Содержание

[АННОТАЦИЯ 5](#_Toc532849706)

[ВВЕДЕНИЕ 6](#_Toc532849707)

[1. ПОСТАНОВКА ЗАДАЧИ 10](#_Toc532849708)

[3. МАТЕМАТИЧЕСКОЕ РЕШЕНИЕ ЗАДАЧИ 12](#_Toc532849709)

[4. МОДЕЛИРУЮЩАЯ ПРОГРАММА 18](#_Toc532849710)

[4.1. Описание программы 18](#_Toc532849711)

[4.2. Текст моделирующей программы 20](#_Toc532849712)

[5. РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ПРОГРАММЫ 27](#_Toc532849713)

[ЛИТЕРАТУРА 30](#_Toc532849714)

# АННОТАЦИЯ

В данной работе отображены вопросы планирования траектории для плоскостного трехзвенного манипулятора, в частности планирование траектории в обобщенных координатах методом линейной интерполяции. Результатом выполнения работы является построение графиков положения, скорости и ускорения звеньев манипулятора в обобщенной системе координат и визуализация движения робота в декартовой системе координат.

# ВВЕДЕНИЕ

Опираясь на законы кинематики и динамики манипулятора, можно так управлять приводами сочленений, чтобы манипулятор двигался вдоль некоторой траектории, обеспечивающей выполнение поставленной задачи. Перед началом движения манипулятора важно знать: во-первых, существуют на его пути какие-то препятствия, и, во-вторых, накладываются ли какие-то ограничения на характер его траектории.

Кривую, вдоль которой схват манипулятора движется из начальной точки в конечную, называют его траекторией. Задача состоит в разработке математического аппарата для выбора и описания желаемого движения манипулятора между начальной и конечной точками траектории. Как правило, траектория, соединяющая начальное и конечное положение схвата, не единственна. Возможно, например, движение схвата вдоль прямой, соединяющей начальную и конечную точки (прямолинейная траектория), а также вдоль некоторой гладкой кривой, удовлетворяющей ряду ограничений на положение и ориентацию схвата на начальном, конечном и промежуточных участках траектории (сглаженная траектория). При этом не смотря на то, что виды траектории будут различными между опорными точками, тем не менее в опорных точках положения будут одинаковыми, равными заданным значениям.

Кроме того, что могут различаться виды траекторий, способы получения траекторий также могут быть различными. Различают способы планирования в декартовых координатах (ДК) и обобщенных координатах (ОК).

Первый состоит в том, что исследователь задает точный набор ограничений для звеньев (заданные значения положения, скорости, ускорения и непрерывность их) в некоторых узловых точках траектории. Планировщик траектории выбирает функцию, проходящую через узловые точки и удовлетворяющую заданным ограничениям, таким образом формирую все промежуточные положения звеньев. После этого решается прямая задача кинематики( ПЗК), т.е. по полученным значениям положений сочленений (звеньев) определяется положение схвата относительно базовой системы координат во всех точках траектории в ДК.

Второй подход состоит в том, что исследователь задает желаемую траекторию манипулятора в виде некоторой аналитически описываемой функции в декартовых координатах. Планировщик после этого решает обратную задачу кинематики (ОЗК), т.е. по заданному положение схвата относительно базовой системы координат определяются положения сочленений (звеньев) в ОК, используемые для управления приводами движения.

Известно, что отработка траектории манипулятора производится в пространстве обобщенных координат. Поэтому, если планирование траектории осуществляется в декартовых координатах, то для ее отработки необходимо решать обратную задачу кинематики о положении манипулятора. Если такое преобразование необходимо выполнять в реальном масштабе времени, то оно является серьезной вычислительной нагрузкой для системы управления. Кроме того законы движения звеньев манипулятора будут далеки от идеальных, возможны обрывы и скачки траектории в ОК. Преимуществом планирования траектории в пространстве декартовых координат является наглядность и предсказуемость траектории движения схвата манипулятора, так как непосредственно планируется траектория, отрабатываемая схватом.

Преимуществом планирования траектории в пространстве обобщенных координат является:

* планирование непосредственно траектории, которую должны отрабатывать приводы звеньев манипулятора;
* планирование траектории требует небольшого количества вычислений;
* траекторию легче планировать, так как отсутствует понятие ориентации.

Недостатком планирования в ОК является то, что итоговая траектория, получаемая после решения ПЗК, является непредсказуемой и может сильно отличатся от желаемой.

Сейчас в микропроцессорных системах управления промышленных роботов наиболее часто применяются линейная и круговая интерполяция в декартовой системе координат и линейная интерполяция в пространстве обобщенных координат. При линейной интерполяции в обобщенных координатах манипулятор совершает движение за минимально необходимое время за счет минимально необходимых перемещений звеньев. Закон движения звеньев трапецеидальный для скорости. При линейной интерполяции в декартовых координатах схват манипулятора движется по прямой линии, ориентация его также меняется по линейному закону. В последнее время все большее применение находит сплайн-интерполяция. Сущность ее заключается в представлении траектории между узловыми точками кривой, описываемой степенными полиномами третьей, четвертой или пятой степени. При этом коэффициенты полинома выбираются так, чтобы обеспечить отработку ограничений, накладываемых на траекторию.

