

**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»

Лабораторная работа № *4*

**Дисциплина*:*** Моделирование

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

**Студент** Мирзоян С. А.

**Группа** ИУ7-65Б

**Оценка (баллы)**

**Преподаватель** Градов В.М.

Москва2020 г.

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

Исходные данные*.*

1. Уравнение для функции T(x):
2. Краевые условия: 
3. Разностная схема с разностным краевым условием x = 0  
     
    (1)  
     
   Разностный аналог краевого условия при *х*=*l* интегро-интерполяционным методом, (интегрируя на отрезке уравнение (1))

Приведя к общему виду, получаем **(2)**:

Простая аппроксимация:

Если принять *c*(u) = 0, сократить τ, формула (2) перейдёт в формулу для разностного краевого условия при *x*=*l* из предыдущей лабораторной работы.

### Физическое смысл задачи*.*

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

### 2. Свойства материала стержня привязаны к температуре, т.е. теплоемкость и коэффициент теплопроводности *c(T)*, *k(T)* зависят от *T* , тогда как в работе №3 *k(x)* зависит от координаты, а *c* = 0.

### 3. При *x* = 0 цилиндр нагружается тепловым потоком *F(t)* , в общем случае зависящим от времени, а в работе №3 поток был постоянный.

### Если в настоящей работе задать поток постоянным, т.е. *F(t) =const*, то будет происходить формирование температурного поля от начальной температуры *T0* до некоторого установившегося (стационарного) распределения *T(x,t)* . Это поле в дальнейшем с течением времени меняться не будет и должно совпасть с температурным распределением *T(x)* , получаемым в лаб. работе №3, если все параметры задач совпадают, в частности, вместо *k(T)* надо использовать *k(x)* из лаб. работы №3. 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 *T0* .

### 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.

### Листинг.

1. **from** numpy **import** arange
2. **import** matplotlib.pyplot as plt
4. **class** Data:
5. x0 **=** 0
6. l **=** 10           # Длина стержня (cm)
7. R **=** 0.5          # Радиус стержня (cm)
8. Tenv **=** 300       # Температура окружающей среды (K)
9. F0 **=** 100        # Плотность теплового потока (W / (cm^2 \* K))
10. k0 **=** 0.1         # Коэффициент теплопроводности в начале стержня (W / (cm \* K))
11. kN **=** 0.2         # Коэффициент теплопроводности в конце стержня (W / (cm \* K))
12. alpha0 **=** 1e**-**2    # Коэффициент теплоотдачи в начале стержня (W / (cm^2 \* K))
13. alphaN **=** 9e**-**2  # Коэффициент теплоотдачи в конце стержня (W / (cm^2 \* K))
14. h **=** 1e**-**2
15. bk **=** (kN **\*** l) **/** (kN **-** k0)
16. ak **=** **-** k0 **\*** bk
17. b\_alpha **=** (alphaN **\*** l) **/** (alphaN **-** alpha0)
18. a\_alpha **=** **-** alpha0 **\*** b\_alpha

21. @staticmethod
22. **def** k(x):
23. **return** Data.ak **/** (x **-** Data.bk)
25. @staticmethod
26. **def** alpha(x):
27. **return** Data.a\_alpha **/** (x **-** Data.b\_alpha)
29. @staticmethod
30. **def** Xn\_plus\_half(x):
31. **return** (2 **\*** Data.k(x) **\*** Data.k(x **+** Data.h)) **/** \
32. (Data.k(x) **+** Data.k(x **+** Data.h))
34. @staticmethod
35. **def** Xn\_minus\_half(x):
36. **return** (2 **\*** Data.k(x) **\*** Data.k(x **-** Data.h)) **/** \
37. (Data.k(x) **+** Data.k(x **-** Data.h))
39. @staticmethod
40. **def** p(x):
41. **return** 2 **\*** Data.alpha(x) **/** Data.R
43. @staticmethod
44. **def** f(x):
45. **return** 2 **\*** Data.alpha(x) **/** Data.R **\*** Data.Tenv

48. **def** thomas\_algorithm(A, B, C, D, K0, M0, P0, KN, MN, PN):  # Tridiagonal matrix algorithm
49. # Initial values
50. xi **=** [None, **-** M0 **/** K0]
51. eta **=** [None, P0 **/** K0]
53. # Straight running
54. **for** i **in** range(1, len(A)):
55. x **=** C[i] **/** (B[i] **-** A[i] **\*** xi[i])
56. e **=** (D[i] **+** A[i] **\*** eta[i]) **/** (B[i] **-** A[i] **\*** xi[i])
58. xi.append(x)
59. eta.append(e)
61. # print(xi)
62. # print(eta)
64. # Reverse running
65. y **=** [(PN **-** MN **\*** eta[**-**1]) **/** (KN **+** MN **\*** xi[**-**1])]
67. **for** i **in** range(len(A) **-** 2, **-**1, **-**1):
68. y\_i **=** xi[i **+** 1] **\***  y[0] **+** eta[i **+** 1]
70. y.insert(0, y\_i)
72. **return** y




