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 №6 «УПРАВЛЕНИЕ МНОГОЗВЕННЫМ МАНИПУЛЯТОРОМ»
по дисциплине «Введение в профессиональную деятельность»

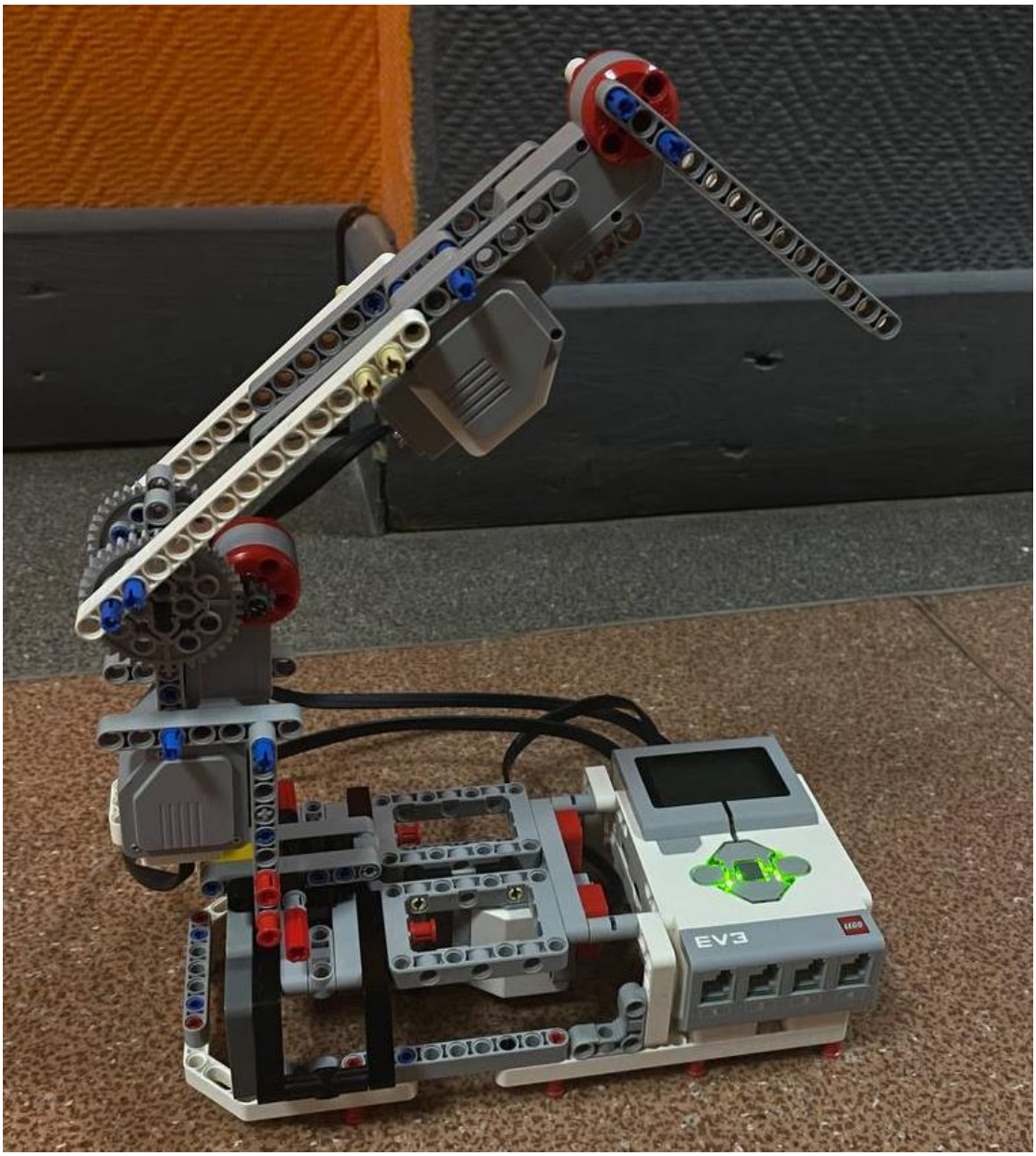
Выполнили: студенты гр. **R3142**
Рогозина В. С.
Петрищев А. С.
Подзоров А.В.
Лоскутова И.В.
Преподаватель: Перегудин А. А.,
ассистент фак.СУиР

Санкт-Петербург
2022

1. Цель работы

Реализовать алгоритм управления многозвенным манипулятором. Решить прямую и обратную задачи кинематики.

2. Фотография манипулятора



3. Материалы работы

3.1 Результаты необходимых измерений

1. Описание процесса расчёта параметров Денавита-Хартенберга для собранного манипулятора

Для начала, мы присвоили нашему манипулятору некоторые оси по алгоритму, приведённому в методических указаниях данной лабораторной работы. Затем мы определили DH-параметры(с помощью линейки измерили расстояния a_i и d_i) для собранного нами манипулятора по алгоритму, представленному на рисунке:

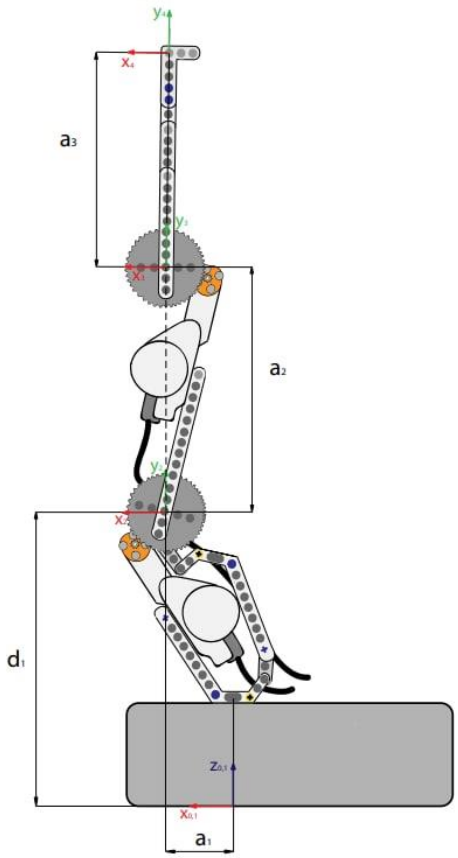


Рисунок 2.1.3 Система с обозначенными DH параметрами.

2. Таблица DH-параметров

звено	a_i	α_i	d_i	θ_i
1	0	$\pi/2$	0.16	θ_1
2	0.15	0	0	θ_2
3	0.09	0	0	θ_3

4. Решение задач

В методических указаниях представлены формулы для решения прямой и обратной задачи кинематики. Однако, опытным путём мы выяснили, что данные нам формулы справедливы лишь для Г-образного начального положения манипулятора. В нашем случае начальное положение отличается, поэтому мы самостоятельно вывели некоторые формулы для решения прямой и обратной задачи кинематики. Так же стоит учитывать, что начальное положение нашего манипулятора не соответствует точке (0,0,0). В нашем случае начальное положение робота (0.2,0.5,0.2). Это одна из главных причин, почему наши графики не выходят из начала координат.

4.1 ПРЯМАЯ ЗАДАЧА КИНЕМАТИКИ

1. Решение прямой задачи кинематики

```
%DH-ПАРАМЕТРЫ
% Расстояния вдоль оси xi(текущая ось) от zi-1 до zi
A1=0;
A2=0.14;
A3=0.09;
```

```
% Угол вращения вокруг оси xi (текущая ось) от zi–1 до zi
a1=pi/2;
a2=0;
a3=0;
% Расстояния вдоль оси zi–1 (предыдущая ось) от xi–1 до xi
d1=0.16;
d2=0;
d3=0;
% Угол вращения вокруг оси zi–1 (предыдущая ось) от xi–1 до xi
Q1=90
Q2=60;
Q3=45;

T01=[cos(Q1) -cos(a1)*sin(Q1) sin(a1)*sin(Q1) A1*cos(Q1);
     sin(Q1) cos(a1)*cos(Q1) -sin(a1)*cos(Q1) A1*sin(Q1);
     0 sin(a1) cos(a1) d1;
     0 0 0 1];

T02=[cos(Q2) -cos(a2)*sin(Q2) sin(a2)*sin(Q2) A2*cos(Q2);
     sin(Q2) cos(a2)*cos(Q2) -sin(a2)*cos(Q2) A2*sin(Q2);
     0 sin(a2) cos(a2) d2;
     0 0 0 1];

T03=[cos(Q3) -cos(a3)*sin(Q3) sin(A3)*sin(Q3) A3*cos(Q3);
     sin(Q3) cos(a3)*cos(Q3) -sin(A3)*cos(Q3) A3*sin(Q3);
     0 sin(a3) cos(a3) d3;
     0 0 0 1];

T=T01*T02*T03;

%Получение координат
disp(T(1,4))
disp(T(2,4))
disp(T(3,4))
```

2. Результаты построений

1. Заданные углы поворота – 90,60,45. Начальная координата робота - (0.2,0.5,0.2).
   Конечная координата робота(теоретическая) – (0,01, 0,04, 0,36).
   Конечная координата робота(практическая) – (0,04, 0,04, 0,34)

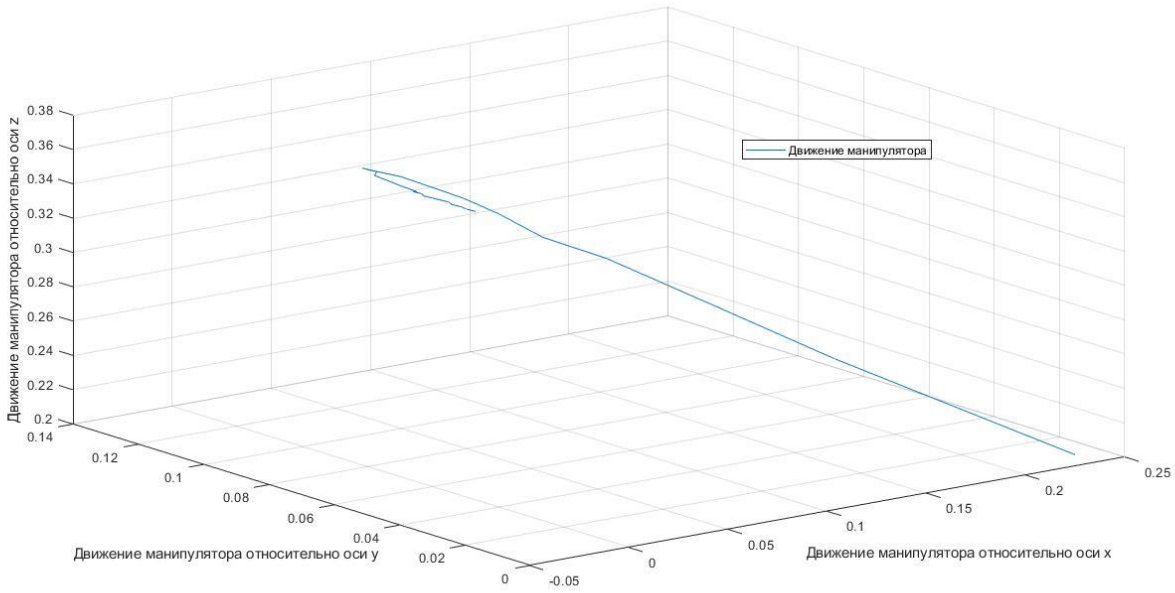


Рисунок 1. График изменения траектории движения манипулятора

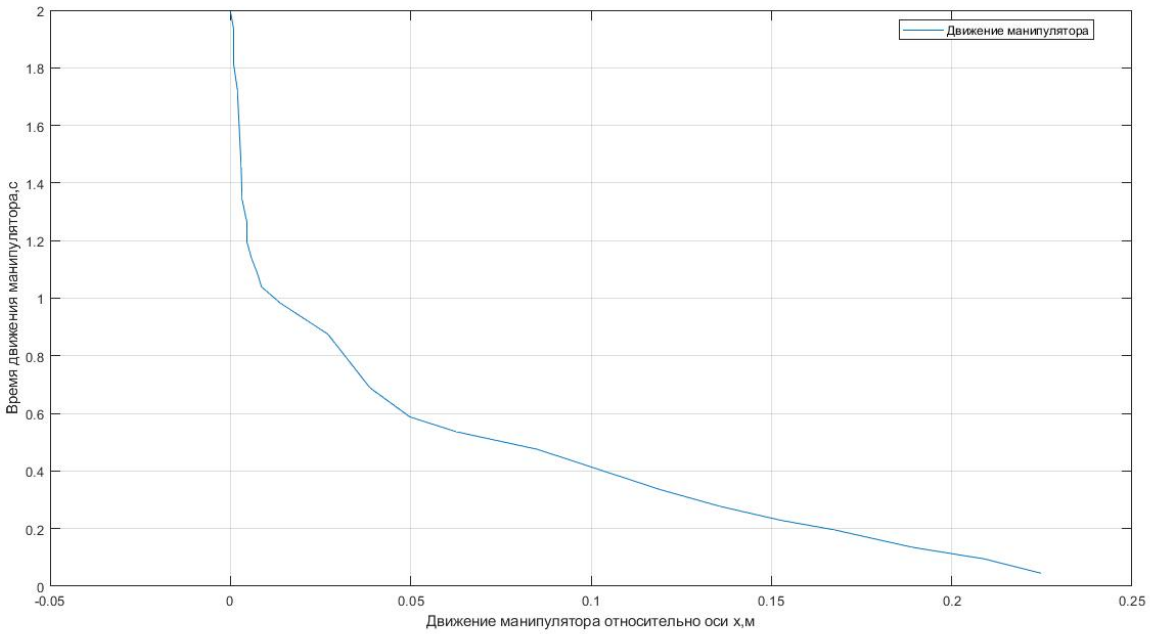


Рисунок 2. График зависимости изменения координаты от времени x(t)

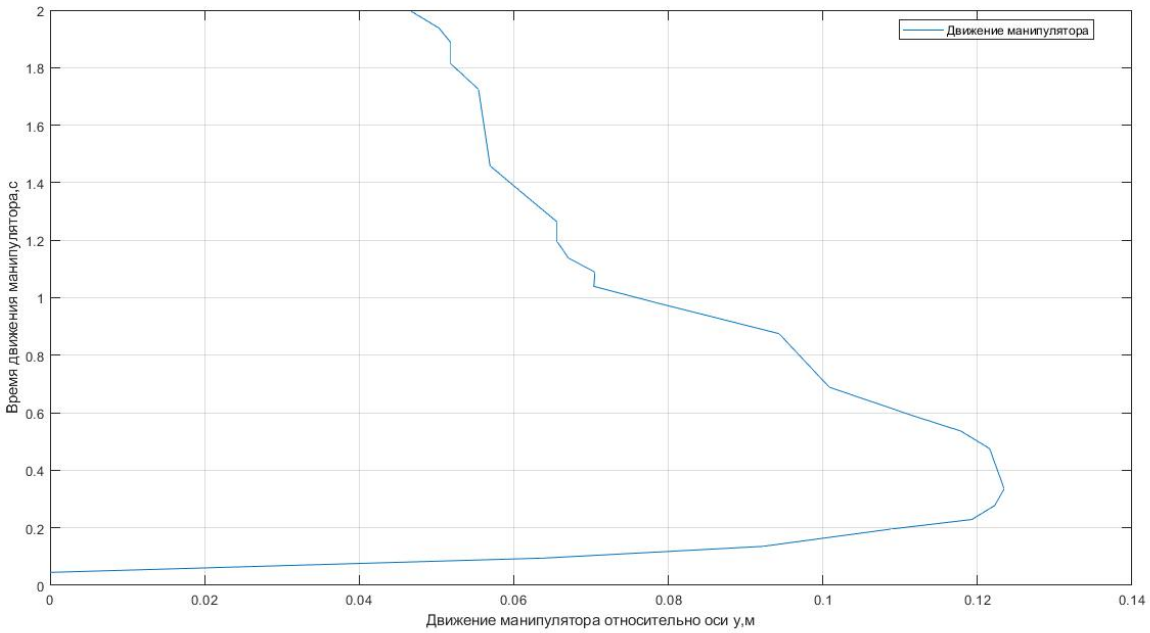


Рисунок 3. График зависимости изменения координаты от времени y(t)

