Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Факультет систем управления и робототехники

Отчет по лабораторной работе №7

«Прямая и обратная задача кинематики. DH параметры»

по дисциплине «Введение в профессиональную деятельность»

Выполнили: студенты гр. R3137
Кирбаба Д.Д.
Кравченко Д.В.
Курчавый В.В.
Мысов М.С.

Преподаватель: Перегудин А.А.,
ассистент фак. СУиР

Санкт-Петербург 2021

1. Цель работы

Ознакомиться со способом нахождения параметров манипулятора и научиться переходить из декартовых координат в обобщенные и обратно.

2. Материалы работы

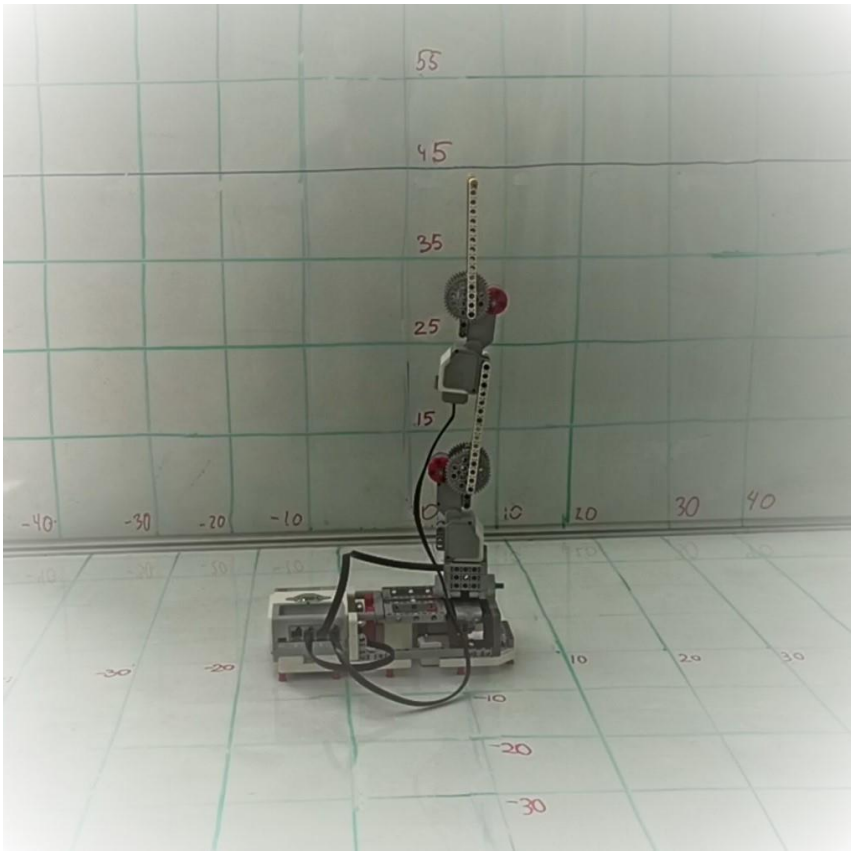

Рисунок 1. Фотография собранного манипулятора

Звено i	a_i	α_i	d_i	θ_i
1	0.3	$\pi/2$	17.3	θ_1
2	16	0	0	$\theta_2 + \pi/2$
3	10	0	0	θ_3

