|  |  |
| --- | --- |
| 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**

**Лабораторная работа № 4**

**По курсу Моделирование**

**Тема: Программно- алгоритмическая реализация моделей на основе дифференциальных уравнений в частных производных с краевыми условиями II и III рода.**

**Студент: Фам Куанг Ань**

**Группа: ИУ7И-66Б**

**Преподаватель: Градов В.М.**

Москва

2021г

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

**I. Аналитическая часть**

Исходные данные.

**1. Задана математическая модель**

Уравнение для функции T(x, t)

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

Краевые условия

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

В обозначениях уравнения лекции

|  |  |  |
| --- | --- | --- |
|  |  | (3) |

**2. Разностная схема с разностным краевым условием** при x = 0 получена в лекции и может быть использована в данной работе. Самостоятельно надо получить интегро -интерполяционным методом разностный аналог краевого условия при x = l , точно так же, как это сделано при x = 0.

Для этого надо проинтегрировать на отрезке выписанное выше уравнение (1) и учесть, что поток , а

**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**

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

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

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

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

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

При произвольной зависимости потока F(t) 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**

|  |  |  |
| --- | --- | --- |
|  |  | (4) |

Где

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

|  |  |  |
| --- | --- | --- |
|  |  | (3) |

**6. Краевые условия**

Обозначим

|  |  |
| --- | --- |
|  | (4) |

**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

**II. Код программы**

1. import numpy as np
2. import math
3. import matplotlib.pyplot as plt
4. from matplotlib import cm
5. from mpl\_toolkits import mplot3d
7. #constants
8. a1 = 0.0134
9. b1 = 1
10. c1 = 4.35e-4
11. m1 = 1
12. a2 = 2.049
13. b2 = 0.563e-3
14. c2 = 0.528e5
15. m2 = 1
17. alpha0 = 0.05
18. alphaN = 0.01
19. l = 10
20. d = alphaN \* l / (alphaN - alpha0)
21. cc = -alpha0 \* d
23. T0 = 300
24. R = 0.5
25. Ft = 50
26. eps1 = 10e-4
27. eps2 = 10e-4
29. def plot2d(x, y, xlabel, ylabel, title):
30. plt.plot(x, y, 'c')
31. plt.xlabel(xlabel)
32. plt.ylabel(ylabel)
33. plt.title(title)
34. plt.grid(True)
35. plt.show()
37. def plot3d(x, y, z):
38. fig = plt.figure()
39. ax = plt.axes(projection='3d')
40. ax.plot\_surface(x, y, z, rstride=1, cstride=1, cmap='viridis', edgecolor='none')
41. #ax.set\_title("")
42. fig.show()
43. plt.show()
45. #ax[n-1]+bx[n]+cx[n+1]=d
46. def tridiagonal\_method(a, b, c, d):
47. n = len(a) - 1;
48. xi = [None for i in range(n+1)]
49. xi[1] = -c[0]/b[0]
50. eta = [None for i in range(n+1)]
51. eta[1] = d[0]/b[0]
52. for i in range(1, n):
53. xi[i+1] = -c[i]/(a[i]\*xi[i]+b[i])
54. eta[i+1] = (d[i]-a[i]\*eta[i])/(a[i]\*xi[i]+b[i])
56. res = [None for i in range(n+1)]
57. res[n] =(d[n]-a[n]\*eta[n])/(a[n]\*xi[n]+b[n])
58. for i in range(n-1, -1, -1):
59. res[i] = xi[i+1]\*res[i+1]+eta[i+1]
60. return res

63. def k(T):
64. return a1 \* (b1 + c1 \* T\*\*m1)
66. def c(T):
67. return a2 + b2 \* T\*\*m2 - c2 / T\*\*2
69. def alpha(x):
70. return cc / (x - d)
72. def p(x):
73. return 2/R \* alpha(x)
75. #f(T) = f(x)
76. def f(x):
77. return 2\*T0/R \* alpha(x)