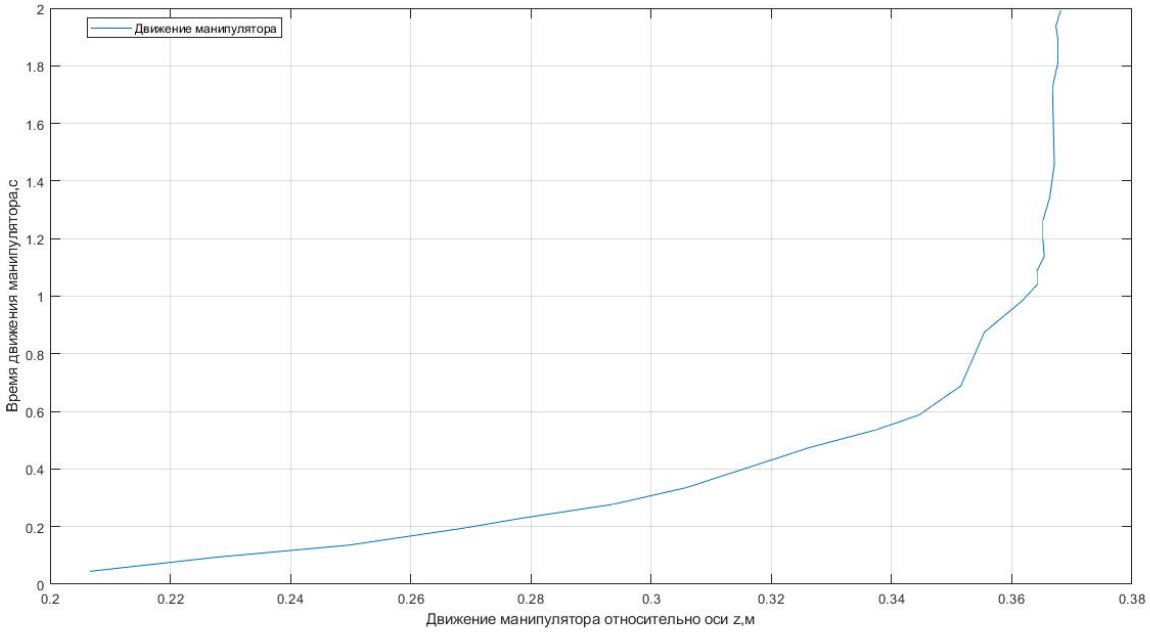


Рисунок 4. График зависимости изменения координаты от времени z(t)

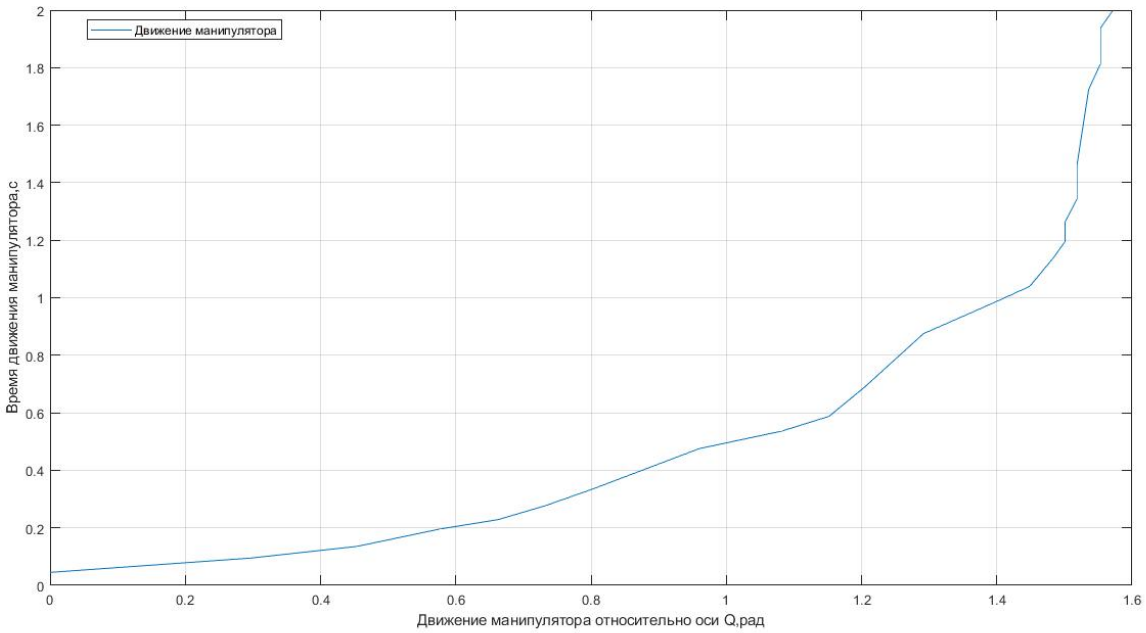


Рисунок 5. График зависимости угла поворота q1(t)

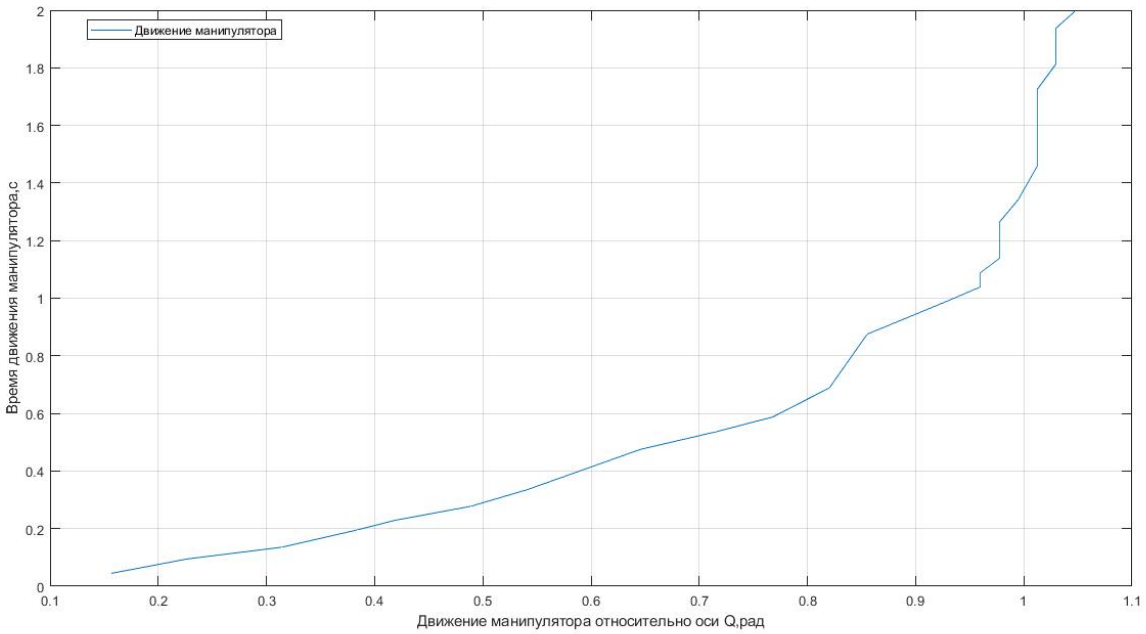


Рисунок 6. График зависимости угла поворота q2(t)

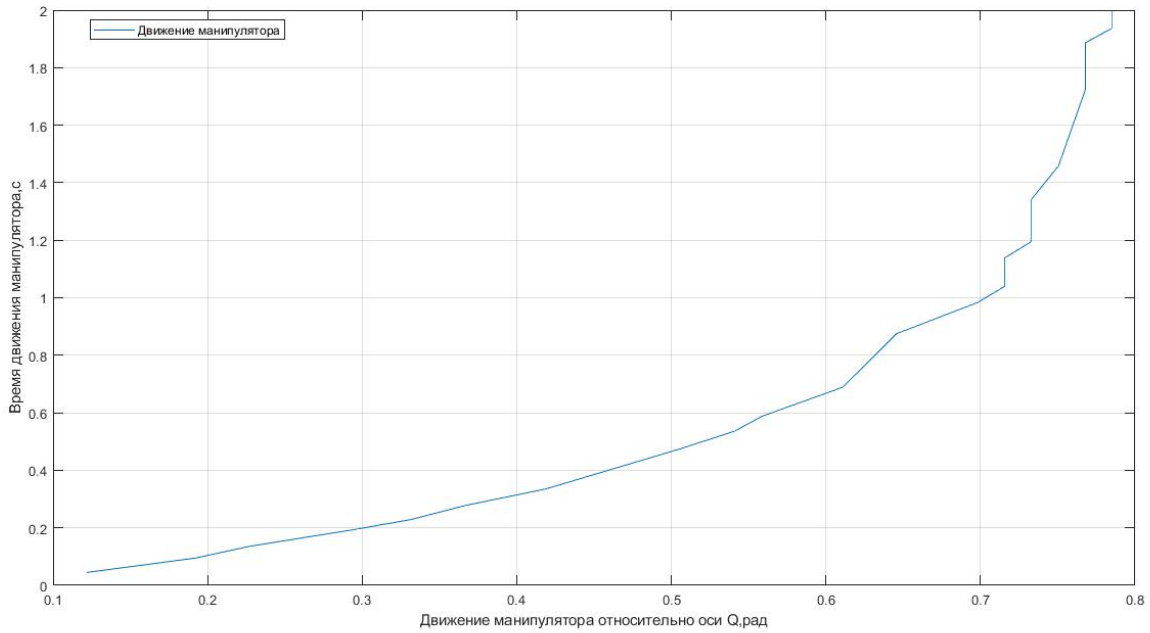


Рисунок 7. График зависимости угла поворота q3(t)

2. Заданные углы поворота – -90,60,45. Начальная координата робота - (0.2,0.5,0.2). Конечная координата робота(теоретическая) – (0,01, -0,04, 0,36).
Конечная координата робота(практическая) – (0,02, -0,06, 0,32)

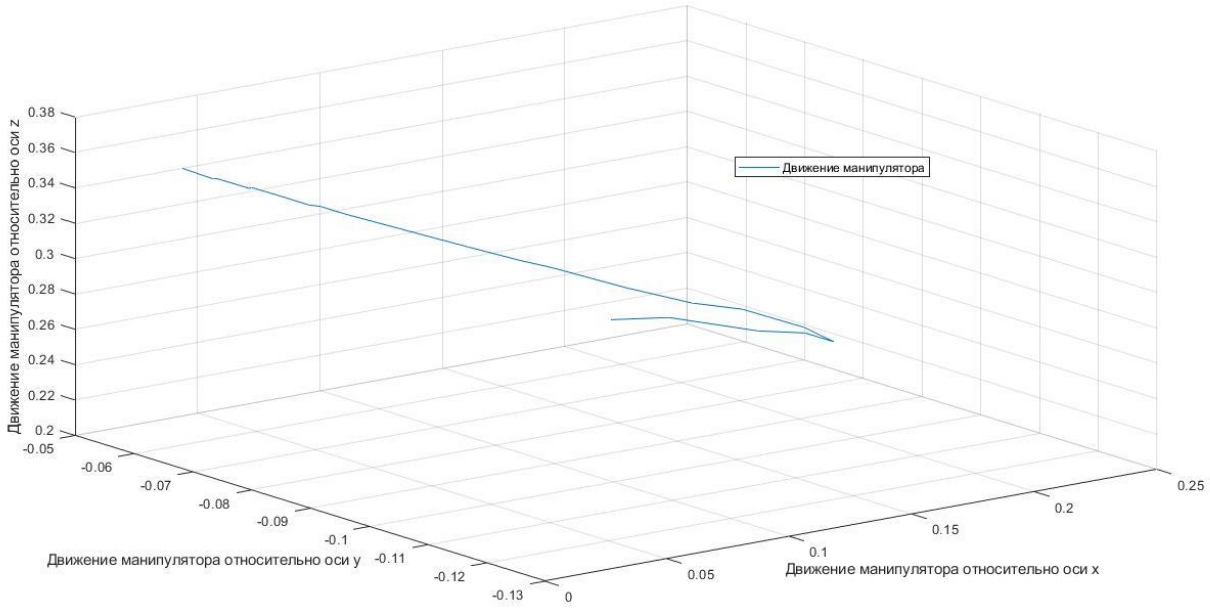


Рисунок 8. График изменения траектории движения манипулятора

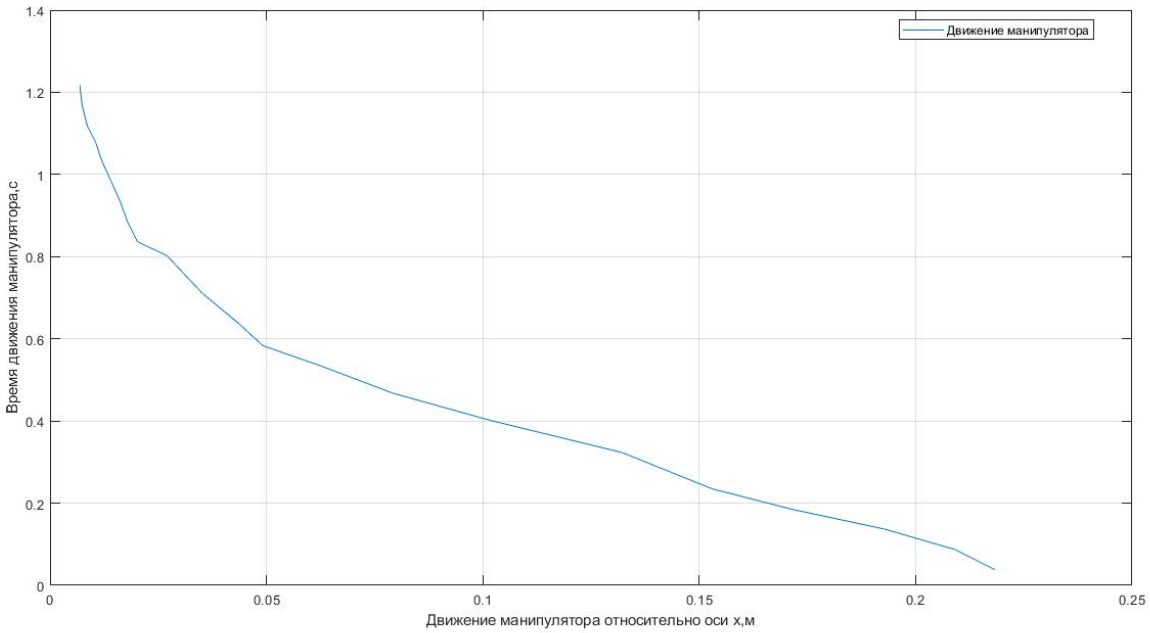


Рисунок 9. График зависимости изменения координаты от времени x(t)

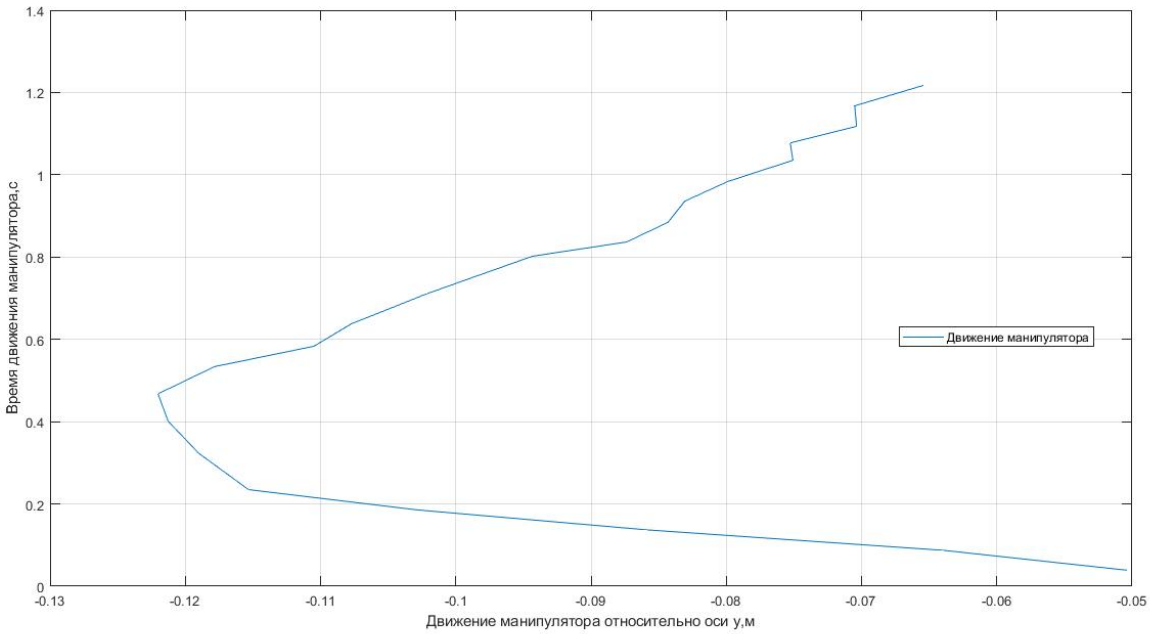


Рисунок 10. График зависимости изменения координаты от времени y(t)

