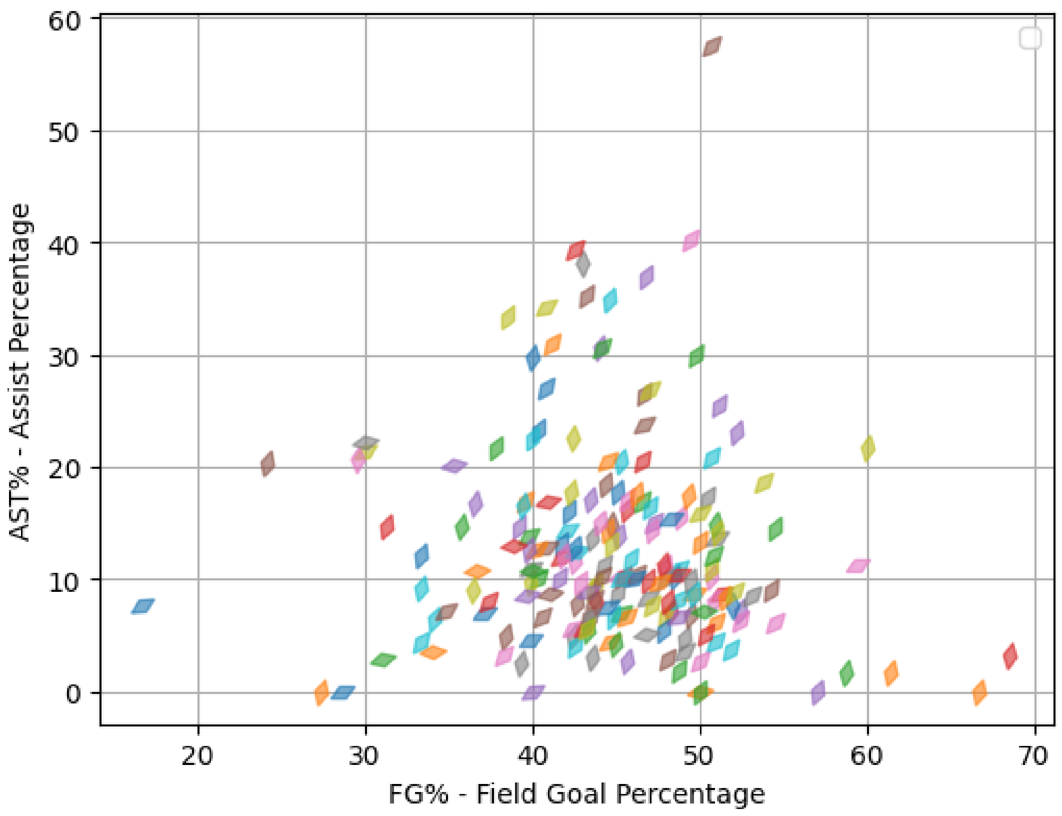
Exercício 3.1

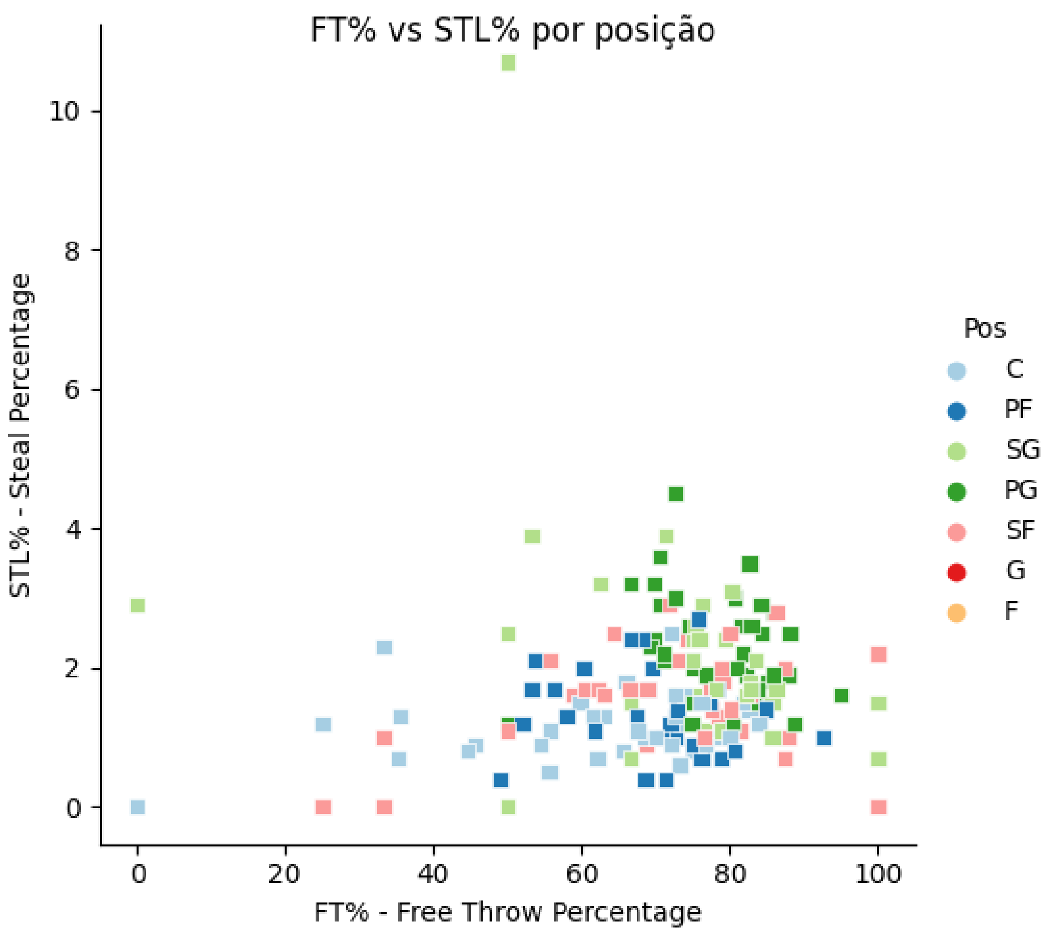
```
import pandas as pd
import numpy as np
import io
from matplotlib import pyplot as plt
from google.colab import files
uploaded = files.upload()
nba = pd.read_csv('/content/Seasons_Stats.csv')
nba = nba[(nba['FG%'] > 0) & (nba['FG%'] < 1) & (nba['TS%'] > 0) &
     (nba['TS%'] < 1) & (nba['2P%'] > 0) & (nba['2P%'] < 1)]
nba = nba.iloc[1000:, :].sample(100)