В общем случае выбор типа движения определяется решаемой технологической задачей. Если требуется выполнить дуговую сварку, окраску или нанесение покрытия, то необходимо отрабатывать траекторию в декартовых координатах. В этом случае траектория движения манипулятора имеет первостепенное значение, хотя некоторые звенья манипулятора совершают большие перемещения, чем необходимо для перемещения в конечную точку, и могут возникнуть излишние ускорения (торможения) звеньев. При этом скорость движения самого схвата должна поддерживаться постоянной, игнорируя ограничения на скорость и ускорение в шарнирах манипулятора. С другой стороны, если манипулирование объектами осуществляется в упорядоченной детерминированной среде, когда отрабатываются только опорные точки и не важна траектория между ними, планирование в декартовых координатах можно не использовать, поскольку удобнее применять интерполяцию в обобщенных координатах, как более быструю и удобную для приводов.

# 1. ПОСТАНОВКА ЗАДАЧИ

Целью задания является моделирование движения манипулятора, приведенного на рисунке 1.1 в обобщенной системе координат (СК) методом линейной интерполяции траектории и графическое отображение его движения на дисплее ПК.

Задание приведено в таблице.1, исходные данные - в таблице.2.

Рисунок.1.1 Кинематическая схема манипулятора*.*

###### Таблица.1. Задание

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер варианта | Система координат | | Интерполяция | | | | | Исходные данные |
|  | Обобщенная | Декартовая | Линейная | Круговая | со сглаживанием  отрезков | Лагранжа | Сплайн |  |
| 9 | - | + | + | - | - | - | - | п/вар N3 |

Таблица.2. Исходные данные

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N  подварианты | a | v | | L1 | | L2 | ϕ-  const | θ-  const | | z-  const |
| 3 | 6500 | 70 | | 6 | | 3 | 0 | 0 | | 0 |
| N  подварианты | ψн | xн | yн | | ψк | Xк | Yк | ψпр | Xпр | Yпр |
| 3 | 0 | 5 | 5.6 | | 90 | 4.3 | -1.2 | - | - | - |

# 3. МАТЕМАТИЧЕСКОЕ РЕШЕНИЕ ЗАДАЧИ

**Первый этап**

Составление матриц однородных преобразований для начальной, конечной и промежуточных точек.

Матрица однородного преобразования, описывающая положение и ориентацию схвата в базовой системе координат ***Qi(t)***, имеет вид:

.

(2.1)

Подставляя значения углов Эйлера и координат схвата в формулу (2.1), получаем матрицы однородных преобразований для начальной, конечной и промежуточных точек траектории.

**Второй этап**

Решение обратной задачи кинематики для каждого звена в начальной, конечной и промежуточных точках. Решим ее с точки зрения алгебраического подхода.

Добавляем к манипулятору еще одну степень подвижности (поворот схвата на угол Q3). Кинематические параметры полученного трехзвенного манипулятора приведены в таблице.2.1.

Таблица.2.1. Кинематические параметры манипулятора

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***I*** | ***α*** | ***a*** | ***d*** | ***Q*** |
| 1 | 0 |  | 0 | Q1 |
| 2 | 0 |  | 0 | Q2 |
| 3 | 0 | 0 | 0 | Q3 |

Матрица, описывающая положение третьего звена (схвата) относительно базовой системы координат, выглядит следующим образом:

,

(2.2)

где *с123=cos(Q1+ Q2+ Q3);*

*s123=sin(Q1+ Q2+ Q3);*

*c1=cos(Q1);*

*c12=cos(Q1+ Q2);*

*s1=sin(Q1);*

*s12=sin(Q1+ Q2).*

Для рассматриваемого манипулятора матрицу  можно записать в виде:

 = ,

(2.3)

где *сψ=cosψ;*

*sψ=sinψ;*

Приравняем выражения (2.2) и (2.3) и получим четыре нелинейных уравнения, которые необходимо решить относительно Q1, Q2, Q3.



(2.4)

Возведя в квадрат, левые и правые части двух последних уравнений системы (2.4) и складывая их, получим:

.

(2.5)

где *с*2*=cosQ2.*

Решая уравнение (2.5) относительно *с*2, получим

.

(2.6)

Для существования решения правая часть уравнения (2.6) должна лежать в пределах [-1…+1]. Если условие выполнено, то находим

(2.7)

.

Выбор знака у *sinQ2* соответствует одному из возможных решений. Принимаем положение манипулятора “локоть вниз”, для которого *sinQ2* в данном случае принимает положительные значения. При определении угла *Q2* воспользуемся функцией *ATAN2,* которая обеспечивает выбор всех решений и выбор квадранта.