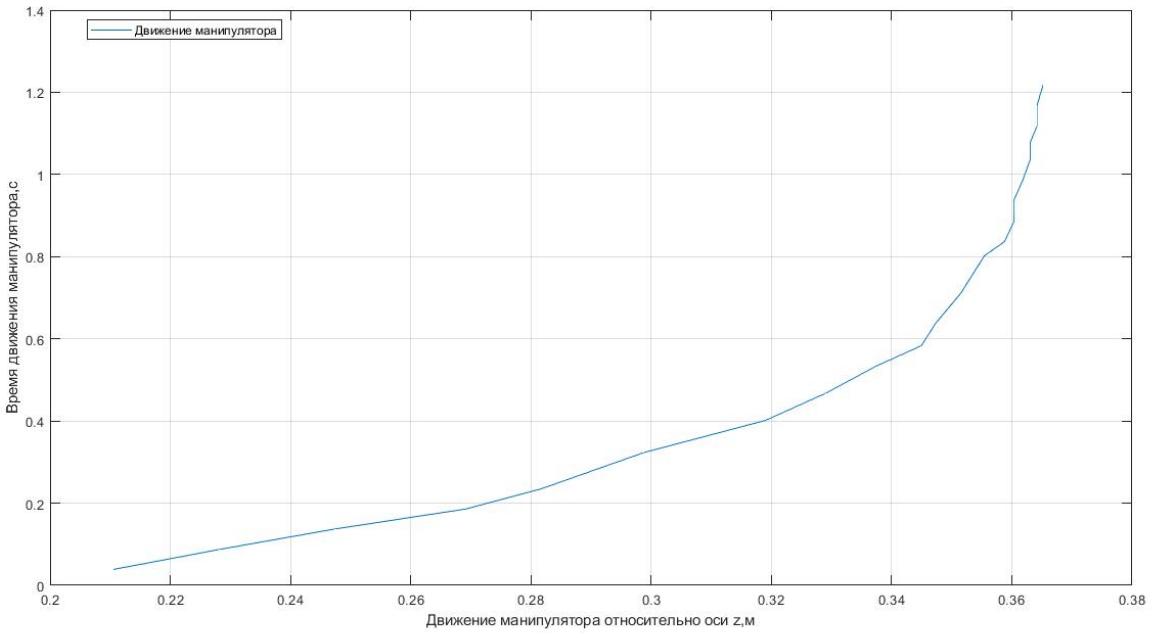


Рисунок 11. График зависимости изменения координаты от времени z(t)

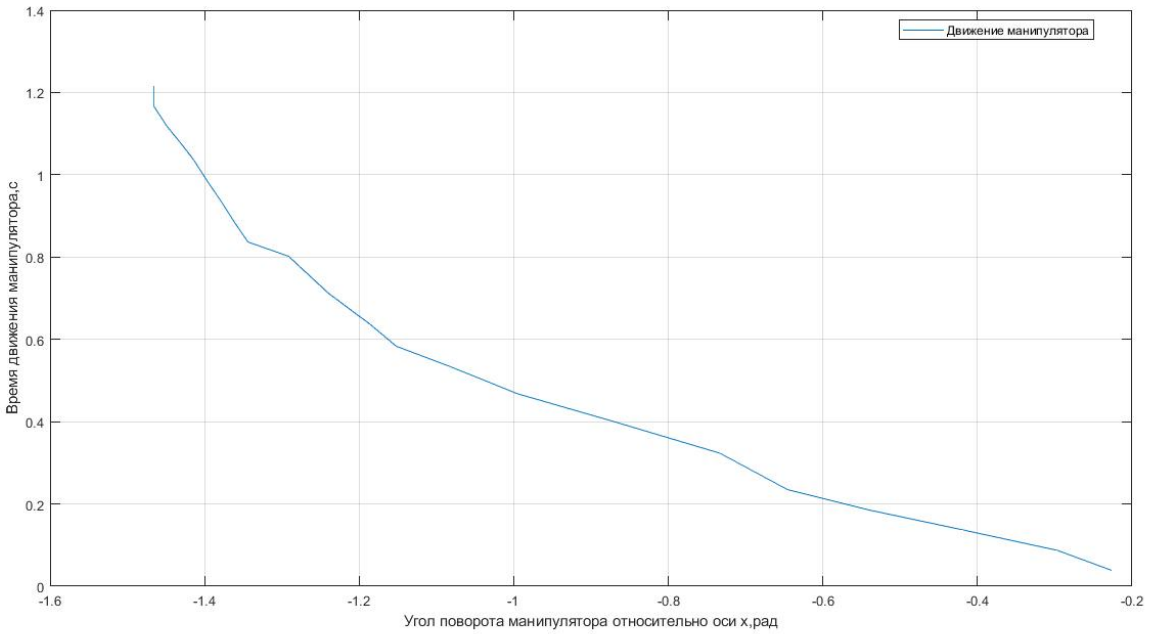


Рисунок 12. График зависимости угла поворота q1(t)

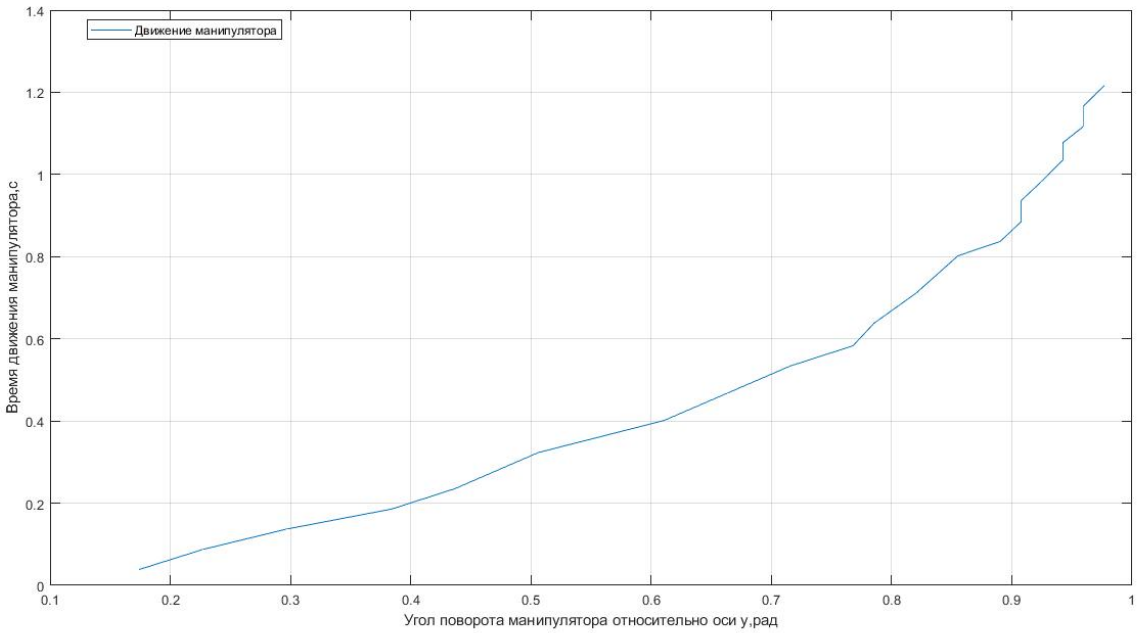


Рисунок 13. График зависимости угла поворота q2(t)

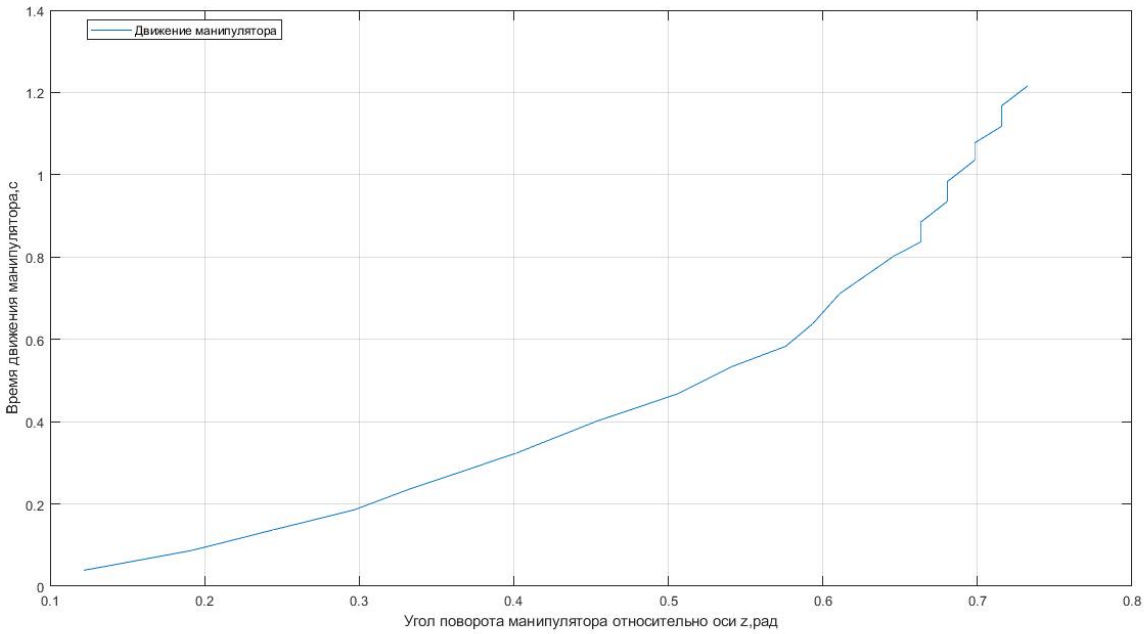


Рисунок 14. График зависимости угла поворота q3(t)

3. Заданные углы поворота – 30,30,60. Начальная координата робота - (0.2,0.05,0.2). Конечная координата робота(теоретическая) – (0,01, 0,06, 0,32). Конечная координата робота(практическая) – (0,01, 0,1, 0,35)

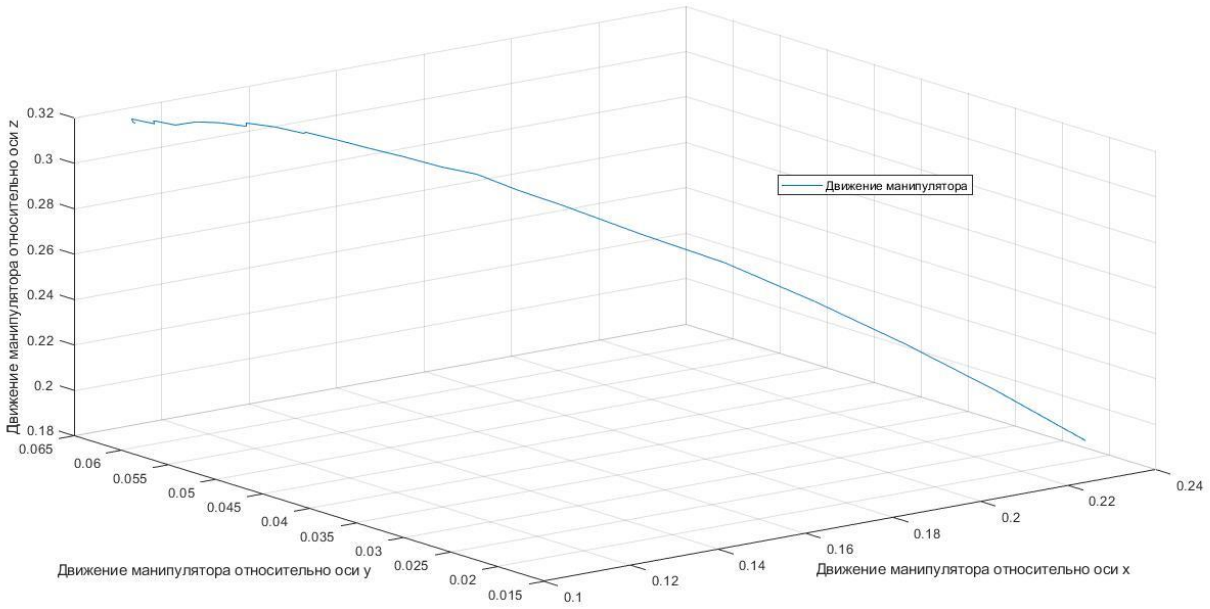


Рисунок 15. График изменения траектории движения манипулятора

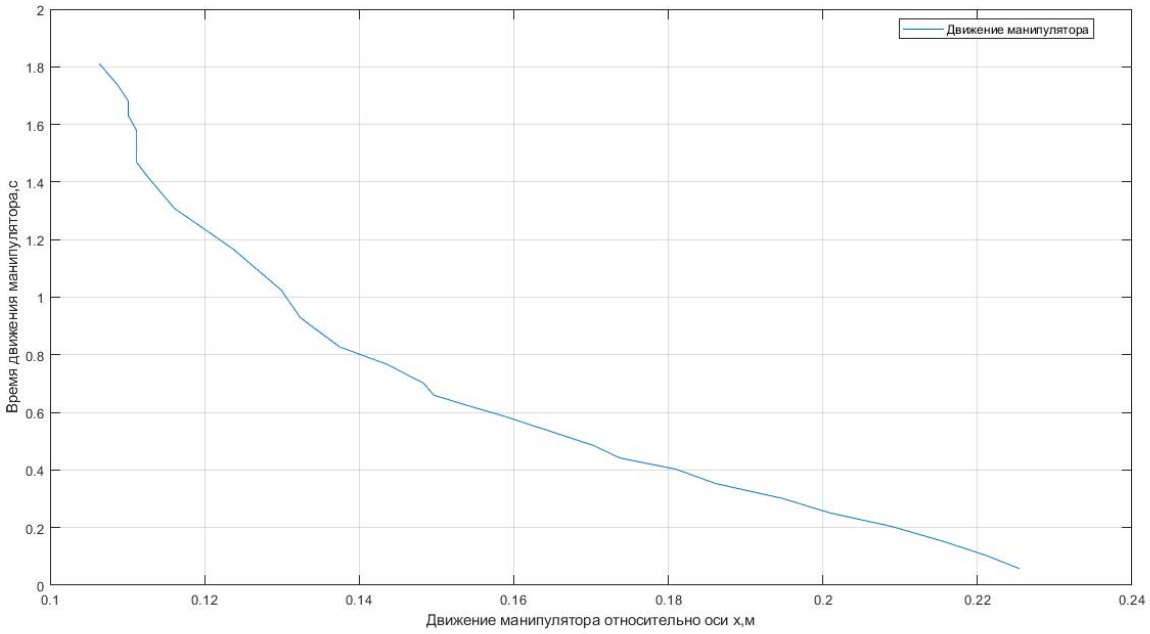


Рисунок 16. График зависимости изменения координаты от времени x(t)

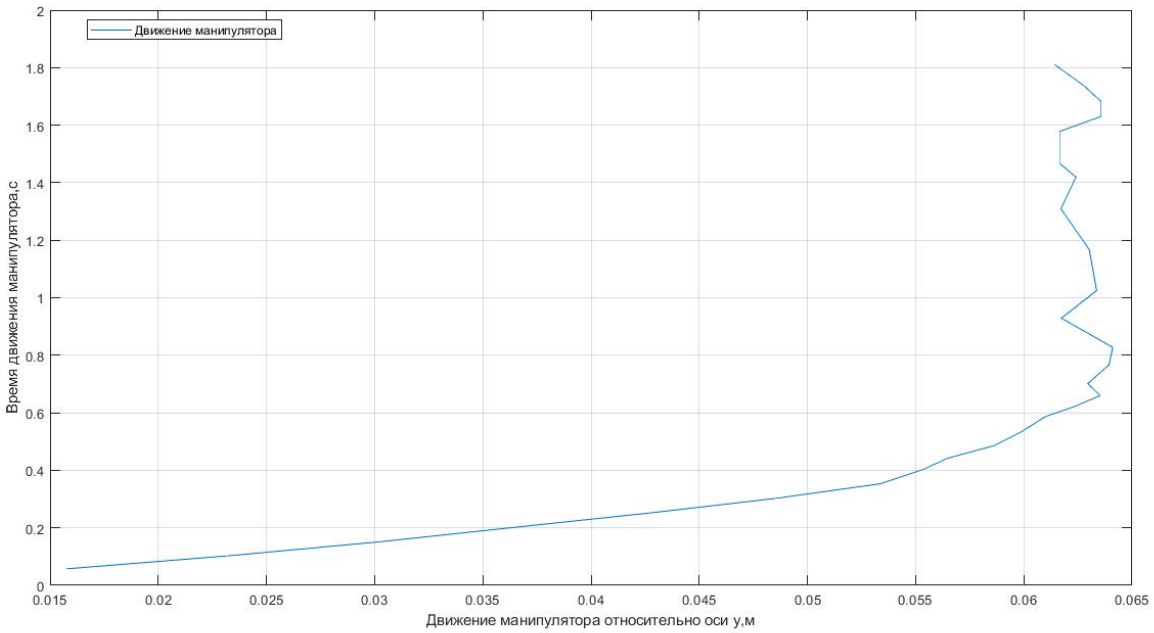


Рисунок 17. График зависимости изменения координаты от времени y(t)