incl = np.array(nba.loc[:,'Age'])
x = np.array(nba.loc[:,'FG%'])*100
y = np.array(nba.loc[:,'AST%'])
z = np.array(nba.loc[:,'Tm'])

i = np.arange(0, len(incl), 1)
j = (incl - min(incl))/(max(incl) - min(incl))
w = -90*j
j = 0

for k in i:
  marker = (2, 1, w[k])
  if z[k] == z[k-1]:
    plt.plot(x[k], y[k], marker = marker, markersize = 10, alpha = 0.6)
  else:
    j +=1
    plt.plot(x[k], y[k], marker = marker, markersize = 10, alpha = 0.6)
plt.legend(scatterpoints = 1)
plt.xlabel('FG% - Field Goal Percentage')
plt.ylabel('AST% - Assist Percentage')
plt.fig.suptitle('FG% vs AST% com idade na inclinação')
plt.grid()
plt.show()
```

FG% vs AST% com idade na inclinação

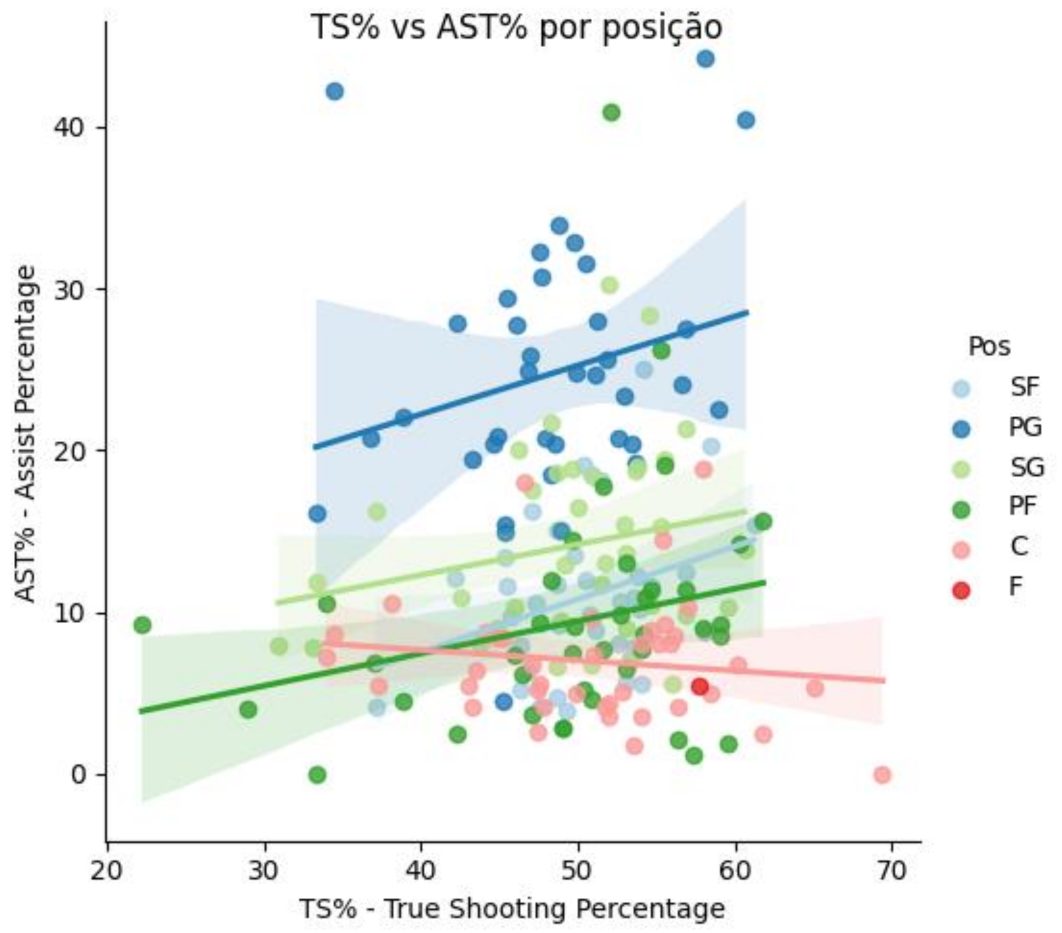


Na inclinação é apresentado a variável idade e nas cores a variável que representa o time de cada jogador.

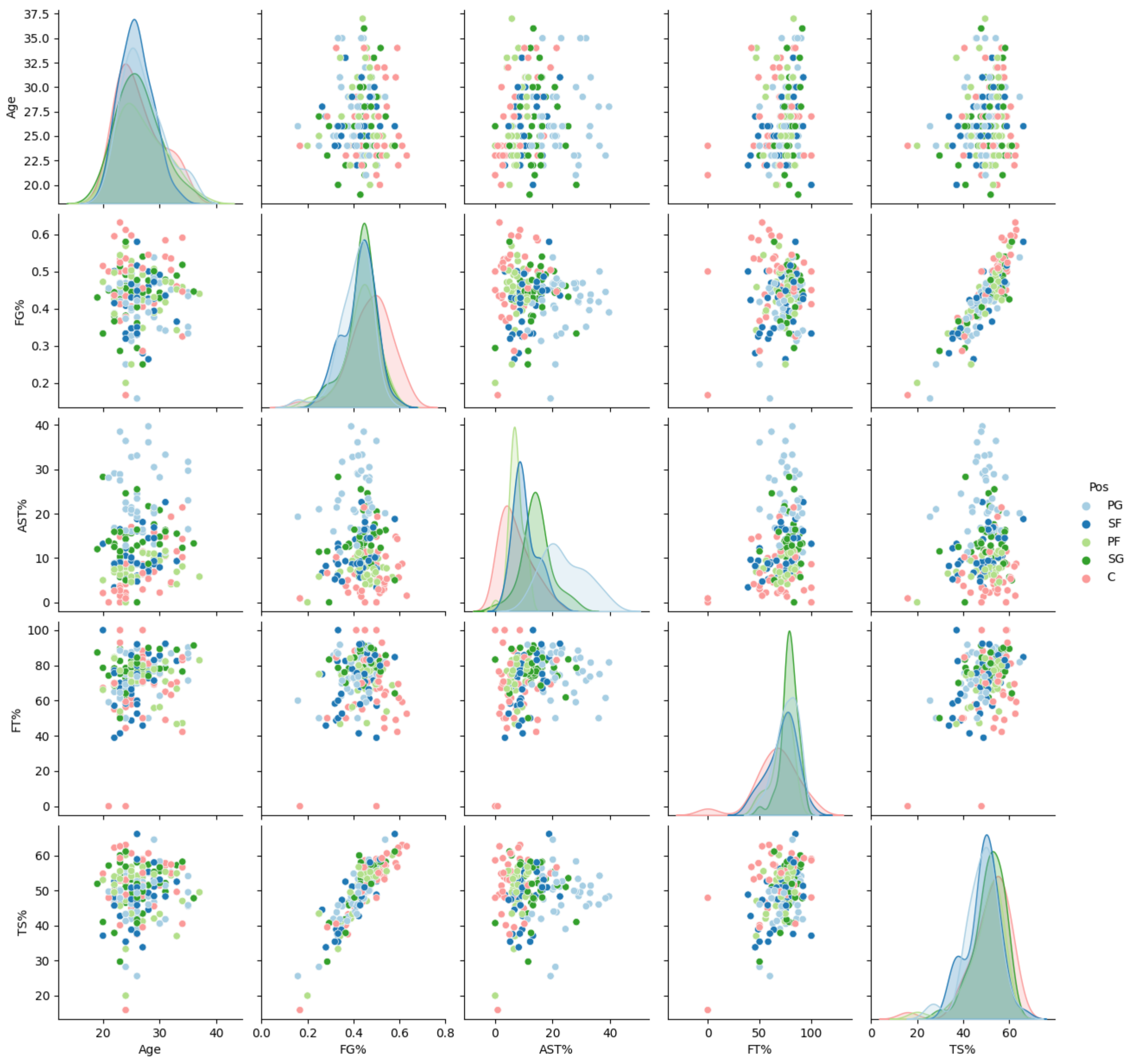
```
import seaborn as sns
nba['FT%'] = nba['FT%'] * 100
