Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

**Национальный исследовательский университет «МЭИ»**

Кафедра вычислительных технологий

Практическое задание по дисциплине

**«Геометрическое моделирование в САПР»**

Создание моделей кривых и поверхностей

Выполнила:

Лиханова Е.Е.

Группа А-09-19

Вариант 8

Проверила:

Лешихина И.Е.

**Москва, 2022**

**ОГЛАВЛЕНИЕ**

[**1.** **ЗАДАНИЕ** 3](#_Toc122576648)

[**2.** **ВЫПОЛНЕНИЕ** 4](#_Toc122576649)

[**2.1.** **ПОСТРОЕНИЕ СОСТАВНОЙ КРИВОЙ ЭРМИТА** 4](#_Toc122576650)

[**2.2.** **ПОСТРОЕНИЕ ПОВЕРХНОСТИ ВРАЩЕНИЯ** 8](#_Toc122576651)

[**2.3.** **ИЗОМЕТРИЧЕСКАЯ ПРОЕКЦИЯ** 9](#_Toc122576652)

[**ПРИЛОЖЕНИЕ А** 10](#_Toc122576653)

# **ЗАДАНИЕ**

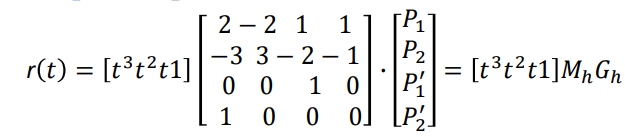
Построить поверхность вращения относительно оси Z составной кривой Эрмита. Число опорных точек – 4 (3 сегмента). Для визуализации использовать изометрическую проекцию.

# **ВЫПОЛНЕНИЕ**

## **ПОСТРОЕНИЕ СОСТАВНОЙ КРИВОЙ ЭРМИТА**

Кривая Эрмита является частным случаем кубического сплайна, а именно нормализованным кубическим сплайном. Параметр t на каждом интервале составной кривой изменяется от 0 до 1.

С помощью математического представления кривой Эрмита (1) получим уравнение, представленное на рисунке 1.



(1)

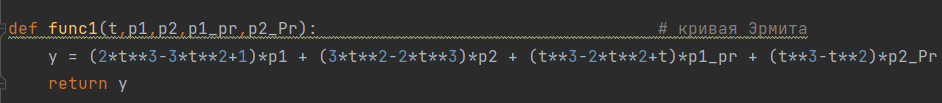


Рисунок 1. Уравнение кривой Эрмита

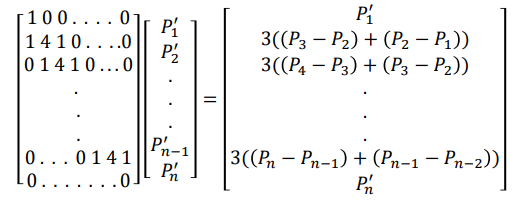
Для того, чтобы составная кривая Эрмита была непрерывна, необходимо обеспечить геометрическую непрерывность нулевого, первого и второго порядка в местах стыка сегментов (2).

(2)

,

– первая точка второго сегмента

Для расчета значений производных в промежуточных точках воспользуемся матричным уравнением (3) из фрагмента лекции.



(3)

В данном случае n = 4. Матричное уравнение примет вид:

(4)

\* =

Преобразуем левую часть уравнения, перемножив матрицы:

(5)

\* =

Данное преобразование позволяет приравнять матрицы уравнений (4) и (5).

=

Получаем систему уравнений (6), в которой неизвестны .

(6)

Найдем неизвестные переменные.

Из 1) получаем:

(7)

Подставим в 1):

Отсюда получаем

(8)

Фрагмент кода с заданием и с помощью формул (7) и (8) представлен на рисунке 2.