Отсюда *Q2* будет равно

.

(2.8)

Зная *Q2*,воспользуемся двумя последними уравнениями системы (2.4) для определения угла *Q1*. Перепишем их в следующем виде:



(2.9)

где ;

.

Для решения уравнений системы (2.9) выполним замену переменных:

; ,

тогда

; .

Теперь систему уравнений (2.9) можно записать следующим образом:



(2.10)

Откуда находим



(2.11)

Используя функцию *ATAN2*, имеем



(2.12)

Из первых двух уравнений системы (2.4) находим

.

(2.13)

Подставляя исходные данные в выражения (2.6) – (2.13) и проведя соответствующий расчет, находим значения обобщенных координат для начальной, конечной и промежуточных точек.

**Третий этап**

Простейшим и часто используемым видом локальной интерполяции является **линейная интерполяция**. Она состоит в том, что заданные точки  при (i = 0. 1, ..., n) соединяются прямолинейными отрезками, и функция f(x) приближается ломаной с вершинами в данных точках.

    Уравнения каждого отрезка ломаной в общем случае разные. Поскольку имеется n интервалов http://econom.misis.ru/s/Hel/Matem/XX.gif, то для каждого из них в качестве уравнения интерполяционного многочлена используется уравнение прямой, проходящей через две точки. В частности, для i-го интервала можно написать уравнение прямой, проходящей через точки http://econom.misis.ru/s/Hel/Matem/XYY.gif и http://econom.misis.ru/s/Hel/Matem/XY.gif, в виде

http://econom.misis.ru/s/Hel/Matem/Fo1.gif

    Отсюда

|  |  |
| --- | --- |
| http://econom.misis.ru/s/Hel/Matem/Fo2.gif | (\*) |

http://econom.misis.ru/s/Hel/Matem/Fo3.gif

    Следовательно, при использовании линейной интерполяции сначала нужно определить интервал, в который попадает значение аргумента х, а затем подставить его в формулу (\*) и найти приближенное значение функции в этой точке.

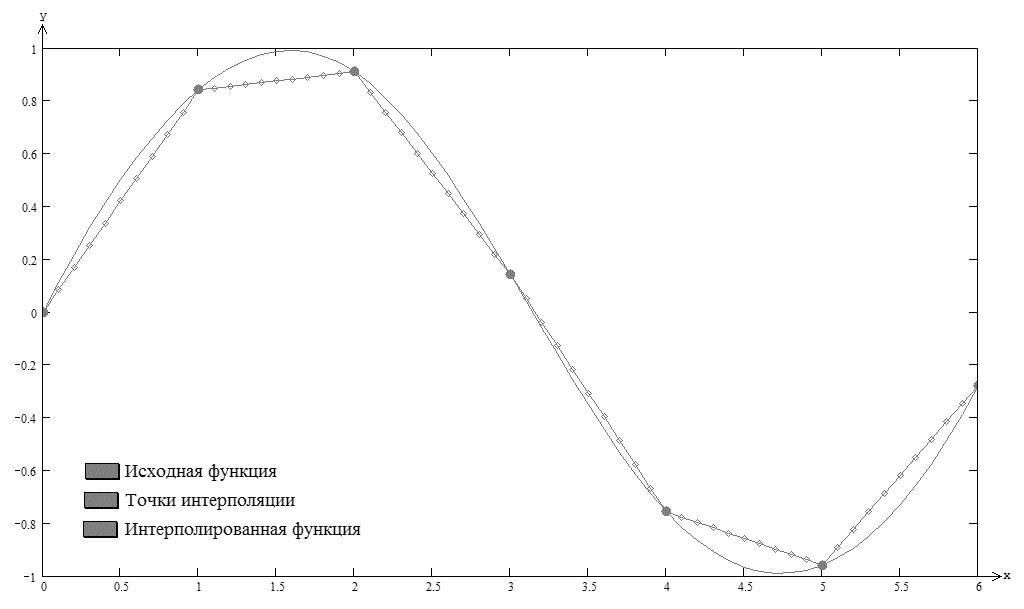


Рис. 2.1. Вид линейной интерполяции

**Четвертый этап**

Поточечное преобразование спланированной в обощенной системе координат траектории в систему декартовых координат путем решения прямой задачи кинематики. Прямую задачу будем решать геометрическим методом.

Декартовы координаты точек A(*x1,y1*) и B(*x2,y*2) относительно центра базовой системы координат O(0,0) вычисляются с помощью геометрических зависимостей (рис.2.3):

*x1=l1cosQ1 ; x2=x1+l2cos(Q1+Q2) ;*

*y1=l1sinQ1 ; y2=y1+l2sin(Q1+Q2) .*

Таким образом, определили все необходимые для написания программы и моделирования движения манипулятора формулы.

