

**Министерство науки и высшего образования**

**Российской Федерации**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГБОУ ВО «МГТУ «СТАНКИН»)**

Институт цифровых интеллектуальных систем

Кафедра робототехники и мехатроники

Учебный курс «Моделирование и исследование робототехнических систем»

**ОТЧЁТ**

**по лабораторной работе**

**на тему:**

**«Решение прямой задачи кинематики для манипулятора SCARA, PUMA»**

Выполнил:

студент группы АДМ-21-05 \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_ Пивкин Д.П.

(дата) (подпись) (ФИО)

Принял

преподаватель: \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_ Порунов М.Ю.

(дата) (подпись) (ФИО)

Оценка: \_\_\_\_\_\_\_\_ Дата:\_\_\_\_\_\_\_\_\_

Москва 2022

**Решение прямой задачи кинематики для манипулятора SCARA, PUMA**

**Задание 1.** Разобраться в примере решения прямой задачи кинематики SCARA (рис. 1). Решить прямую задачу кинематики SCARA для своего закона изменения обобщенных координат.

• Оценить диапазон X, Y и Z координат для данной траектории;

• Оценить максимальную скорость по осям;

• Оценить абсолютную максимальную скорость.

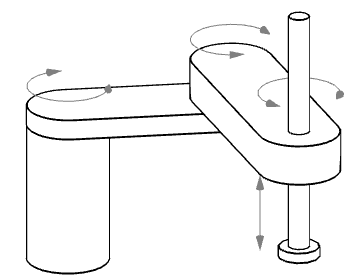


Рисунок 1. Робот SCARA.

Выполнение задания 1 представлено в листинге 1. Результат работы программы представлен на рис. 2-3.

Листинг 1 – Реализация задания 1

**from** matplotlib **import** pyplot **as** plt

**from** matplotlib **import** animation

**import** numpy **as** np

**from** IPython.display **import** HTML

**%matplotlib** notebookIn [2]:

**from** kinematics **import** Vector, Quaternion, Transform

**import** graphics

**def** scara\_chain(q, l, math\_source**=**np):

base **=** Transform**.**identity()

column **=** base **+** Transform(

Vector(0, 0, l[0]),

Quaternion**.**from\_angle\_axis(q[0], Vector(0, 0, 1), math\_source)

)

elbow **=** column **+** Transform(

Vector(l[1], 0, 0),

Quaternion**.**from\_angle\_axis(q[1], Vector(0, 0, 1), math\_source)

)

tool **=** elbow **+** Transform(

Vector(l[2], 0, 0),

Quaternion**.**from\_angle\_axis(q[2], Vector(0, 0, 1), math\_source)

)

flange **=** tool **+** Transform(

Vector(0, 0, **-**q[3]),

Quaternion**.**identity()

)

**return** [

base,

column,

elbow,

tool,

flange

]

//Закон изменения обобщенных координат

**def** scara\_q(t, total):

omega **=** t **/** total **\*** np**.**pi **\*** 2

**return** [

np**.**pi **/** 4 **\*** np**.**cos(omega),

np**.**pi **/** 2,

omega,

3 **+** 3 **\*** np**.**sin(omega)

]

//Длины звеньев

scara\_l **=** [8, 4, 3]

scara\_fig **=** plt**.**figure()

ax **=** scara\_fig**.**add\_subplot(projection**=**"3d")

ax**.**set\_xlim(**-**6, 6); ax**.**set\_ylim(**-**6, 6); ax**.**set\_zlim(0, 12)

lines, **=** ax**.**plot([], [], [], color**=**"#000000")

graphics**.**axis(ax, Transform**.**identity(), 2)

r, g, b **=** graphics**.**axis(ax, Transform**.**identity(), 1)

total **=** 100

**def** animate(frame):

chain **=** scara\_chain(scara\_q(frame, total), scara\_l)

(x, y, z) **=** graphics**.**chain\_to\_points(chain)

lines**.**set\_data\_3d(x, y, z)