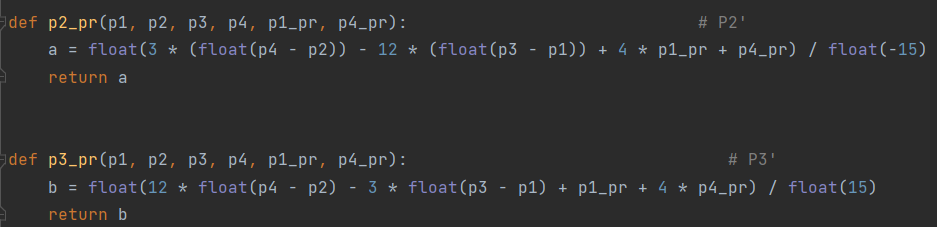


Рисунок 2. Функции для и

Для удобства дальнейших вычислений создадим два массива: в первый поместим производные точек, являющихся началом сегментов, а во второй поместим производные точек, являющихся концами сегментов. Данная логика представлена на рисунке 3.

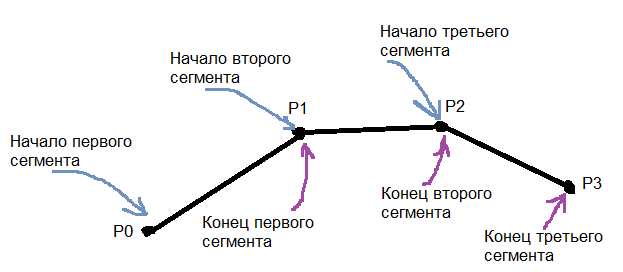


Рисунок 3. Логика создания массивов

В первый массив поочередно добавляются значения , , по X, Y, Z. Во второй массив поочередно добавляются значения , , по X, Y, Z. По умолчанию примем как 1, а как -1. Фрагмент кода с выше описанным процессом представлен на рисунке 4.

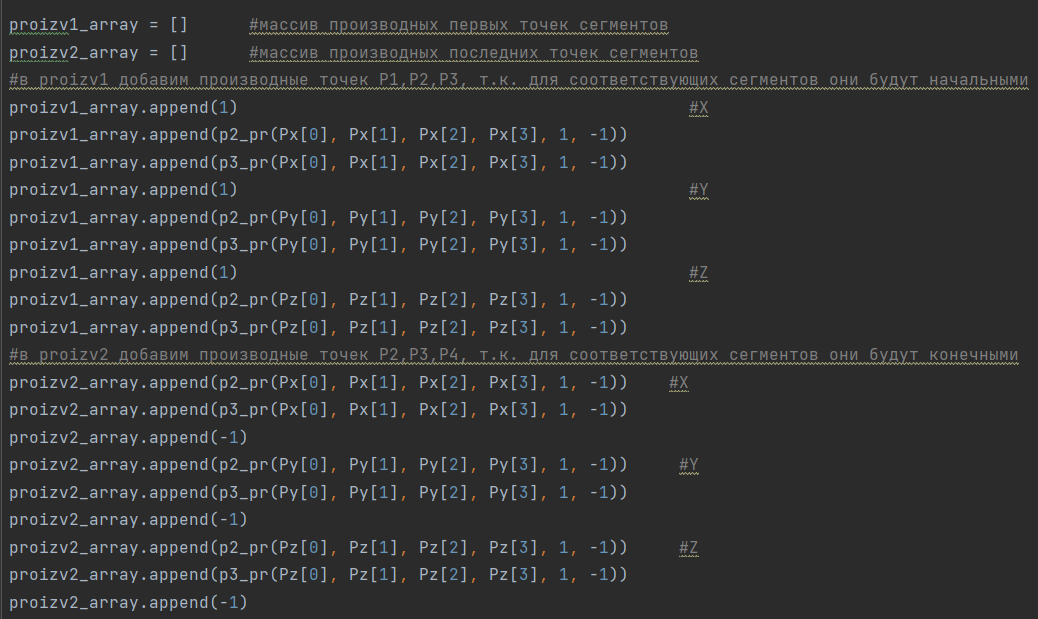


Рисунок 4. Создание массивов с производными

Далее переходим к построению составной кривой Эрмита.

