**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

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

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

по дисциплине **«Методы моделирования сложных систем»**

на тему: «**Построение статических моделей**»

Выполнил: студент гр. ИП-32

Суховенко Э.С.

Принял: преподаватель

Трохова Т.А.

Гомель 2022

**Цель работы**: 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;

- функция хода ползуна в зависимости от угла поворота кривошипа.

Проверить условие существования механизма.

Результаты моделирования представить в численном и графическом виде.

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

- для функции хода ползуна вычислить минимальное значение и угол поворота кривошипа, при котором это значение минимально;

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

***Исходными данными для построения модели 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

##### Таблица 1.1 - Варианты исходных данных

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | (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 |

***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).

**Листинг решения задачи 1 в python:**

import numpy as np  
import matplotlib.pyplot as plt  
import math  
  
FI1 = 45 \* math.pi / 180  
FI2 = 22.5 \* math.pi / 180  
FI3 = 67.5 \* math.pi / 180  
S1 = 1.2  
S2 = 1.4  
S3 = .95  
A4 = .1  
BETA = 100  
  
  
def is\_exist(a1, a2, a3):  
 if a1 < a2 - a3:  
 return True  
 return False  
  
  
def find\_parameters(f, s):  
 matrix = []  
 vector = []  
  
 for i in range(len(f)):  
 matrix.append([s[i] \* math.cos(f[i]), math.sin(f[i]), - 1])  
 vector.append([s[i] \*\* 2])  
  
 K = np.linalg.solve(matrix, vector)  
  
 return K  
  
  
def s\_f(i, A1, A2, A3, H):  
 return A1 \* math.cos(i) + math.sqrt(A2 \*\* 2 - (A3 \* H - A1 \* math.sin(i)) \*\* 2)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 K = find\_parameters([FI1, FI2, FI3], [S1, S2, S3])  
 print(K)  
  
 A1 = K[0] / 2  
 A3 = K[1] / (2 \* A1)  
 A2 = math.sqrt(A1 \*\* 2 + A3 \*\* 2 - K[2])  
 print(A1, A2, A3)  
  
 if A3 > 0:  
 H = 1  
 else:  
 H = -1  
 print(H)  
 print(is\_exist(A1, A2, A3))  
  
 FI = np.arange(0, 2 \* np.pi, 0.01)  
  
 S = []  
 for i in FI:  
 value = s\_f(i, A1, A2, A3, H)  
 S.append(value[0])  
 print(S)  
  
 yy\_i = 0  
 res = []  
 while yy\_i < len(FI):  
 eps = 1e-2  
 if np.abs(S[yy\_i] - 1.2) < eps:  
 res.append(FI[yy\_i])  
 yy\_i += 100  
 yy\_i += 1  
  
 plt.plot(FI, S)  
 plt.scatter(res[0], 1.2, color='orange', s=40, marker='\*')  
 plt.scatter(res[1], 1.2, color='orange', s=40, marker='\*')  
 plt.grid()  
 plt.show()

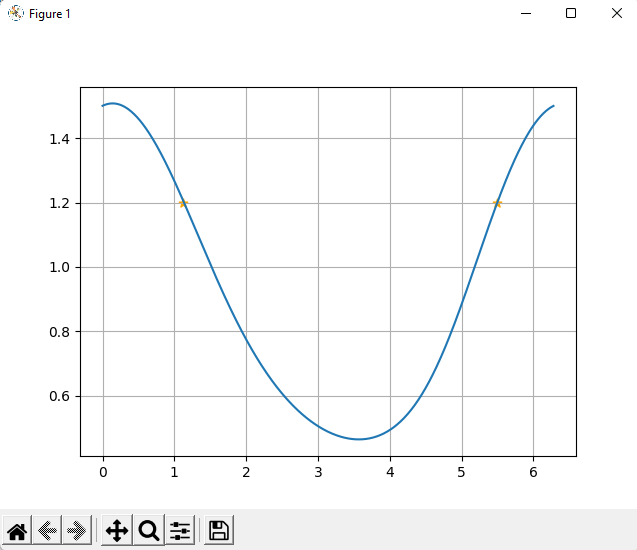


Рисунок 1 – Результат задачи 1 в графическом виде

**Листинг решения задачи 2 в python:**

import numpy as np  
import matplotlib.pyplot as plt  
import math  
  
alpha = 3.2  
betta = 109 \* math.pi / 180  
k1 = 0.314  
k2 = 0.122  
v = 1.3  
  
  
def psi(phi):  
 return k1 \* np.sin(phi) + k2  
  
  
phi\_m = 2 \* math.pi  
phi1 = 0.25 \* phi\_m  
phi2 = 0.75 \* phi\_m  
phi3 = phi\_m  
  