plot = sns.relplot(data=nba, x='FT%', y='STL%', hue='Pos', marker='s',
palette='Paired')
plot.set_axis_labels('FT% - Free Throw Percentage', 'STL% - Steal Percentage')
plot.fig.suptitle('FT% vs STL% by Position')
plt.show()
```



```
nba['TS%'] = nba['TS%'] * 100
plot = sns.lmplot(data=nba, x='TS%', y='AST%', hue='Pos', palette='Paired')
plot.set_axis_labels('TS% - True Shooting Percentage', 'AST% - Assist Percentage')
plot.fig.suptitle('TS% vs AST% por posição')
plt.show()
```

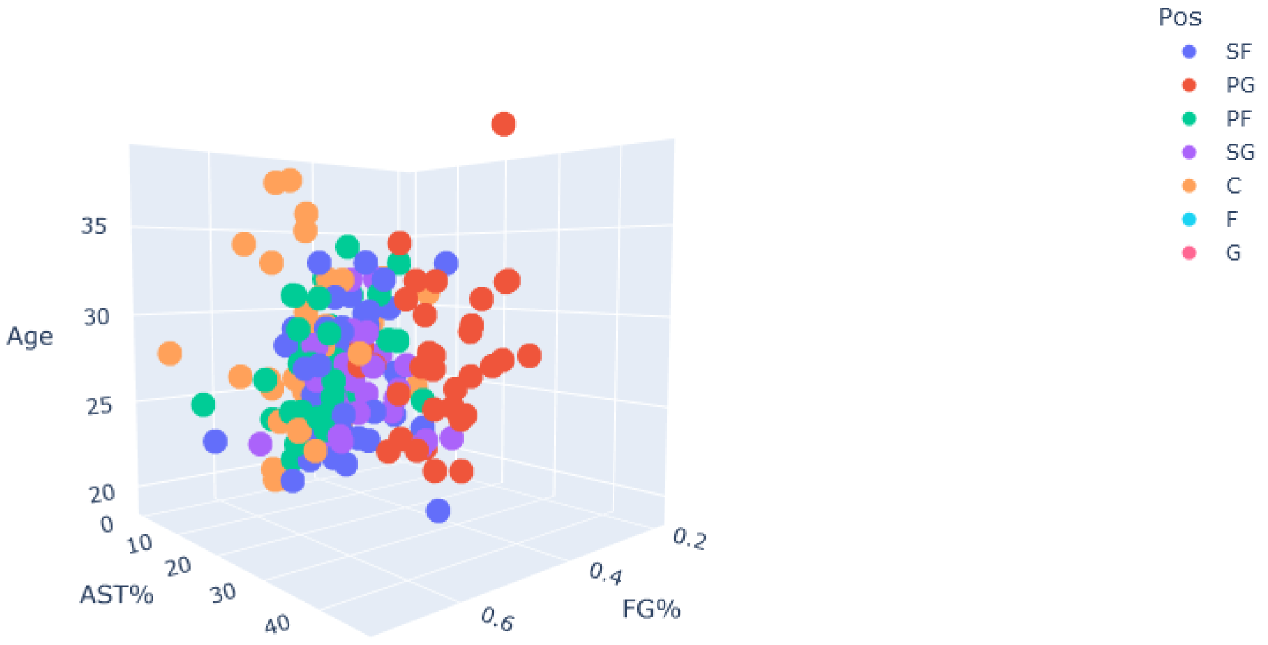


```
cols = ['Age', 'FG%', 'AST%', 'Pos', 'FT%', 'TS%']
nba = nba[cols]
sns.pairplot(data = nba, hue = 'Pos', palette = 'Paired')
```

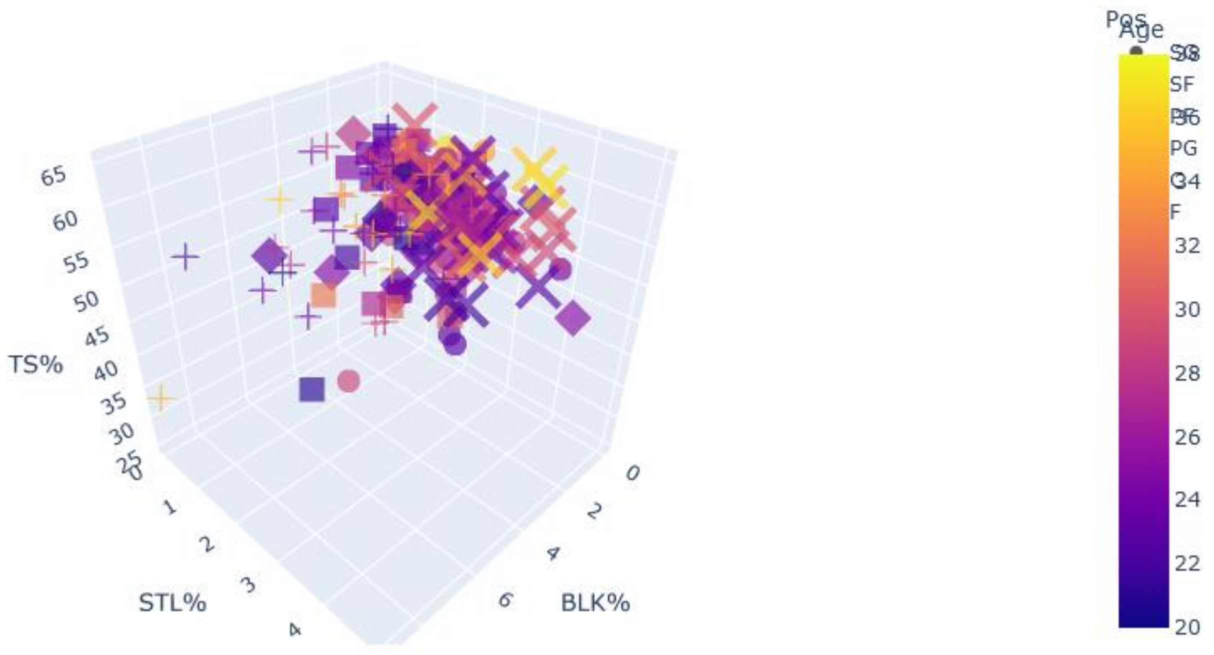


Exercício 3.2

```
import plotly.express as px

fig = px.scatter_3d(nba, x = 'FG%', y = 'AST%', z = 'Age', color = 'Pos')
fig.show()
```

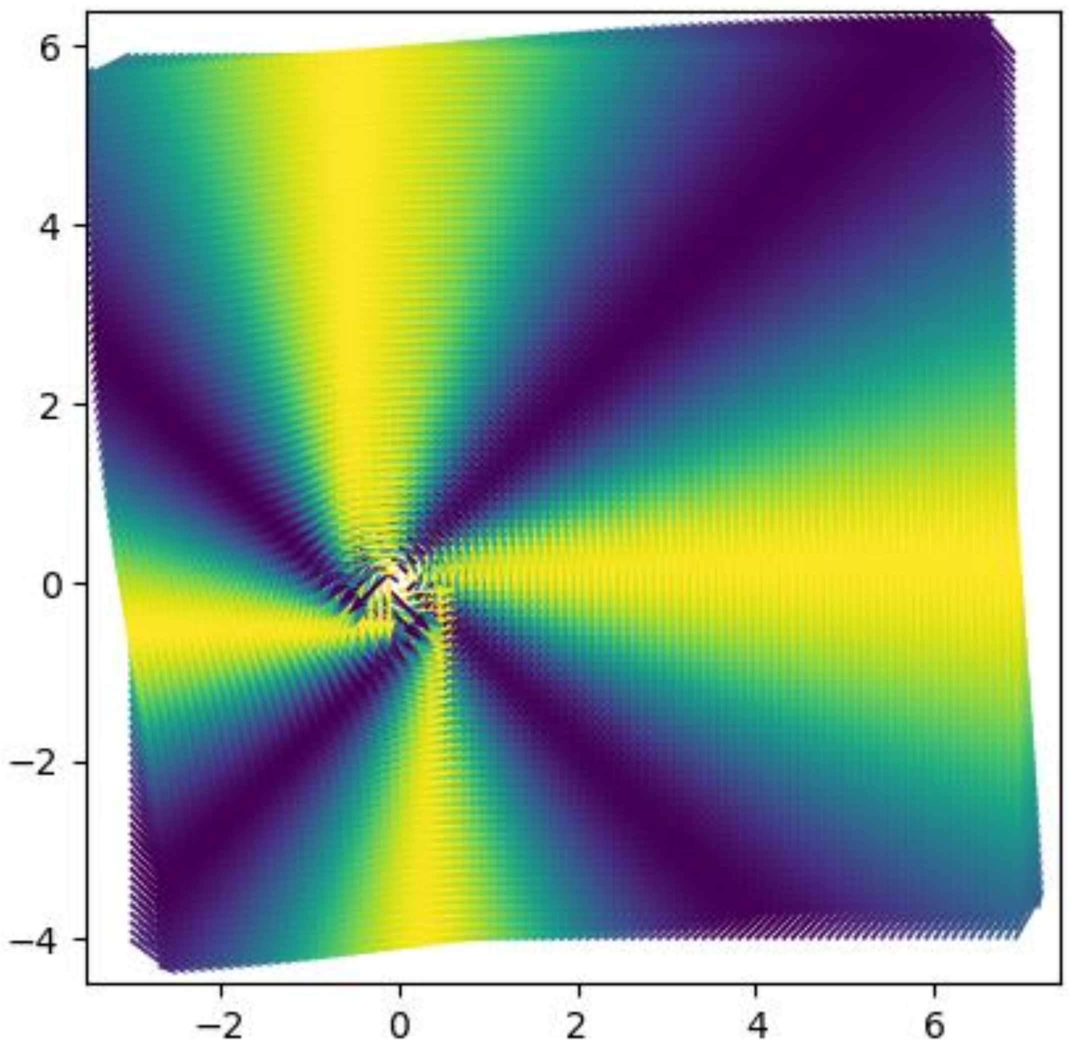


```
fig = px.scatter_3d(nba, x = 'BLK%', y = 'STL%', z = 'TS%', color = 'Age', symbol