Циклично получаем значения точек по X, Y, Z, заносим их в соответствующие массивы и после отстраиваем рассчитанную кривую. Данный фрагмент кода и результат построения кривой представлены на рисунках 5 и 6 соответственно.

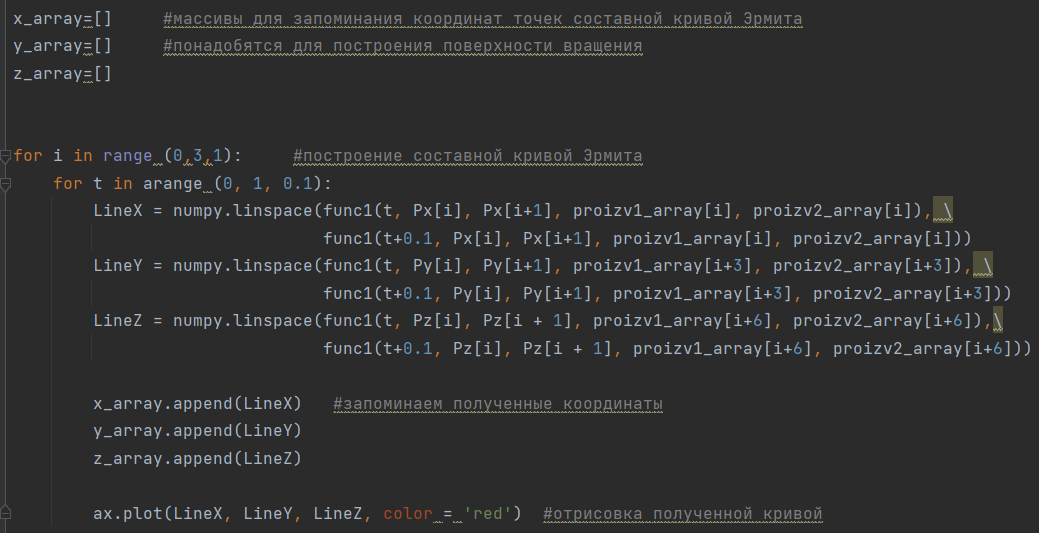


Рисунок 5. Построение составной кривой Эрмита

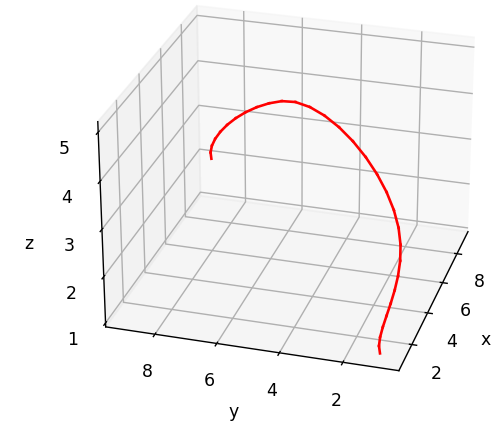
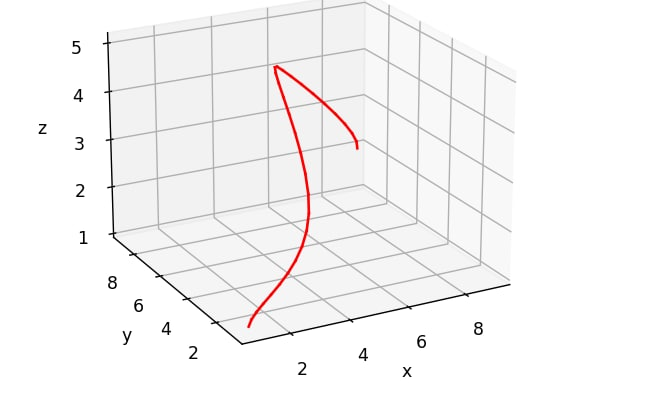
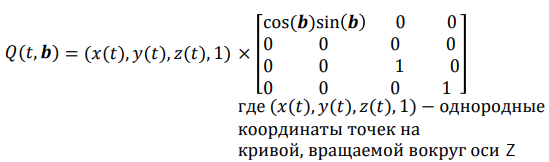
 

Рисунок 6. Результат построения кривой

## **ПОСТРОЕНИЕ ПОВЕРХНОСТИ ВРАЩЕНИЯ**

Поверхность вращения – это поверхность, построенная в результате вращения двумерного объекта (в данном случае кривой) вокруг оси в пространстве.