**global** r, g, b

r**.**remove(); g**.**remove(); b**.**remove()

r, g, b **=** graphics**.**axis(ax, chain[**-**1], 0.5)

animate(0)

fps **=** 25

scara\_ani **=** animation**.**FuncAnimation(

scara\_fig,

animate,

frames**=**total,

interval**=**1000.0**/**fps

)

HTML(scara\_ani**.**to\_jshtml())

// отображение траектории

t\_end **=** 10

step **=** 0.01

t **=** np**.**arange(0, t\_end, step)

chain **=** scara\_chain(scara\_q(t, t\_end), scara\_l)

x, y, z **=** graphics**.**chain\_to\_points(chain)

fig **=** plt**.**figure()

ax **=** fig**.**add\_subplot(projection**=**"3d")

ax**.**set\_xlim(**-**6, 6); ax**.**set\_ylim(**-**6, 6); ax**.**set\_zlim(0, 12)

ax**.**plot(x[2], y[2], z[2], color**=**"#c0c0c0")

ax**.**plot(x[3], y[3], z[3], color**=**"#c0c0c0")

ax**.**plot(x[4], y[4], z[4], color**=**"#000000")

graphics**.**axis(ax, Transform**.**identity(), 2)

fig**.**show()

// Диапазон изменения Z координаты фланца:

print(

"Z координата менялась в диапазоне от",

np**.**min(z[**-**1]),

"до",

np**.**max(z[**-**1])

)

//Диапазон изменения X координаты фланца:

print(

"X координата менялась в диапазоне от",

np**.**min(x[**-**1]),

"до",

np**.**max(x[**-**1])

)

//Диапазон изменения Y координаты фланца:

print(

"Y координата менялась в диапазоне от",

np**.**min(y[**-**1]),

"до",

np**.**max(y[**-**1])

)

// отображение изменения скорости движения

t\_end **=** 10

step **=** 0.01

t **=** np**.**arange(0, t\_end, step)

chain **=** scara\_chain(scara\_q(t, t\_end), scara\_l)

end **=** chain[**-**1]

velocity\_x **=** np**.**diff(end**.**translation**.**x) **/** step

velocity\_y **=** np**.**diff(end**.**translation**.**y) **/** step

velocity\_z **=** np**.**diff(end**.**translation**.**z) **/** step

velocity\_total **=** (velocity\_x **\*\*** 2 **+** velocity\_y **\*\*** 2 **+** velocity\_z **\*\*** 2) **\*\*** 0.5

fig **=** plt**.**figure()

ax **=** fig**.**add\_subplot()

ax**.**plot(t[:**-**1], velocity\_x, color**=**"#ff0000", label**=**"$V\_x$")

ax**.**plot(t[:**-**1], velocity\_y, color**=**"#00ff00", label**=**"$V\_y$")

ax**.**plot(t[:**-**1], velocity\_z, color**=**"#0000ff", label**=**"$V\_z$")

ax**.**plot(t[:**-**1], velocity\_total, color**=**"#000000", label**=**"$V$")

fig**.**legend()

fig**.**show()

//Максимальная скорость на траектории

index\_max **=** np**.**argmax(velocity\_total)

print(

"Максимальная скорость была",

velocity\_total[index\_max],

"в момент",

index\_max **\*** step,

"сек"

)

//Максимальная скорость по оси X

index\_max **=** np**.**argmax(velocity\_x)

print(

"Максимальная скорость по оси X была",

velocity\_x[index\_max],

"в момент",

index\_max **\*** step,

"сек"

)

//Максимальная скорость по оси Y

index\_max **=** np**.**argmax(velocity\_y)

print(

"Максимальная скорость по оси Y была",

velocity\_y[index\_max],

"в момент",

index\_max **\*** step,

"сек"

)

//Максимальная скорость по оси Z

index\_max **=** np**.**argmax(velocity\_z)

print(

"Максимальная скорость по оси Z была",

velocity\_z[index\_max],

"в момент",

index\_max **\*** step,

"сек"

)