# 4. МОДЕЛИРУЮЩАЯ ПРОГРАММА

## 4.1. Описание программы

Моделирующая программа написана на языке Python3 и выполняет следующие операции:

-решение обратной задачи кинематики для заданных контрольных точек (начальной, конечной и промежуточных);

-планирование сглаженной траектории в обобщенных координатах методом интерполяции линейной;

-решение прямой задачи кинематики в каждой точке геометрическим методом;

-визуализацию движения трехзвенного манипулятора;

-построение графиков зависимостей обобщенных координат q1, q2, q3, а также их скоростей и ускорений от времени.

Программа построена по блочно-модульному типу с использованием многократно вызываемых методов и событий. Моделирующая программа включает в себя следующие основные методы и события:

- Time – метод создает массив значений времени, в соответствии со временем прохождения каждого участка (всего 100 элементов);

- ozk – метод для решения обратной задачи кинематики, с точки зрения алгебраического подхода, для каждого звена в начальной, конечной и промежуточных точках;

- interp – метод для реализации линейной интерполяции с помощью функции библиотеки scipy: функции interp1d;

- diff\_1\_interp – метод для реализации линейной интерполяции с помощью формулы(нахождения скорости – первая производная полинома по времени);

- diff\_2\_interp – метод для реализации линейной интерполяции с помощью формулы (нахождения ускорения – вторая производная полинома по времени);

- pzk – метод для решения прямой задачи кинематики в каждой точке геометрическим методом;

- Array\_parameters –метод, который разбивает заданный интервал X, Y, PSI на 100 равных промежутков;

- Plot\_trajectory – графический метод осуществляет визуализацию движения трехзвенного манипулятора в обобщенных координатах;

- Plot – графический метод осуществляет построение графиков положения, скорости и ускорения всех звеньев в обобщенных координатах;

- Solution – метод осуществляющий ввод всех начальных значений и подсчет X, Y, PSI, T, Qr1, Qr2, Qr3, x1, y1, x2, y2;

- Moving\_plot *–* графический метод осуществляет построение графиков положений всех звеньев в обобщенных координатах;

- Velosity\_plot *–* графический метод осуществляет построение графиков скоростей всех звеньев в обобщенных координатах;

- Axeleration\_plot *–* графический метод осуществляет построение графиков ускорений всех звеньев в обобщенных координатах;

- Manipulator\_plot *–* графический метод осуществляет построение траектории движения манипулятора в обобщенных координатах;

## 4.2. Текст моделирующей программы

import sys

import numpy as np

import manipulator as mn

import Manipulator\_app as M\_app

from PyQt5 import QtWidgets, QtGui, QtCore

def Solution():

x = np.array([5, 4.3])

y = np.array([-5.6, -1.2])

psi = np.radians(np.array([0,90]))

V = 0.7

A = 6.5

T = mn.Time(x, y, V, A)

L1 = 6

L2 = 4

X, Y, PSI = mn.Array\_parameters(x, y, psi)

Qr1, Qr2, Qr3 = mn.ozk(X, Y, L1, L2, PSI)

x1, y1, x2, y2 = mn.pzk(L1, L2, Qr1, Qr2)

return x, y, X, Y, PSI, T, Qr1, Qr2, Qr3, x1, y1, x2, y2, L1, L2

class MainWindow(QtWidgets.QMainWindow, M\_app.Ui\_MainWindow):

def \_\_init\_\_(self, callback):

super().\_\_init\_\_()

x, y, X, Y, PSI, T, Qr1, Qr2, Qr3, x1, y1, x2, y2, L1, L2 = callback()

self.T = T

self.X = X

self.Y = Y

self.L1 = L1

self.L2 = L2

self.Qr1 = Qr1

self.Qr2 = Qr2

self.Qr3 = Qr3

self.x1 = x1

self.x2 = x2

self.y1 = y1

self.y2 = y2

self.PSI = PSI

self.x = x

self.y = y

self.home()

def home(self):

self.setupUi(self)

self.Velosity\_plot\_btn.clicked.connect(self.Velosity\_plot)

self.Axeleration\_plot\_btn.clicked.connect(self.Axeleration\_plot)

self.Moving\_plot\_btn.clicked.connect(self.Moving\_plot)

self.Trajectory\_plot\_btn.clicked.connect(self.Manipulator\_plot)

self.Close\_btn.clicked.connect(sys.exit)

self.show()

def Velosity\_plot(self):

Qr1\_, Qr\_2, Qr\_3 = mn.diff\_1\_interp(self.Qr1, self.Qr2, self.Qr3, self.T)

return mn.Plot(Qr1\_, Qr\_2, Qr\_3, self.T,

legend=['Q1\'(t)', 'Q2\'(t)', 'Q3\'(t)'],

title='График угловой скорости')

def Axeleration\_plot(self):