Для построения поверхности вращения относительно оси Z воспользуемся уравнением из лекции.



(9)

Из уравнения (9) следует, что для получения поверхности вращения относительно оси Z необходимо умножить X и Y координаты на косинус и синус β соответственно. β изменяется от 0 до 2π. Фрагмент кода с реализацией данного алгоритма представлен на рисунке 7.

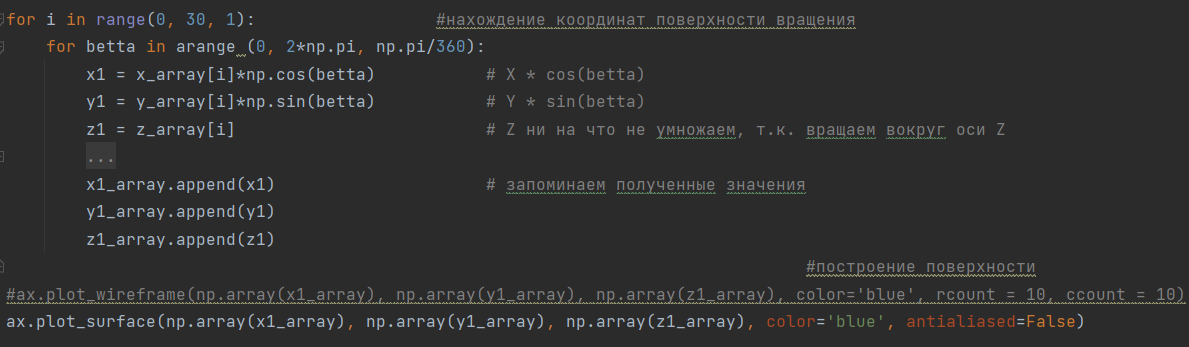


Рисунок 7. Создание поверхности вращения относительно Z

Результат создания поверхности вращения представлен на рисунке 8.

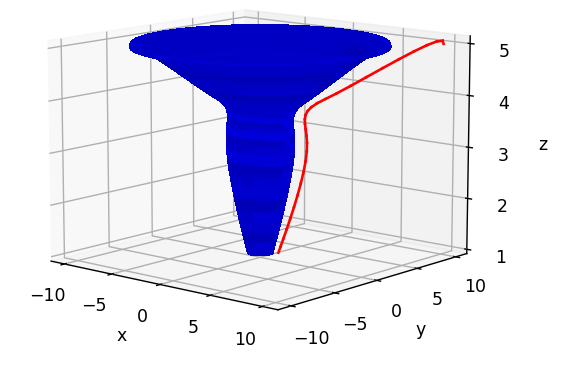
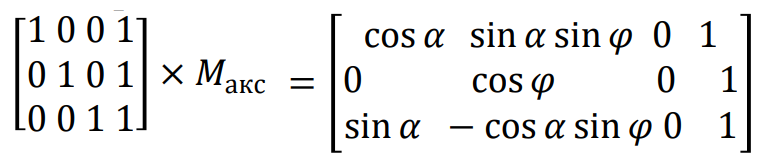


Рисунок 8. Результат построения поверхности вращения.

## **ИЗОМЕТРИЧЕСКАЯ ПРОЕКЦИЯ**

Изометрическая проекция – это подвид аксонометрической проекции, при которой укорачивание по всем координатным осям одинаковое.

Для построения воспользуемся уравнением аксонометрической проекции из лекции.



(10)

Из одинакового укорачивания по всем координатным осям следует, что , (угол поворота относительно оси X).

Фрагмент кода представлен на рисунке 9.