Таблица 1. DH параметры манипулятора

2.1. Графики эксперимента в конфигурационном пространстве

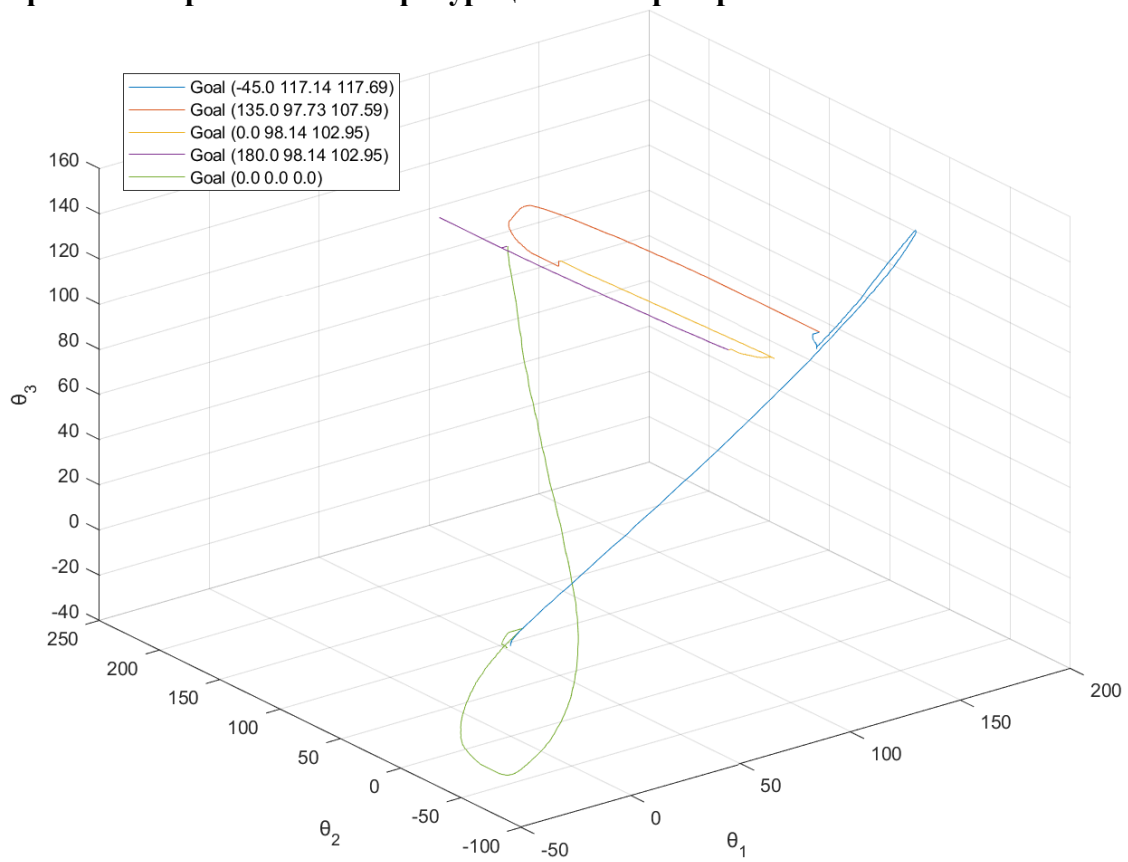

Рисунок 2. График движения при ОЗК

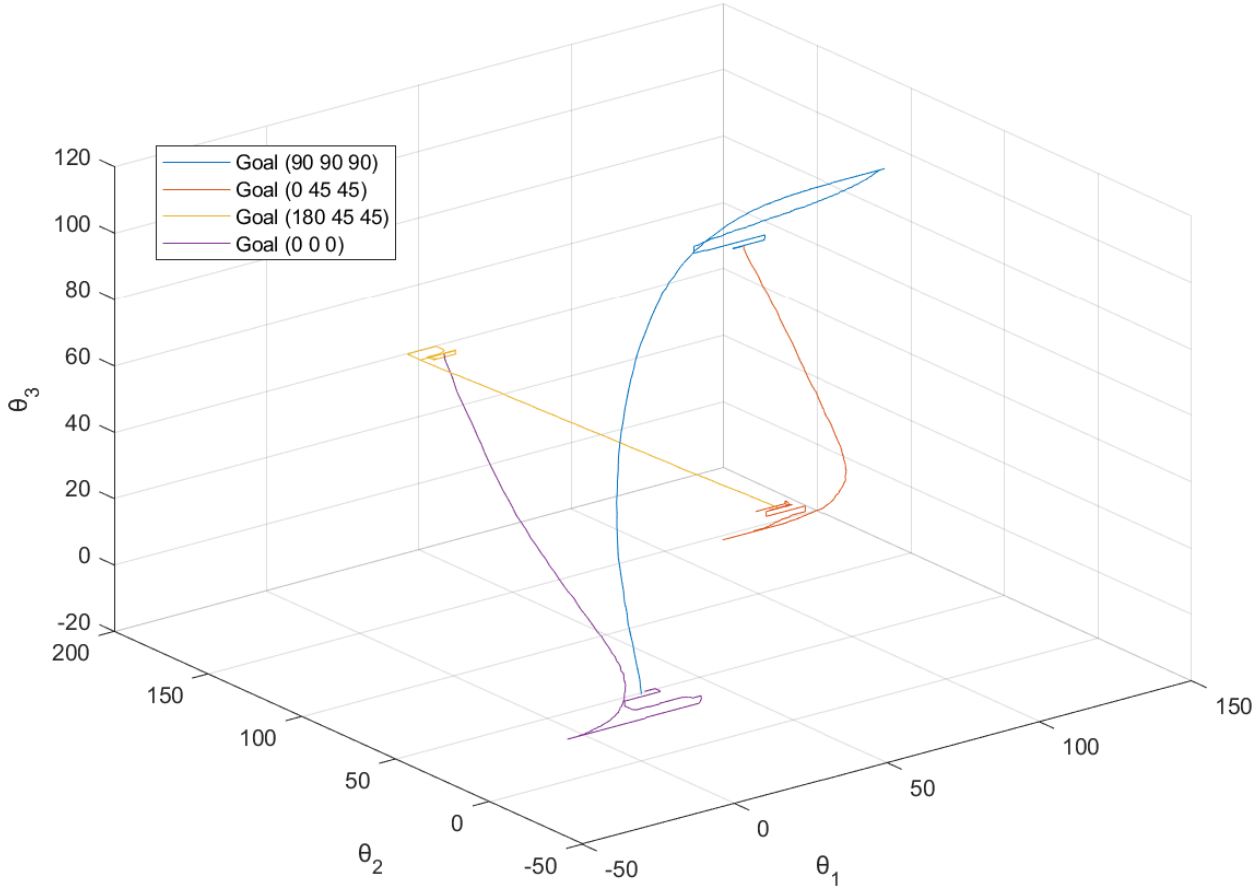

Рисунок 3. График движения при ПЗК

2.2. Графики эксперимента в операционном пространстве

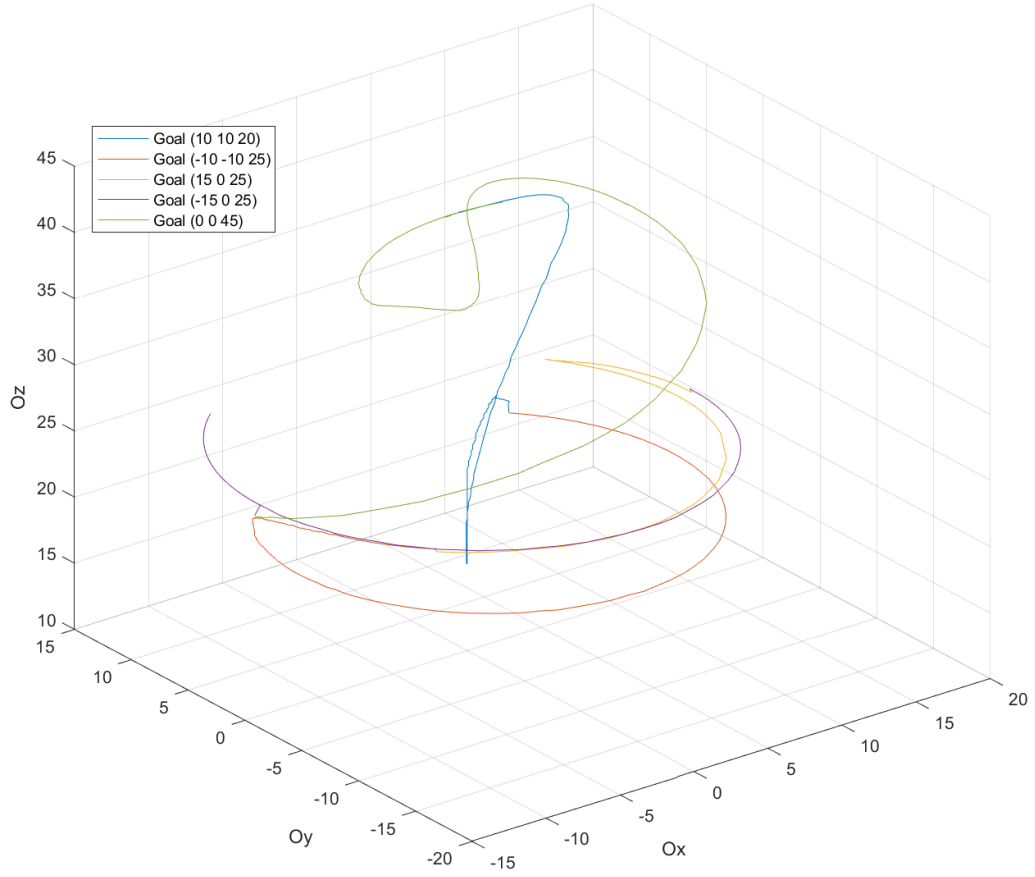

Рисунок 4. График движения при ОЗК

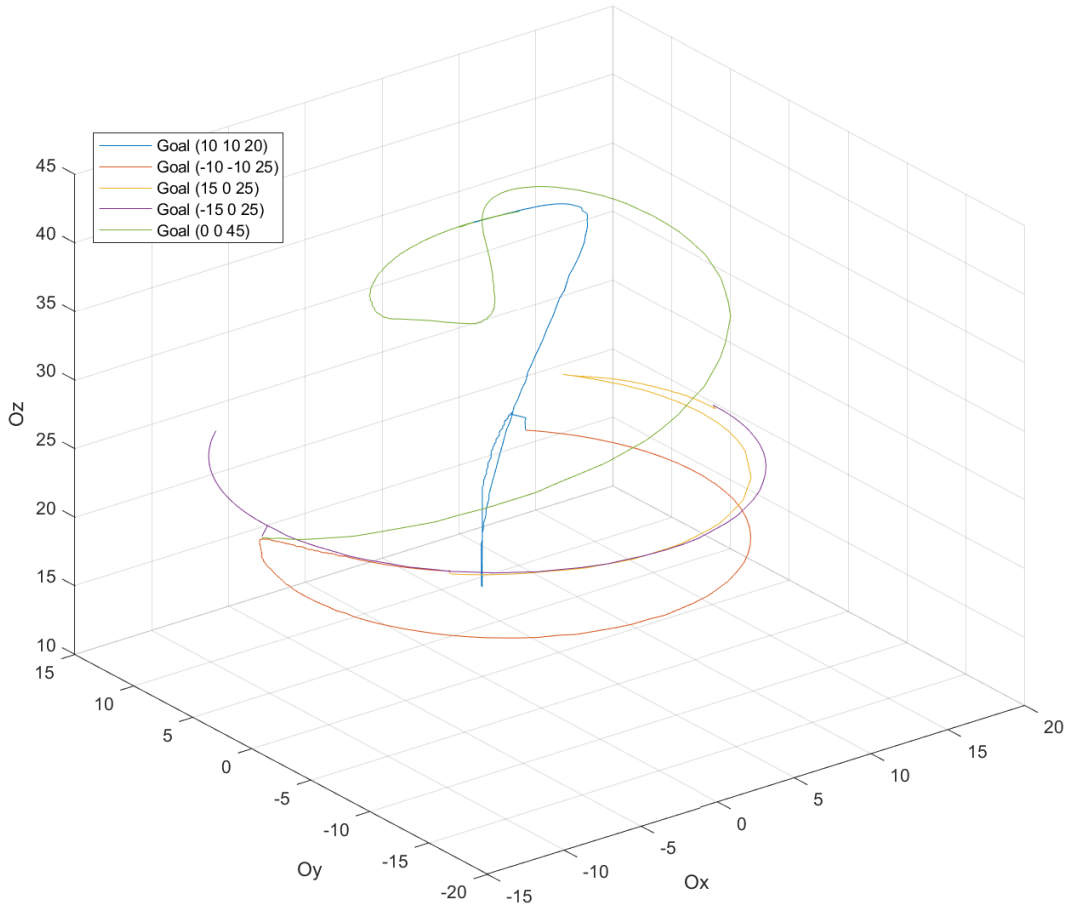

Рисунок 5. График движения при ПЗК

3. Программа на языке Python

3.1. Код для управления манипулятором

```
#!/usr/bin/env python3
from ev3dev.ev3 import *
from math import pi, sin, cos, acos, asin, atan2, atan, sqrt
import time

kp = [10, 10, 10] # proportional gain
ki = [0.5, 0.5, 0.5] # integral gain
kd = [10, 10, 10] # differential gain
geerRatio = [0.6, 0.21, 0.19] # geer ratio

uMax = 7 # max voltage

motor1 = LargeMotor('outA') # first motor declaration
motor2 = LargeMotor('outB') # second motor declaration
motor3 = LargeMotor('outC') # third motor declaration

# system DH-parameters
a = [0.06, 0.16, 0.10]
alpha = [pi/2, 0, 0]
d = [0.133, 0, 0]
tetta = [0, pi/2 - 0, -1 * 0]

def sign(x):
	if x > 0:
		return 1
	elif x < 0:
		return -1
	else:
		return 0

def getGoalTettaByGoalCords(goalCords):

	goalTetta = [0, 0, 0]

	goalTetta[0] = atan2(goalCords[1], goalCords[0])