81. def solve\_equation\_system(old\_T, prev\_T\_by\_time, tau, N):
82. \_a = []
83. \_b = []
84. \_d = []
85. \_f = []
86. h = l / N
88. def cal\_plus\_half(func, start, end):
89. return (func(start) + func(end)) / 2
91. def c\_(n):
92. return c(old\_T[n])
94. def c\_plus(n):
95. return cal\_plus\_half(c, old\_T[n], old\_T[n+1])
97. def chi\_plus(n):
98. return cal\_plus\_half(k, old\_T[n], old\_T[n+1])
100. def p\_plus(n):
101. return (p(n) + p(n+h)) / 2
103. def f\_plus(n):
104. return (f(n) + f(n+h)) / 2
106. def p\_(n):
107. return p(h\*n)
109. def f\_(n):
110. return f(h\*n)
112. #the left boundary condition x = 0
113. \_a.append(0)
114. \_b.append(h/8\*c\_plus(0)
115. + h/4\*c\_(0)
116. + tau/h\*chi\_plus(0)
117. + tau\*h/8\*p\_plus(0)
118. + tau\*h/4\*p(0))
119. \_d.append(h/8\*c\_plus(0)
120. - tau/h\*chi\_plus(0)
121. + tau\*h/8\*p\_plus(0))
122. \_f.append(h/8\*c\_plus(0)\*(prev\_T\_by\_time[0]+prev\_T\_by\_time[1])
123. + h/4\*c\_(0)\*prev\_T\_by\_time[0]
124. + tau\*Ft
125. + tau\*h/4\*(f\_plus(0) + f(0)))
127. #1 <= n <= N-1
128. for i in range(1, N):
129. \_a.append(tau/h\*chi\_plus(i-1))
130. \_d.append(tau/h\*chi\_plus(i))
131. \_b.append(-(\_a[-1] + \_d[-1] + c\_(i)\*h + p\_(i)\*h\*tau))
132. \_f.append(-(f\_(i)\*h\*tau + c\_(i)\*prev\_T\_by\_time[i]\*h))
134. #the right boudary condition x = l
135. \_a.append(h/8\*c\_plus(N-1)
136. - tau/h\*chi\_plus(N-1)
137. + tau\*h/8\*p\_plus(N-1))
138. \_b.append(h/4\*c\_(N)
139. + h/8\*c\_plus(N-1)
140. + tau\*alphaN
141. + tau/h\*chi\_plus(N-1)
142. + tau\*h/4\*p\_(N)
143. + tau\*h/8\*p\_plus(N-1))
144. \_d.append(0)
145. \_f.append(h/4\*c\_(N)\*prev\_T\_by\_time[N]
146. + h/8\*c\_plus(N-1)\*prev\_T\_by\_time[N]
147. + h/8\*c\_plus(N-1)\*prev\_T\_by\_time[N-1]
148. + tau\*alphaN\*T0
149. + tau\*h/4\*(f\_(N)+f\_plus(N-1)))
151. return tridiagonal\_method(\_a, \_b, \_d, \_f)

154. def check(res1, res2, eps):
155. tmp = 0
156. for i in range(len(res2)):
157. tmp = max(tmp, math.fabs((res2[i]-res1[i])/res2[i]))
158. if tmp <= eps:
159. return True
160. else:
161. return False
163. #simple iteration metho
164. def solve(tau, N):
165. h = l / N
166. time = 0
167. old\_T = [T0] \* (N+1)
168. new\_T = [None] \* (N+1)
169. res = [old\_T]
170. while True:
171. old\_T = res[-1]
172. while True:
173. new\_T = solve\_equation\_system(old\_T, res[-1], tau, N)
174. if check(old\_T, new\_T, eps1):
175. break
176. old\_T = new\_T
177. time += tau
178. res.append(new\_T)
179. if check(res[-2], res[-1], eps2):
180. break
181. return res
183. if \_\_name\_\_ == "\_\_main\_\_":
184. tau = 1
185. N = 100
186. res = np.array(solve(tau, N))
188. #3d surface T(x, t)
189. x = np.linspace(0, l, N+1)
190. y = np.arange(len(res))
191. X, Y = np.meshgrid(x, y)
192. Z = res
193. plot3d(X, Y, Z)
195. #Зависимость температуры от координаты стержня
196. x = np.linspace(0, l, N+1)
197. for T in res[:10]:
198. plt.plot(x, T)
199. plt.xlabel("Длина, см")
200. plt.ylabel("Температура, K")
201. plt.grid(True)
202. plt.show()
204. #Зависимость температуры от времени
205. t = np.arange(len(res))
206. for T in res.transpose()[:10]:
207. plt.plot(t, T)
208. plt.xlabel("Время, сек")
209. plt.ylabel("Температура, K")
210. plt.grid(True)
211. plt.show()