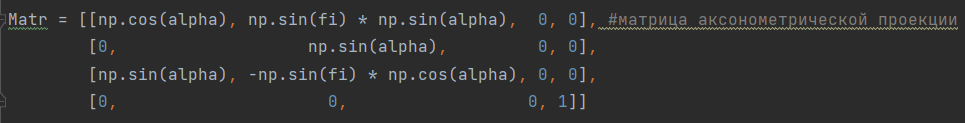


Рисунок 9. Матрица аксонометрической проекции

Изображение в изометрической проекции представлено на рисунке 10.

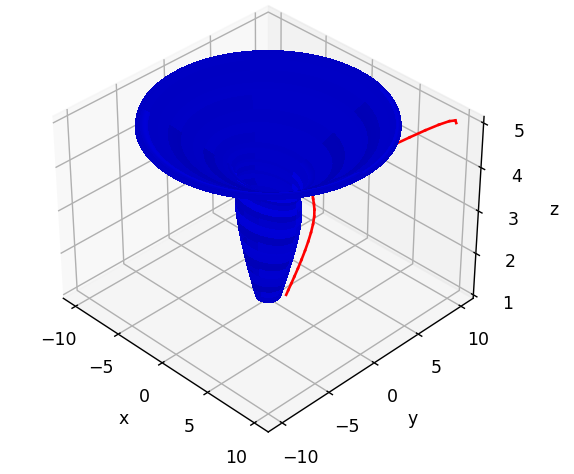


Рисунок 10. Изображение в изометрической проекции

# **ПРИЛОЖЕНИЕ А**

Код программы:

import numpy as np

import matplotlib.pyplot as plt

from numpy.array\_api import arange

fi = 35\*np.pi/180 # угол фи равен ±35 градусов 26′

alpha = 45\*np.pi/180 # угол альфа равен ±45 градусов

Matr = [[np.cos(alpha), np.sin(fi) \* np.sin(alpha), 0, 0], #матрица аксонометрической проекции

[0, np.sin(alpha), 0, 0],

[np.sin(alpha), -np.sin(fi) \* np.cos(alpha), 0, 0],

[0, 0, 0, 1]]

def func1(t,p1,p2,p1\_pr,p2\_Pr): # кривая Эрмита

y = (2\*t\*\*3-3\*t\*\*2+1)\*p1 + (3\*t\*\*2-2\*t\*\*3)\*p2 + (t\*\*3-2\*t\*\*2+t)\*p1\_pr + (t\*\*3-t\*\*2)\*p2\_Pr

return y

def p2\_pr(p1, p2, p3, p4, p1\_pr, p4\_pr): # P2'

a = float(3 \* (float(p4 - p2)) - 12 \* (float(p3 - p1)) + 4 \* p1\_pr + p4\_pr) / float(-15)

return a

def p3\_pr(p1, p2, p3, p4, p1\_pr, p4\_pr): # P3'

b = float(12 \* float(p4 - p2) - 3 \* float(p3 - p1) + p1\_pr + 4 \* p4\_pr) / float(15)

return b

Px = [1, 3, 4, 10] #точки по оси Х

Py = [1, 2, 3, 10] #точки по оси Y

Pz = [1, 3, 4, 5] #точки по оси Z

proizv1\_array = [] #массив производных первых точек сегментов

proizv2\_array = [] #массив производных последних точек сегментов

#в proizv1 добавим производные точек Р1,Р2,Р3, т.к. для соответствующих сегментов они будут начальными

proizv1\_array.append(1) #X

proizv1\_array.append(p2\_pr(Px[0], Px[1], Px[2], Px[3], 1, -1))

proizv1\_array.append(p3\_pr(Px[0], Px[1], Px[2], Px[3], 1, -1))

proizv1\_array.append(1) #Y

proizv1\_array.append(p2\_pr(Py[0], Py[1], Py[2], Py[3], 1, -1))

proizv1\_array.append(p3\_pr(Py[0], Py[1], Py[2], Py[3], 1, -1))

proizv1\_array.append(1) #Z

proizv1\_array.append(p2\_pr(Pz[0], Pz[1], Pz[2], Pz[3], 1, -1))

proizv1\_array.append(p3\_pr(Pz[0], Pz[1], Pz[2], Pz[3], 1, -1))