	goalCords[0] = goalCords[0] + a[0] * cos(goalTetta[0])
	goalCords[1] = goalCords[1] + a[0] * sin(goalTetta[0])

	aaa = atan2((goalCords[2] - d[0]), sqrt((goalCords[0]**2) + (goalCords[1]**2)))
	bbb = a[1]**2 + goalCords[0]**2 + goalCords[1]**2 + (goalCords[2] - d[0])**2 -
a[2]**2
	ccc = 2 * a[1] * sqrt(goalCords[0]**2 + goalCords[1]**2 + (goalCords[2] -
d[0])**2)

	if bbb > ccc:
		ddd = 1
	else:
		ddd = bbb / ccc

	goalTetta[1] = - (pi/2 - (aaa + acos(ddd)))

	goalTetta[2] = pi - acos((a[1]**2 + a[2]**2 - goalCords[0]**2 - goalCords[1]**2
- (goalCords[2] - d[0])**2) / (2 * a[1] * a[2]))

	return goalTetta

def run(curTetta, goalTetta):

	# create file for output
	dataOutput = open("7LabMeasurementsDKP [" + str(round(curCords[0])) + " " +
str(round(curCords[1])) + " " + str(round(curCords[2])) + "] -> [" + str(goalCords[0])
+ " " + str(goalCords[1]) + " " + str(goalCords[1]) + "].txt","w+")
```

```
	motor1.position = 0 # in grad
	motor2.position = 0 # in grad
	motor3.position = 0 # in grad
	angle1 = 0 # angle of rotation of the first motor in radians
	angle2 = 0 # angle of rotation of the second motor in radians
	angle3 = 0 # angle of rotation of the third motor in radians

	prevTetta = [0, 0, 0]
	curAngleError = [0, 0, 0]
	prevAngleError = [0, 0, 0]
	integralSum = [0, 0, 0]