\_Qr\_1,\_Qr\_2,\_Qr\_3 = mn.diff\_2\_interp(self.Qr1, self.Qr2, self.Qr3, self.T)

return mn.Plot(\_Qr\_1,\_Qr\_2,\_Qr\_3, self.T,

legend=['Q1\"(t)', 'Q2\"(t)', 'Q3\"(t)'],

title='График углового ускорения')

def Moving\_plot(self):

return mn.Plot(self.Qr1, self.Qr2, self.Qr3, self.T,

legend=['Q1(t)', 'Q2(t)', 'Q3(t)'],

title='График углового перемещения')

def Manipulator\_plot(self):

return mn.Plot\_trajectory(self.x1, self.y1,

self.x2, self.y2)

def main():

app = QtWidgets.QApplication(sys.argv)

window = MainWindow(Solution)

window.show()

app.exec\_()

if \_\_name\_\_ == '\_\_main\_\_':

main()

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.animation as animation

import sys

from scipy.interpolate import interp1d

from math import \*

def Array\_parameters(x, y, psi, num=100):

X = np.linspace(x[0], x[1], num=num)

Y = np.linspace(y[0], y[1], num=num)

PSI = np.linspace(psi[0], psi[1], num=num)

return X, Y, PSI

def ozk(x,y,L1,L2,psi):

c2 = (x\*\*2 + y\*\*2 - L1\*\*2- L2\*\*2)/(2\*L1\*L2)

s2 = np.sqrt(1 - c2\*\*2)

Qr2 = np.arctan2(s2,c2)

Qr1 = np.arctan2(y,x) - np.arctan2(L2\*s2,L1+L2\*c2)

Qr3 = psi - Qr1 - Qr2

return Qr1, Qr2, Qr3

def pzk(L1,L2,Qr1, Qr2):

x1 = L1 \* np.cos(Qr1)

y1 = L1 \* np.sin(Qr1)

x2 = x1 + L2 \* np.cos(Qr1+Qr2)

y2 = y1 + L2 \* np.sin(Qr1+Qr2)

return x1,y1,x2,y2

def interp(x,y,X):

f = interp1d(x,y)

Y\_interp = f(X)

return Y\_interp

def diff\_1\_interp(Qr1, Qr2, Qr3, T):

T\_vector = np.linspace(0, T, len(Qr1))

dt = np.ediff1d(T\_vector)

Qr\_1 = np.ediff1d(Qr1)/dt

Qr\_2 = np.ediff1d(Qr2)/dt

Qr\_3 = np.ediff1d(Qr3)/dt

return Qr\_1,Qr\_2,Qr\_3

def diff\_2\_interp(Qr1, Qr2, Qr3, T):

T\_vector = np.linspace(0, T, len(Qr1))

dt = np.ediff1d(T\_vector)

\_Qr\_1 = np.ediff1d((np.ediff1d(Qr1) / dt) / dt)

\_Qr\_2 = np.ediff1d((np.ediff1d(Qr2) / dt) / dt)

\_Qr\_3 = np.ediff1d((np.ediff1d(Qr3) / dt) / dt)

return \_Qr\_1,\_Qr\_2,\_Qr\_3

def Plot\_trajectory(x1, y1, x2, y2):

fig = plt.figure()

ax = fig.add\_subplot(111, xlim=(0, 10), ylim=(-10, 0))

ax.grid()

line, = ax.plot([], [], 'o-', lw=2)

line2, = ax.plot([], [], '-', lw=2)

def init():

line.set\_data([], [])

line2.set\_data([], [])

return line,line2

def animate(i):

thisx = [0, x1[i], x2[i]]

thisy = [0, y1[i], y2[i]]

line.set\_data(thisx, thisy)

line2.set\_data(x2[:i],y2[:i])

return line,line2

ani = animation.FuncAnimation(fig, animate, frames= np.arange(1,len(x1)),

interval=25, blit=True, init\_func=init, repeat=False)

plt.show()

return ani

def Plot(Qr1, Qr2, Qr3, T, legend=[], title=''):

T = np.linspace(0, T, len(Qr1))

fig = plt.figure()

ax = fig.add\_subplot(211, autoscale\_on = True)

ax.grid()

f1, f2, f3 = ax.plot(T, Qr1,'-', T, Qr2, '--', T, Qr3, '-.', linewidth=2)

ax.set\_xlabel('Time(sec)')

ax.set\_ylabel('Radians')

fig.legend((f1, f2, f3), (legend[0], legend[1], legend[2]), 'upper left')

plt.title(title)

plt.show()

def Time(x, y, Speed, Axeleration):

Distance = np.sqrt((x[1]-x[0])\*\*2 + (y[1]-y[0])\*\*2)

T = Distance/Speed

return T

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui\_MainWindow(object):

def setupUi(self, MainWindow):

MainWindow.setObjectName("MainWindow")