**III. Результат работы**

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

См Разностный аналог краевого условия при x = l

**2. График зависимости температуры**

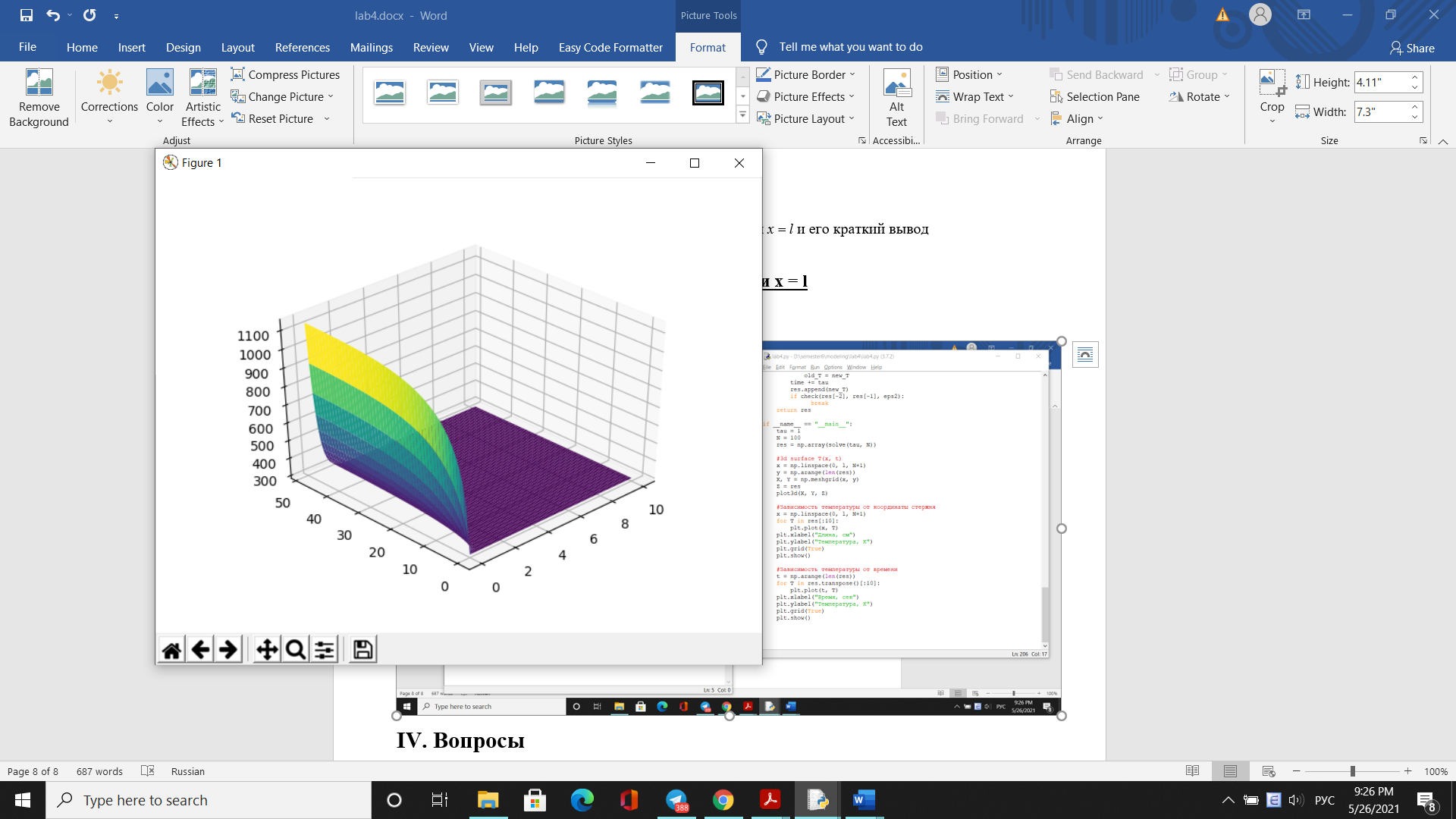


Рис. 1 : Трехмерный график изменения температуры по времени и длине

