**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

по дисциплине «Математические модели сложных систем»

на тему: «Построение статических моделей»

Выполнил: студент гр. ИП-31

Коваленко А.И.

Принял: преподаватель

Трохова Т.А.

Гомель 2023

**Цель:** Получить навыки компьютерного моделирования технических объектов, представленных в виде статических модели с выводом результатов моделирования в численном и графическом виде.

***Задача 1. Компьютерная модель кривошипно-ползунного механизма***

## *Постановка задачи моделирования*

***1) Разработать компьютерную модель манипулятора в виде кривошипно-ползунного механизма, которая имеет следующие выходные параметры:***

- длины звеньев кривошипно-ползунного механизма по заданным исходным данным;

- проверить условие существования механизма.

- рассчитать функции координат характерных точек механизма в зависимости от угла поворота кривошипа. Построить графики этих функций.

- функция хода ползуна в зависимости от угла поворота кривошипа.

Результаты моделирования представить в численном и графическом виде.

***2) Исследовать модель, для чего:***

- вычислить значение угла поворота кривошипа, при котором функция хода ползуна пересекает пороговое значение (пороговое значение подобрать самостоятельно).

***3) Дополнительно.***

Построить графически вид механизма в нескольких положениях или разработать анимацию движения механизма.

**Исходные данные**

Исходными данными для работы являются:

φ1, φ2, φ3 – начальные значения угла поворота кривошипа

*S1, S2, S3*  – начальные значения перемещения ползуна

*a4* – длина звена механизма

*β* – угол между звеньями механизма

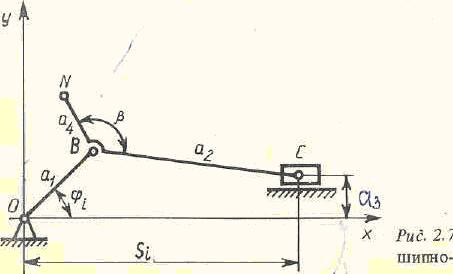


Рисунок 2.1 – Кинематическая схема кривошипно-ползунного механизма

Таблица 2.1 - Таблица исходных данных

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N варианта | (град) | (град) | (град) | *S1*  *(м)* | *S2*  *(м)* | *S3*  *(м)* | *a4*  *(м)* | *Β*  (град) |
| 1 | 45 | 22.5 | 67.5 | 1.2 | 1.4 | 0.95 | 0,1 | 100 |

**Листинг:**

import math

from scipy.optimize import fsolve

import numpy as np

import matplotlib.pyplot as mp

f11=math.radians(43)

f2=math.radians(82)

f3=math.radians(103)

S1=1

S2=0.8

S3=0.67

a4=0.09

B=math.radians(120)

def F(X):

K1,K2,K3=X

K = np.array([0, 0, 0], float)

K[0] = K1\*S1\*np.cos(f11)+K2\*np.sin(f11)-K3-S1\*\*2

K[1] = K1\*S2\*np.cos(f2)+K2\*np.sin(f2)-K3-S2\*\*2

K[2]=K1\*S3\*np.cos(f3)+K2\*np.sin(f3)-K3-S3\*\*2

return K

K=fsolve(F,[1,1,1])

a1=K[0]/2

a3=K[1]/(2\*a1)

a2=math.sqrt(a1\*\*2+a3\*\*2-K[2])

print("a1: " + str(a1) + "\na2: " + str(a2) + "\na3: " + str(a3))

if a1< a2-a3:

print("Решение существует")

else:

print("Решение не существует")

H=0

if a3>0:

H=1

elif a3<0:

H=-1

f1=np.linspace(0,math.pi,100)

Xc=np.zeros(100)

Xc=a1\*np.cos(f1)+np.sqrt(a2\*\*2-(a3\*H-a1\*np.sin(f1))\*(a3\*H-a1\*np.sin(f1)))

Yc=a3\*H

Xv=a1\*np.cos(f1)

Yv=a1\*np.sin(f1)

f6=f2+B

Xn=a1\*np.cos(f1)+a4\*np.cos(f6)

Yn=a1\*np.sin(f1)+a4\*np.sin(f6)

f22=np.arccos((Xc-a1\*np.cos(f1))/a2)

mp.plot(Xv, Yv,Xn,Yn)

mp.scatter(Xn[30],Yn[30])

mp.scatter(Xv[20],Yv[20])

mp.scatter(Xc[10],a3)

mp.scatter(0,0)

XD=[0,Xv[20],Xn[30]]

YD=[0,Yv[20],Yn[30]]

XCD=[Xc[10],Xv[20]]

