曲柄滑块机构Python设计代码：

import math

import numpy as np

import matplotlib.pyplot as plt

from numpy import sin, cos, pi

from scipy.optimize import fsolve

from scipy.misc import derivative

##求解曲柄长和连杆长

H = float(input("滑块行程H(mm): "))

e = float(input("偏心距e(mm): "))

A\_deg = float(input("极位夹角theta(°): "))

A\_rad = math.radians(A\_deg) #极位夹角(rad)

def equations(vars, H, e, A\_rad):

    b, c = vars

    eq1 = H \* e - b \* c \* sin(A\_rad)

    eq2 = H\*\*2 - (b\*\*2 + c\*\*2 - 2 \* b \* c \* cos(A\_rad))

    return [eq1, eq2]

initial\_guess = [1, 1]

b, c = fsolve(equations, initial\_guess, args=(H, e, A\_rad))

l=(b+c)/2 #连杆长

r=abs(b-c)/2 #曲柄长

alpha=math.degrees(pi/2-math.acos((e+r)/l)) #最大压力角

print(f"连杆长度为: {l:.2f}mm")

print(f"曲柄长度为: {r:.2f}mm")

print(f"最大压力角为: {alpha:.2f}°")

##绘制图表

rpm = float(input("曲柄转速(r/min):"))

omega = (2\*pi\*rpm)/60 #曲柄转速(rad/s)

#输入时间范围

t0 = float(input("机构运动秒数:"))

time = np.linspace(0,t0,1000)

def disp(t): #关于时间的滑块位移函数

    d1 = r\*cos(omega\*t)

    d2 = np.sqrt((l\*\*2-(e-r\*sin(omega\*t))\*\*2))

    d3=np.sqrt(((l+r)\*\*2-e\*\*2))

    return d3-d1-d2

def vel(t): #关于时间的滑块速度函数

    return -derivative(disp,t)

def acc(t): #关于时间的滑块加速度函数

    return -derivative(vel,t)

x = np.array([disp(ti) for ti in time])

v = np.array([vel(ti) for ti in time])

a = np.array([acc(ti) for ti in time])

theta = omega\*time\*180/pi #曲柄角度(°)

#绘制曲线图

plt.figure(figsize=(5,15)) #新建画布

plt.subplot(3,1,1)

plt.plot(theta,x,color="green")

plt.title("Graphs of a slider-crank mechanism")

plt.ylabel("displacement(mm)")

plt.grid(color="blue")

plt.subplot(3,1,2)

plt.plot(theta,v,color="orange")

plt.ylabel("velocity(mm/s)")

plt.grid(color="blue")

plt.subplot(3,1,3)

plt.plot(theta,a,color="red")

plt.xlabel("angle(°)")

plt.ylabel("acceleration(mm/s^2)")

plt.grid(color="blue")

plt.show() #显示图表

凸轮机构Python设计代码：

import math

import numpy as np

import matplotlib.pyplot as plt

from numpy import sin, cos, pi

from scipy.optimize import minimize\_scalar

#定义区间

x=np.linspace(0,2\*pi,1000)

theta = np.linspace(0, 2 \* np.pi, 1000)

#定义函数

def f1(x):

    return -18.09\*cos(6/5\*x)+18.09

def f2(x):

    return -8.85\*cos(3\*x)+27.33

def f3(x):

    return 9.24\*sin(3\*x)+9.24

def s(x):

    return np.piecewise(x,

                        [ (0 <= x) & (x < 5/6\*pi),

                          (5/6\*pi <= x) & (x < pi),

                          (pi <= x) & (x < 4/3\*pi),

                          (4/3\*pi <= x) & (x < 3/2\*pi),

                          (3/2\*pi <= x) & (x < 11/6\*pi),

                          (11/6\*pi <= x) & (x <= 2\*pi)],

                        [f1, 36.18, f2, 18.48, f3, 0])

def diff(func, x):

    dx = x[1] - x[0]

    return np.gradient(func(x), dx)

v = diff(s, x)

a = diff(lambda t: diff(s, t), x)

##求基圆半径

def f(x):

    return abs(v)\*1.73205-s(x)

min= minimize\_scalar(lambda t: f(np.array([t]))[0], bounds=(0, 2\*pi), method='bounded')

print(f"最大值出现在 x = {min.x}")

print(f"最大值为 f(x) = {-min.fun}")

r0=-min.fun

y1=s(x)

y2=v

y3=a

r1=np.full\_like(theta,r0)

r2=r0+s(x)

##绘制曲线图

#plt.plot(x\*180/pi,f(x))

#plt.show()

plt.figure(figsize=(5,15))

plt.subplot(3,1,1)

plt.plot(x\*180/pi,y1)

plt.ylabel('s(mm)')

plt.grid(True)

plt.subplot(3,1,2)

plt.plot(x\*180/pi,y2)

plt.ylabel('v(mm/°)')

plt.grid(True)

plt.subplot(3,1,3)

plt.plot(x\*180/pi,y3)

plt.ylabel('a(mm/°^2)')

plt.xlabel('delta(°)')

plt.grid(True)

plt.show()

