|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

*ФАКУЛЬТЕТ «Информатика и системы управления»*

*КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»*

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 04 |

**Дисциплина:  *Mоделирование***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ***ИУ7И-66Б*** |  |  | **Нгуен Ф. С.** |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | **Градов В. М.** |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

*Москва, 2021*

***Цель работы*** *: Получение навыков разработки алгоритмов решения смешанной краевой задачи при реализации моделей, построенных на квазилинейном уравнении параболического типа.*

***Исходные данные.***

***1. Задана математическая модель.***

*Уравнение для функции*

*(1)*

*Краевые условия*

*(2)*

*В обозначениях уравнения (14.1) лекции №14*

*(3)*

***2. Разностная схема с разностным краевым условием при получена в Лекции может быть использована в данной работе.***

*Разностная схема:*

*(4)*

*(5)*

*(6)*

*(7)*

*(8)*

*Разностные аналоги краевых условий при х=0:*

*(9)*

*Самостоятельно надо получить интегро-интерполяционным методом разностный аналог краевого условия при . Для этого надо проинтегрировать на отрезке [xN-1/2, xN] выписанное выше уравнение (1) и учесть, что поток*

*(10)*

*(11)*

***3. Значения параметров для отладки (все размерности согласованы)***

*, Вт/см К,*

*, Дж/см3К.*

*=0.0134, =1, =4.35 10-4, =1,*

*=2.049, =0.563 10-3, =0.528 105, =1.*

*,*

*0.05 Вт/см2 К,*

*0.01 Вт/см2 К,*

*10 см,*

*300К,*

*0.5 см,*

*50 Вт/см2 (для отладки принять постоянным).*

***4. Физическое содержание задачи***

*Постановки задач в данной лабораторной работе и работе №3 во многом совпадают. Отличия заключаются в следующем:*

*1. Сформулированная в данной работе математическая модель описывает нестационарное температурное поле , зависящее от координаты x и меняющееся во времени.*

*2. Свойства материала стержня привязаны к температуре, т.е. теплоемкость и коэффициент теплопроводности зависят от .*

*3. При цилиндр нагружается тепловым потоком , в общем случае зависящим от времени.*

*Если в настоящей работе задать поток постоянным, т.е. =const, то будет происходить формирование температурного поля от начальной температуры до некоторого установившегося (стационарного) распределения . Это поле в дальнейшем с течением времени меняться не будет Это полезный факт для тестирования программы.*

*Если после разогрева стержня положить поток =0, то будет происходить остывание, пока температура не выровняется по всей длине и не станет равной .*

*При произвольной зависимости потока от времени температурное поле будет как-то сложным образом отслеживать поток.*

***Замечание****. Варьируя параметры задачи, следует обращать внимание на то, что решения, в которых температура превышает примерно 2000К, физического смысла не имеют и практического интереса не представляют.*

***5. Разностная схема***

|  |  |  |
| --- | --- | --- |
|  |  | *(12)* |

*Где*

*Для величин можно получить различные приближенные выражения, численно вычисляя интеграл методом трапеций*

|  |  |  |
| --- | --- | --- |
|  |  | *(13)* |

***6. Краевые условия***

*Обозначим*

|  |  |
| --- | --- |
|  | *(14)* |

***Разностные аналоги краевых условий при x = 0***

*Тогда*

***Разностный аналог краевого условия при x = l***

*Проинтегрируем уравнение (1) на отрезке и на временном интервале*

*при вычислении внутренних интегралов по справа в уравнении применен метод правых прямоугольников*

*Интегралы по x в вычислим методом средних, а первый интеграл в правой части (по времени) - по-прежнему, методом правых прямоугольников, получим*

*Подставим в полученное уравнение*

*Получим*

*Тогда*

***7. Метод простых итераций***

*Для решения разностной системы исползуется метод простых итераций. Обозначим текующую итерацию s, а предыдущую s-1, тогда процесс организуется по схеме*

*Все коэффтциенты берутся на (s-1)-ой итерации, т.е они известны. Получили обычную линейную схему, решение которой осуществляется методом прогонки, показанным ниже.*