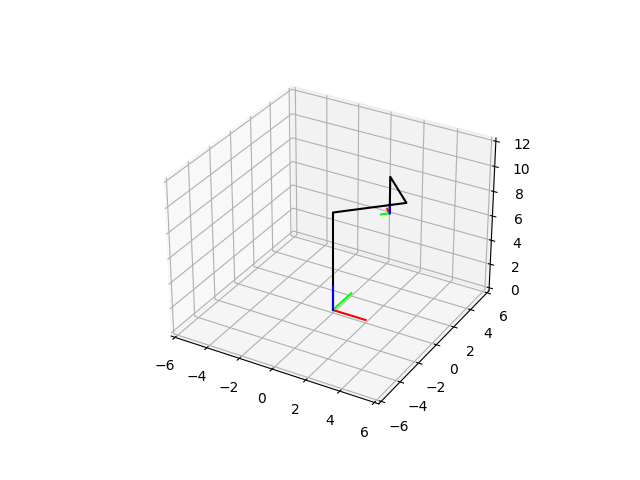


Рисунок 2. Результат работы программы

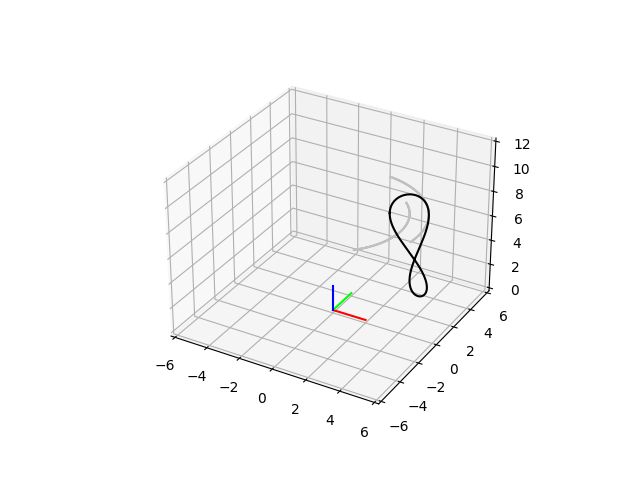


Рисунок 3. Траектория движения робота

График изменения скоростей показан на рис.4.

*Диапазон X, Y и Z координат для данной траектории.*

Z координата менялась в диапазоне от 2.0 до 8.0.

X координата менялась в диапазоне от 0.7071067811865479 до 4.999999405196338.

Y координата менялась в диапазоне от -0.7071067811865483 до 4.949747468305834.

*Максимальная скорость по осям.*

Максимальная скорость по оси X была 1.8515594722111839 в момент 1.79 сек.

Максимальная скорость по оси Y была 2.1606956560390467 в момент 7.0 сек.

Максимальная скорость по оси Z была 1.884943189667787 в момент 4.99 сек.

*Абсолютная максимальная скорость -* 2.4673894680005657 в момент 2.5 сек.

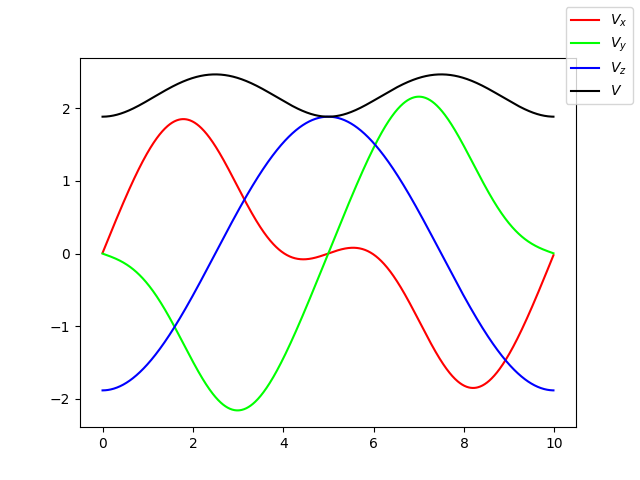


Рисунок 4. График изменения скоростей

**Задание 2.** Разобраться в примере решения прямой задачи кинематики PUMA (рис.5). Решить прямую задачу кинематики PUMA для своего закона изменения обобщенных координат.