MainWindow.resize(372, 480)

self.centralwidget = QtWidgets.QWidget(MainWindow)

self.centralwidget.setObjectName("centralwidget")

self.Trajectory\_plot\_btn = QtWidgets.QPushButton(self.centralwidget)

self.Trajectory\_plot\_btn.setGeometry(QtCore.QRect(10, 10, 351, 61))

self.Trajectory\_plot\_btn.setObjectName("Trajectory\_plot\_btn")

self.Moving\_plot\_btn = QtWidgets.QPushButton(self.centralwidget)

self.Moving\_plot\_btn.setGeometry(QtCore.QRect(10, 80, 351, 61))

self.Moving\_plot\_btn.setObjectName("Moving\_plot\_btn")

self.Velosity\_plot\_btn = QtWidgets.QPushButton(self.centralwidget)

self.Velosity\_plot\_btn.setGeometry(QtCore.QRect(10, 150, 351, 61))

self.Velosity\_plot\_btn.setObjectName("Velosity\_plot\_btn")

self.Axeleration\_plot\_btn = QtWidgets.QPushButton(self.centralwidget)

self.Axeleration\_plot\_btn.setGeometry(QtCore.QRect(10, 220, 351, 61))

self.Axeleration\_plot\_btn.setObjectName("Axeleration\_plot\_btn")

self.Close\_btn = QtWidgets.QPushButton(self.centralwidget)

self.Close\_btn.setGeometry(QtCore.QRect(10, 380, 351, 91))

self.Close\_btn.setAutoFillBackground(False)

self.Close\_btn.setObjectName("Close\_btn")

MainWindow.setCentralWidget(self.centralwidget)

self.retranslateUi(MainWindow)

QtCore.QMetaObject.connectSlotsByName(MainWindow)

def retranslateUi(self, MainWindow):

\_translate = QtCore.QCoreApplication.translate

MainWindow.setWindowTitle(\_translate("MainWindow", "MainWindow"))

self.Trajectory\_plot\_btn.setText(\_translate("MainWindow", "Manipulator moving"))

self.Moving\_plot\_btn.setText(\_translate("MainWindow", "Q(t)"))

self.Velosity\_plot\_btn.setText(\_translate("MainWindow", "Q\'(t)"))

self.Axeleration\_plot\_btn.setText(\_translate("MainWindow", "Q\'\'(t)"))

self.Close\_btn.setText(\_translate("MainWindow", "Close"))

# 5. РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ПРОГРАММЫ

При запуске программы main.py (программа написана на языке Python) появляется диалоговое окно. В представленном меню мы выбираем раздел с нужными нам сведениями.

