 1, 0])) # se chutar no lado direito da bola, vai pra␣
 ↪esquerda, Hz, velocidade de rotação
omega=array([0,-10,0])



# força de arrasto
# CA coefiente de arrasto
# rho densidade
# A área da bola
# nv módulo da velocidade da bola
# vuni vetor unitário na mesma direção e sentido que a velocidade
FArrasto = lambda CA,rho,A,nv,vuni: 0.5*CA*rho*A*nv*nv*(-vuni)

# força de sustentação, fórmula (6)
# CM coefiente de Magnus
# rho densidade do ar
```

```
# A área da bola
# r raio da bola
# omega velocidade angular
# vel velocidade
FSustentacao = lambda CM,rho,A,r,omega,vel: 0.5*CM*rho*A*r*cross(omega,vel)


# método de Euler
# para calcular a velocidade e posição a partir da aceleração
tini=0
tfin=10 # s
h=1e-3
n=int((tfin-tini)/h)
vel=v0
pos=zeros((3,n))
pos[:,0]=p0
tempo=0
for i in range(1,n-1):
    nv=norm(vel)
    if nv<1e-6: # nv=norma da velocidade, nv<1e-6 significa velocidade nula
        # Isto pode acontecer se quisermos simular a queda da bola de basquete␣
↪na represa: (velocidade inicial=0)
        # por exemplo https://www.youtube.com/watch?v=2OSrvzNW9FE
        # velocidade=0 produz Re=0 e CAMorrison(0) está indefinido!
        CA=0 # se velocidade é zero, não temos arrasto.
        vuni=vel*0
    else: # velocidade não nula
        CA=CAMorrison(Re(rho,D,nv,visco))
        vuni=vel/nv
    FA=FArrasto(CA,rho,A,nv,vuni)
    FS=FSustentacao(CM,rho,A,r,omega,vel)
    FG=massa*g
    vel=vel+(h/massa)*(FA+FS+FG)
    pos[:,i+1]=pos[:,i]+h*vel
    if pos[2,i+1]<0: # se a posição da bola estiver abaixo do chão
        #print(pos[2,i+1])
        break
    tempo=tempo+h
print('Tempo de voo: %.1e s.' %(tempo))
pos=pos[:,:i+2]

fig = plt.figure()
ax = plt.axes(projection='3d')
ax.plot3D(pos[0,:],pos[1,:],pos[2,:],'red')


# # desenha os eixos de coordenadas (0,X,Y,Z)
```

```
# maxpos=max(max(abs(pos)))
# c=maxpos/4
# cor='r'
# tamanho=3
# line([0 c],[0 0],[0 0],'Color',cor,'LineWidth',tamanho)
# h=text(c,0,0,'X')
# set(h,'fontsize',20)
# line([0 0],[0 c],[0 0],'Color',cor,'LineWidth',tamanho)
# h=text(0,c,0,'Y')
# set(h,'fontsize',20)
# line([0 0],[0 0],[0 c],'Color',cor,'LineWidth',tamanho)
# h=text(0,0,c,'Z')
# set(h,'fontsize',20)

# #view([0 0])
# axis equal
```

```
Tempo de voo: 2.2e-02 s.
```

[2]:
```
[<mpl_toolkits.mplot3d.art3d.Line3D at 0x2be713115e0>]
```

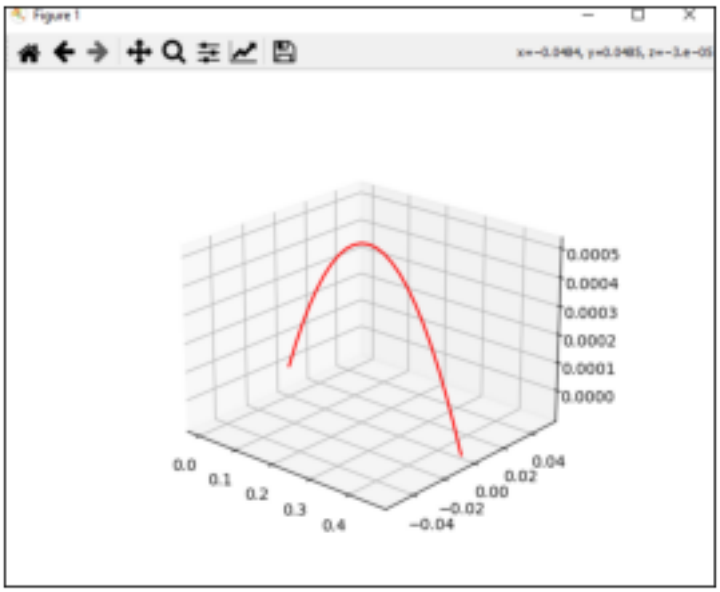


0.4 4. Resultados

É necessário dizer que as imagens para os resultados para os testes apresentados neste documento foram feitas pela IDE Spyder, de forma que pudéssemos alcançar uma visão melhor do que foi simulado.

0.4.1 Primeiro caso:

[17]:
```
imagem_path = 'primeiro caso.png'