**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

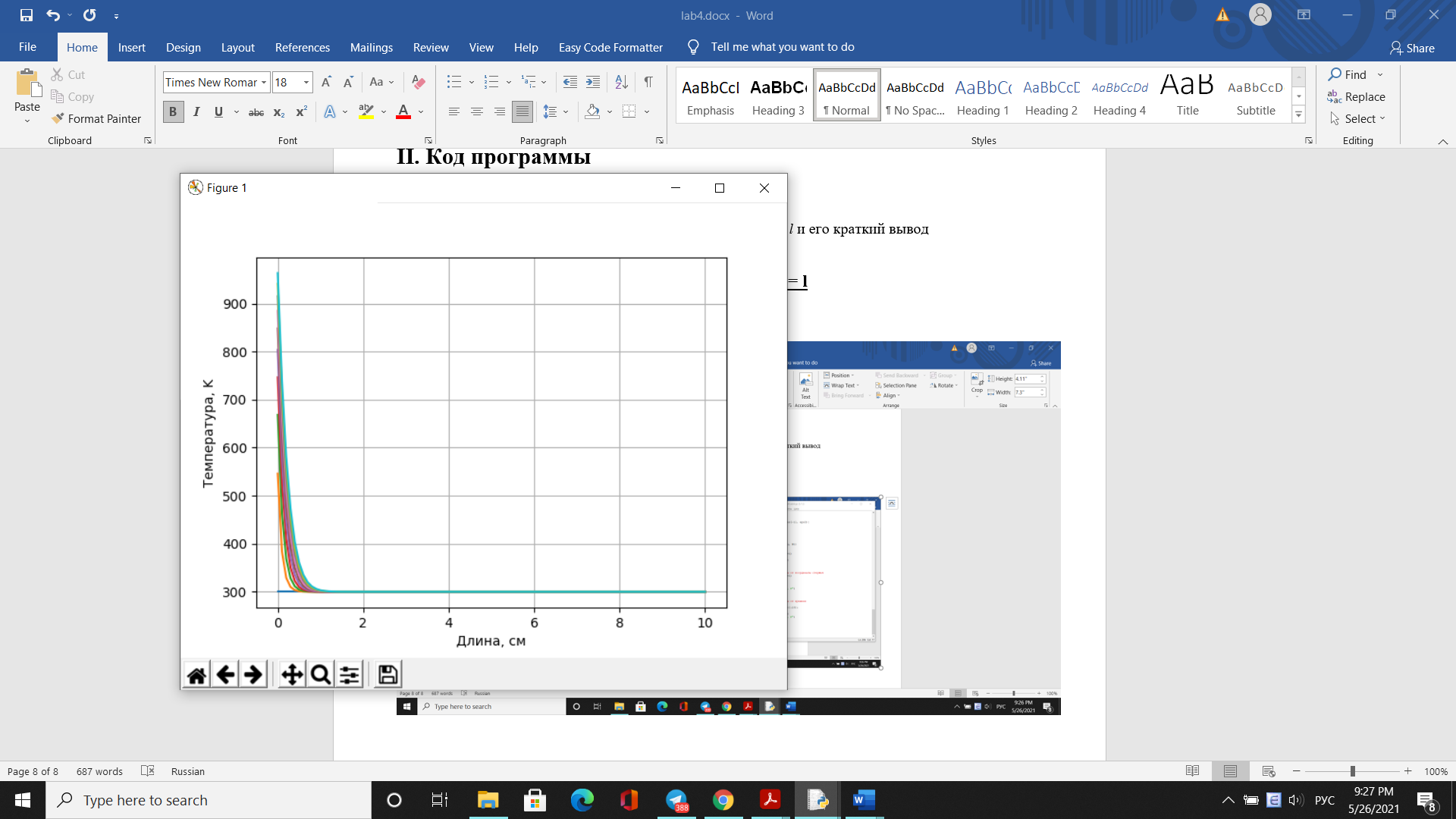


Рис. 2 : Зависимость температуры от координаты стрержня

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

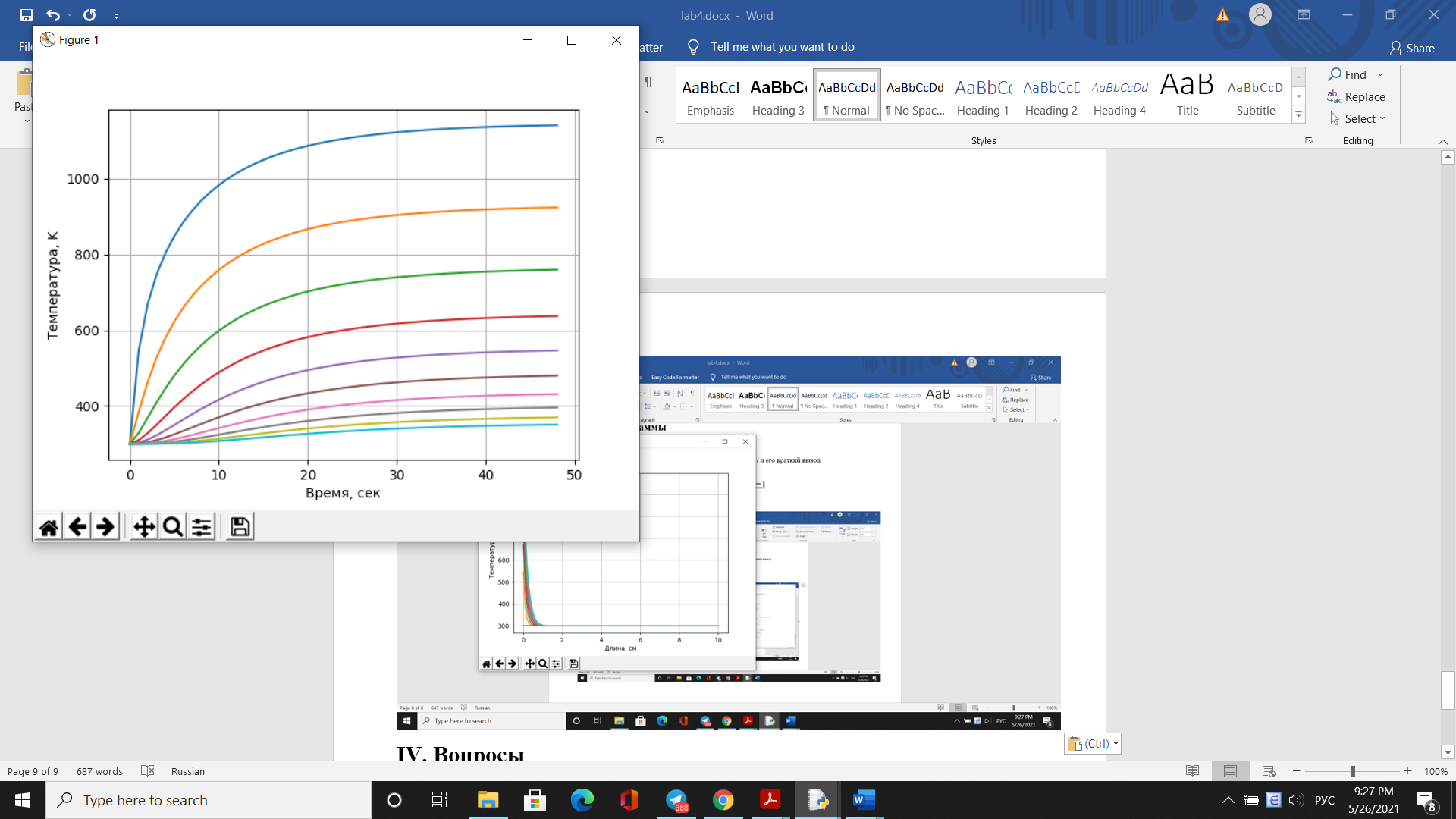


Рис. 3 : Зависимость температуры от координаты времени

Самый верхний синий график – при x = 0

**IV. Вопросы**

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

- Ввести (тепловой поток) равным нулю. Это значит, что тепловое нагружение отсутствует, поэтому температура стержня будет везде равна температуре окружающей среды