YCD=[a3,Yv[20]]

print(math.sqrt((Xv[0]-Xc[0])\*\*2+(Yv[0]-a3)\*\*2))

mp.plot(XD,YD)

mp.plot(XCD,YCD)

mp.grid()

mp.show()

**Задание 1:**

**Результат выполнения задания 1**

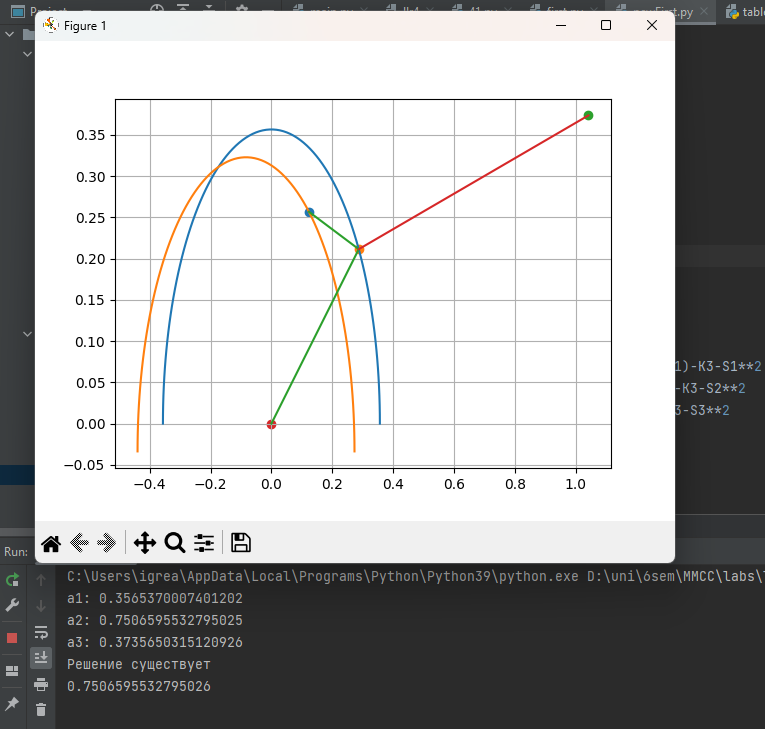
****

Рисунок 1 – Результат выполнения программы

***Задача 2.Компьютерная модель шарнирного четырехзвенника***

## Постановка задачи моделирования

1. Рассчитать длины звеньев шарнирного кривошипно-коромыслового четырехзвенника.

2. Проверить, соответствуют ли вычисленные значения параметров ***a*** (длина кривошипа), ***b*** (длина шатуна) и ***c*** (длина коромысла) с заданными значениями α и β условиям существования механизма и ограничениям:

υ ≤ υд и a < d

3. Рассчитать значение функции погрешности ΔΨ(ϕ), построить графики зависимости Ψ(φ) и Ψ(φ)+ΔΨ(ϕ), сделать выводы по полученным результатам.

4. Вычислить максимальное значение функции погрешности и значение угла ϕ, при котором погрешность максимальна. Дать графическую интерпретацию результатов.

Исходными данными для задачи являются:

* звено AD (стойка) имеет длину d=1;
* углы α и β, определяющие взаимное расположение звеньев AB и CD относительно стойки, заданы в таблице 1;
* вид функции закона движения коромысла, заданный графически;
* допустимое значение угла давления шатуна на коромысло υд;
* пределы изменения угла ϕ ( 0 ≤ ϕ ≤ 2π).

##### Таблица 3.1 - Варианты исходных данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер варианта | α, рад | k1, k2 | υд, рад | β |
| 1 | 3 | 0.313, 0.13 | 1,3 | 140° |

**Листинг программы:**

import sys  
from functools import reduce  
import numpy as np  
import matplotlib.pyplot as plt  
  
def psi(phi):  
 return k1 \* np.sin(phi) + k2  
  
def f0(phi):  
 return np.cos(phi + alpha)  
  
def f1(phi):  
 return np.cos(psi(phi) + betta)  
  
def f2():  
 return 1  
  
def F(phi):  
 return np.cos(phi + alpha - psi(phi) - betta)  
  
def max\_error():  
 return reduce(lambda x, y: x if np.abs(x) > np.abs(y) else y, delta\_psi), \  
 list(delta\_psi).index(reduce(lambda x, y: x if np.abs(x) > np.abs(y) else y, delta\_psi))  
  