##绘制凸轮

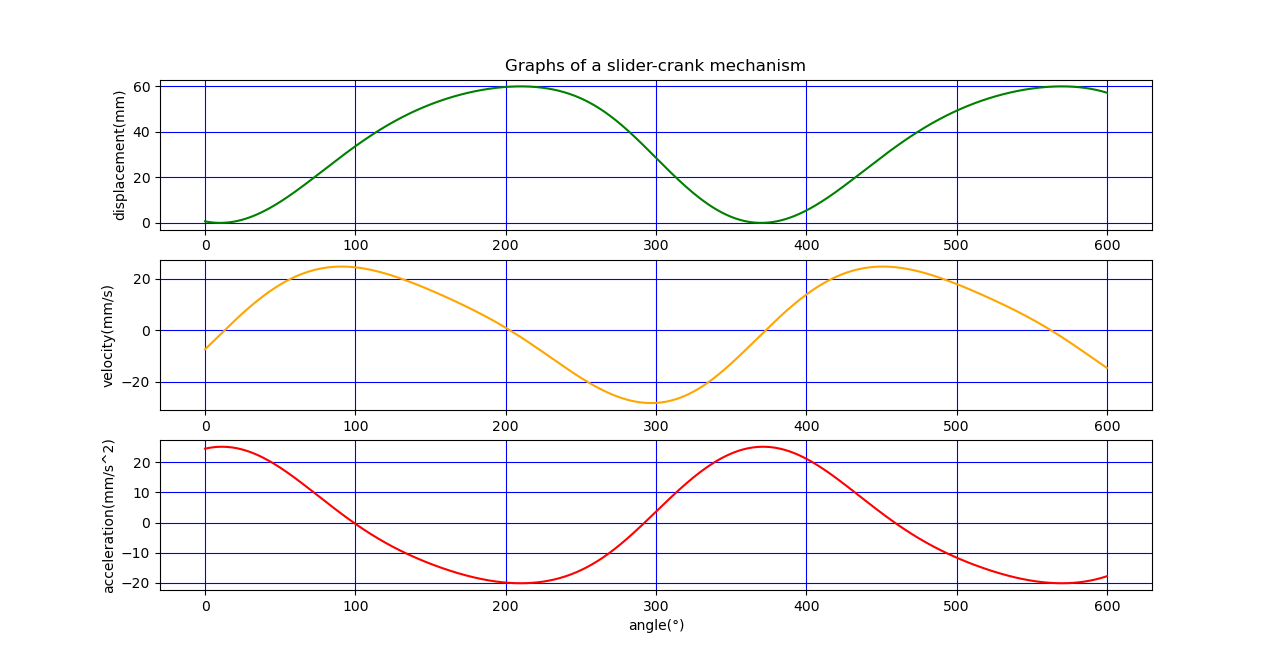
plt.polar(theta,r1,color='green')

plt.polar(theta,r2,color='red')

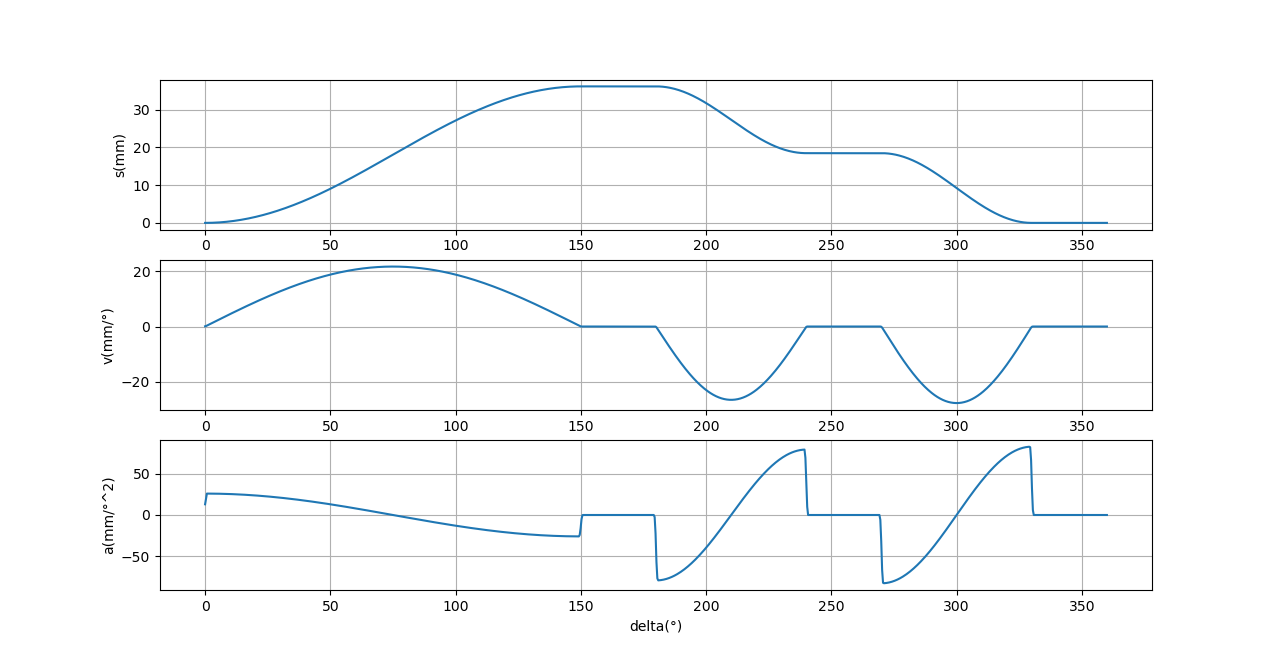
plt.grid(False)

plt.show()

曲柄滑块机构位移、速度、加速度曲线图：



凸轮机构位移、速度、加速度曲线图：



凸轮理论轮廓线：