*Итерации прекращаются при условии*

***8. Метод прогонки***

*Прямой ход : предполагаем, что система уравнений имеет вид*

*Все прогоночные коэффициенты можно найти по формулам*

*и*

*Обратный ход :*

*Находим начальное значение*

*Остальные значения находятся по формуле*

***Код программы***

import numpy as np

import math

import matplotlib.pyplot as plt

from matplotlib import cm

from mpl\_toolkits import mplot3d

#constants

a1 = 0.0134

b1 = 1

c1 = 4.35e-4

m1 = 1

a2 = 2.049

b2 = 0.563e-3

c2 = 0.528e5

m2 = 1

alpha0 = 0.05

alphaN = 0.01

l = 10

d = alphaN \* l / (alphaN - alpha0)

cc = -alpha0 \* d

T0 = 300

R = 0.5

Ft = 50

eps1 = 10e-4

eps2 = 10e-4

def plot2d(x, y, xlabel, ylabel, title):

plt.plot(x, y, 'c')

plt.xlabel(xlabel)

plt.ylabel(ylabel)

plt.title(title)

plt.grid(True)

plt.show()

def plot3d(x, y, z):

fig = plt.figure()

ax = plt.axes(projection='3d')

ax.plot\_surface(x, y, z, rstride=1, cstride=1, cmap='viridis', edgecolor='none')

#ax.set\_title("")

fig.show()

plt.show()

#ax[n-1]+bx[n]+cx[n+1]=d

def tridiagonal\_method(a, b, c, d):

n = len(a) - 1;

xi = [None for i in range(n+1)]

xi[1] = -c[0]/b[0]

eta = [None for i in range(n+1)]

eta[1] = d[0]/b[0]

for i in range(1, n):

xi[i+1] = -c[i]/(a[i]\*xi[i]+b[i])

eta[i+1] = (d[i]-a[i]\*eta[i])/(a[i]\*xi[i]+b[i])

res = [None for i in range(n+1)]

res[n] =(d[n]-a[n]\*eta[n])/(a[n]\*xi[n]+b[n])

for i in range(n-1, -1, -1):

res[i] = xi[i+1]\*res[i+1]+eta[i+1]

return res

def k(T):

return a1 \* (b1 + c1 \* T\*\*m1)

def c(T):

return a2 + b2 \* T\*\*m2 - c2 / T\*\*2

def alpha(x):

return cc / (x - d)

def p(x):

return 2/R \* alpha(x)

#f(T) = f(x)

def f(x):

return 2\*T0/R \* alpha(x)

def solve\_equation\_system(old\_T, prev\_T\_by\_time, tau, N):

\_a = []

\_b = []

\_d = []

\_f = []

h = l / N

def cal\_plus\_half(func, start, end):

return (func(start) + func(end)) / 2

def c\_(n):

return c(old\_T[n])

def c\_plus(n):

return cal\_plus\_half(c, old\_T[n], old\_T[n+1])

def chi\_plus(n):

return cal\_plus\_half(k, old\_T[n], old\_T[n+1])

def p\_plus(n):

return (p(n) + p(n+h)) / 2

def f\_plus(n):

return (f(n) + f(n+h)) / 2

def p\_(n):

return p(h\*n)

def f\_(n):

return f(h\*n)

#the left boundary condition x = 0

\_a.append(0)

\_b.append(h/8\*c\_plus(0)

+ h/4\*c\_(0)

+ tau/h\*chi\_plus(0)

+ tau\*h/8\*p\_plus(0)

+ tau\*h/4\*p(0))

\_d.append(h/8\*c\_plus(0)

- tau/h\*chi\_plus(0)

+ tau\*h/8\*p\_plus(0))

\_f.append(h/8\*c\_plus(0)\*(prev\_T\_by\_time[0]+prev\_T\_by\_time[1])

+ h/4\*c\_(0)\*prev\_T\_by\_time[0]

+ tau\*Ft

+ tau\*h/4\*(f\_plus(0) + f(0)))

#1 <= n <= N-1

for i in range(1, N):

\_a.append(tau/h\*chi\_plus(i-1))

\_d.append(tau/h\*chi\_plus(i))

\_b.append(-(\_a[-1] + \_d[-1] + c\_(i)\*h + p\_(i)\*h\*tau))

\_f.append(-(f\_(i)\*h\*tau + c\_(i)\*prev\_T\_by\_time[i]\*h))

#the right boudary condition x = l

\_a.append(h/8\*c\_plus(N-1)

- tau/h\*chi\_plus(N-1)

+ tau\*h/8\*p\_plus(N-1))

\_b.append(h/4\*c\_(N)