imagem = abrir_imagem(imagem_path)
imagem = np.array(imagem, dtype='int')
imagem  = np.clip(imagem, a_min = 0, a_max = 255)

fig = imshow(imagem)
fig.axes.get_xaxis().set_visible(False)
fig.axes.get_yaxis().set_visible(False)
```



Neste caso fundamental, vemos que o trajeto da bola forma uma parábola praticamente perfeita. Para isso, temos os seguintes valores de variáveis e tempo de vôo resultante:

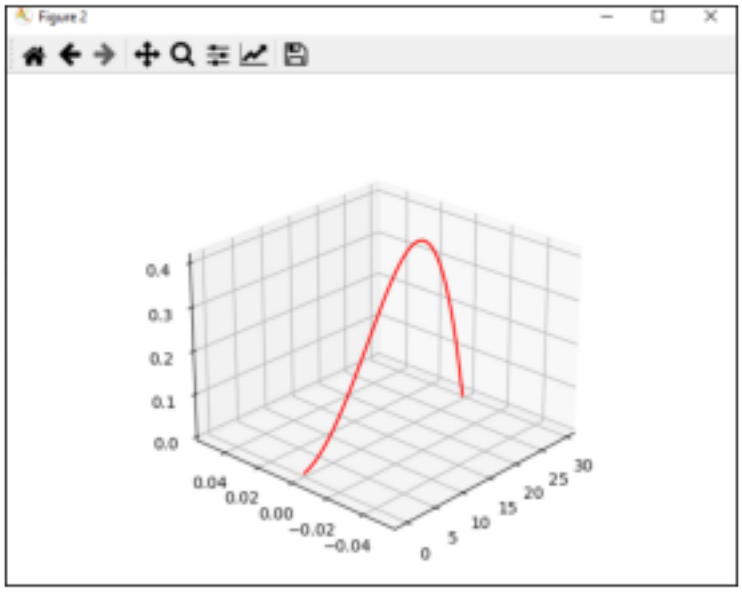
$Omega = array([0, -10, 0])$

$T_{voo} = 2,2x10^{-2}s$

0.4.2 Segundo caso (caso Pelé):

[18]:
```
imagem_path = 'segundo caso.png'

imagem = abrir_imagem(imagem_path)
imagem = np.array(imagem, dtype='int')
imagem  = np.clip(imagem, a_min = 0, a_max = 255)

fig = imshow(imagem)
```

```
fig.axes.get_xaxis().set_visible(False)
fig.axes.get_yaxis().set_visible(False)
```



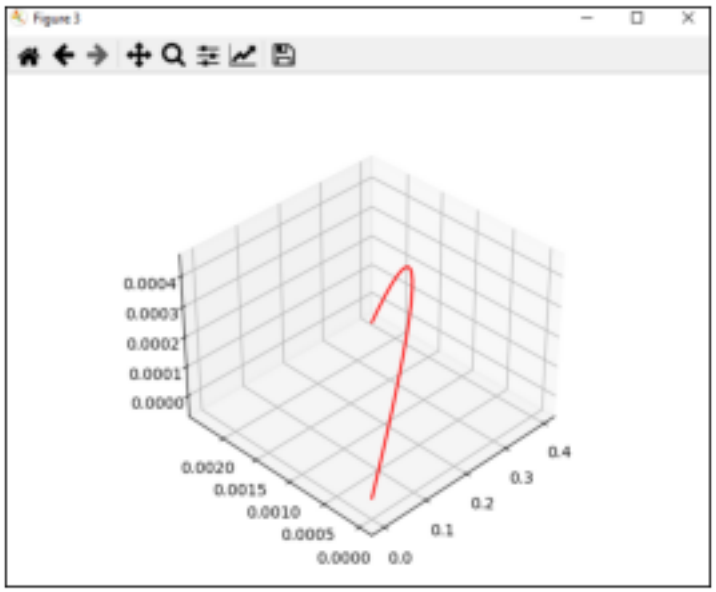
Para este caso, demonstramos a situação encontrada com o gol de Pelé, onde ele chuta na parte de baixo da bola e cria este efeito para ultrapassar o goleiro adversário por cima dele.

$Omega = array([0, -100, 0])$

$T_{voo} = 1,8s$

0.4.3 Terceiro caso (caso Roberto Carlos):

[19]:
```
imagem_path = 'terceiro caso.png'