	uIntegral = [0, 0, 0] # uIntegral - integral component of voltage

	totalAngleError = 0
	for i in range(3):
		totalAngleError += abs(goalTetta[i] - curTetta[i])

	currentTime = time.time()
	prevTime = time.time()

	# if total angle error less than 10 degrees, then we consider that the
manipulator has achieved his goal
	while (totalAngleError > 1.74):

		prevTime = currentTime
		currentTime = time.time()
		dt = currentTime - prevTime

		for i in range(3):
			prevAngleError[i] = curAngleError[i]

		angle1Prev = angle1
		angle2Prev = angle2
		angle3Prev = angle3

		angle1 = geerRatio[0] * motor1.position * pi / 180
		angle2 = geerRatio[1] * motor2.position * pi / 180
		angle3 = geerRatio[2] * motor3.position * pi / 180

		prevTetta[0] = curTetta[0]
		prevTetta[1] = curTetta[1]
		prevTetta[2] = curTetta[2]

		curTetta[0] = prevTetta[0] + angle1
		curTetta[1] = prevTetta[1] + angle2
		curTetta[2] = prevTetta[2] + angle3

		for i in range(3):
			curAngleError[i] = goalTetta[i] - curTetta[i]

			integralSum[i] += curAngleError[i] * dt
			uIntegral[i] = ki[i] * integralSum[i]

			# anti-windup for integral component (limited to 15% of maximum
power)
			if abs(uIntegral[i] / uMax * 100) > 15:
				uIntegral[i] = sign(uIntegral[i]) * uMax * 0.15

		u1 = kp[0] * curAngleError[0] + uIntegral[0] + kd[0] * (curAngleError[0]