+ h/8\*c\_plus(N-1)

+ tau\*alphaN

+ tau/h\*chi\_plus(N-1)

+ tau\*h/4\*p\_(N)

+ tau\*h/8\*p\_plus(N-1))

\_d.append(0)

\_f.append(h/4\*c\_(N)\*prev\_T\_by\_time[N]

+ h/8\*c\_plus(N-1)\*prev\_T\_by\_time[N]

+ h/8\*c\_plus(N-1)\*prev\_T\_by\_time[N-1]

+ tau\*alphaN\*T0

+ tau\*h/4\*(f\_(N)+f\_plus(N-1)))

return tridiagonal\_method(\_a, \_b, \_d, \_f)

def check(res1, res2, eps):

tmp = 0

for i in range(len(res2)):

tmp = max(tmp, math.fabs((res2[i]-res1[i])/res2[i]))

if tmp <= eps:

return True

else:

return False

#simple iteration metho

def solve(tau, N):

h = l / N

time = 0

old\_T = [T0] \* (N+1)

new\_T = [None] \* (N+1)

res = [old\_T]

while True:

old\_T = res[-1]

while True:

new\_T = solve\_equation\_system(old\_T, res[-1], tau, N)

if check(old\_T, new\_T, eps1):

break

old\_T = new\_T

time += tau

res.append(new\_T)

if check(res[-2], res[-1], eps2):

break

return res

if \_\_name\_\_ == "\_\_main\_\_":

tau = 1

N = 100

res = np.array(solve(tau, N))

#3d surface T(x, t)

x = np.linspace(0, l, N+1)

y = np.arange(len(res))

X, Y = np.meshgrid(x, y)

Z = res

plot3d(X, Y, Z)

#Зависимость температуры от координаты стержня

x = np.linspace(0, l, N+1)

for T in res[:10]:

plt.plot(x, T)

plt.xlabel("Длина, см")

plt.ylabel("Температура, K")

plt.grid(True)

plt.show()

#Зависимость температуры от времени

t = np.arange(len(res))

for T in res.transpose()[:10]:

plt.plot(t, T)

plt.xlabel("Время, сек")

plt.ylabel("Температура, K")

plt.grid(True)

plt.show()

**Результат работы**

**1. Представить разностный аналог краевого условия при и его краткий вывод интегро -интерполяционным методом.**

Для получения разностного аналога краевого условия при , надо проинтегрировать на отрезке [xN-1/2, xN] выписанное выше уравнение (1) и учесть (10) и (11)

Обозначим (15)

Тогда (1) можно записать в виде:

(16)

где

Проитегрируем (13):

(14)

Вычисляем интегралы

(17)

Приведем к виду . Получим:

(16)

Для функций c, X, p будет принята простая аппроксимация.

(18)

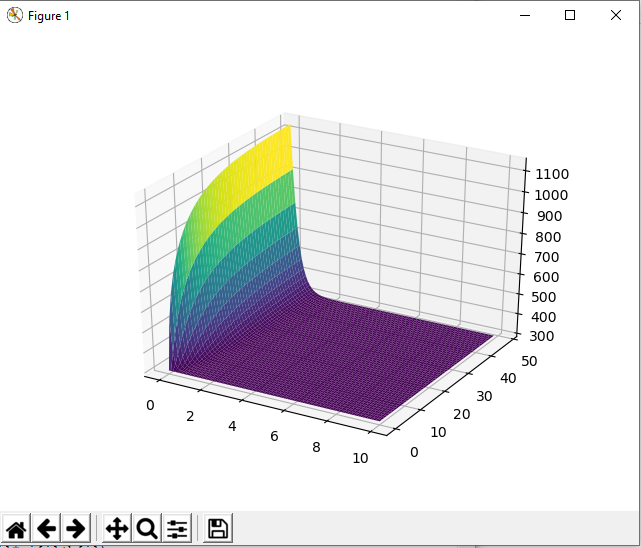
Из (9) и (16) получим .