#в proizv2 добавим производные точек Р2,Р3,Р4, т.к. для соответствующих сегментов они будут конечными

proizv2\_array.append(p2\_pr(Px[0], Px[1], Px[2], Px[3], 1, -1)) #X

proizv2\_array.append(p3\_pr(Px[0], Px[1], Px[2], Px[3], 1, -1))

proizv2\_array.append(-1)

proizv2\_array.append(p2\_pr(Py[0], Py[1], Py[2], Py[3], 1, -1)) #Y

proizv2\_array.append(p3\_pr(Py[0], Py[1], Py[2], Py[3], 1, -1))

proizv2\_array.append(-1)

proizv2\_array.append(p2\_pr(Pz[0], Pz[1], Pz[2], Pz[3], 1, -1)) #Z

proizv2\_array.append(p3\_pr(Pz[0], Pz[1], Pz[2], Pz[3], 1, -1))

proizv2\_array.append(-1)

fig = plt.figure() #задание поля для отрисовки

ax = fig.add\_subplot(111, projection='3d')

ax.set\_xlabel('x') # подпись оси x

ax.set\_ylabel('y') # подпись оси y

ax.set\_zlabel('z') # подпись оси z

x\_array=[] #массивы для запоминания координат точек составной кривой Эрмита

y\_array=[] #понадобятся для построения поверхности вращения

z\_array=[]

for i in range (0,3,1): #построение составной кривой Эрмита

for t in arange (0, 1, 0.1):

LineX = numpy.linspace(func1(t, Px[i], Px[i+1], proizv1\_array[i], proizv2\_array[i]), \

func1(t+0.1, Px[i], Px[i+1], proizv1\_array[i], proizv2\_array[i]))

LineY = numpy.linspace(func1(t, Py[i], Py[i+1], proizv1\_array[i+3], proizv2\_array[i+3]), \

func1(t+0.1, Py[i], Py[i+1], proizv1\_array[i+3], proizv2\_array[i+3]))

LineZ = numpy.linspace(func1(t, Pz[i], Pz[i + 1], proizv1\_array[i+6], proizv2\_array[i+6]),\

func1(t+0.1, Pz[i], Pz[i + 1], proizv1\_array[i+6], proizv2\_array[i+6]))

x\_array.append(LineX) #запоминаем полученные координаты

y\_array.append(LineY)

z\_array.append(LineZ)

ax.plot(LineX, LineY, LineZ, color = 'red') #отрисовка полученной кривой

x1\_array=[] #массивы для запоминания координат точек поверхности вращения

y1\_array=[]

z1\_array=[]

for i in range(0, 30, 1): #нахождение координат поверхности вращения

for betta in arange (0, 2\*np.pi, np.pi/360):

x1 = x\_array[i]\*np.cos(betta) # X \* cos(betta)

y1 = y\_array[i]\*np.sin(betta) # Y \* sin(betta)

z1 = z\_array[i] # Z ни на что не умножаем, т.к. вращаем вокруг оси Z

# if (betta >= 2 \* np.pi + np.pi / 180): # условие для отсечения 10 градусов, которые были взяты для выравнивания края

# x1 = x\_array[i] \* np.cos(betta) \* np.nan

# y1 = y\_array[i] \* np.sin(betta) \* np.nan

# z1 = z\_array[i] \* np.nan

x1\_array.append(x1) # запоминаем полученные значения

y1\_array.append(y1)

z1\_array.append(z1)

#построение поверхности

#ax.plot\_wireframe(np.array(x1\_array), np.array(y1\_array), np.array(z1\_array), color='blue', rcount = 10, ccount = 10)

ax.plot\_surface(np.array(x1\_array), np.array(y1\_array), np.array(z1\_array), color='blue', antialiased=False)

ax.view\_init(elev=35.433333, azim=-45) #визуализация в изометрической проекции

plt.show() #функция для отображения построенных элементов