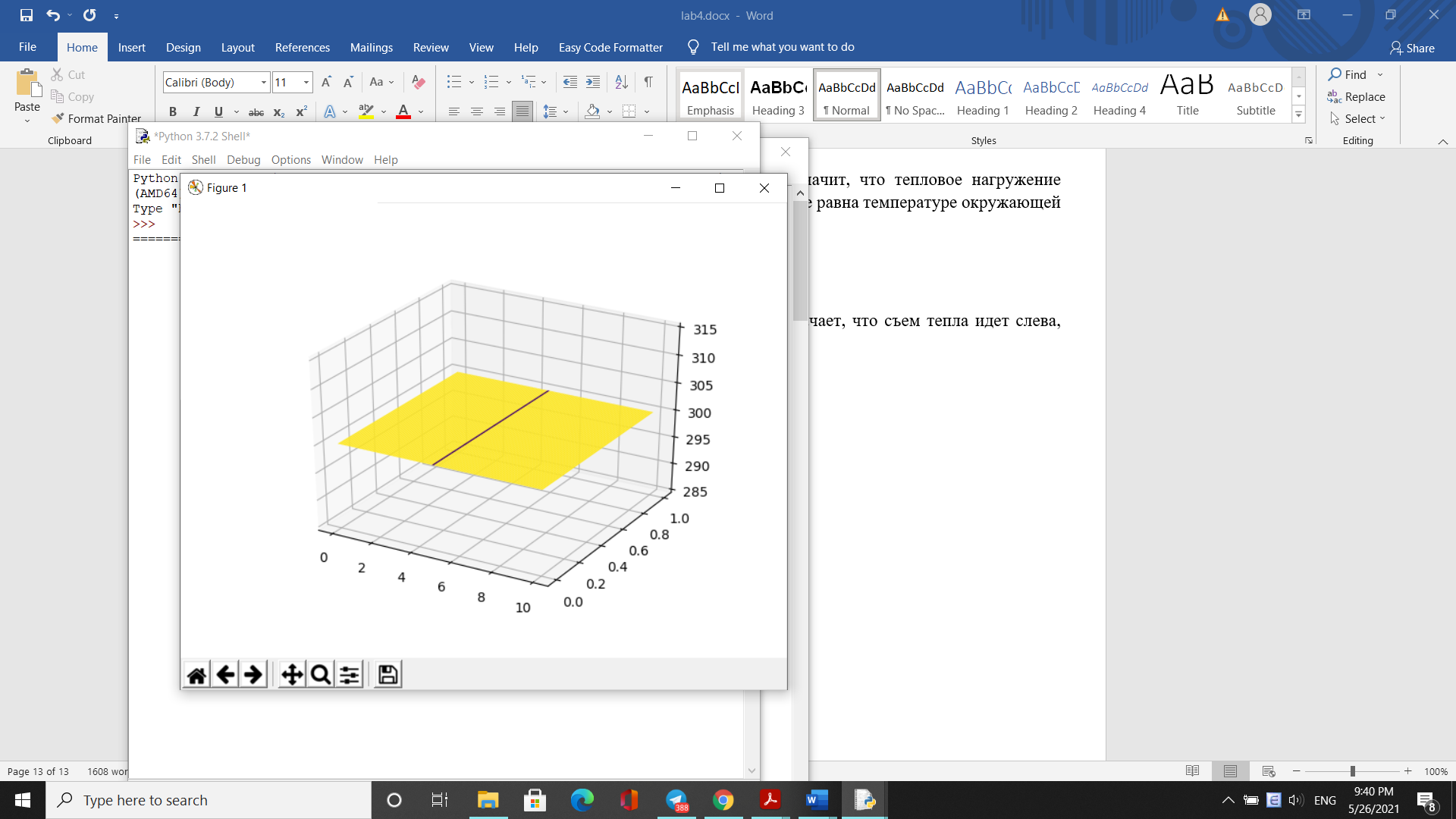


Рис. 4 : T(x, t) при

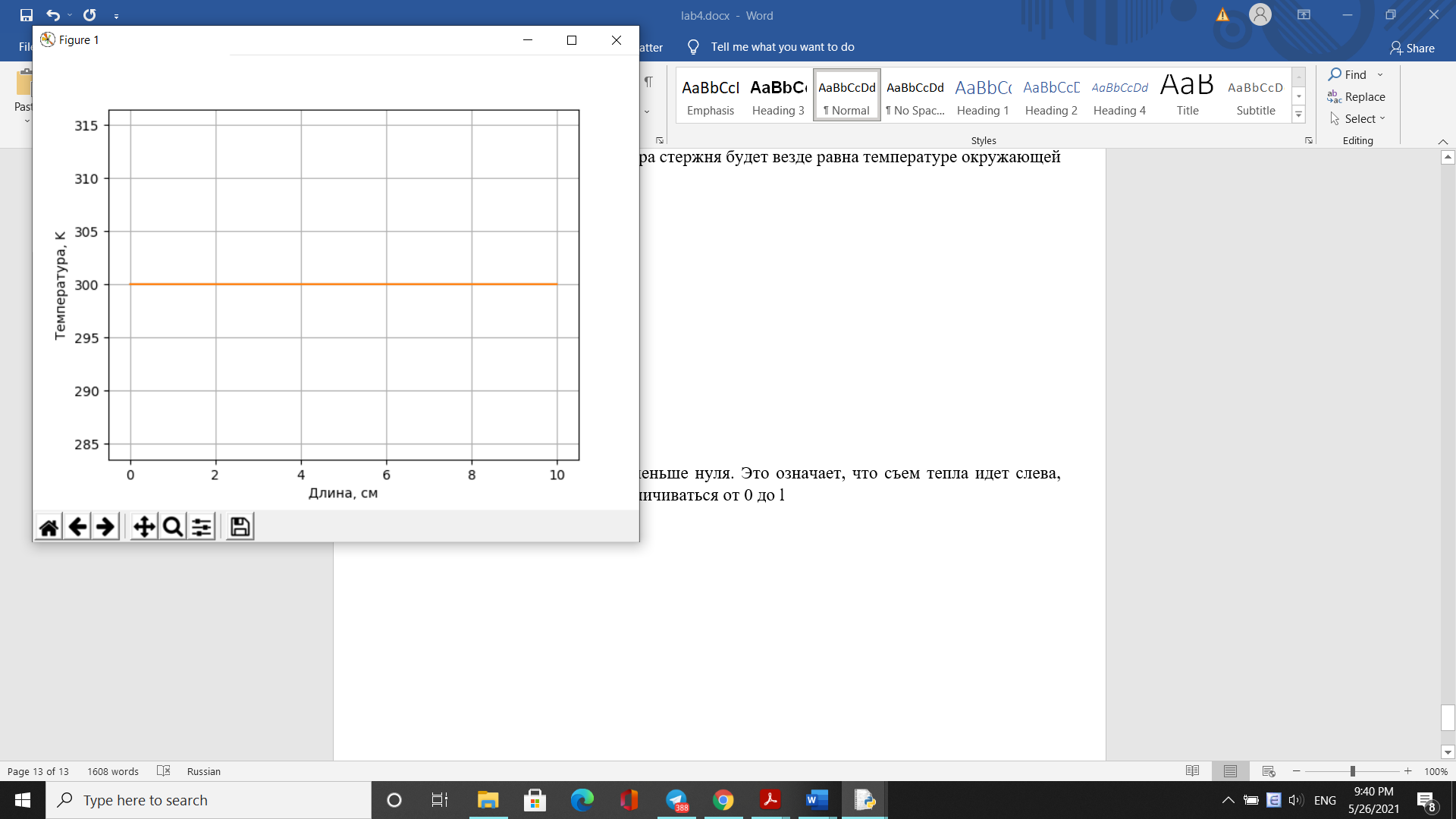


Рис. 5 : Зависимость