Получим систему:

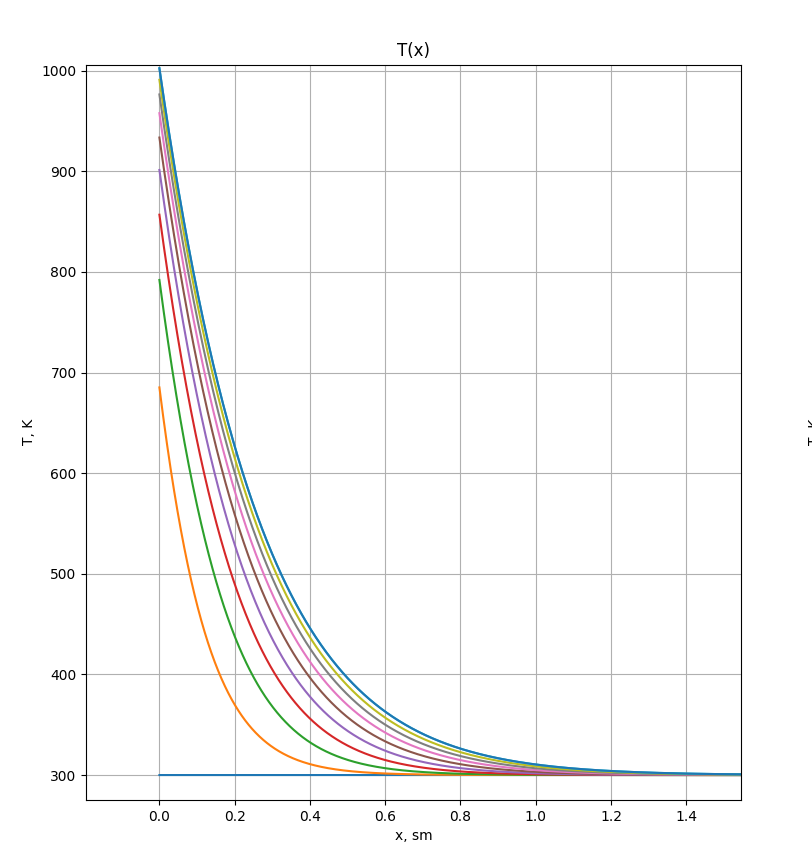
(18)