imagem = abrir_imagem(imagem_path)
imagem = np.array(imagem, dtype='int')
imagem  = np.clip(imagem, a_min = 0, a_max = 255)

fig = imshow(imagem)
fig.axes.get_xaxis().set_visible(False)
fig.axes.get_yaxis().set_visible(False)
```



Neste terceiro caso, tentamos replicar a situação que chegasse próximo ao chute de falta feito pelo jogador Roberto Carlos, onde ele chuta a parte esquerda da bola e cria este efeito que, aos olhos dele, parece que a bola foi para a direita e termina o trajeto chegando à esquerda, quando, na verdade, em relação ao eixo y, a bola está fazendo um efeito de movimento para a esquerda. Os dados para esta aproximação são:

$Omega = array([0, 0, 100])$

$T_{voo} = 1,9x10^{-2}s$

0.5 5. Conclusão

Esta atividade prática, em termos de aplicação computacional, foi satisfatória no sentido de que pudemos juntar os conhecimentos aprendidos em sala de aula e aplicar em situações que levantam interesse (inclusive à quem vos escreve) aos que acompanham, neste caso específico, o futebol. Nos deparamos em algo que presenciamos toda semana na rotina de quem acompanha as partidas e, agora, podemos entender o que está por trás de tantos cruzamentos, chutes à gol e passes com efeitos tão diferentes que parecem ser mágica.

Há espaço para revisitar o programa que fora utilizado para diversas aplicações profissionais, bem como aplicações em esportes diferentes, como tênis, vôlei, basquete, entre outros. Além disso, esta aplicação teórica pode servir de aprendizado tanto para programadores e físicos, quanto à atletas que querem entender o que está por trás do que os mesmos aprenderam durante a prática, vivência, indução e instinto, de forma que possam melhorar suas habilidades focando na parte mais concreta que os cercam: a ciência que está por trás de tantas jogadas alcançadas por diversos super-atletas.

Quanto aos resultados, pudemos ver, com clareza, o que acontece durante a trajetória da bola durante situações diversas que foram abordadas. Vale dizer, para quem tem interesse de uma visão melhor, que utilize o código em alguma IDE que permita rotacionar os gráficos gerados em cada

situação, desta forma, a análise e visualização estará mais completa do que pôde ser apresentada neste relatório.

[15]:
```
imagem_path =␣
↪'20181208-636799053671283529_20181208223920557-kadD-U453442347832FVC-992x558@LaVanguardia-W
↪jpg'

imagem = abrir_imagem(imagem_path)
imagem = np.array(imagem, dtype='int')
imagem  = np.clip(imagem, a_min = 0, a_max = 255)

fig = imshow(imagem)
fig.axes.get_xaxis().set_visible(False)
fig.axes.get_yaxis().set_visible(False)
```

0.6 6. Referências bibliográficas

AGUIAR e RUBINI. A aerodinâmica da bola de futebol (Aerodynamics of the soccer ball). Instituto de Física, Universidade do Federal do Rio de Janeiro, Brasil, 2004;

MORRISON, Faith A. Data correlation for Drag Coefficient for Sphere. Michigan Technological University, Houghton, 2016;

Notas de aula do professor Miguel Aroztegui, na disciplina de Introdução à Mecânica dos Fluidos, ofertada pela Universidade Federal da Paraíba - UFPB, Brasil, 2020.