def print\_result():  
 print(f'\nДлина кривошипа: {a:.3f} \nДлина шатуна: {b:.3f} \nДлина коромысла: {c:.3f} \nДлина стойки: {d:.3f}')  
 print(f'Максимальный угол давления: {max(v):.3f} при delta\_phi = {phi\_change[np.argmax(v)]:.3f}')  
 print(f'Максимальное значение погрешности: {np.abs(max\_error):.3f} при delta\_phi = {phi\_change[index\_max\_error]:.3f}')  
  
def print\_graphics():  
 plt.plot(phi\_change, psi\_f)  
 plt.plot(phi\_change, [psi\_f[i] + delta\_psi[i] for i in range(len(phi\_change))], color='pink')  
 plt.plot([phi\_change[index\_max\_error], phi\_change[index\_max\_error]],  
 [psi\_f[index\_max\_error], psi\_f[index\_max\_error] + max\_error], color='purple', linewidth='3')  
 plt.grid()  
 plt.show()  
  
  
alpha = 3  
betta = np.radians(140)  
k1 = 0.313  
k2 = 0.13  
v\_d = 1.3  
  
phi\_m = 2 \* np.pi  
phi3 = phi\_m  
phi2 = 0.75 \* phi\_m  
phi1 = 0.25 \* phi\_m  
phi\_change = np.arange(0, phi\_m, 0.01)  
d = 1  
  
matrix\_coefficients = [[np.cos(phi1 + alpha), np.cos(psi(phi1) + betta), 1],  
 [np.cos(phi2 + alpha), np.cos(psi(phi2) + betta), 1],  
 [np.cos(phi3 + alpha), np.cos(psi(phi3) + betta), 1]]  
column\_tree\_members = [np.cos(phi1 + alpha - psi(phi1) - betta),  
 np.cos(phi2 + alpha - psi(phi2) - betta),  
 np.cos(phi3 + alpha - psi(phi3) - betta)]  
p = np.linalg.solve(matrix\_coefficients, column\_tree\_members)  
  
  
a = 1 / p[1]  
c = -1 / p[0]  
b = np.sqrt(a \* a + c \* c + 1 - 2 \* c \* a \* p[2])  
#l\_sqr = a\*\*2 + 1 - 2 \* a \* np.cos(phi\_elem);  
v = [np.arcsin((b\*\*2 + c\*\*2 - (a\*\*2 + 1 - 2 \* a \* np.cos(phi\_elem))) / (2 \* b \* c)) for phi\_elem in phi\_change]  
  
psi\_f = [psi(phi\_elem) for phi\_elem in phi\_change]  
  
delta\_q = [-2 \* a \* c \* (p[0] \* f0(phi\_elem) + p[1] \* f1(phi\_elem) +  
 p[2] \* f2() - F(phi\_elem)) for phi\_elem in phi\_change]  
delta\_psi = [delta\_q[i] / (2 \* b \* c \* np.cos(v[i])) for i in range(len(phi\_change))]  
  
max\_error, index\_max\_error = max\_error()  
  
print\_result()  
print\_graphics()

**Результат выполнения задания 2**

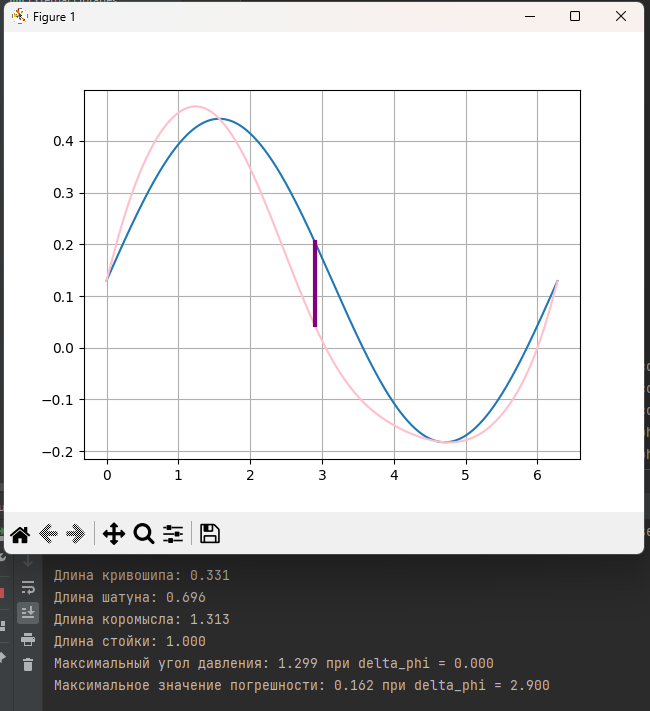
****

Рисунок 2 – Результат выполнения программы

Вывод: получили навыки компьютерного моделирования технических объектов, представленных в виде статических моделей с выводом результатов моделирования в численном и графическом виде.