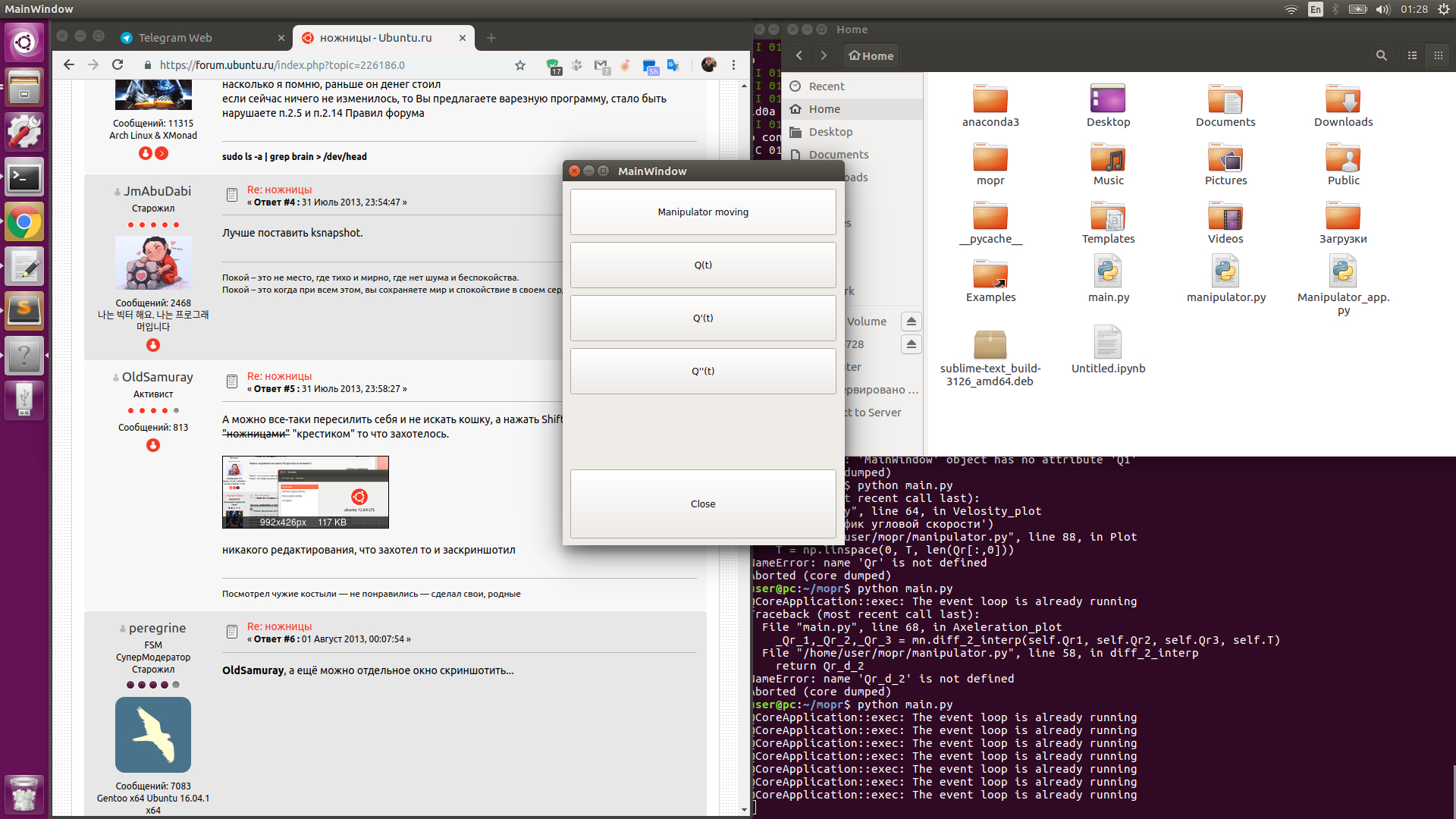


Рисунок 5.1 Диалоговое окно

Результаты выполнения программы представлены на рисунках 5.1 – 5.4. На рис.5.2 показан экран программы после планирования траектории с изображенным манипулятором, траекторией движения, промежуточными значениями и исходными данными.

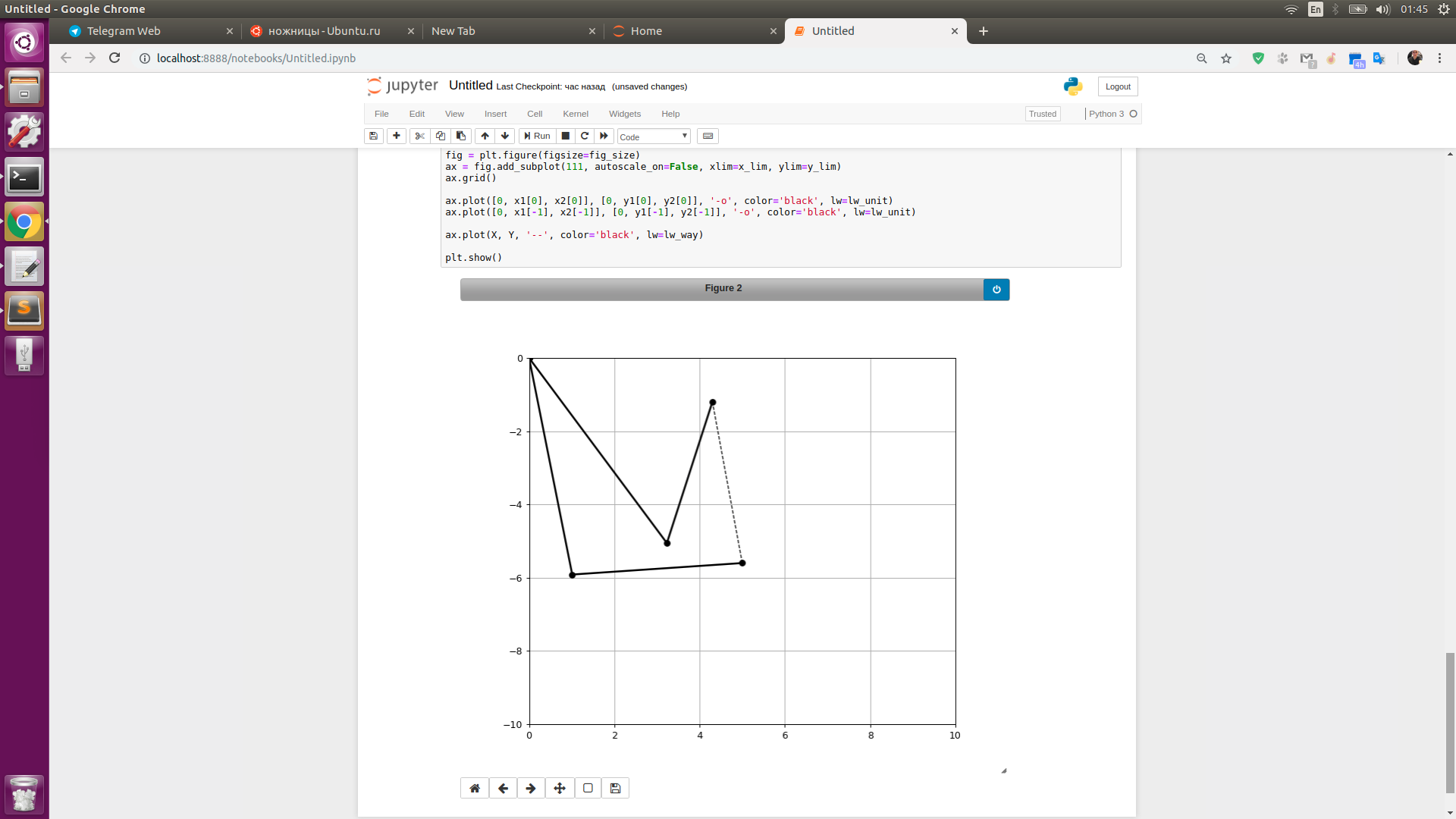
****

Рисунок 5.2. Визуализация движения манипулятора в системе декартовых координат (начальное положение, конечное и траектория движения)