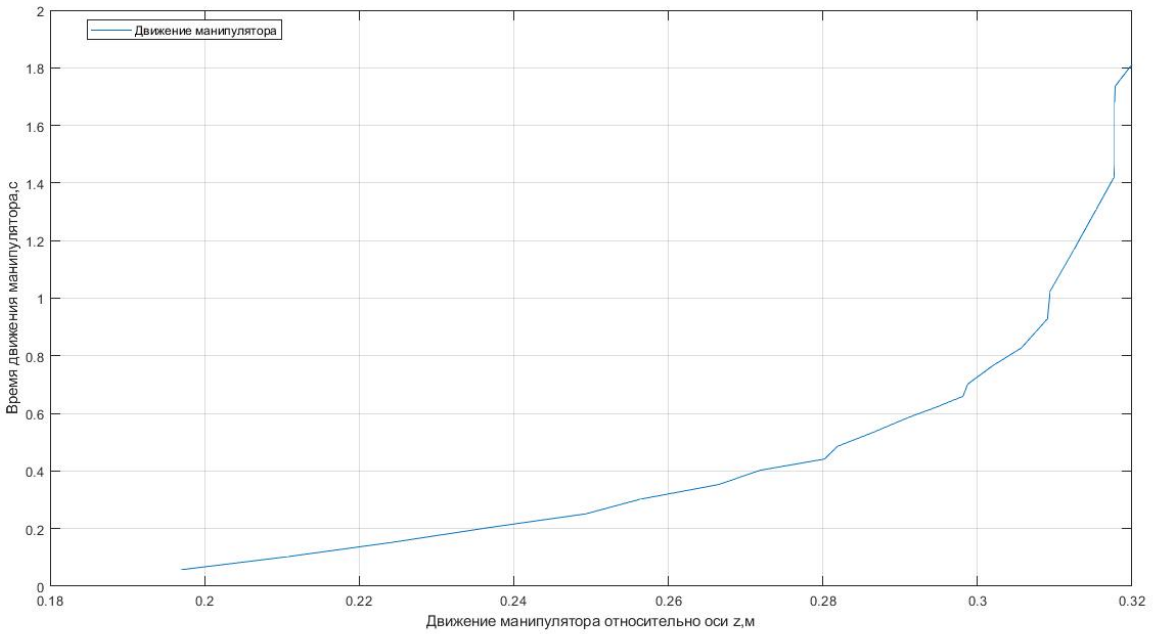


Рисунок 18. График зависимости изменения координаты от времени z(t)

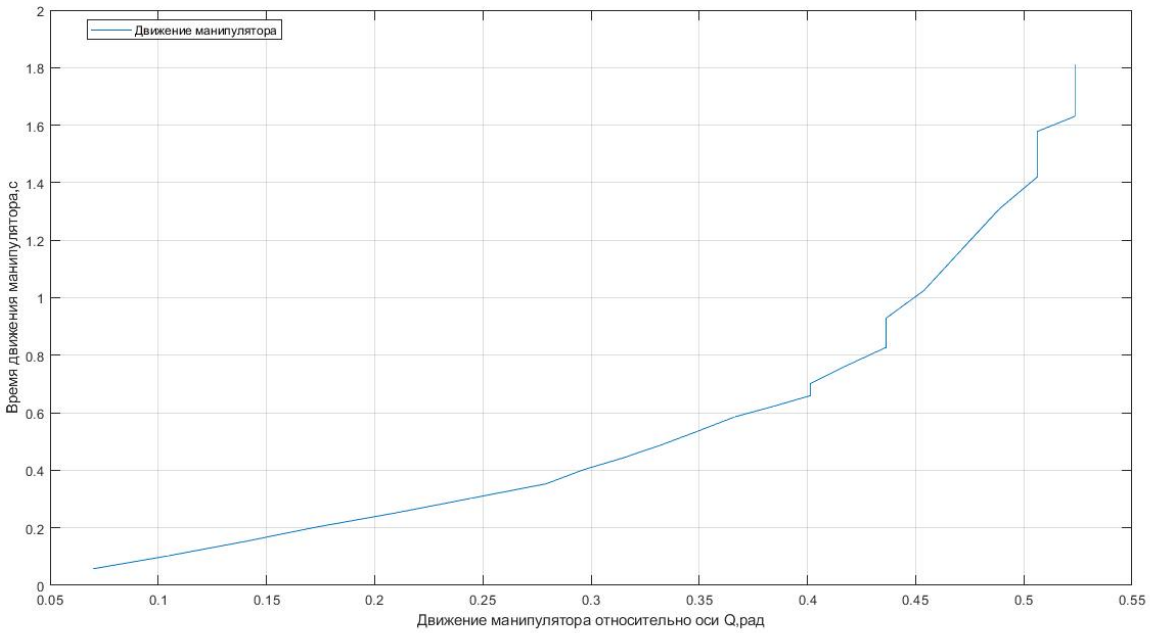


Рисунок 19. График зависимости угла поворота q1(t)

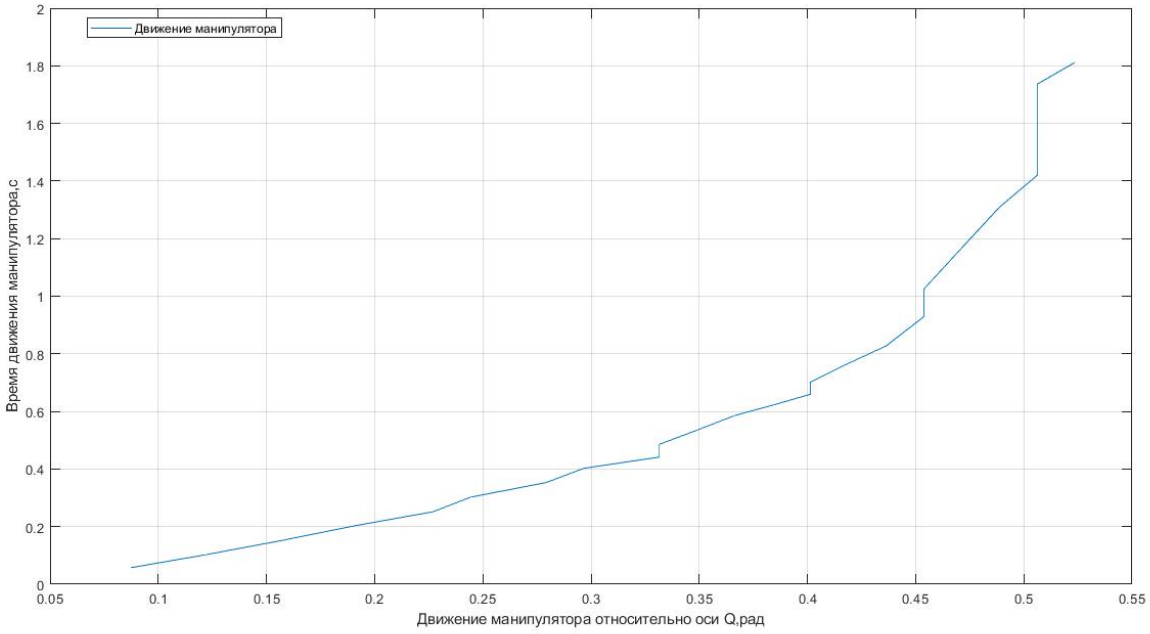


Рисунок 20. График зависимости угла поворота q2(t)

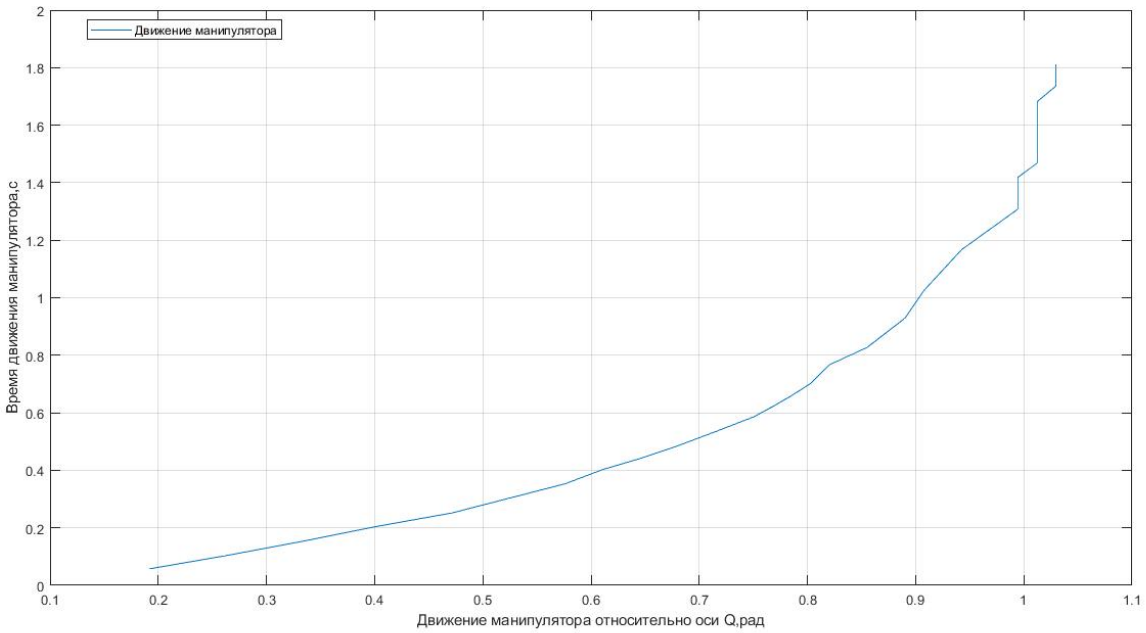


Рисунок 21. График зависимости угла поворота q3(t)

4.2 ОБРАТНАЯ ЗАДАЧА КИНЕМАТИКИ

1. Решение обратной задачи кинематики

```
X=0.11;
Y=0.06;
Z=0.32;
%DH-ПАРАМЕТРЫ
    % Расстояния вдоль оси xi(текущая ось) от zi-1 до zi
    A1=0;
    A2=0.14;
    A3=0.09;
    % Угол вращения вокруг оси xi (текущая ось) от zi-1 до zi
    a1=pi/2;
    a2=0;
    a3=0;
    % Расстояния вдоль оси zi-1 (предыдущая ось) от xi-1 до xi
    d1=0.16;
    d2=0;
r1=sqrt(X^2+Y^2);
r2=Z-d1;
r3=sqrt(r1^2+r2^2);
% Угол вращения вокруг оси zi-1 (предыдущая ось) от xi-1 до xi
Q1=(atan2(Y,X));
Q2= ((pi/2) - (acos((A2^2+r3^2-A3^2)/(2*A2*r3))+atan2(r2,r1)));
Q3=(pi-acos((A2^2+A3^2-r3^2)/(2*A2*A3)));
disp(Q1)
disp(Q2)
disp(Q3)
```

2. Результаты построений

Полученные углы поворота – 90,50,50. Теоретически движение манипулятора осуществляется в точки (0.05, 0.04, 0.26). Полученные практическим путем координаты манипулятора (0.1,0.07,0.27)

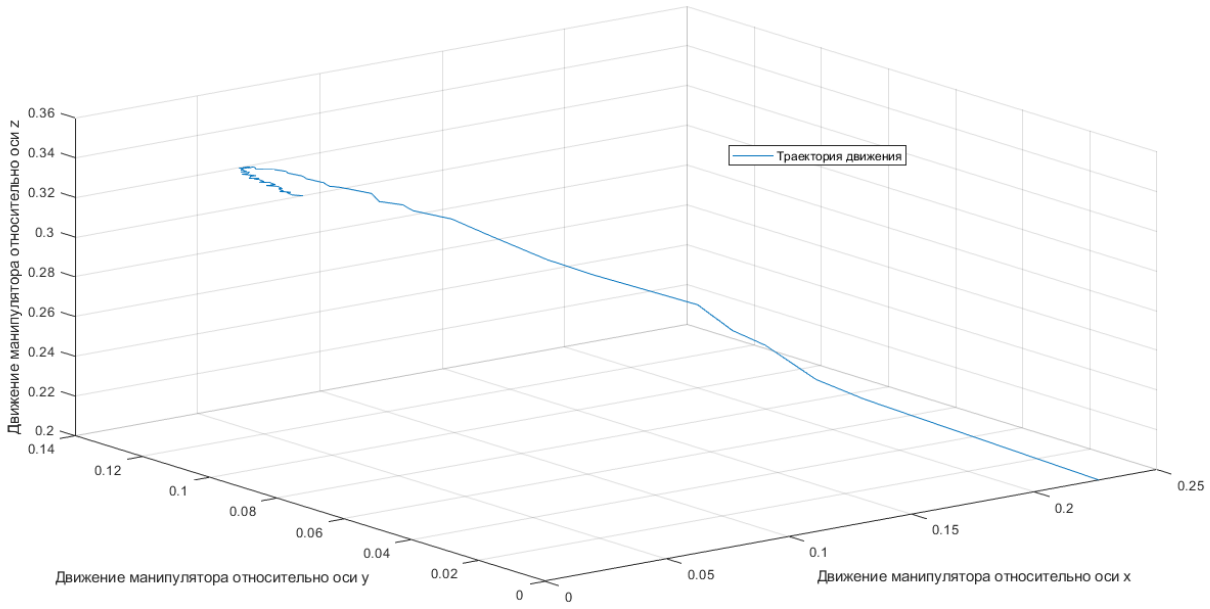


Рисунок 22. График траектории движения манипулятора

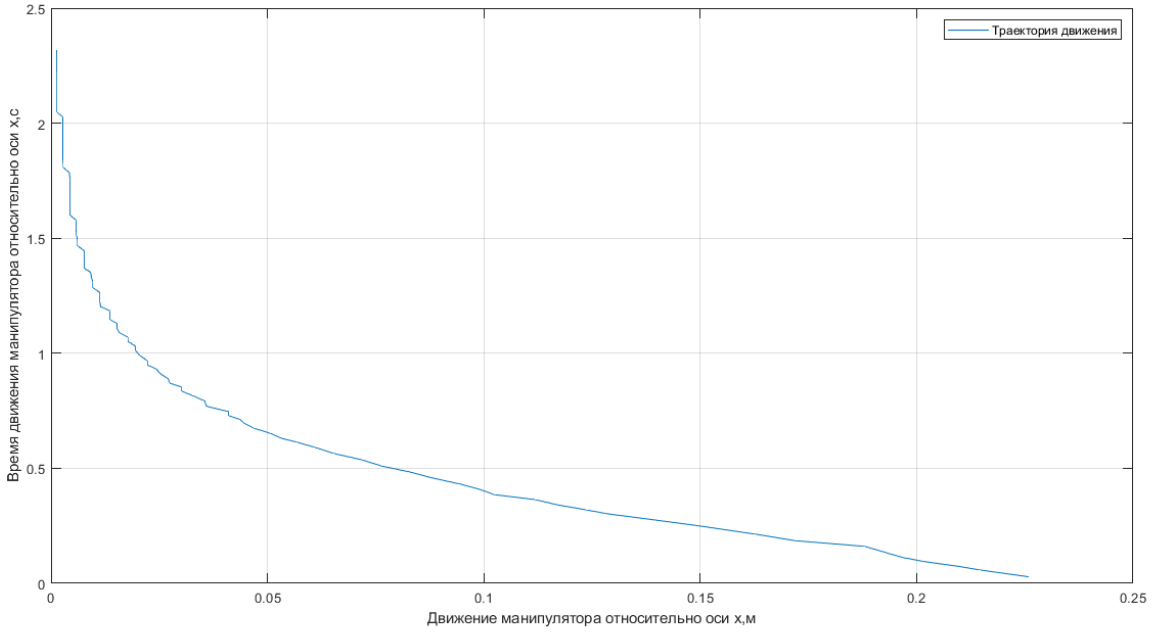


Рисунок 23. График зависимости координаты от времени x(t)

