import numpy as np

import pandas as pd

from function import f1 # 用于计算龙头位置随时间的变化

from function import f3 # 用于计算注入流量后的位置迭代

from function import f4 # 用于计算模具上速度的增长

from zeropoint import zero2 # 用于计算函数f3的零点

from number import number# 用于保留6位小数

from crushjudge import judge # 用于判断是否碰撞

# 参数初始化

d = 0.56 # 龙头直径

v0 = 1.0 # 初始速度

theta\_init = 32 \* np.pi / 180 # 初始角度（32度转弧度）

# 第一部分：精确碰撞检测，在极角逐渐递减到0的过程中，检测会不会发生碰撞

theta\_collision = None

# 粗检测（步长0.01）

for theta in np.arange(theta\_init, 0, -0.01): #逆向搜索逻辑

if judge(theta, d, v0): #碰撞检测函数

theta\_collision = theta

break

# 中等精度检测（步长0.0001）

if theta\_collision is not None:

for theta in np.arange(theta\_collision + 0.01, theta\_collision - 0.01, -0.0001):

if judge(theta, d, v0):

theta\_collision = theta

break

# 高精度检测（步长0.000001）

if theta\_collision is not None:

for theta in np.arange(theta\_collision + 0.0001, theta\_collision - 0.0001, -0.000001):

if judge(theta, d, v0):

theta\_collision = theta

break

# 计算碰撞时间（修正公式）

if theta\_collision is not None: #如果第一部分检测到会发生碰撞

t\_collision = (f1(theta\_init) - f1(theta\_collision)) / (4 \* np.pi \* v0)

let\_chain\_theta = [theta\_collision] #从碰撞点开始记录后续的运动

else:

let\_chain\_theta = [theta\_init]

# 第二部分：计算把手轨迹

for i in np.arange(223):

if i == 0:

d0 = 3.41 - 0.275 - 2 # 初始设计参数

else:

d0 = 2.2 - 0.275 - 2 # 后续设计参数

theta\_last = let\_chain\_theta[-1]

theta = zero2(f3, theta\_last, theta\_last + np.pi / 2, 1e-8, d, d0, theta\_last)

let\_chain\_theta.append(theta)

# 第三部分：计算坐标和速度场

# 将极坐标转换为直角坐标系

let\_chair\_xyv = []

for i in np.arange(224):

theta = let\_chain\_theta[i]

x = d \* theta \* np.cos(theta) / (2 \* np.pi) # 修正坐标计算公式

y = d \* theta \* np.sin(theta) / (2 \* np.pi)

let\_chair\_xyv.append([x, y, v0]) # 初始速度为v0

let\_chair\_xyv = np.array(let\_chair\_xyv) #将结果转换成Numpy数组

# 第四部分：速度场迭代计算

for i in np.arange(223):

v\_last = let\_chair\_xyv[i, 2] # 上一个速度,遍历步长为2

theta\_last = let\_chain\_theta[i]

theta = let\_chain\_theta[i + 1]

# 获取坐标

x\_last, y\_last = let\_chair\_xyv[i, 0], let\_chair\_xyv[i, 1]

x, y = let\_chair\_xyv[i + 1, 0], let\_chair\_xyv[i + 1, 1]

# 计算斜率

k\_chair = (y\_last - y) / (x\_last - x) if (x\_last - x) != 0 else np.inf

k\_v\_last = f4(theta\_last)

k\_v = f4(theta)

# 计算角度

alpha1 = np.arctan(np.abs((k\_v\_last - k\_chair) / (1 + k\_v\_last \* k\_chair)))

alpha2 = np.arctan(np.abs((k\_v - k\_chair) / (1 + k\_v \* k\_chair)))