= 'Pos', opacity = 0.65)
fig.show()
```

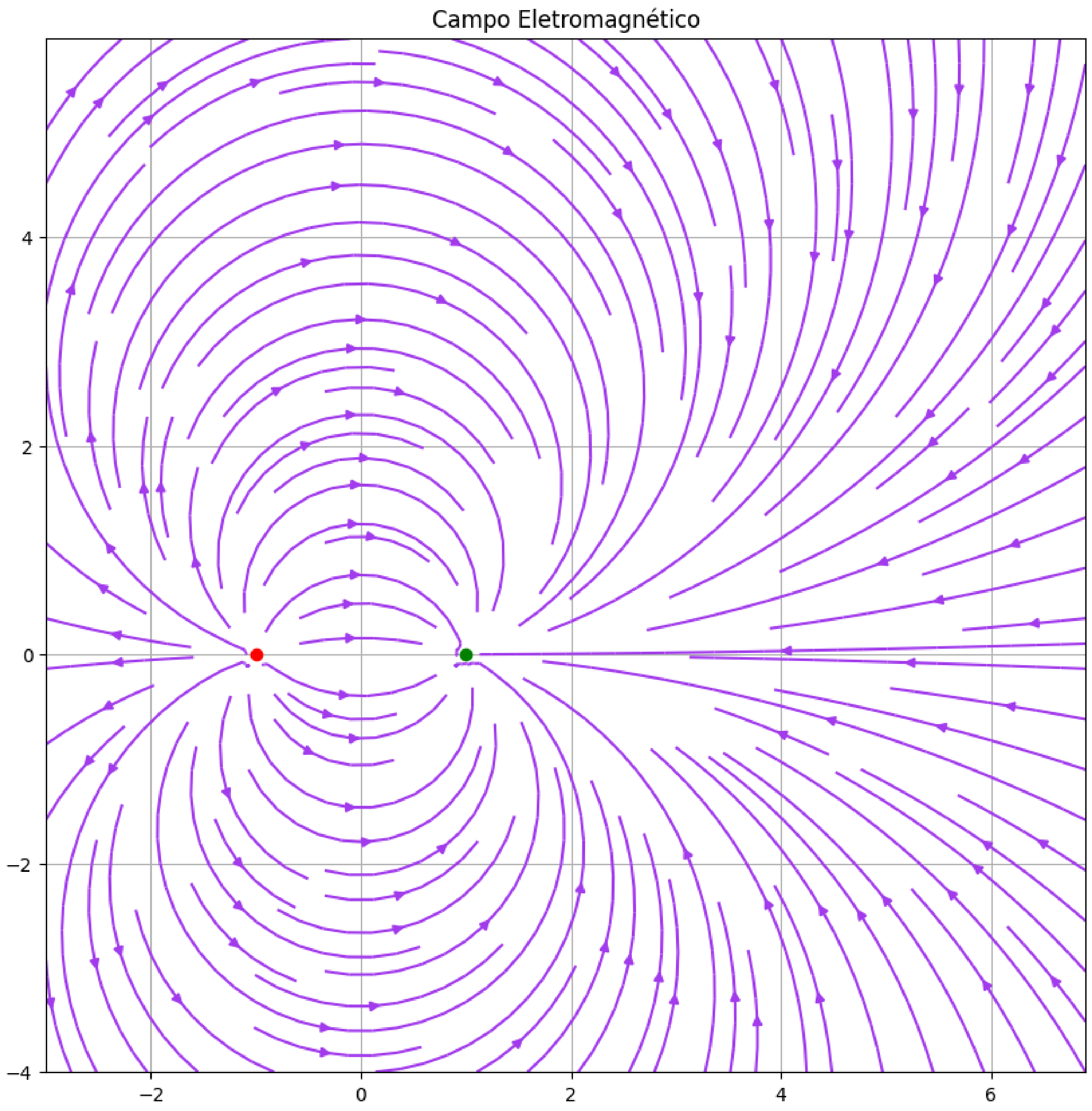


Exercício 3.3

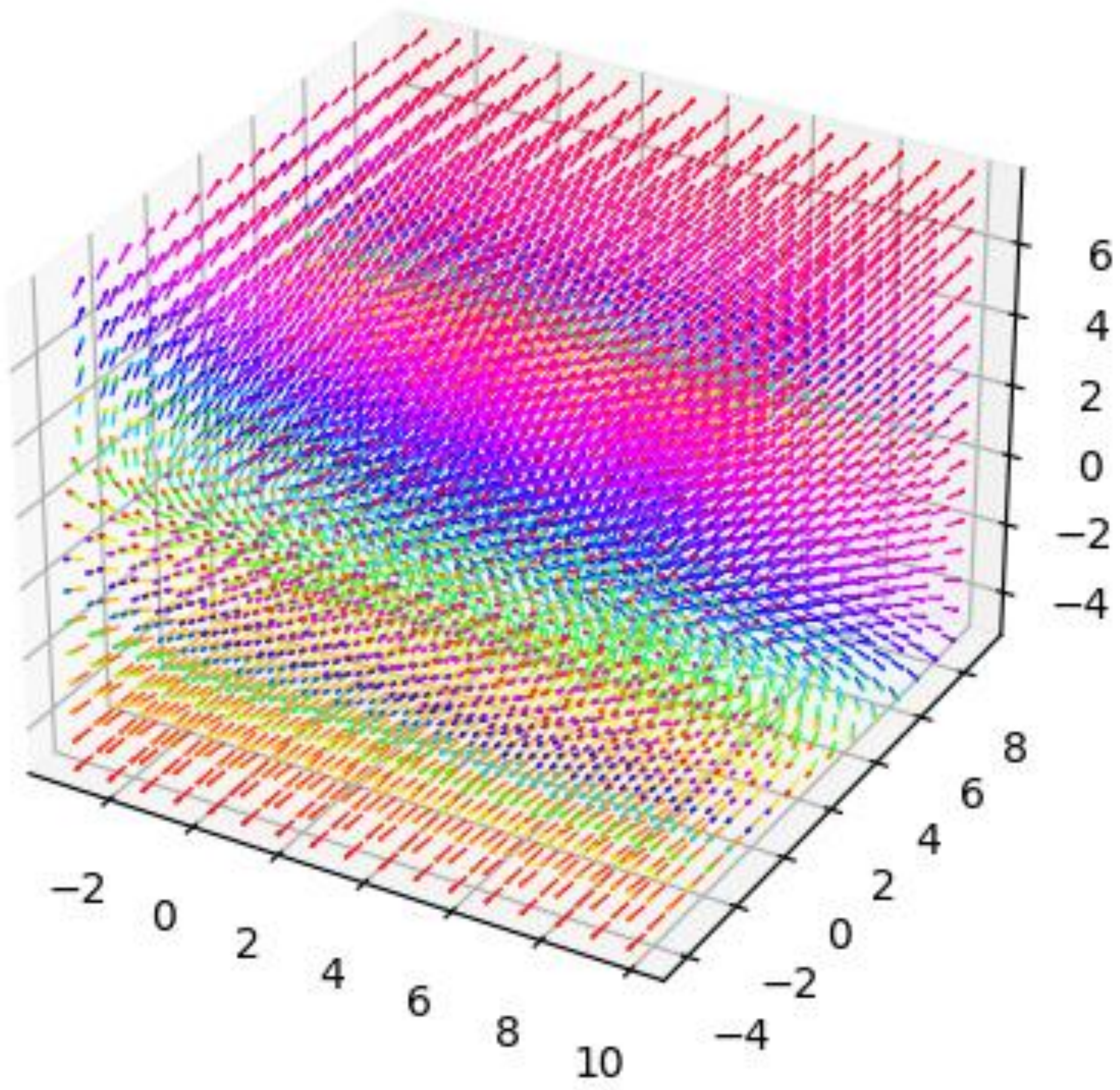
```
X, Y = np.meshgrid(np.arange(-3,7,0.1),np.arange(-4,6,0.1))
U = -Y/np.hypot(X, Y)
V = X/np.hypot(X, Y)
M = np.hypot(U**6, V**6)
fig, ax = plt.subplots()
Q = ax.quiver(X, Y, U, V, M, units = 'xy', width = 0.05, scale = 1.5)
ax.set_aspect('equal', 'box')
```



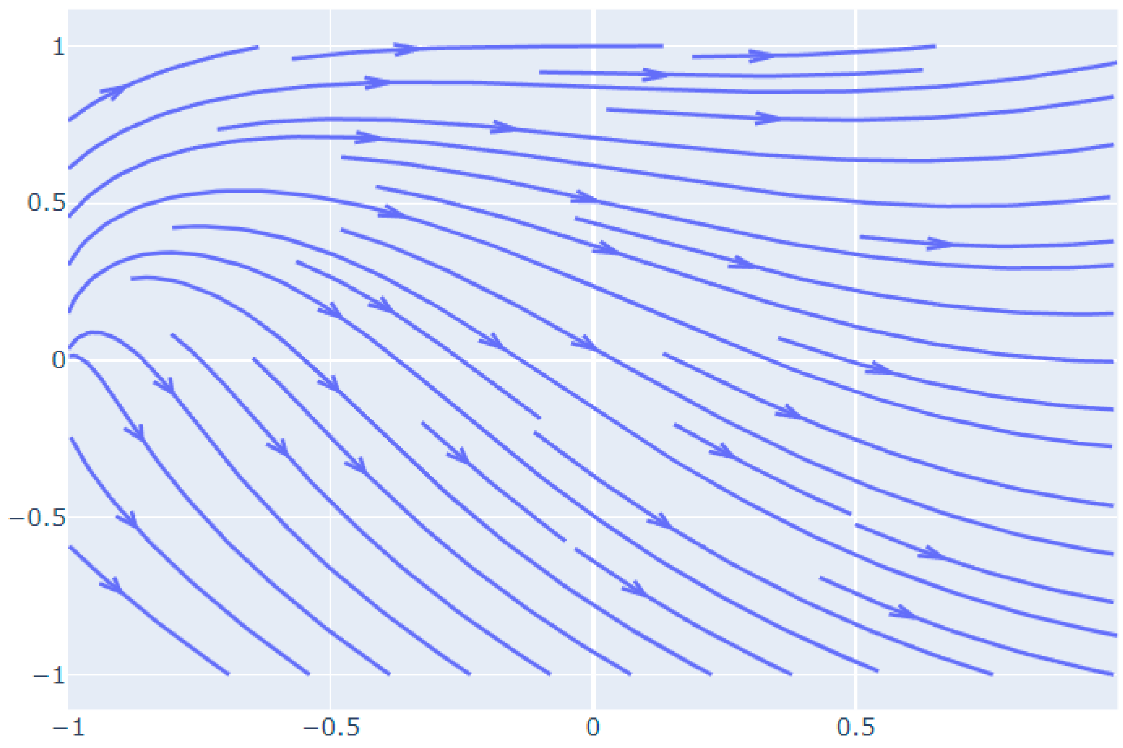
```