• Оценить диапазон X, Y и Z координат для данной траектории;

• Оценить максимальную скорость по осям;

• Оценить абсолютную максимальную скорость.

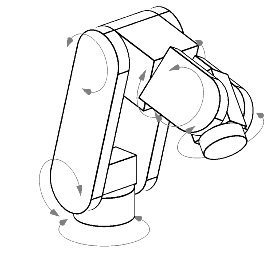


Рисунок 5. Робот PUMA

Реализация задания 2 показана в листинге 2. Результаты работы представлены ниже на рис. 6-7. Изменение скоростей показано на рис. 8.

Листинг 2 – Реализация задания 2

from matplotlib import pyplot as plt

from matplotlib import animation

import numpy as np

from IPython.display import HTML

%matplotlib notebook

from kinematics import Vector, Quaternion, Transform

import graphics

def puma\_chain(q, l, math\_source=np):

base = Transform.identity()

shoulder = base + Transform(

Vector(0, 0, l[0]),

Quaternion.from\_angle\_axis(q[0], Vector(0, 0, 1), math\_source) \*

Quaternion.from\_angle\_axis(q[1], Vector(0, 1, 0), math\_source)

)

elbow = shoulder + Transform(

Vector(0, 0, l[1]),

Quaternion.from\_angle\_axis(q[2], Vector(0, 1, 0), math\_source)

)

wrist = elbow + Transform(

Vector(0, 0, l[2]),

Quaternion.from\_angle\_axis(q[3], Vector(0, 0, 1), math\_source) \*

Quaternion.from\_angle\_axis(q[4], Vector(0, 1, 0), math\_source)

)

flange = wrist + Transform(

Vector(0, 0, l[3]),

Quaternion.from\_angle\_axis(q[5], Vector(0, 0, 1), math\_source)

)

return [base, shoulder, elbow, wrist, flange]

def puma\_q(t, total):

omega = t / total \* np.pi \* 2

return [

np.pi / 6 \* np.cos(omega),

np.pi / 3,

np.pi / 2,

omega,

np.pi / 2,

0

]

puma\_l = [2, 2, 1, 0.5]

In [19]:

puma\_fig = plt.figure()

ax = puma\_fig.add\_subplot(projection="3d")

ax.set\_xlim([-2, 2]); ax.set\_ylim([-2, 2]); ax.set\_zlim([0, 4])

lines, = ax.plot([], [], [], color="#000000")

graphics.axis(ax, Transform.identity(), 2)

r, g, b = graphics.axis(ax, Transform.identity(), 1)

total = 100

def animate(frame):

chain = puma\_chain(puma\_q(frame, total), puma\_l)

(x, y, z) = graphics.chain\_to\_points(chain)

lines.set\_data\_3d(x, y, z)

global r, g, b

r.remove(); g.remove(); b.remove()

r, g, b = graphics.axis(ax, chain[-1], 0.5)

animate(0)

fps = 25

puma\_ani = animation.FuncAnimation(

puma\_fig,

animate,

frames=total,

interval=1000.0/fps

)

HTML(puma\_ani.to\_jshtml())

t\_end = 10

step = 0.1

t = np.arange(0, t\_end, step)

chain = puma\_chain(puma\_q(t, t\_end), puma\_l)

x, y, z = graphics.chain\_to\_points(chain)

fig = plt.figure()

ax = fig.add\_subplot(projection="3d")

ax.set\_xlim([-2, 2]); ax.set\_ylim([-2, 2]); ax.set\_zlim([0, 4])

ax.plot(x[2], y[2], z[2], color="#c0c0c0")

ax.plot(x[3], y[3], z[3], color="#c0c0c0")

ax.plot(x[4], y[4], z[4], color="#000000")

graphics.axis(ax, Transform.identity(), 2)

fig.show()

print(

"Z координата менялась в диапазоне от",

np.min(z[-1]),

"до",

np.max(z[-1])

)

print(

"X координата менялась в диапазоне от",

np.min(x[-1]),

"до",

np.max(x[-1])