coeffs = [  
 [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]  
]  
  
right = [  
 np.cos(phi1 + alpha - psi(phi1) - betta),  
 np.cos(phi2 + alpha - psi(phi2) - betta),  
 np.cos(phi3 + alpha - psi(phi3) - betta)  
]  
  
ps = np.linalg.solve(coeffs, right)  
a = 1 / ps[1]  
c = -1 / ps[0]  
b = np.sqrt(a \*\* 2 + c \*\* 2 + 1 - 2 \* c \* a \* ps[2])  
  
  
def vu(phi):  
 one\_square = a \*\* 2 + 1 - 2 \* a \* np.cos(phi)  
 return np.arcsin((b \*\* 2 + c \*\* 2 - one\_square) / (2 \* b \* c))  
  
  
def f0(phi):  
 return np.cos(phi + alpha)  
  
  
def f1(phi):  
 return np.cos(psi(phi) + betta)  
  
  
def f2(phi):  
 return 1  
  
  
def big\_f(phi):  
 return np.cos(phi + alpha - psi(phi) - betta)  
  
  
def delta\_q(phi):  
 return -2 \* a \* c \* (ps[0] \* f0(phi) + ps[1] \* f1(phi) + ps[2] \* f2(phi) - big\_f(phi))  
  
  
def delta\_psi(phi):  
 return delta\_q(phi) / (2 \* b \* c \* np.cos(vu(phi)))  
  
*# exist because vu(0) < 1.38*print(a, b, c)  
phis = np.arange(0, 2 \* np.pi, 0.01)  
psis = []  
delta\_psis = []  
  
mx\_phi = 0  
mx\_i = 0  
mx\_delt = -1e9  
  
for i in range(len(phis)):  
 phi = phis[i]  
 psis.append(psi(phi))  
 delta\_psis.append(psi(phi) + delta\_psi(phi))  
 delta = np.abs(psis[len(psis) - 1] - delta\_psis[len(delta\_psis) - 1])  
 if mx\_delt < delta:  
 mx\_delt = delta  
 mx\_phi = phi  
 mx\_i = i  
  
print(mx\_phi, mx\_delt)  
  
plt.plot(phis, psis)  
plt.plot(phis, delta\_psis)  
  
plt.scatter(mx\_phi, psis[mx\_i])  
plt.scatter(mx\_phi, delta\_psis[mx\_i])  
  
plt.show()

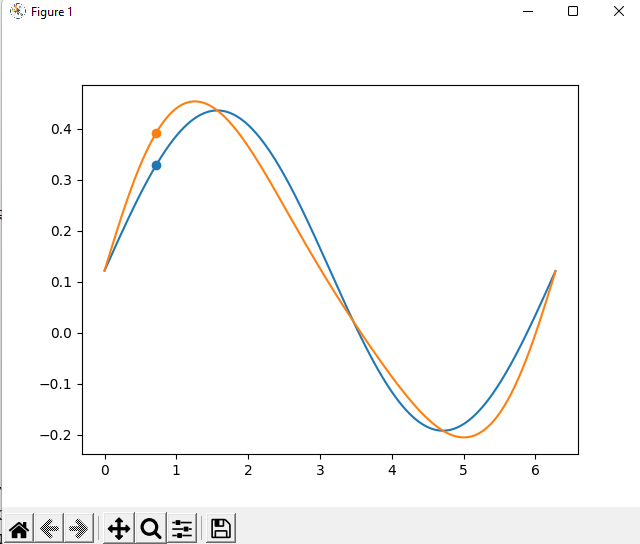
****

Рисунок 2 ­– Результат задачи 2 в графическом виде

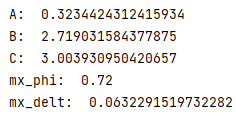
****

Рисунок 3 – Результат задачи 2 в текстовом виде

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