# 更新速度

v = v\_last \* np.cos(alpha1) / np.cos(alpha2)

let\_chair\_xyv[i + 1, 2] = v

# 数据后处理

let\_chair\_xyv = number(let\_chair\_xyv, 6) # 保留6位小数

df = pd.DataFrame(let\_chair\_xyv, columns=['x', 'y', 'velocity'])

df.to\_excel('result2.xlsx', index=False)

def f1(theta):

result = theta + np.sqrt(theta\*\*2 + 1) + np.log(theta + np.sqrt(theta\*\*2 + 1))

return result

def f3(theta, d, d0, theta\_last):

t = theta #当前板凳的极角

t\_1 = theta\_last #前一节板凳的极角

result = t\*\*2 \* t\_1\*\*2 - 2 \* t \* t\_1 - np.cos(t - t\_1) - 4 \* np.pi\*\*2 \* d0\*\*2 / d\*\*2

return result

#用于计算螺线上速度的斜率

def f4(theta):

t = theta

result = (np.sin(t) + t \* np.cos(t)) / (np.cos(t) - t \* np.sin(t))

return result

def judge(theta, d, v0):

d1 = 0.275 # 内侧距离阈值

d2 = 0.15 # 外侧距离阈值

let\_theta = [theta]

# 第一部分：计算角度轨迹

for i in np.arange(223):

if i == 0:

d0 = 3.41 - 0.275 + 2

else:

d0 = 2.2 - 0.275 + 2

theta\_last = let\_theta[-1] # 获取上一个角度值

a = theta\_last

b = theta\_last + np.pi / 2

theta\_new = zero2(f3, a, b, 1e-9, d, d0, theta\_last)

let\_theta.append(theta\_new)

if theta\_new - theta > 3 \* np.pi:

break # 角度变化过大时终止

# 第二部分：计算坐标轨迹

lst\_x, lst\_y = [], []

for i in np.arange(len(let\_theta)):

p = let\_theta[i] \* d / (2 \* np.pi)

x = p \* np.cos(let\_theta[i])

y = p \* np.sin(let\_theta[i])

lst\_x.append(x)

lst\_y.append(y)

# 第三部分：计算斜率

lst\_k = []

for i in np.arange(len(let\_theta) - 1):

k = (lst\_y[i] - lst\_y[i + 1]) / (lst\_x[i] - lst\_x[i + 1])

lst\_k.append(k)

# 第四部分：碰撞检测

flag = 0

k1 = lst\_k[0]

x1 = lst\_x[0]

y1 = lst\_y[0]

# 计算参考线参数

k2 = (d2 / d1 + k1) / (1 + d2 \* k1 / d1) # 修正斜率计算公式

b = d2 \* np.sqrt(k1 \*\* 2 + 1) + y1 - k1 \* x1 # 截距计算

# 碰撞检测主循环

for i in np.arange(len(lst\_k)):

if let\_theta[i + 1] - theta > np.pi:

ki = lst\_k[i]

xi = lst\_x[i]

yi = lst\_y[i]

# 计算距离（修正距离公式）

numerator = np.abs(ki \* (x1 - xi) + (y1 - yi))

denominator = np.sqrt(ki \*\* 2 + 1)

d\_chair = numerator / denominator

if d\_chair < d2:

flag = 1

break

return flag

def zero2(f, a, b, e, d, d0, theta\_last):

while b - a > e:

c = (a + b) / 2 # 取中点

if f(a, d, d0, theta\_last) \* f(c, d, d0, theta\_last) < 0:

b = c

else:

a = c

return (a + b) / 2

# 用于计算函数f的零点（带theta\_last参数的版本）

def number(A,n):

#遍历A数组的行和列

for i in np.arange(A.shape[0]):

for j in np.arange(A.shape[1]):

a=A[i,j] #a用来存储这个二维数组的值

b=int(a\*10\*\*n)\*10\*\*(-n)

if a-b>=5\*10\*\*(-n-1):

b=b+10\*\*(-n)

#四舍五入

A[i,j]=b

return A

帮我可视化运行这个代码