)

print(

"Y координата менялась в диапазоне от",

np.min(y[-1]),

"до",

np.max(y[-1])

)

t\_end = 10

step = 0.01

t = np.arange(0, t\_end, step)

chain = puma\_chain(puma\_q(t, t\_end), puma\_l)

end = chain[-1]

velocity\_x = np.diff(end.translation.x) / step

velocity\_y = np.diff(end.translation.y) / step

velocity\_z = np.diff(end.translation.z) / step

velocity\_total = (velocity\_x \*\* 2 + velocity\_y \*\* 2 + velocity\_z \*\* 2) \*\* 0.5

fig = plt.figure()

ax = fig.add\_subplot()

ax.plot(t[:-1], velocity\_x, color="#ff0000", label="$V\_x$")

ax.plot(t[:-1], velocity\_y, color="#00ff00", label="$V\_y$")

ax.plot(t[:-1], velocity\_z, color="#0000ff", label="$V\_z$")

ax.plot(t[:-1], velocity\_total, color="#000000", label="$V$")

fig.legend()

fig.show()

index\_max = np.argmax(velocity\_total)

print(

"Максимальная скорость была",

velocity\_total[index\_max],

"в момент",

index\_max \* step,

"сек"

)

index\_max = np.argmax(velocity\_x)

print(

"Максимальная скорость по оси X была",

velocity\_x[index\_max],

"в момент",

index\_max \* step,

"сек"

)

index\_max = np.argmax(velocity\_y)

print(

"Максимальная скорость по оси Y была",

velocity\_y[index\_max],

"в момент",

index\_max \* step,

"сек"

)

index\_max = np.argmax(velocity\_z)

print(

"Максимальная скорость по оси Z была",

velocity\_z[index\_max],

"в момент",

index\_max \* step,

"сек"

)