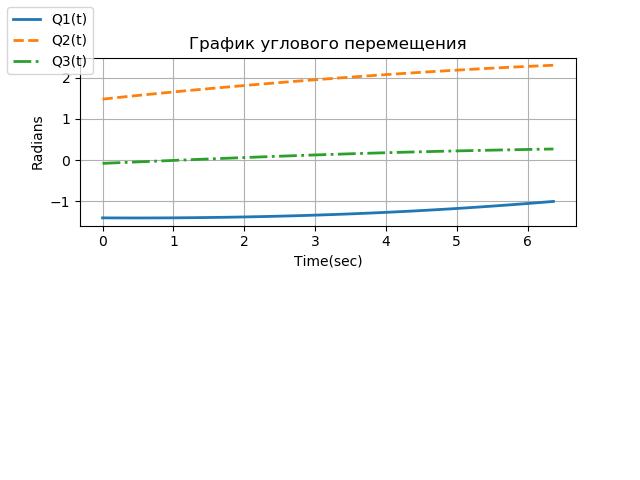


Рисунок 5.3. Графики обобщенных координат звеньев.

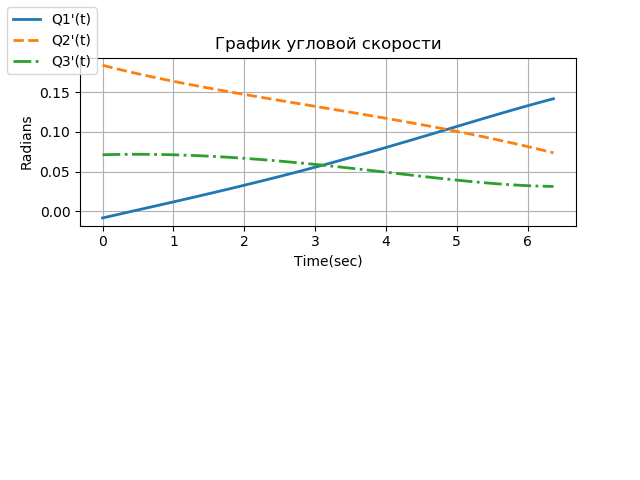
******

Рисунок 5.4. Графики скоростей звеньев.

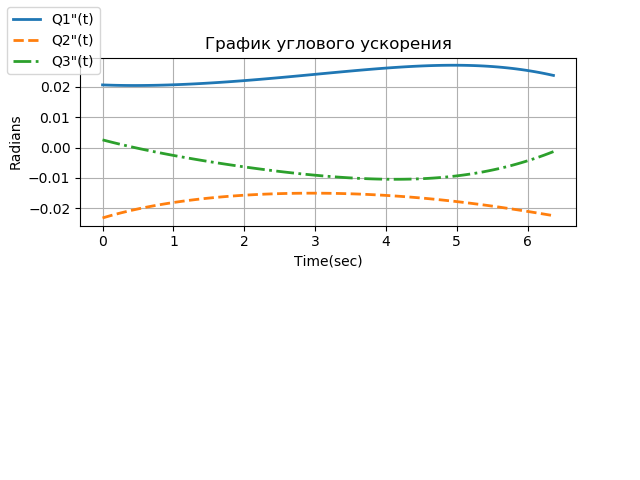


Рисунок 5.5. Графики ускорений звеньев.

# ЛИТЕРАТУРА

1. К. Фу, Р. Гонсалес, К. Ли. Робототехника. - М.: Мир,

1989. - 624 с.

2. М. Шахинпур. Курс робототехники. - М.: Мир, 1990.- 527 с.

3. Механика промышленных роботов. Под ред. К.В. Фролова,

Е.И. Воробьева. Кн. 1. Кинематика и динамика. - М.: Высшая школа, 1988. - 304 с.

4. Г. Корн, Т. Корн. Справочник по математике. - М.: Мир,

1974. - 832 с.

5. Околов А.Р. Математическое обеспечение промышленных роботов: учебно-методический комплекс для студентов специальности 1-53 01 06 «Промышленные роботы и робототехнические комплексы»: в 2 ч./ А.Р.Околов, Е.Р.Новичихина, Г.С.Свидерский. – Минск: БНТУ, 2012. – Ч.1: Лабораторные работы. – 80 с.