- prevAngleError[0]) / dt
		u2 = kp[1] * curAngleError[1] + uIntegral[1] + kd[1] * (curAngleError[1]
- prevAngleError[1]) / dt
		u3 = kp[2] * curAngleError[2] + uIntegral[2] + kd[2] * (curAngleError[2]
- prevAngleError[2]) / dt

		if abs(u1) > uMax:
			voltage1 = sign(u1) * 100
```

```
            else:
                  voltage1 = u1 / uMax * 100

            if abs(u2) > uMax:
                  voltage2 = sign(u2) * 100
            else:
                  voltage2 = u2 / uMax * 100

            if abs(u3) > uMax:
                  voltage3 = sign(u3) * 100
            else:
                  voltage3 = u3 / uMax * 100

            # power supply to the motors
            motor1.run_direct(duty_cycle_sp = (voltage1))
            motor2.run_direct(duty_cycle_sp = (voltage2))
            motor3.run_direct(duty_cycle_sp = (voltage3))

            # output positions
            dataOutput.write(str(curTetta[0]) + '\t' + str(curTetta[1]) + '\t' +
str(curTetta[2]) + '\n')

            totalAngleError = 0
            for i in range(3):
                  totalAngleError += abs(goalTetta[i] - curTetta[i])

      dataOutput.close()

      motor1.stop(stop_action = 'brake')
      motor2.stop(stop_action = 'brake')
      motor3.stop(stop_action = 'brake')

      return curTetta

run()
```