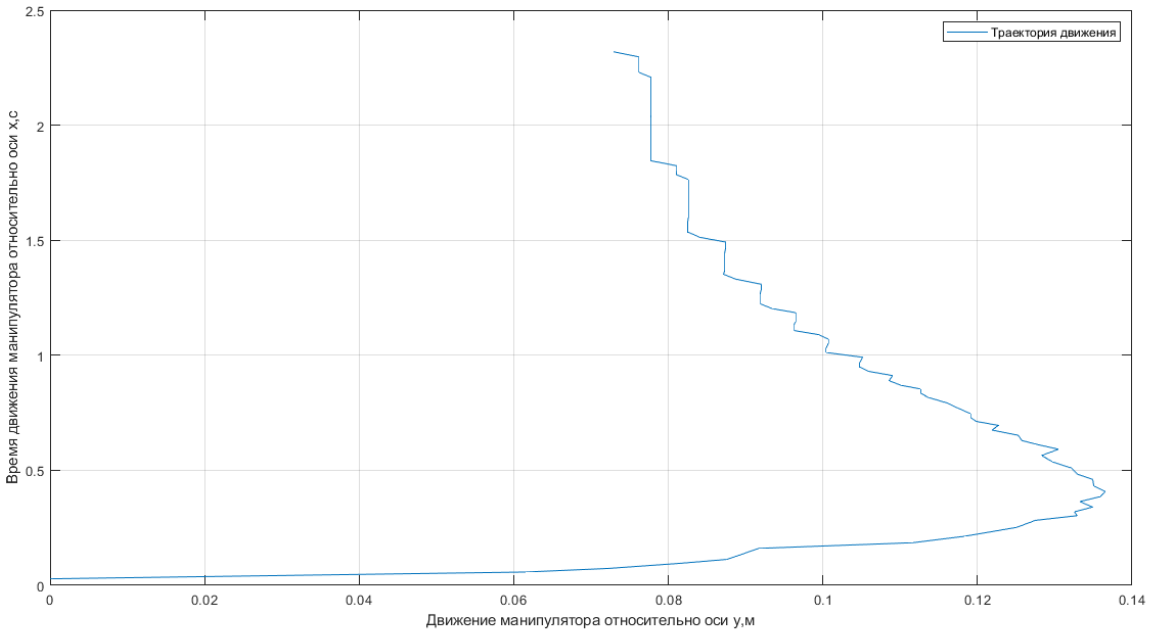


Рисунок 24. График зависимости координаты от времени y(t)

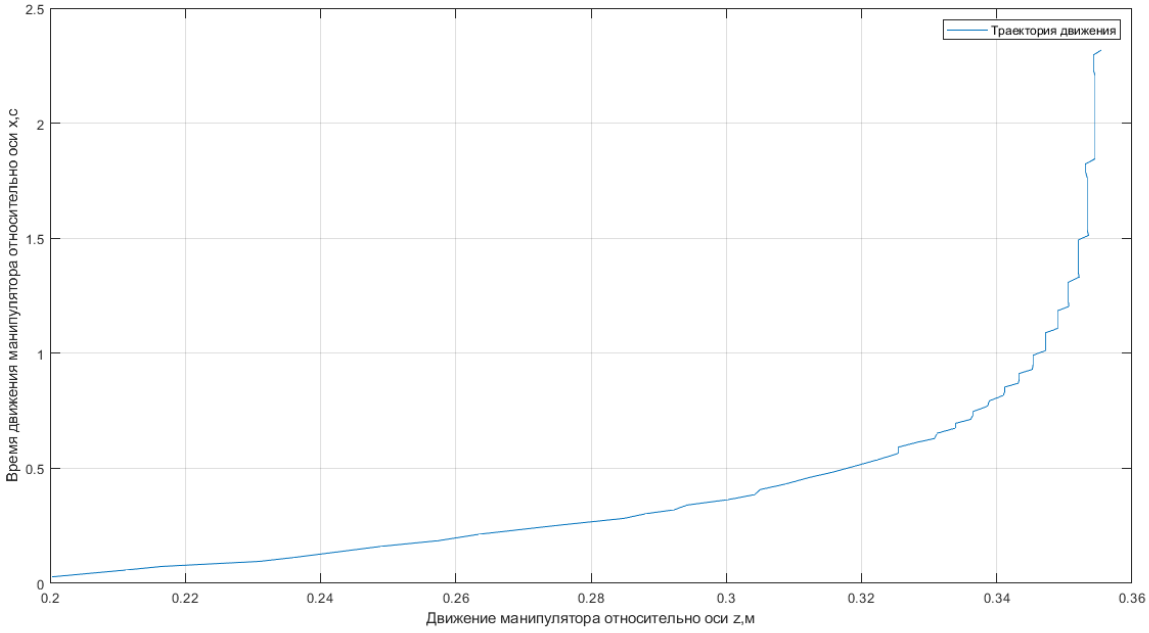


Рисунок 25. График зависимости координаты от времени z(t)

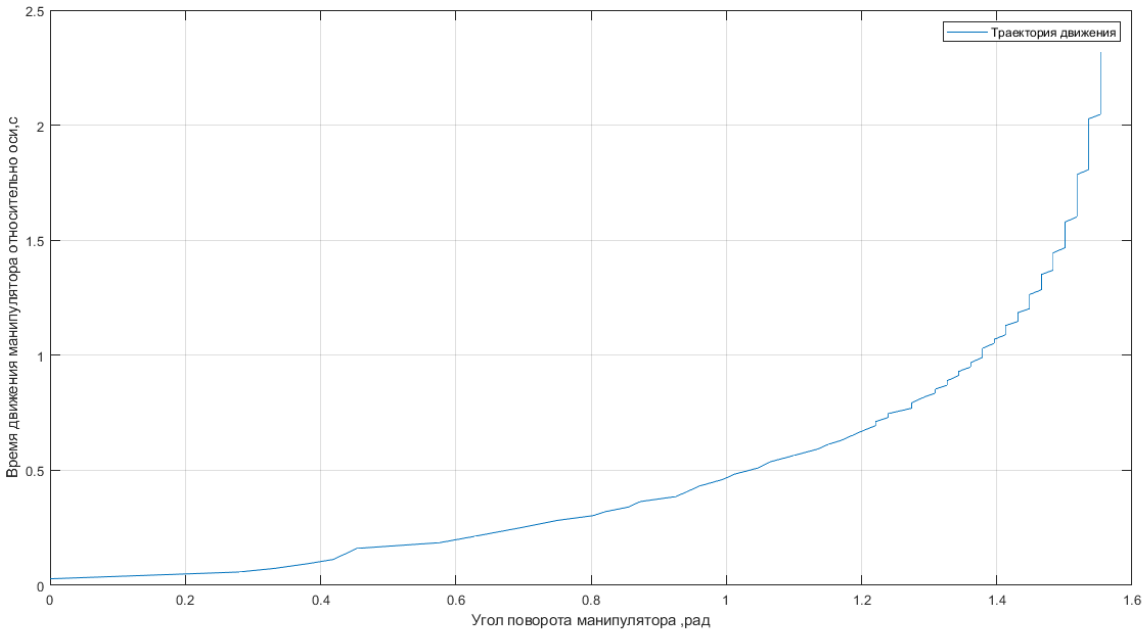


Рисунок 26. График зависимости угла поворота от времени q1(t)

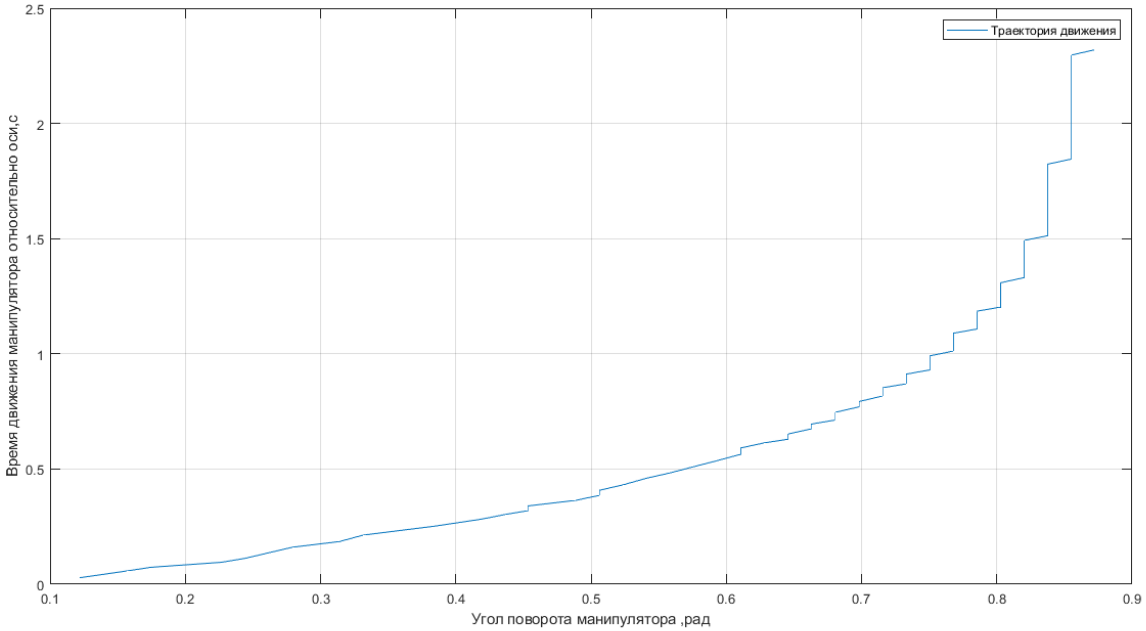


Рисунок 27. График зависимости угла поворота от времени q2(t)

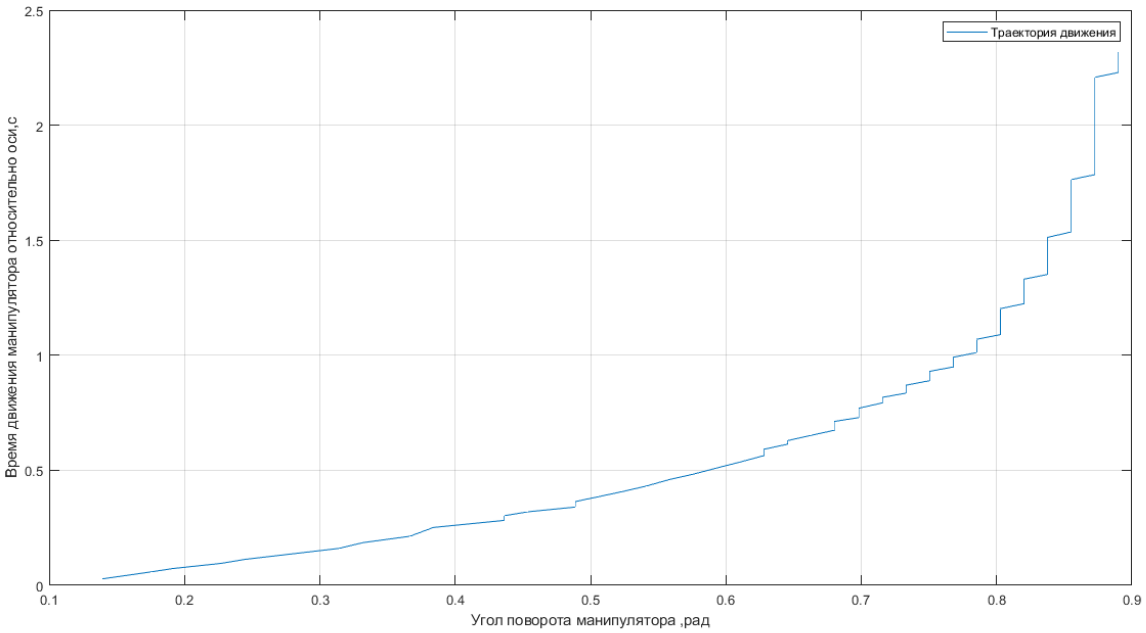


Рисунок 28. График зависимости угла поворота от времени q3(t)

Полученные углы поворота – 60,30,40. Теоретически движение манипулятора осуществляется в точки (0.07, 0.02, 0.17). Полученные практическим путем координаты манипулятора (0.1,0.04,0.21)

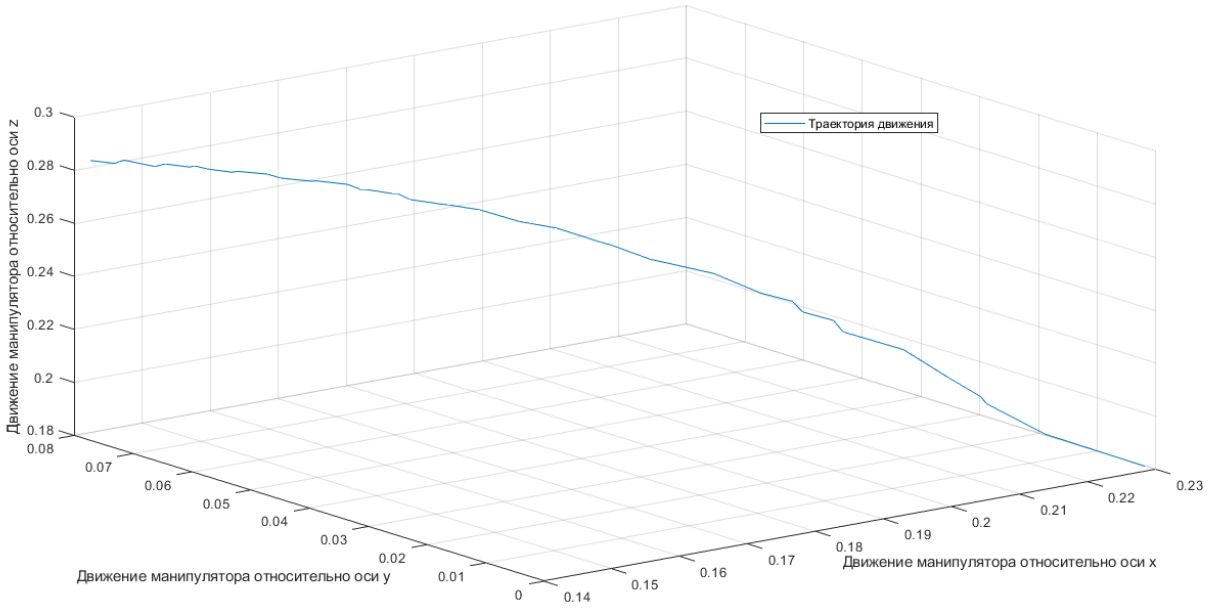


Рисунок 29. График траектории движения манипулятора

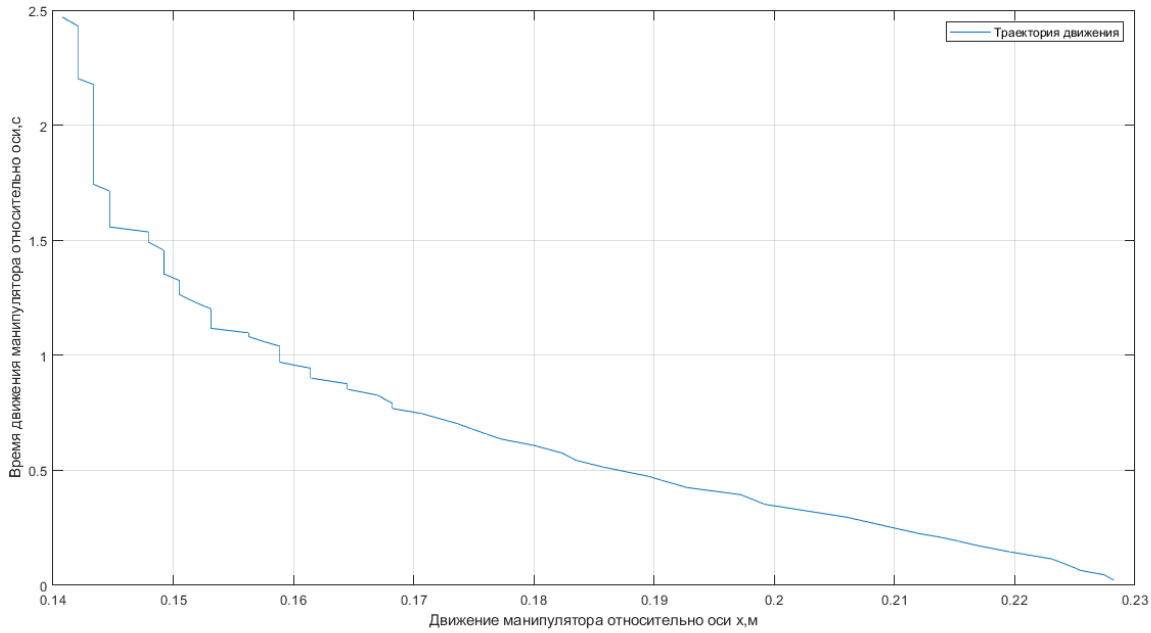


Рисунок 30. График зависимости координаты от времени x(t)