температуры от координаты стрержня при

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

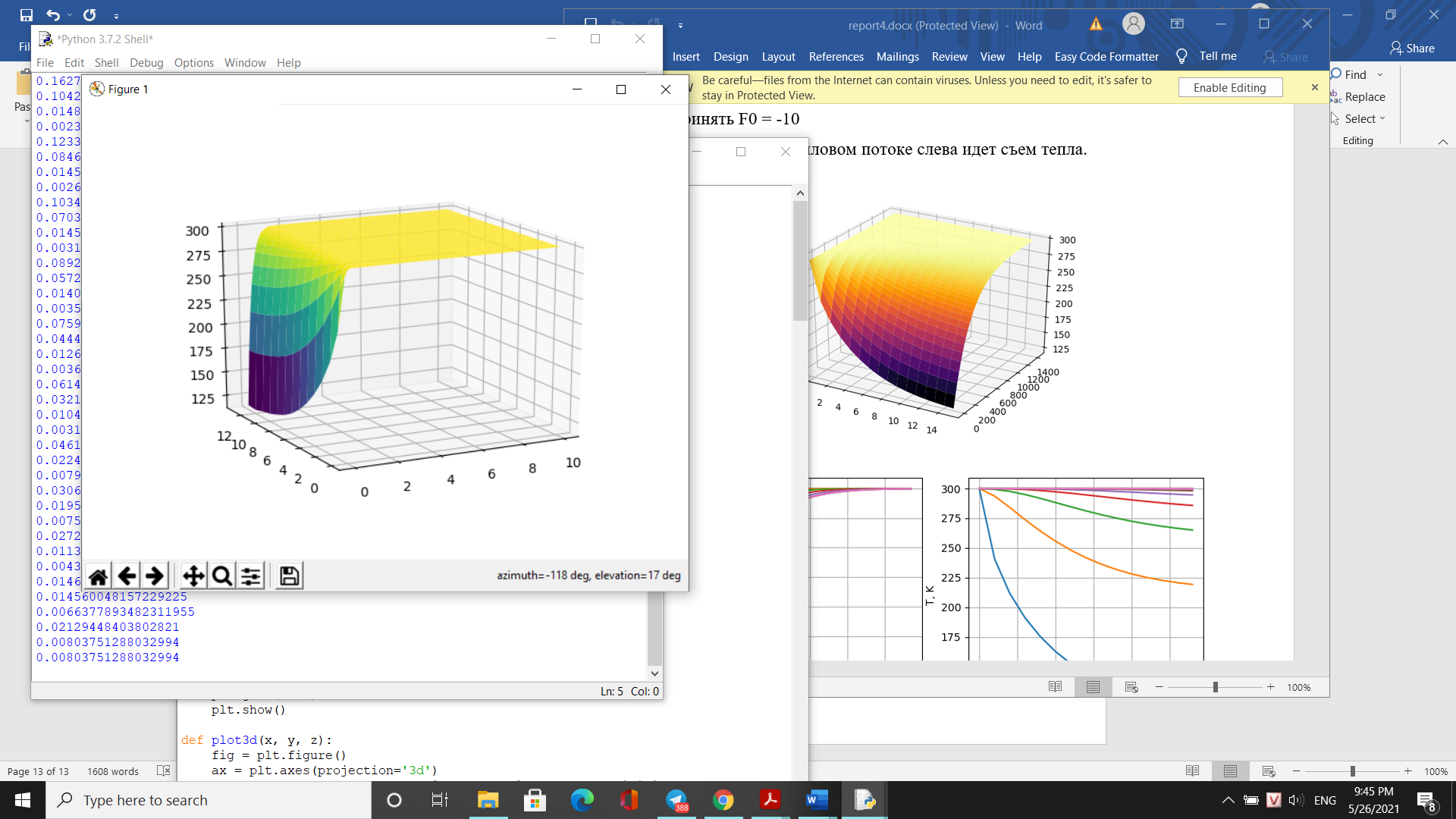