Ex = (X + 1)/((X+1)**2 + Y**2) - (X - 1)/((X-1)**2 + Y**2)
Ey = Y/((X+1)**2 + Y**2) - Y/((X-1)**2 + Y**2)

plt.figure(figsize=(10, 10))
plt.streamplot(X,Y,Ex,Ey, density=1.4, linewidth=None, color='#A23BEC')
plt.plot(-1,0,'-or')
plt.plot(1,0,'-og')
plt.title('Campo Eletromagnético')
plt.grid()
plt.show()
```



```
ax = plt.figure().add_subplot(projection = '3d')
x, y, z = np.meshgrid(np.arange(-3,10,.8), np.arange(-4,9,.8), np.arange(-5,8,.8))
u = x - z
v = y + z
w = z + 1
cor = np.arcsinh(v, u)
cor = (cor.flatten() - cor.min()) / cor.ptp()
cor = plt.cm.hsv(cor)
a = ax.quiver(x, y, z, u, v, w, length = 0.5, normalize = True, colors = cor,
lw = .8)
plt.show()
```



```
import plotly.figure_factory as ff

x = np.linspace(-1, 1, 10)
y = np.linspace(-1, 1, 10)
Y, X = np.meshgrid(x, y)
u = 1 - X**2 + Y
v = -1 + X - Y**2

fig = ff.create_streamline(x, y, u, v, arrow_scale = 0.05)
fig.show()
```


1
0.5
0
−0.5
−1
−1
−0.5
0
0.5