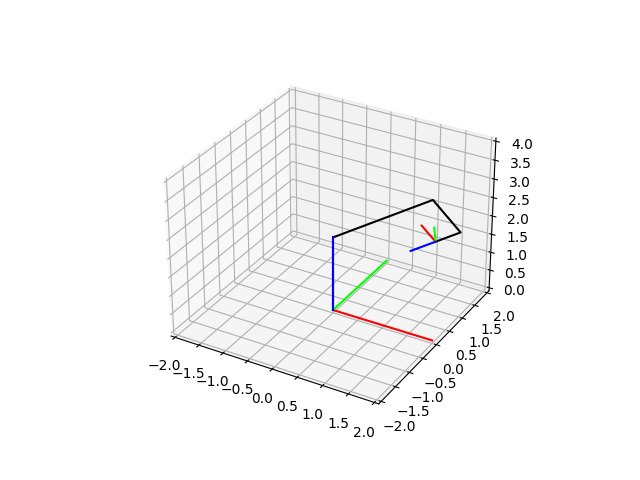


Рисунок 6. Результат работы программы

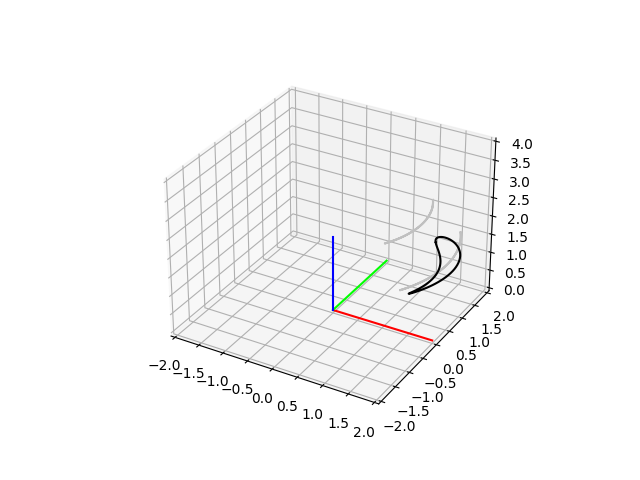


Рисунок 7. Траектория движения робота

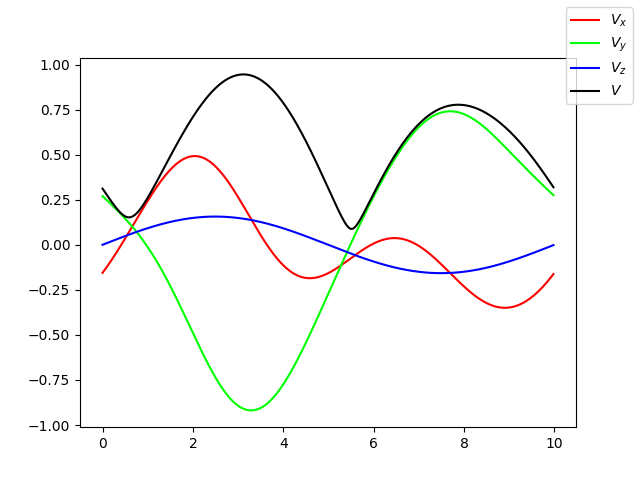


Рисунок 8. График изменения скоростей

*Диапазон X, Y и Z координат для данной траектории.*

Z координата менялась в диапазоне от 1.8839745962155616 до 2.3839745962155616.

X координата менялась в диапазоне от 1.5240970474083972 до 2.49587346279243.

Y координата менялась в диапазоне от -1.3985749429997412 до 1.0413955451425434.

*Максимальная скорость по осям.*

Максимальная скорость по оси X была 0.4928930628488981 в момент 2.04 сек.

Максимальная скорость по оси Y была 0.7414834937571335 в момент 7.69 сек.

Максимальная скорость по оси Z была 0.15707859913902666 в момент 2.49 сек.

*Абсолютная максимальная скорость -* 0.9462288965185905 в момент 3.12 сек.

**Задание 3.** Решить прямую задачу кинематики для своего манипулятора (рис.9). Реализация решения представлена в листинге 3. Ниже на рис. 10 представлена схема манипулятора, а на рис. 11 отображение траектории движения.

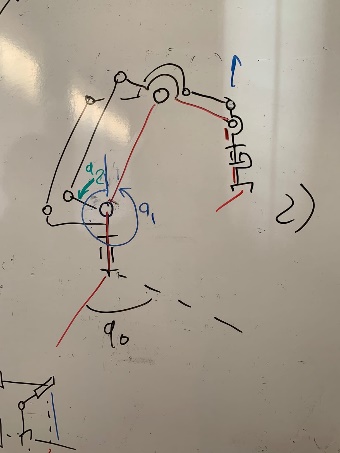


Рис. 9 Манипулятор для выполнения самостоятельного задания

Листинг 3 – Решение задания 3

|  |
| --- |
| def custom\_chain(q, l, math\_source=np):  base = Transform.identity()  end = base + Transform(  Vector(0, 0, l[0]),  Quaternion.from\_angle\_axis(q[0], Vector(0, 0, 1), math\_source) \*  Quaternion.from\_angle\_axis(q[1], Vector(0, 1, 0), math\_source)  )  temp = end + Transform(  Vector(0, 0, l[1]),  Quaternion.from\_angle\_axis(q[2] - q[1], Vector(0, 1, 0), math\_source)  )  wrist = temp + Transform(  Vector(0, 0, l[2]),  Quaternion.from\_angle\_axis(math\_source.pi - q[2], Vector(0, 1, 0), math\_source)  )  flange = wrist + Transform(  Vector(0, 0, l[3]),  Quaternion.from\_angle\_axis(q[3], Vector(0, 0, 1), math\_source)  )  return [ base, end, temp, wrist, flange ] |

Закон изменения обобщенных координат:

|  |
| --- |
| def custom\_q(t, total):  omega = t / total \* np.pi \* 2  return [  1/2 \* np.pi \* np.sin(omega)\* np.cos(omega),  -np.pi / 4 + np.pi/2 \* np.sin(omega/4),  np.pi/4 + np.pi /2 \* t/total,  omega,  ] |