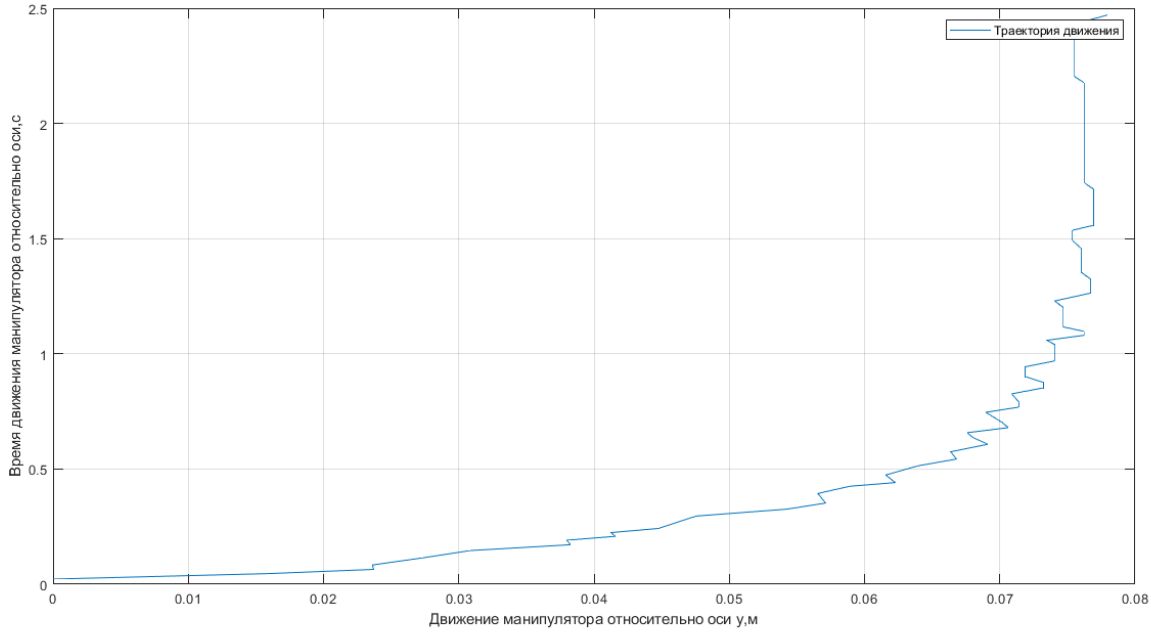


Рисунок 31. График зависимости координаты от времени y(t)

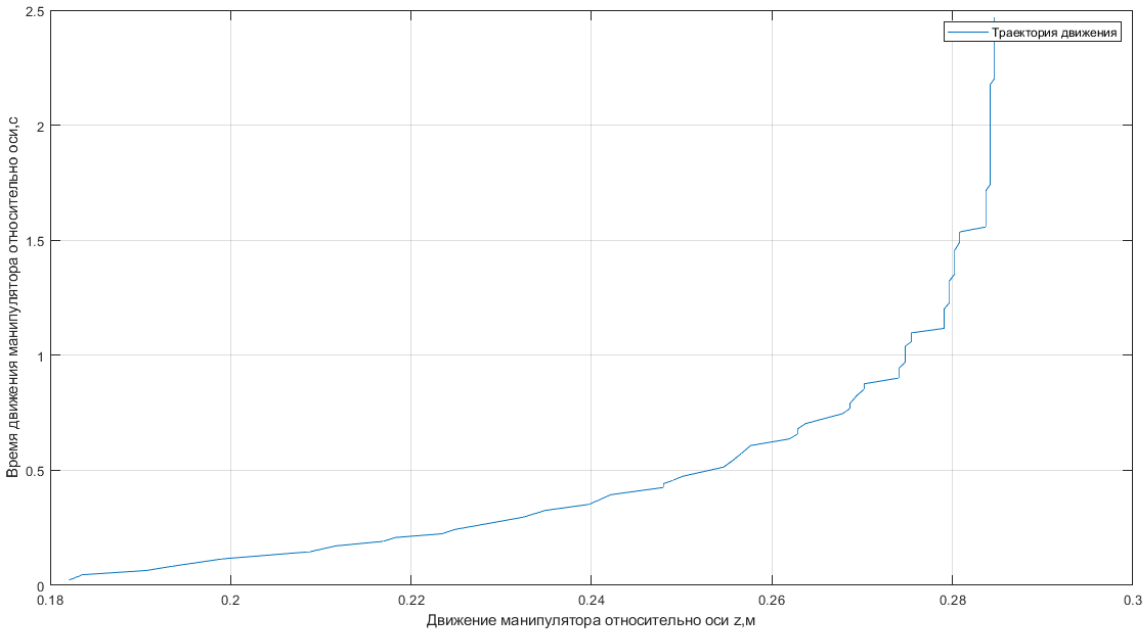


Рисунок 32. График зависимости координаты от времени z(t)

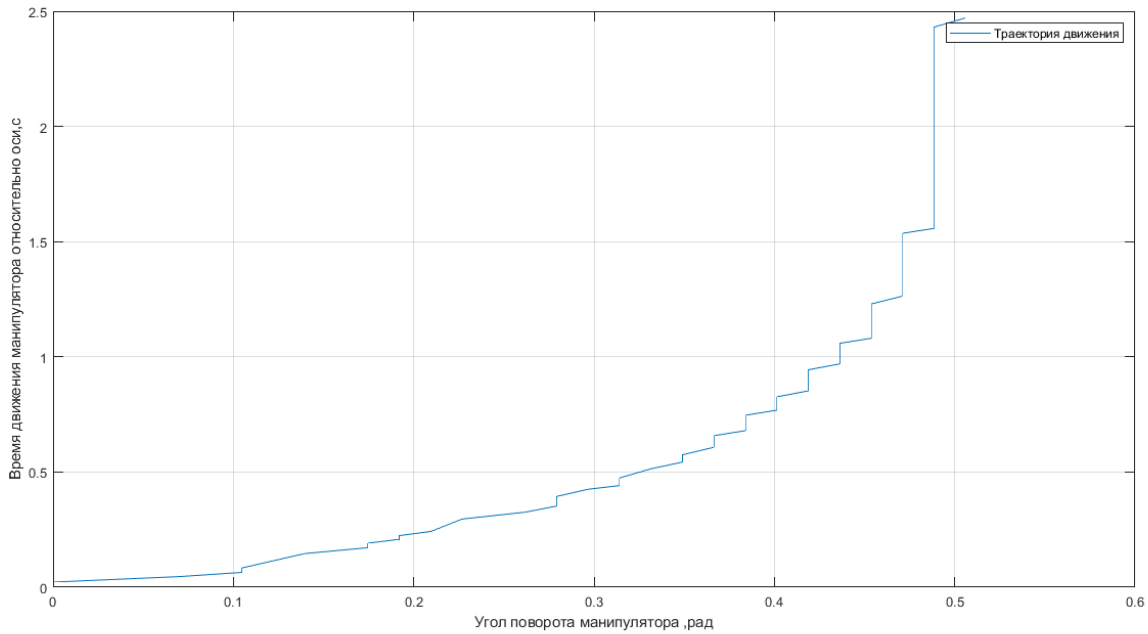


Рисунок 33. График зависимости угла поворота от времени q1(t)

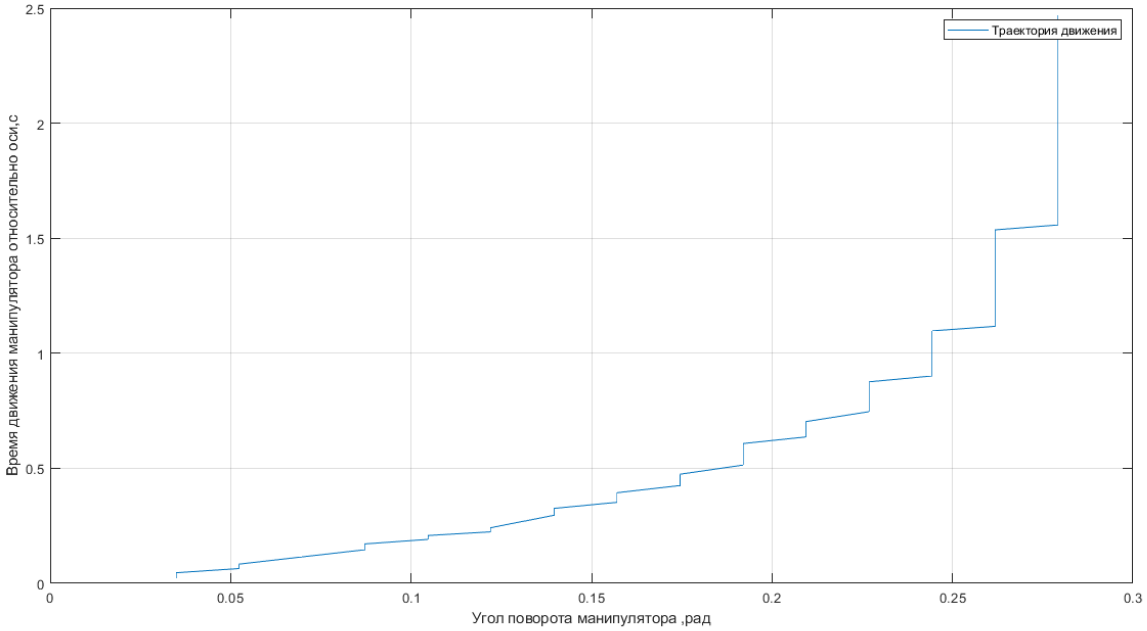


Рисунок 34. График зависимости угла поворота от времени q2(t)

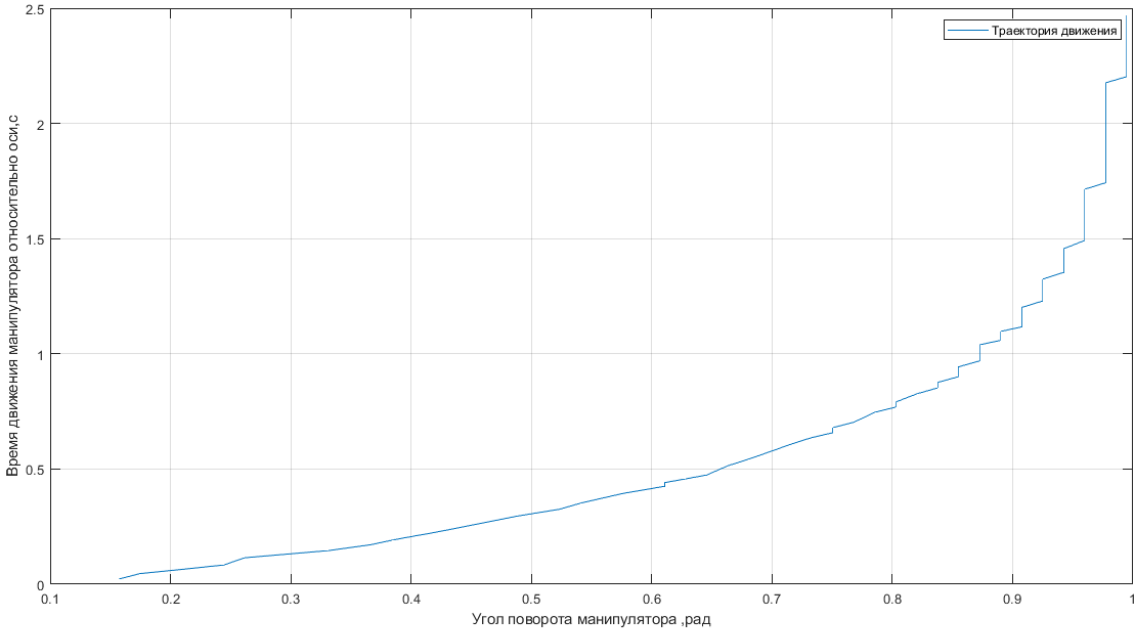


Рисунок 35. График зависимости угла поворота от времени q3(t)

Полученные углы поворота – 30,20,60. Теоретически движение манипулятора осуществляется в точки (0.02,0.07,0,25). Полученные практическим путем координаты манипулятора (0.04,0.07,0.19)

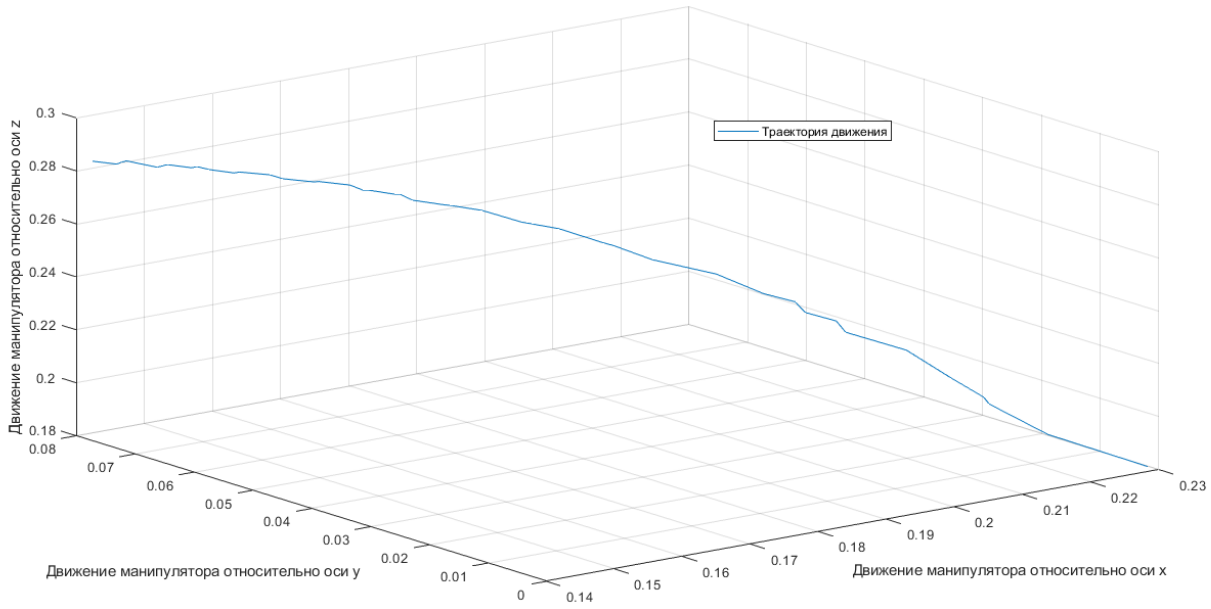


Рисунок 36. График траектории движения манипулятора в точку

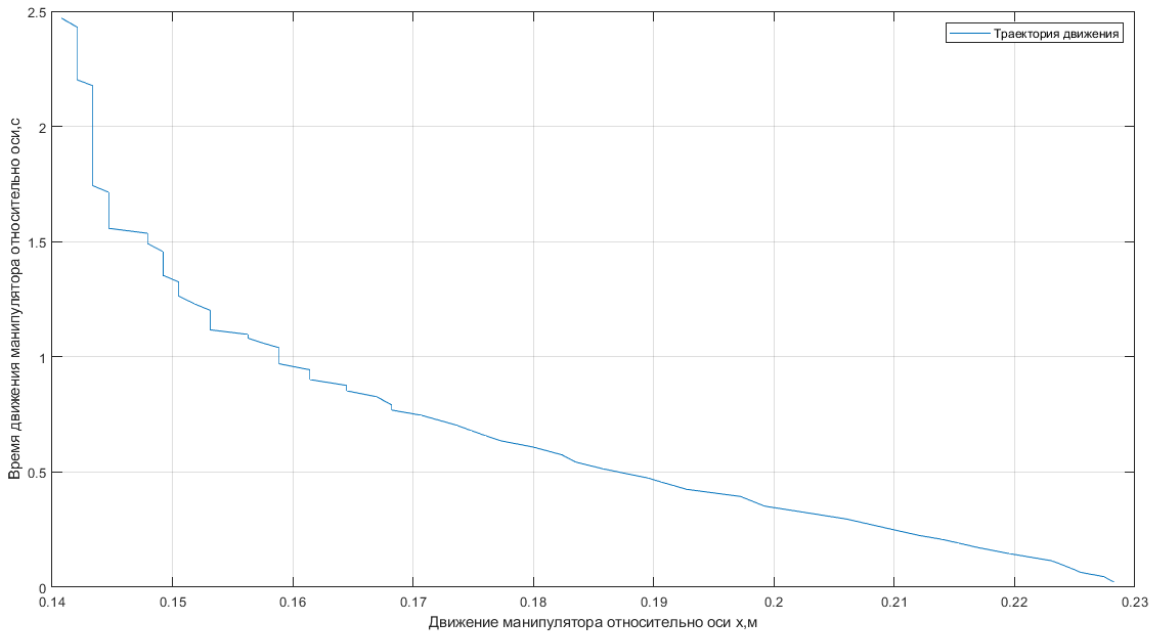


Рисунок 37. График зависимости координаты от времени x(t)