Эту систему можно решить методом итераций. Пусть i — номер итерации

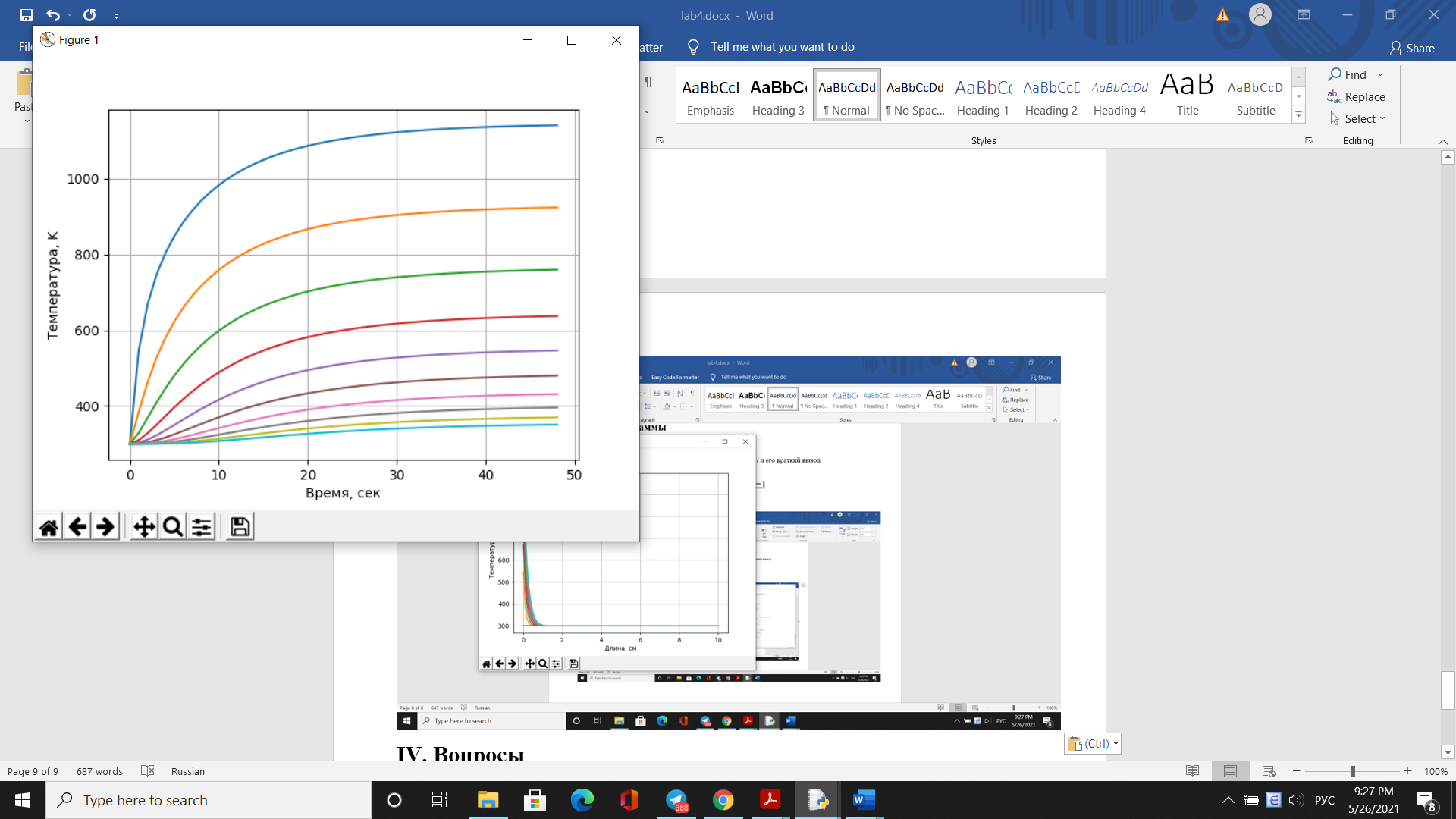
**2. График зависимости температуры :**

****

**3. График зависимости температуры от координаты при нескольких фиксированных значениях времени (аналогично рисунку в лекции) при заданных выше параметрах. Обязательно представить распределение в момент времени, соответствующий установившемуся режиму, когда поле перестает меняться с некоторой точностью (например ), т.е. имеет место выход на стационарный режим. На этой стадии левая часть дифференциального уравнения близка к нулю**



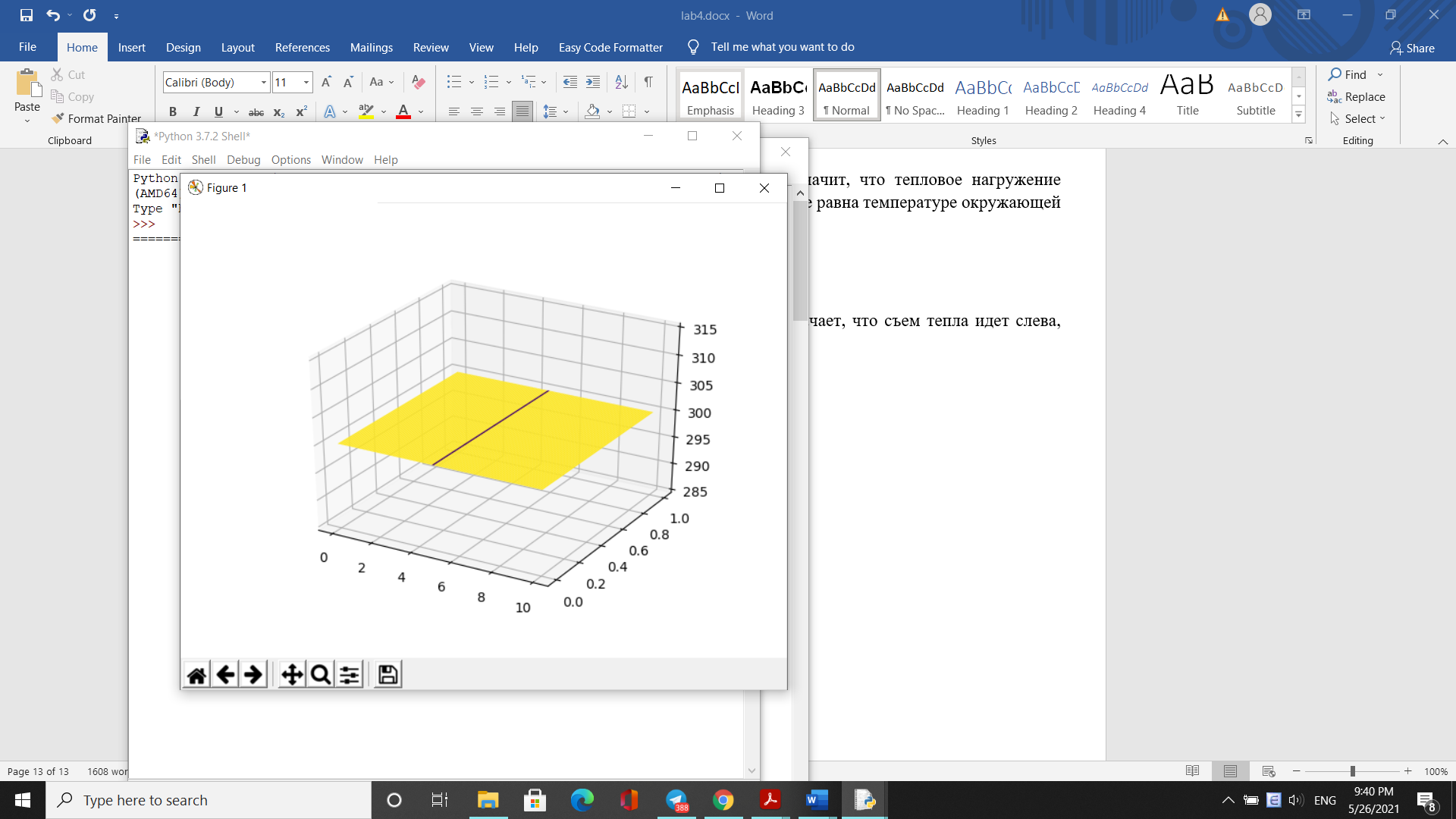
**4. График зависимости при нескольких фиксированных значениях координаты** . Обязательно представить случай n=0, т.е

