Chart, diagram, pie chart

Description automatically generated

**Python Code**

#Program to create Figure 3.22 Circle from Giordano

import pandas as pd

import numpy as np

from matplotlib import pyplot as plt

from math import cos,sin,exp,sqrt,pi,radians

import sys

xClist=[] #list of x positions for circle

yClist=[] #list of y positions for circle

xSlist=[] #list of x positions for stadium

ySlist=[] #list of y positions for stadium

r = 1 #radius in m

xC = 0.2 #initial x position in m for circle

yC = 0 #initial y position in m for circle

v0 = 1 #initial velocity in m/s

phi = pi/7.85 #initial angle of velocity in radians

vxC = v0\*cos(phi) #x component of initial velocity in m/s for circle

vyC = v0\*sin(phi) #y component of initial velocity in m/s for circle

dt = 0.0001 #time step

alpha = 0.01 #ratio of seperation distance of the circle halves

xS = 0 #initial x position in m for stadium

yS = 0 #initial y position in m for stadium

v0S = 1 #initial velocity in m/s

theta = pi/4.05

vxS = v0S\*cos(theta) #x component of initial velocity in m/s for stadium

vyS = v0S\*sin(theta) #y component of initial velocity in m/s for stadium

dt2 = 0.0001

xSlist.append(xS)

ySlist.append(yS)

xClist.append(xC)

yClist.append(yC)

#Euler-Cromer method to calculate positions and velocities at each interval

for i in range(1, 500000):

d = sqrt(xC\*\*2 + yC\*\*2) #Calculates distance from center

if d < r: #Calculates new position while inside circle

vxC = vxC

xC = xC + vxC\*dt

vyC = vyC

yC = yC + vyC\*dt

elif d >= r: #Calculates new direction of velocity after colliding with the wall

nxC = xC/(sqrt(xC\*\*2 + yC\*\*2))

nyC = yC/(sqrt(xC\*\*2 + yC\*\*2))

vperpxiC = (vxC\*nxC + vyC\*nyC)\*nxC

vperpyiC = (vxC\*nxC + vyC\*nyC)\*nyC

vparxiC = vxC - vperpxiC

vparyiC = vyC - vperpyiC

vperpxfC = -vperpxiC

vperpyfC = -vperpyiC

vparxfC = vparxiC

vparyfC = vparyiC

vxC = vperpxfC + vparxfC

vyC = vperpyfC + vparyfC

xC = xC + vxC\*dt

yC = yC + vyC\*dt

xClist.append(xC)

yClist.append(yC)

#Euler-Cromer method to calculate positions for stadium with alpha=0.01

for i in range(1, 700000):

if abs(yS) < alpha\*r:

if xS >= r: #Ball hits edge of circle

if xS > 0: #right side of circle

nyS = 0

nxS = -1

elif xS < 0: #left side of circle

nxS = 0

nyS = 1

vperpxiS = (vxS\*nxS + vyS\*nyS)\*nxS

vperpyiS = (vxS\*nxS + vyS\*nyS)\*nyS

vparxiS = vxS - vperpxiS

vparyiS = vyS - vperpyiS

vperpxfS = -vperpxiS

vperpyfS = -vperpyiS

vparxfS = vparxiS

vparyfS = vparyiS

vxS = vperpxfS + vparxfS

vyS = vperpyfS + vparyfS

xS += vxS\*dt2

yS += vyS\*dt2

else:

xS += vxS\*dt2

yS += vyS\*dt2

elif yS >= (alpha\*r): #ball hits top

if sqrt((yS-alpha\*r)\*\*2 + xS\*\*2) < r:

vxS = vxS

xS += vxS\*dt2

vyS = vyS

yS += vyS\*dt2

elif sqrt((yS-alpha\*r)\*\*2 + xS\*\*2) >= r:

nyS = (yS - alpha\*r)/sqrt((yS-alpha\*r)\*\*2 + xS\*\*2)

nxS = xS/sqrt((yS-alpha\*r)\*\*2 + xS\*\*2)

vperpxiS = (vxS\*nxS + vyS\*nyS)\*nxS

vperpyiS = (vxS\*nxS + vyS\*nyS)\*nyS

vparxiS = vxS - vperpxiS

vparyiS = vyS - vperpyiS

vperpxfS = -vperpxiS

vperpyfS = -vperpyiS

vparxfS = vparxiS

vparyfS = vparyiS

vxS = vperpxfS + vparxfS

vyS = vperpyfS + vparyfS

xS += vxS\*dt2

yS += vyS\*dt2

elif yS < -alpha\*r: #ball hits bottom

if sqrt((yS+alpha\*r)\*\*2 + xS\*\*2) >= r:

nyS = (yS + alpha\*r)/sqrt((yS+alpha\*r)\*\*2 + xS\*\*2)

nxS = xS/sqrt((yS+alpha\*r)\*\*2 + xS\*\*2)

vperpxiS = (vxS\*nxS + vyS\*nyS)\*nxS

vperpyiS = (vxS\*nxS + vyS\*nyS)\*nyS

vparxiS = vxS - vperpxiS

vparyiS = vyS - vperpyiS

vperpxfS = -vperpxiS

vperpyfS = -vperpyiS

vparxfS = vparxiS

vparyfS = vparyiS

vxS = vperpxfS + vparxfS

vyS = vperpyfS + vparyfS

xS += vxS\*dt2

yS += vyS\*dt2

else :

xS += vxS\*dt2

yS += vyS\*dt2

xSlist.append(xS)

ySlist.append(yS)

#Creates a figure

%matplotlib

fig, (ax1, ax2) = plt.subplots(1, 2)

ax1.margins(0)

angle = [2\*pi/100\*i for i in range(101)]

radius = 1

x = [radius\*cos(i) for i in angle]

y = [radius\*sin(i) for i in angle]

ax1.plot(x, y, 'k:', lw=1.2)

ax1.plot(xClist, yClist, 'k:', lw=1.2)

ax2.plot(x, y, 'k:', lw=1.2)

ax2.plot(xSlist, ySlist, 'k:', lw=1.2)

ax1.set\_title("Circular stadium - trajectory")

ax2.set(xlabel='x', ylabel='y')

ax2.set\_title("Stadium billiard \u03B1 = 0.01")

ax1.set(xlabel='x', ylabel='y')

ax1.set\_xticks([-1,-0.5,0,0.5,1])

ax1.set\_xticklabels([-1,-0.5,0,0.5,1], fontsize=12)

ax1.set\_yticks([-1,-0.5,0,0.5,1])

ax1.set\_yticklabels([-1,-0.5,0,0.5,1], fontsize=12)

ax2.set\_xticks([-1,-0.5,0,0.5,1])

ax2.set\_xticklabels([-1,-0.5,0,0.5,1], fontsize=12)

ax2.set\_yticks([-1,-0.5,0,0.5,1])

ax2.set\_yticklabels([-1,-0.5,0,0.5,1], fontsize=12)

ax1.tick\_params(direction='in', bottom=True, top=True, left=True, right=True)

ax1.tick\_params(labelbottom=True, labeltop=False, labelleft=True, labelright=False)

ax2.tick\_params(direction='in', bottom=True, top=True, left=True, right=True)

ax2.tick\_params(labelbottom=True, labeltop=False, labelleft=True, labelright=False)

#fig.tight\_layout()

#aspectratio=1.0

#ratio\_default=(ax.get\_xlim()[1]-ax.get\_xlim()[0])/(ax.get\_ylim()[1]-ax.get\_ylim()[0])

#ax.set\_aspect(ratio\_default\*aspectratio)

plt.show()