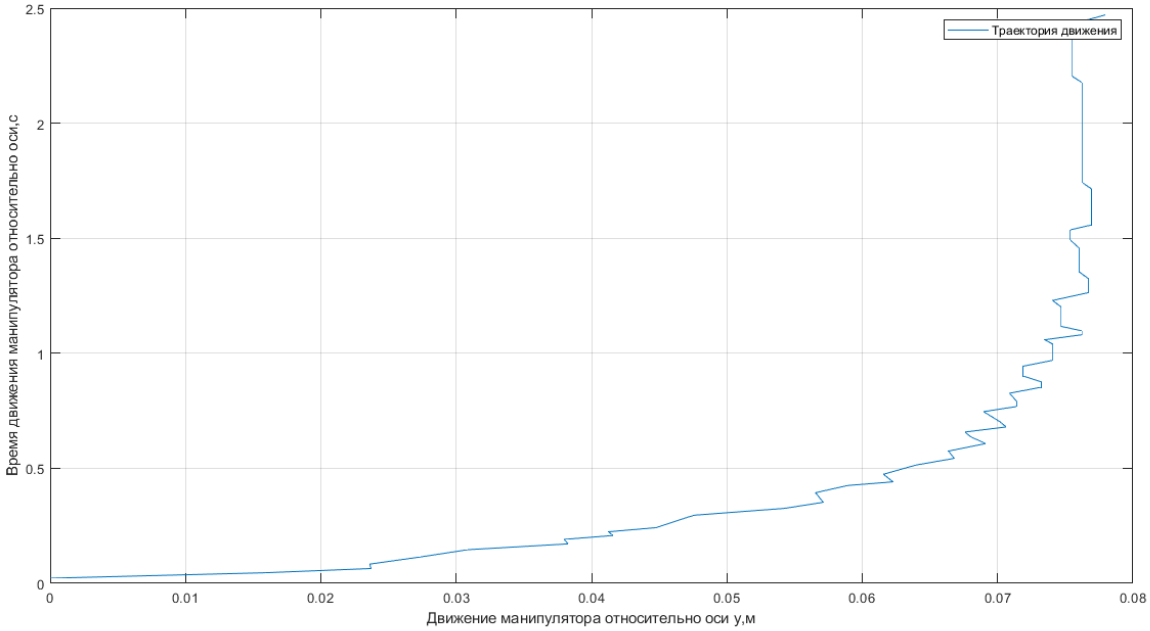


Рисунок 38. График зависимости координаты от времени y(t)

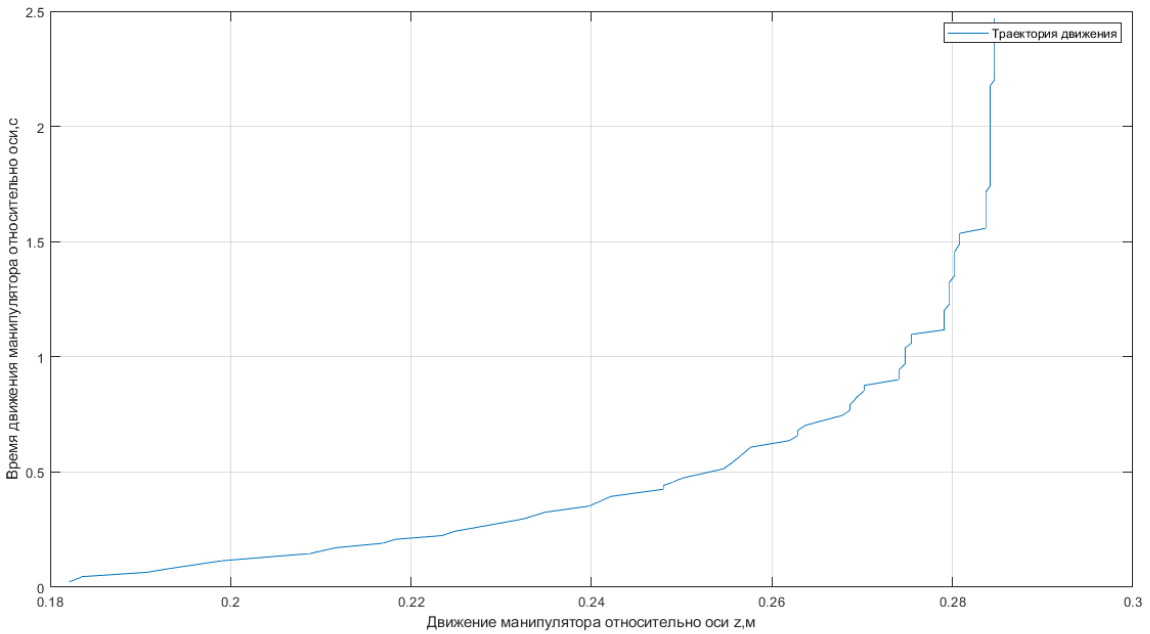


Рисунок 39. График зависимости координаты от времени z(t)

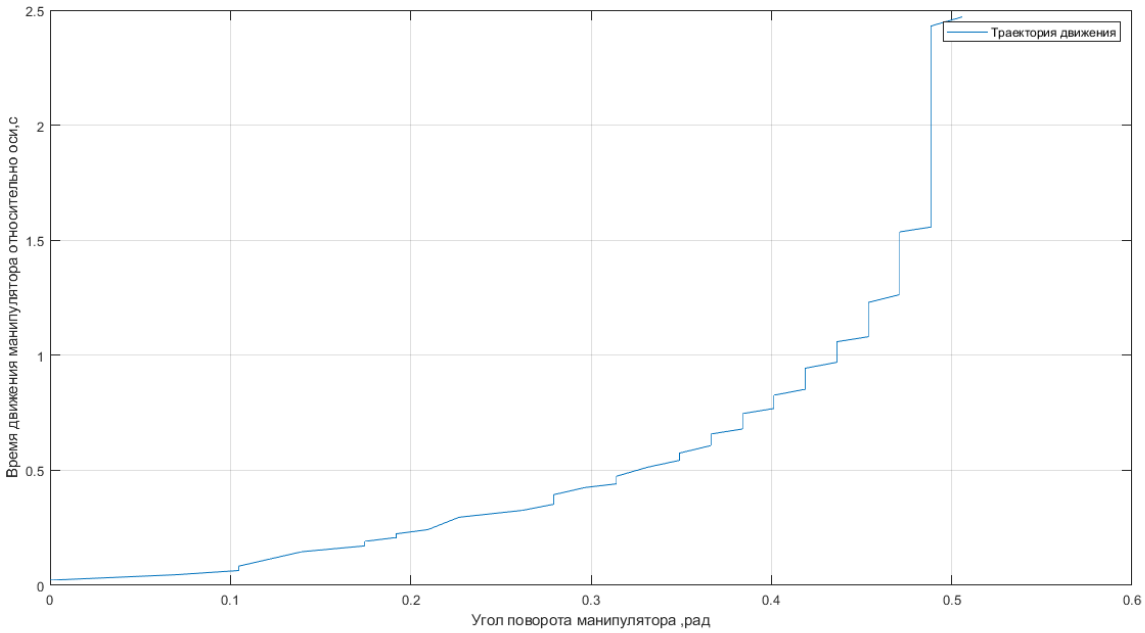


Рисунок 40. График зависимости угла поворота от времени q1(t)

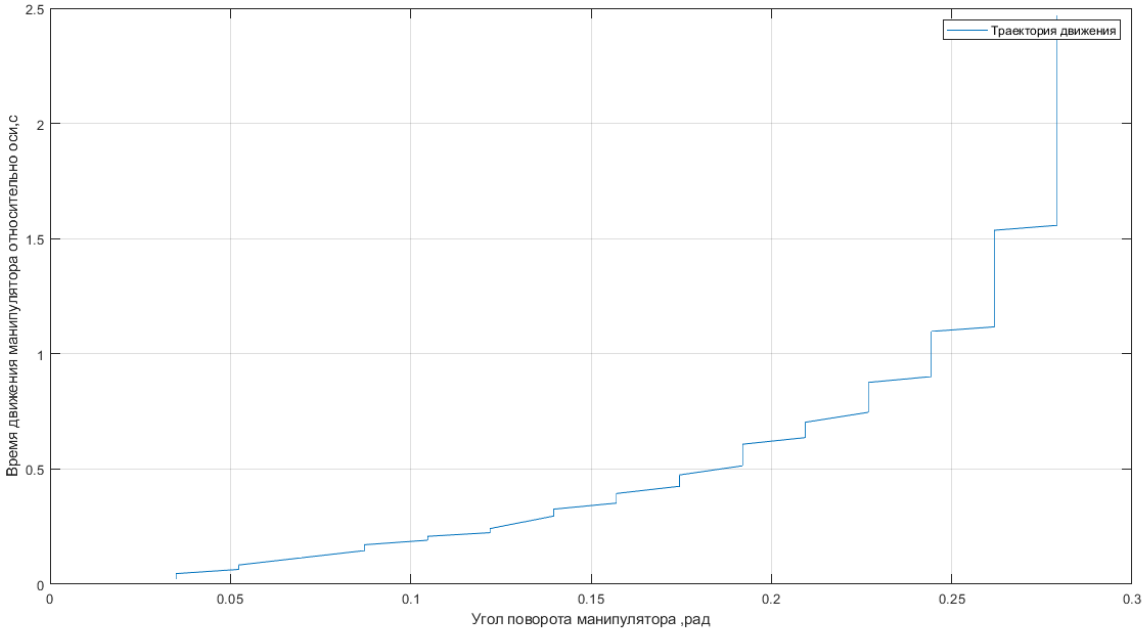


Рисунок 41. График зависимости угла поворота от времени q2(t)

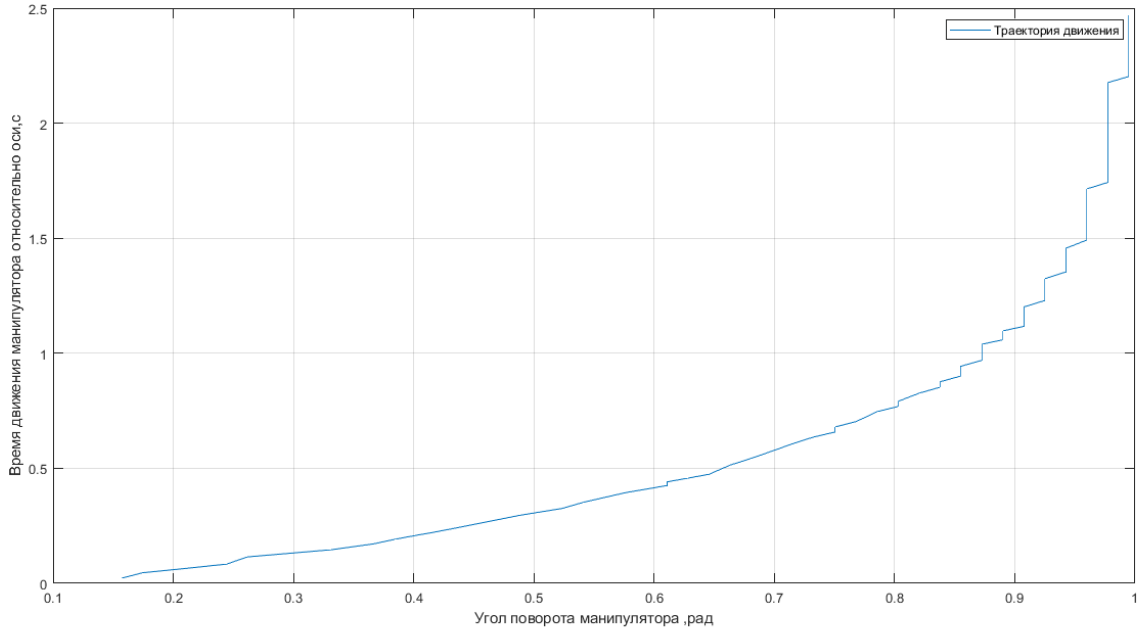


Рисунок 42. График зависимости угла поворота от времени q3(t)

5. Код Python

5.1 Код для решения прямой задачи кинематики

```
#!/usr/bin/env python3

import math
import time


from ev3dev2.motor import *

motorA = LargeMotor(OUTPUT_A)
motorB = LargeMotor(OUTPUT_B)
motorC = LargeMotor(OUTPUT_C)

kp1 = 10
kp2 = 10
kp3 = 10

d1 = 0.16  # meters
a2 = 0.14  # meters
a3 = 0.09  # meters

theta_1_cur, theta_2_cur, theta_3_cur = 0, 0, 0


def to_radians(degrees):
    return degrees / 180 * math.pi


def saturate(value, bounds=(-100, 100)):
    if value > bounds[1]:
        return bounds[1]
```

```
    if value < bounds[0]:
        return bounds[0]
    return value


def calculate_target_angles(x, y, z):
    global d1, a2, a3
    r_1 = (x ** 2 + y ** 2) ** 0.5
    r_2 = z - d1
    r_3 = (r_1 ** 2 + r_2 ** 2) ** 0.5


    t_1 = math.atan2(y, x)

    if (a2 ** 2 + r_3 ** 2 - a3 ** 2) / (2 * a2**2 * r_3**2) >= 1:
        t_2 = math.atan2(r_2, r_1)
    if (a2 ** 2 + r_3 ** 2 - a3 ** 2) / (2 * a2**2 * r_3**2) <= -1:
        t_2 = math.atan2(r2, r1) - pi
    if abs((a2 ** 2 + r_3 ** 2 - a3 ** 2) / (2 * a2**2 * r_3**2)) < 1:
        t_2 = (pi/2)-((math.acos((a2 ** 2 + r_3 ** 2 - a3 ** 2) / (2 * a2**2 * r_3**2) + math.atan2(r2, r1))))

    t_3 = math.pi - math.acos((a2 ** 2 + a3 ** 2 - r_3 ** 2) / (2 * a2 * a3))

    return t_1, t_2, t_3


def handle_reductor(motor, n1, n2):
    return motor.position / n2 * n1


def update_angles():
    global motorA, motorB, motorC, theta_1_cur, theta_2_cur, theta_3_cur

    theta_1_cur = to_radians(handle_reductor(motorA, 1, 1))
    theta_2_cur = to_radians(handle_reductor(motorB, 1, 1))
    theta_3_cur = to_radians(handle_reductor(motorC, 1, 1))


def calculate_errors(target_1, target_2, target_3):
    global theta_1_cur, theta_2_cur, theta_3_cur

    return target_1 - theta_1_cur, target_2 - theta_2_cur, target_3 - theta_3_cur


def move_to_target_angles(theta_1, theta_2, theta_3):
    global theta_1_cur, theta_2_cur, theta_3_cur, kp1, kp2, kp3

    errors = calculate_errors(theta_1, theta_2, theta_3)
    start = time.time()

    while sum(errors) > to_radians(1):
        e1, e2, e3 = errors

        motorA.on(saturate(e1 * kp1))
        motorB.on(saturate(e2 * kp2))
```

```
        motorC.on(saturate(e3 * kp3))

        update_angles()
        errors = calculate_errors(theta_1, theta_2, theta_3)
        print(theta_1_cur, theta_2_cur, theta_3_cur, time.time() - start)


def move_to_coordinates(x, y, z):
    target_1, target_2, target_3 = calculate_target_angles(x, y, z)

    move_to_target_angles(target_1, target_2, target_3)


move_to_coordinates(x, y, z)
```