3.2. Код для перевода углов в координаты

```
from math import cos, pi, sin

import numpy as np

def run(l):
  data = open('test_7_'+ str(l) + '.txt', 'r')
  data_output = open('out_7_'+ str(l) + '.txt', 'w+')
  text = data.read()
  count = text.count("\n")
  text = text[text.find("\n") + 1:]

  a = [0.3, 16, 10]
  alpha = [pi/2, 0, 0]
  d = [17.3, 0, 0]

  tetta = [0, 0, 0]
  # vector in the base (initial) coordinate system
  k0 = np.matrix([0, 0, 0, 1])
  T03 = np.identity(4)
  for i in range(count - 1):
    index = text.find("\n")
    line = text[:index]

    findtime = line.find(" ")
    time = line[:findtime]
    line = line[findtime + 1:]

    findangle = line.find(" ")
    angle1 = line[:findangle]
```

```
    line = line[findangle + 1:]

    findangle = line.find(" ")
    angle2 = line[:findangle]
    line = line[findangle + 1:]

    angle3 = line
    tetta = [-float(angle1) / 180 * pi, pi/2 - float(angle2) / 180 * pi,
float(angle3)/ 180 * pi]

    x0 = np.matrix([0, 0, 0, 1]).transpose()

    z01Rotate = np.matrix([[cos(tetta[0]), sin(tetta[0]), 0, 0], [-sin(tetta[0]),
cos(tetta[0]), 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]])
    x01Rotate = np.matrix([[1, 0, 0, 0], [0, cos(alpha[0]), sin(alpha[0]), 0], [0, -
sin(alpha[0]), cos(alpha[0]), 0], [0, 0, 0, 1]])
    offset01x = np.matrix([[1, 0, 0, -a[0]], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0,
1]])
    offset01z = np.matrix([[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, d[0]], [0, 0, 0, 1]])

    z12Rotate = np.matrix([[cos(tetta[1]), sin(tetta[1]), 0, 0], [-sin(tetta[1]),
cos(tetta[1]), 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]])
    offset12x = np.matrix([[1, 0, 0, a[1]], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]])

    z23Rotate = np.matrix([[cos(tetta[2]), sin(tetta[2]), 0, 0], [-sin(tetta[2]),
cos(tetta[2]), 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]])
    offset23x = np.matrix([[1, 0, 0, a[2]], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]])

    transition01 = np.matmul(z01Rotate, offset01z)
    transition01 = np.matmul(offset01x, transition01)
    transition01 = np.matmul(x01Rotate, transition01)

    transition01 = np.matmul(z12Rotate, transition01)
    transition01 = np.matmul(offset12x, transition01)

    transition01 = np.matmul(z23Rotate, transition01)
    transition01 = np.matmul(offset23x, transition01)

    transition01 = np.linalg.inv(transition01)

    x1 = np.matmul(transition01, x0) * (-1)
    x = float(x1[0][0])
    y = float(x1[1][0])
    z = float(x1[2][0])
    data_output.write(str(x) + " " +
          str(y) + " " +
          str(z) + "\n")

    text = text[index + 1:]
  data.close()
  data_output.close()

for i in range(5):
  run(i)
```

4. Вывод

В ходе лабораторной работы мы рассчитали DH параметры для манипулятора и научились переходить от декартовлй системы координат в обобщенную. Мы успешно рассчитали ОЗК и ПЗК системы и управляли манипулятором.