Укажите длину звеньев:

|  |
| --- |
| custom\_l = [10, 15, 15, 2] |

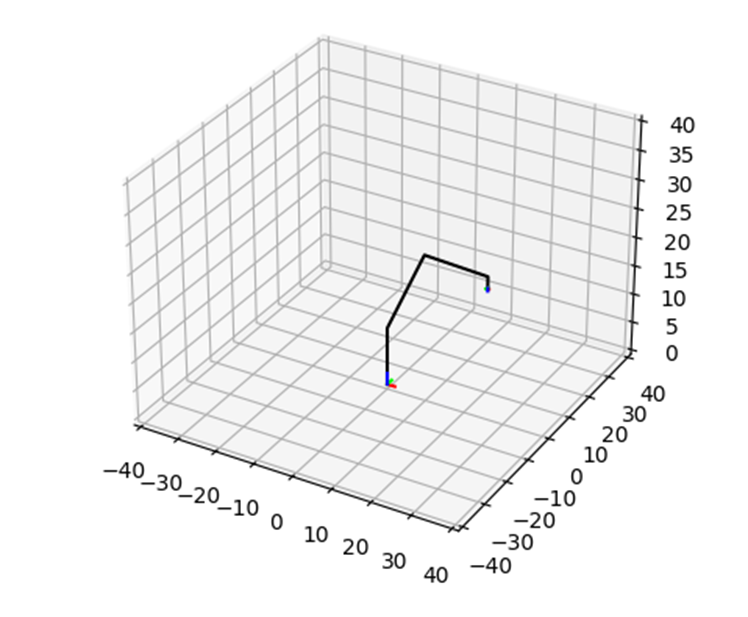


Рис. 10 Схема манипулятора

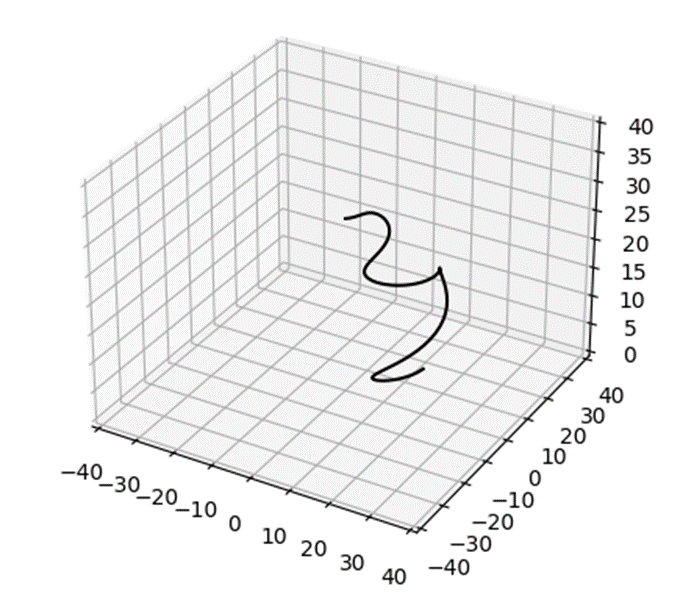


Рис. 11 Отображение траектории движения манипулятора

Оцените диапазон 𝑋, 𝑌 и 𝑍 координат для данной траектории.

Определим диапазон изменения Х координаты фланца (листинг 4):

Листинг 4 – Диапазон изменения Х

print(

"X координата менялась в диапазоне от",

np.min(x[-1]),

"до",

np.max(x[-1])

)

X координата менялась в диапазоне от -2.886579864025407e-15 до 23.123855179017184.

Определим диапазон изменения Y координаты фланца (листинг 5):

Листинг 5 – Диапазон изменения Y

print(

"Y координата менялась в диапазоне от",

np.min(y[-1]),

"до",

np.max(y[-1])

)

Y координата менялась в диапазоне от -16.09135324311646 до 15.70079421404361

Определим диапазон изменения Z координаты фланца (листинг 6):

Листинг 6 – Диапазон изменения Z

print(

"Z координата менялась в диапазоне от",

np.min(z[-1]),

"до",

np.max(z[-1])

)

Z координата менялась в диапазоне от 8.169964972656071 до 29.68494186734532.