5.2 Код для решения обратной задачи кинематики

```
#!/usr/bin/env python3

import math
import time


from ev3dev2.motor import *

motorA = LargeMotor(OUTPUT_A)
motorB = LargeMotor(OUTPUT_B)
motorC = LargeMotor(OUTPUT_C)

kp1 = 10
kp2 = 10
kp3 = 10

d1 = 0.16  # meters
a2 = 0.14  # meters
a3 = 0.09  # meters

theta_1_cur, theta_2_cur, theta_3_cur = 0, 0, 0


def to_radians(degrees):
    return degrees / 180 * math.pi


def saturate(value, bounds=(-100, 100)):
    if value > bounds[1]:
        return bounds[1]
    if value < bounds[0]:
        return bounds[0]
    return value


def calculate_target_angles(x, y, z):
    global d1, a2, a3
    r_1 = (x ** 2 + y ** 2) ** 0.5
    r_2 = z - d1
```

```
    r_3 = (r_1 ** 2 + r_2 ** 2) ** 0.5


    t_1 = math.atan2(y, x)

    if (a2 ** 2 + r_3 ** 2 - a3 ** 2) / (2 * a2**2 * r_3**2) >= 1:
        t_2 = math.atan2(r_2, r_1)
    if (a2 ** 2 + r_3 ** 2 - a3 ** 2) / (2 * a2**2 * r_3**2) <= -1:
        t_2 = math.atan2(r2, r1) - pi
    if abs((a2 ** 2 + r_3 ** 2 - a3 ** 2) / (2 * a2**2 * r_3**2)) < 1:
        t_2 = (pi/2)-((math.acos((a2 ** 2 + r_3 ** 2 - a3 ** 2) / (2 * a2**2 * r_3**2) + math.atan2(r2, r1))))

    t_3 = math.pi - math.acos((a2 ** 2 + a3 ** 2 - r_3 ** 2) / (2 * a2 * a3))

    return t_1, t_2, t_3


def handle_reductor(motor, n1, n2):
    return motor.position / n2 * n1


def update_angles():
    global motorA, motorB, motorC, theta_1_cur, theta_2_cur, theta_3_cur

    theta_1_cur = to_radians(handle_reductor(motorA, 1, 1))
    theta_2_cur = to_radians(handle_reductor(motorB, 1, 1))
    theta_3_cur = to_radians(handle_reductor(motorC, 1, 1))


def calculate_errors(target_1, target_2, target_3):
    global theta_1_cur, theta_2_cur, theta_3_cur

    return target_1 - theta_1_cur, target_2 - theta_2_cur, target_3 - theta_3_cur


def move_to_target_angles(theta_1, theta_2, theta_3):
    global theta_1_cur, theta_2_cur, theta_3_cur, kp1, kp2, kp3

    errors = calculate_errors(theta_1, theta_2, theta_3)
    start = time.time()

    while sum(errors) > to_radians(1):
        e1, e2, e3 = errors

        motorA.on(saturate(e1 * kp1))
        motorB.on(saturate(e2 * kp2))
        motorC.on(saturate(e3 * kp3))

        update_angles()
        errors = calculate_errors(theta_1, theta_2, theta_3)
        print(theta_1_cur, theta_2_cur, theta_3_cur, time.time() - start)


def move_to_coordinates(x, y, z):
    target_1, target_2, target_3 = calculate_target_angles(x, y, z)
```

move_to_target_angles(target_1, target_2, target_3)

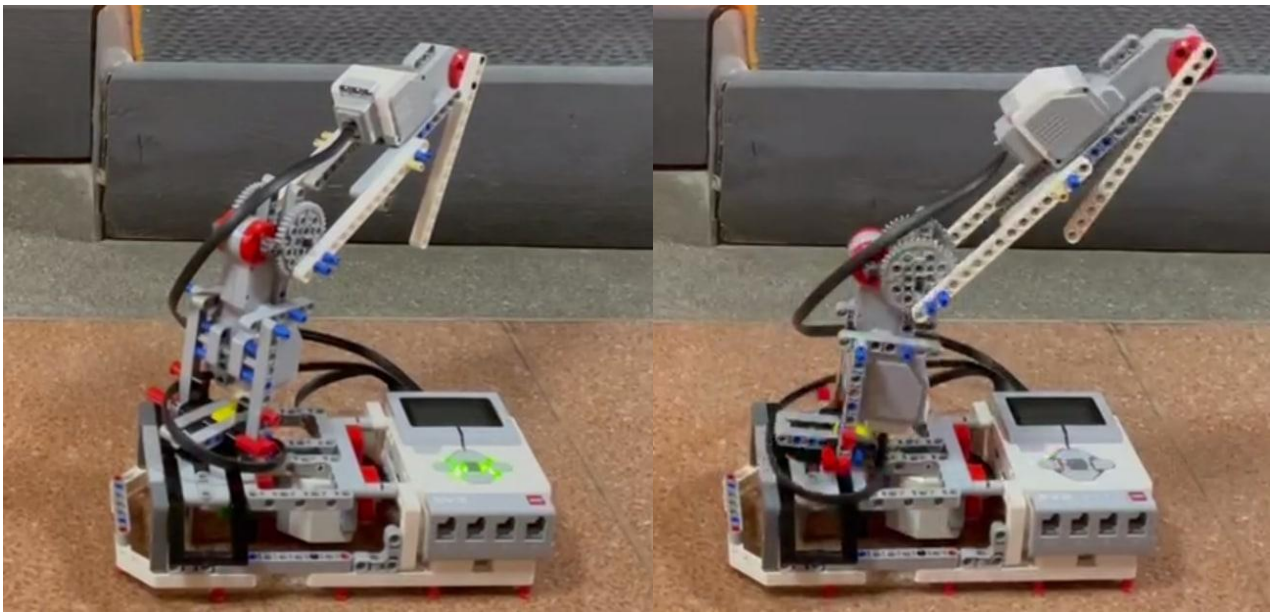
move_to_target_angles(a1, a2, a3)

6. Код Matlab

```
% Открытие и считывание файла
clear all
close all
File =
fopen('C:\Users\79623\OneDrive\Документы\lab6\inverse.30.30.60\inverse.30.30.60.txt','r');
w = fscanf(File, '%f %f %f' , [4 Inf]);
W = transpose(w);

fclose(File);

for i=1:size(W,1)
    q1(i)=W(i,1);
    q2(i)=(W(i,2));
    q3(i)=(W(i,3));
    t(i)=W(i,4);
    %DH-ПАРАМЕТРЫ
    % Расстояния вдоль оси xi(текущая ось) от zi-1 до zi
    A1=0;
    A2=0.14;
    A3=0.09;
    % Угол вращения вокруг оси xi (текущая ось) от zi-1 до zi
    a1=pi/2;
    a2=0;
    a3=0;
    % Расстояния вдоль оси zi-1 (предыдущая ось) от xi-1 до xi
    d1=0.16;
    d2=0;
    d3=0;
    % Угол вращения вокруг оси zi-1 (предыдущая ось) от xi-1 до xi
    Q1=60;
    Q2=30;
    Q3=40;

    T01=[cos(Q1) -cos(a1)*sin(Q1) sin(a1)*sin(Q1) A1*cos(Q1);
         sin(Q1) cos(a1)*cos(Q1) -sin(a1)*cos(Q1) A1*sin(Q1);
         0 sin(a1) cos(a1) d1;
         0 0 0 1];

    T02=[cos(Q2) -cos(a2)*sin(Q2) sin(a2)*sin(Q2) A2*cos(Q2);
         sin(Q2) cos(a2)*cos(Q2) -sin(a2)*cos(Q2) A2*sin(Q2);
         0 sin(a2) cos(a2) d2;
         0 0 0 1];

    T03=[cos(Q3) -cos(a3)*sin(Q3) sin(A3)*sin(Q3) A3*cos(Q3);
         sin(Q3) cos(a3)*cos(Q3) -sin(A3)*cos(Q3) A3*sin(Q3);
         0 sin(a3) cos(a3) d3;
         0 0 0 1];

    T=T01*T02*T03;


end
%plot3(X,Y,Z)
xlabel("Движение манипулятора относительно оси x")
ylabel("Движение манипулятора относительно оси y")
zlabel("Движение манипулятора относительно оси z")
```

```
%legend(["Траектория движения"])

%plot(q3,t)
%xlabel("Угол поворота манипулятора ,рад")
%ylabel("Время движения манипулятора относительно оси,с")
legend(["Траектория движения"])
grid on
```

7. Фотографии схвата манипулятора для прямой и обратной задачи кинематики



8. Вывод

В данной лабораторной мы успешно достигли поставленной цели: схват манипулятора перемещается в нужные координаты, вне зависимости от пространства, в котором мы их задаём. Метод Денавита-Хартенберга работает, однако накапливается небольшая ошибка, связанная с неточностью компьютерных вычислений, но она настолько мала, что не влияет на перемещение манипулятора.

9. Дополнительное задание.

Реализация поиска предмета по цвету.

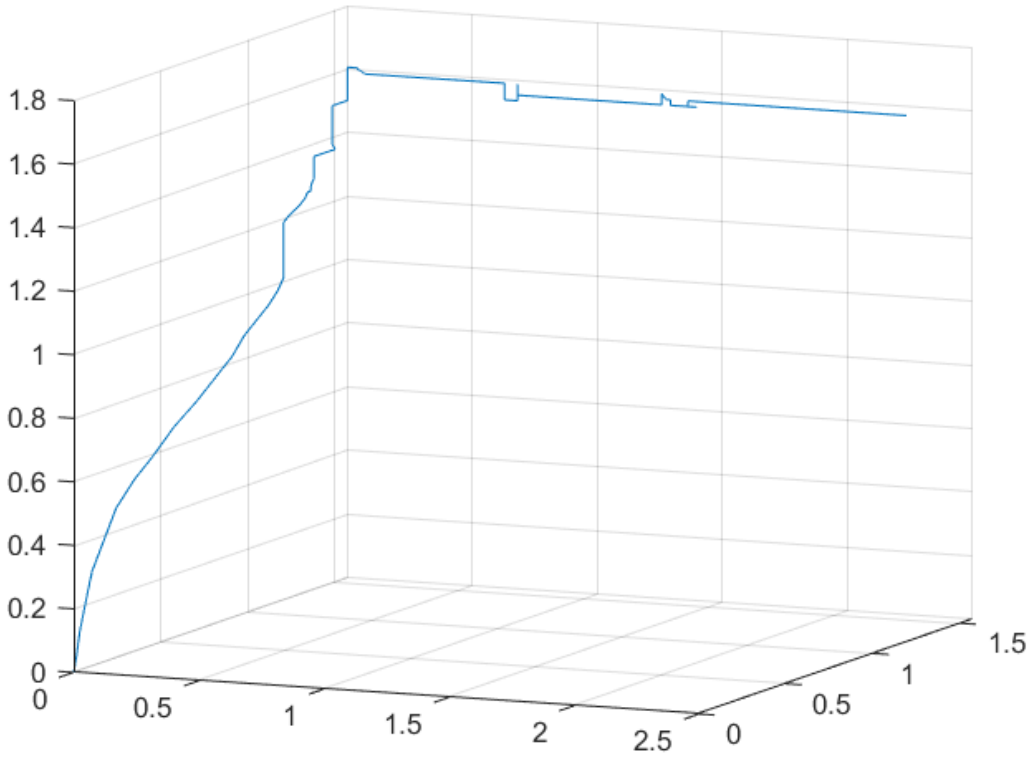
Манипулятор опускается на заданный угол и начинает искать предмет по цвету с помощью сканера цвета. Если сканер не определяет нужный цвет, то робот поворачивается ещё на определенный угол и снова считывает цвет объекта. Таким образом манипулятор будет поворачиваться до тех пор, пока не найдет заданный цвет.

10. Код Python для дополнительного задания

```
#!/usr/bin/env python3
import math
import time
from ev3dev2.motor import *
import ev3dev.ev3 as ev3
motorA = LargeMotor(OUTPUT_A)
motorB = LargeMotor(OUTPUT_B)
motorC = LargeMotor(OUTPUT_C)
motorA.position = 0
motorB.position = 0
motorC.position = 0
kp1 = 10
kp2 = 10
kp3 = 10
d1 = 0.17  # meters
a2 = 0.125  # meters
a3 = 0.05  # meters
theta_1_cur, theta_2_cur, theta_3_cur = 0, 0, 0
def to_radians(degrees):
    return degrees / 180 * math.pi
def saturate(value, bounds=(-100, 100)):
    if value > bounds[1]:
        return bounds[1]
    if value < bounds[0]:
        return bounds[0]
    return value
def calculate_target_angles(x, y, z):
    return math.pi/2, math.pi/4, math.pi/2
    global d1, a2, a3
    r_1 = (x ** 2 + y ** 2) ** 0.5
    r_2 = z - d1
    r_3 = (r_1 ** 2 + r_2 ** 2) ** 0.5
    t_1 = math.atan2(y, x)
    if (a2 ** 2 + r_3 ** 2 - a3 ** 2) / (2 * a2**2 * a3**2) >= 1:
        t_2 = math.atan2(r_2, r_1)
    if (a2 ** 2 + r_3 ** 2 - a3 ** 2) / (2 * a2**2 * a3**2) <= -1:
        t_2 = math.atan2(r_2, r_1) - math.pi
    if abs((a2 ** 2 + r_3 ** 2 - a3 ** 2) / (2 * a2**2 * a3**2)) < 1:
        t_2 = (math.pi/2)-((math.acos((a2 ** 2 + r_3 ** 2 - a3 ** 2) / (2 * a2**2 * a3**2) +
math.atan2(r_2,   r_1))))
    t_3 = math.pi - math.acos((a2 ** 2 + a3 ** 2 - r_3 ** 2) / (2 * a2**2 * a3**2))
    return t_1, t_2, t_3
def handle_reductor(motor, n1, n2, reverse=1):
    return motor.position / n2 * n1 * reverse
def update_angles():
    global motorA, motorB, motorC, theta_1_cur, theta_2_cur, theta_3_cur
    theta_1_cur = to_radians(handle_reductor(motorA, 1, 1))
    theta_2_cur = to_radians(handle_reductor(motorB, 1, 1))
    theta_3_cur = to_radians(handle_reductor(motorC, 1, 1))
def calculate_errors(target_1, target_2, target_3):
    global theta_1_cur, theta_2_cur, theta_3_cur
    return target_1 - theta_1_cur, target_2 - theta_2_cur, target_3 - theta_3_cur
def move_to_target_angles(theta_1, theta_2, theta_3):
```

```
    global theta_1_cur, theta_2_cur, theta_3_cur, kp1, kp2, kp3
    file = open("data_dop.txt", "w")
    errors = calculate_errors(theta_1, theta_2, theta_3)
    start = time.time()
    while time.time() - start < 5.0:
        e1, e2, e3 = errors
        motorA.on(saturate(e1 * kp1))
        motorB.on(saturate(e2 * kp2))
        motorC.on(saturate(e3 * kp3))
        update_angles()
        errors = calculate_errors(theta_1, theta_2, theta_3)
        file.write(str(theta_1_cur) + " " + str(theta_2_cur) + " " + str(theta_3_cur) + "\n")
        print(theta_1_cur, theta_2_cur, theta_3_cur)
def move_to_coordinates(x, y, z):
    target_1, target_2, target_3 = 0, math.pi/2, math.pi/2
    move_to_target_angles(target_1, target_2, target_3)
move_to_coordinates(0, 0.1, 0.1)
colorSensor = ev3.ColorSensor('in1')
cur_angle = 0
step = 20
start = time.time()
motorB.run_direct(duty_cycle_sp=0)
while True:
    if time.time() - start > 10:
        cur_angle += step
        motorA.on_for_degrees(degrees=step, speed=20)
        if abs(cur_angle) >= 180:
            step=-step
        start = time.time()
    time.sleep(2.0)
    currentColor = colorSensor.color
    if currentColor == ev3.ColorSensor.COLOR_BLUE:
        ev3.Sound.speak("Color found")
        time.sleep(1.0)
        break
```

11. Графики для дополнительного задания



12. Видеоотчёт по дополнительному заданию

https://drive.google.com/drive/folders/15JZriBNB8MtD_RfiKwFyd1zP9KD8_AUL