Рис. 6 : T(x, t) при

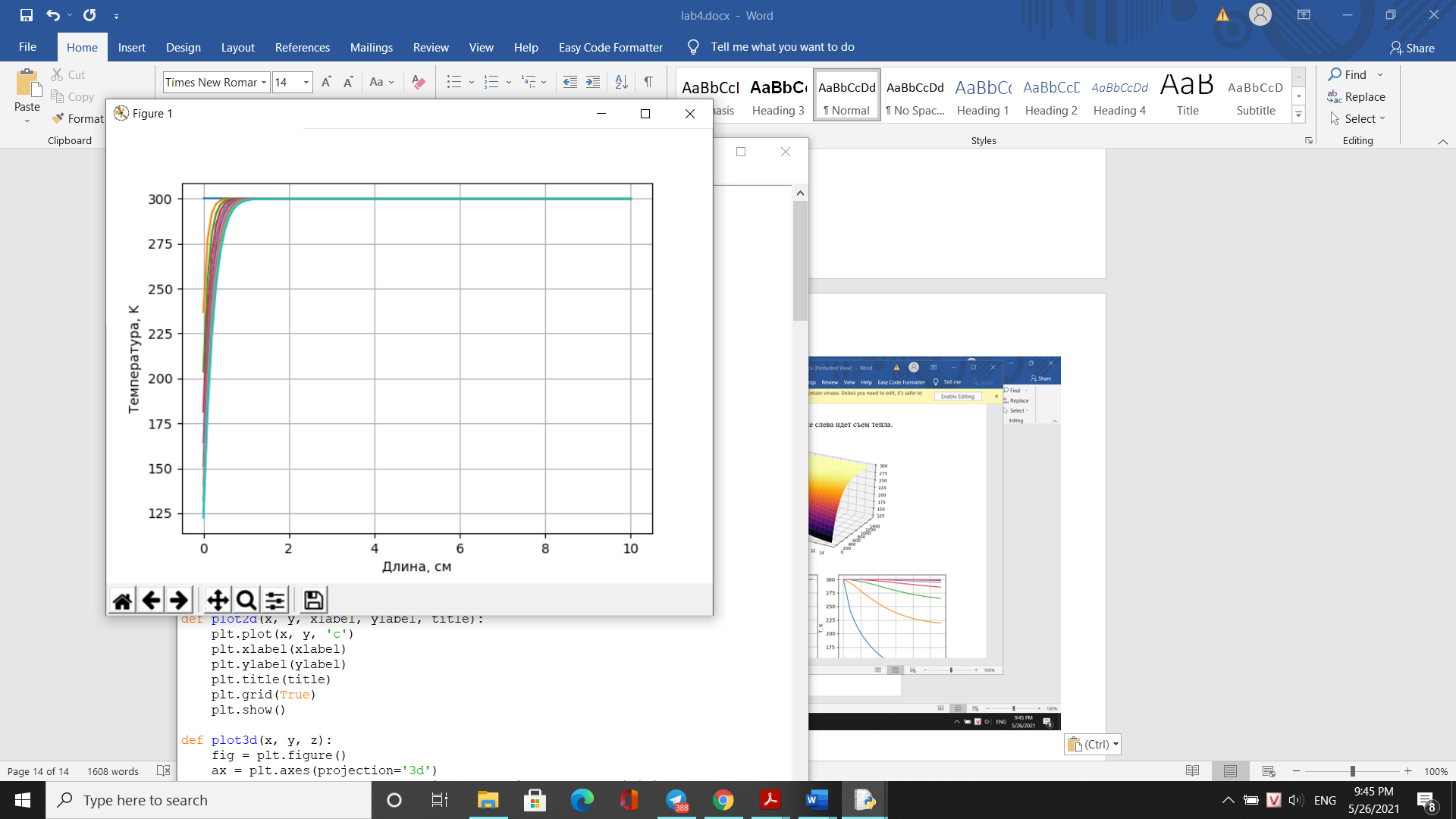


Рис. 7 : Зависимость температуры от координаты стрержня при