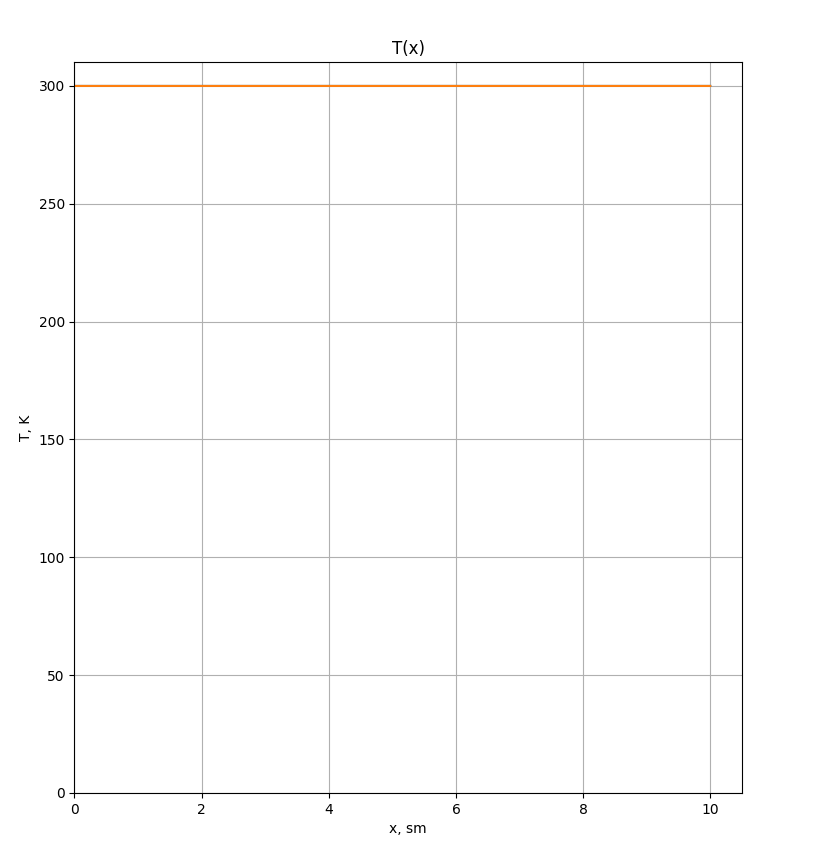


**Вопросы**

**1. Приведите результаты тестирования программы (графики, общие соображения, качественный анализ).**

Если принять F0 =0.Тепловое нагружение отсутствует, причин для нагрева нет, температура стержня должна быть равна температуре окружающей среды T0.





Если (тепловой поток) меньше нуля. Это означает, что съем тепла идет слева, поэтому температура будет увеличиваться от 0 до l