На рис. 12 показан график изменения скоростей движения.

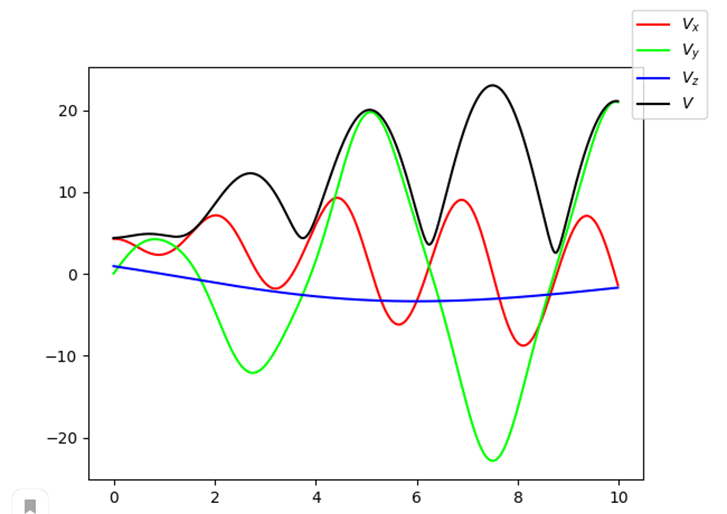


Рис. 12 Графики изменения скоростей движения манипулятора

Оцените максимальную скорость по осям и абсолютную максимальную скорость. Оценим абсолютную максимальную скорость (листинг 7):

Листинг 7 – Оценка абсолютной скорости

index\_max = np.argmax(velocity\_total)

print(

"Максимальная скорость была",

velocity\_total[index\_max],

"в момент",

index\_max \* step,

"сек"

)

Максимальная скорость была 23.026484838731673 в момент 7.5 сек.

Оценим максимальную скорость по оси X (листинг 8):

Листинг 8 – Скорость по оси Х

index\_max = np.argmax(velocity\_x)

print(

"Максимальная скорость по оси Х была",

velocity\_x[index\_max],

"в момент",

index\_max \* step,

"сек"

)

Максимальная скорость по оси Х была 9.287776140468118 в момент 4.42 сек.

Оценим максимальную скорость по оси Y (листинг 9):

Листинг 9 – Скорость по оси Y

index\_max = np.argmax(velocity\_y)

print(

"Максимальная скорость по оси Y была",

velocity\_y[index\_max],

"в момент",

index\_max \* step,

"сек"

)

Максимальная скорость по оси Y была 21.033061478970062 в момент 9.94 сек.

Оценим максимальную скорость по оси Z (листинг 10):

Листинг 10 – Скорость по оси Z

index\_max = np.argmax(velocity\_z)

print(

"Максимальная скорость по оси Z была",

velocity\_z[index\_max],

"в момент",

index\_max \* step,

"сек"

)

Максимальная скорость по оси Z была 0.94645270829119 в момент 0.0 сек.

Аналитическое решение прямой задачи кинематики представлено в листинге 11.

Листинг 11 – Аналитическое решение задания 3

import sympy as sp

l\_0, l\_1, l\_2, l\_3 = sp.symbols("l\_0, l\_1, l\_2, l\_3")

q\_0, q\_1, q\_2, q\_3 = sp.symbols("q\_0, q\_1, q\_2, q\_3")

flange = custom\_chain([q\_0, q\_1, q\_2, q\_3], [l\_0, l\_1, l\_2, l\_3], sp)[-1]

sp.simplify(flange.translation.x)

(l1sin(q1)+l2sin(q2))cos(q0)

sp.simplify(flange.translation.y)

2(l1sin(q1)+l2sin(q2))sin(q0/2)cos(q0/2)

sp**.**simplify(flange**.**translation**.**z)

l0-2l1sin2(q1/2)+l1-2l2sin2(q2/2)+l2-l3

Вывод по лабораторной работе: изучили методы моделирования последовательных кинематических цепей и способы оценки их характеристик.