78. **def** left\_boundary\_conditions():
79. X\_half **=** Data.Xn\_plus\_half(Data.x0)
80. p1 **=** Data.p(Data.x0 **+** Data.h)
81. f1 **=** Data.f(Data.x0 **+** Data.h)
83. p0 **=** Data.p(Data.x0)
84. f0 **=** Data.f(Data.x0)
86. p\_half **=** (p0 **+** p1) **/** 2
88. K0 **=** X\_half **+** Data.h **\*** Data.h **\*** p\_half **/** 8 **+** Data.h **\*** Data.h **\*** p0 **/** 4
89. M0 **=** Data.h **\*** Data.h **\*** p\_half **/** 8 **-** X\_half
90. P0 **=** Data.h **\*** Data.F0 **+** Data.h **\*** Data.h **\*** (3 **\*** f0 **+** f1) **/** 4
92. **return** K0, M0, P0

95. **def** right\_\_boundary\_conditions():
96. X\_half **=** Data.Xn\_minus\_half(Data.l)
98. pN **=** Data.p(Data.l)
99. pN1 **=** Data.p(Data.l **-** Data.h)
100. fN **=** Data.f(Data.l)
101. fN1 **=** (2 **\*** Data.alpha(Data.l **-** Data.h)) **/** Data.R **\*** Data.Tenv
103. KN **=** **-** (X\_half **+** Data.alphaN **\*** Data.h) **/** Data.h **-** Data.h **\*** (5 **\*** pN **+** pN1) **/** 16
104. MN **=** X\_half **/** Data.h **-** Data.h **\*** (pN **+** pN1) **/** 16
105. PN **=** **-** Data.alphaN **\*** Data.Tenv **-** Data.h **\*** (3 **\*** fN **+** fN1) **/** 8
107. **return** KN, MN, PN

110. **def** calc\_coefficients():
111. A **=** []
112. B **=** []
113. C **=** []
114. D **=** []
116. **for** i **in** arange(Data.x0, Data.l, Data.h):
117. An **=** Data.Xn\_minus\_half(i) **/** Data.h
118. Cn **=** Data.Xn\_plus\_half(i) **/** Data.h
119. Bn **=** An **+** Cn **+** Data.p(i) **\*** Data.h
120. Dn **=** Data.f(i) **\*** Data.h
122. A.append(An)
123. B.append(Bn)
124. C.append(Cn)
125. D.append(Dn)
127. **return** A, B, C, D

130. **if** \_\_name\_\_ **==** "\_\_main\_\_":
131. a, b, c, d **=** calc\_coefficients()
132. # print(a)
133. # print(b)
134. # print(c)
135. # print(d)
137. k0, m0, p0 **=** left\_boundary\_conditions()
138. # print(k0)
139. # print(m0)
140. # print(p0)
142. kN, mN, pN **=** right\_\_boundary\_conditions()
143. # print(kN)
144. # print(mN)
145. # print(pN)
147. T **=** thomas\_algorithm(a, b, c, d, k0, m0, p0, kN, mN, pN)
148. print(T)
149. x **=** arange(Data.x0, Data.l, Data.h)
151. plt.title('Heating the rod')
152. plt.grid(True)
153. plt.plot(x, T, 'r', linewidth**=**0.5)
154. plt.xlabel("Length (cm)")
155. plt.ylabel("Temperature (K)")
157. plt.savefig("plot.png")
159. plt.show()

### Результат работы программы*.*

, 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,

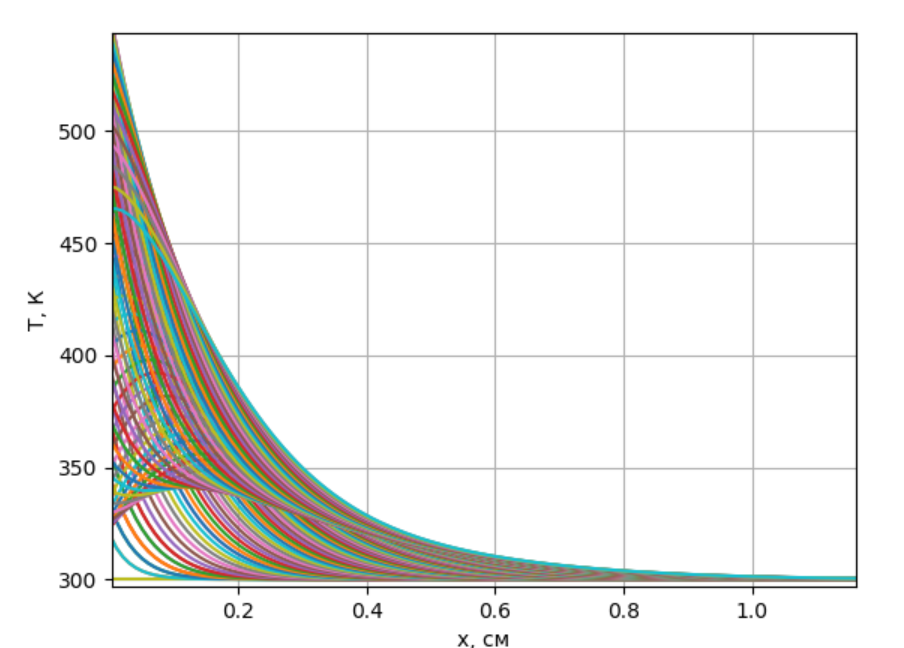
50 Вт/см2 (для отладки принять постоянным). 

Рисунок 1. График зависимости температуры *T* (*x*) от координаты при заданных выше параметрах.

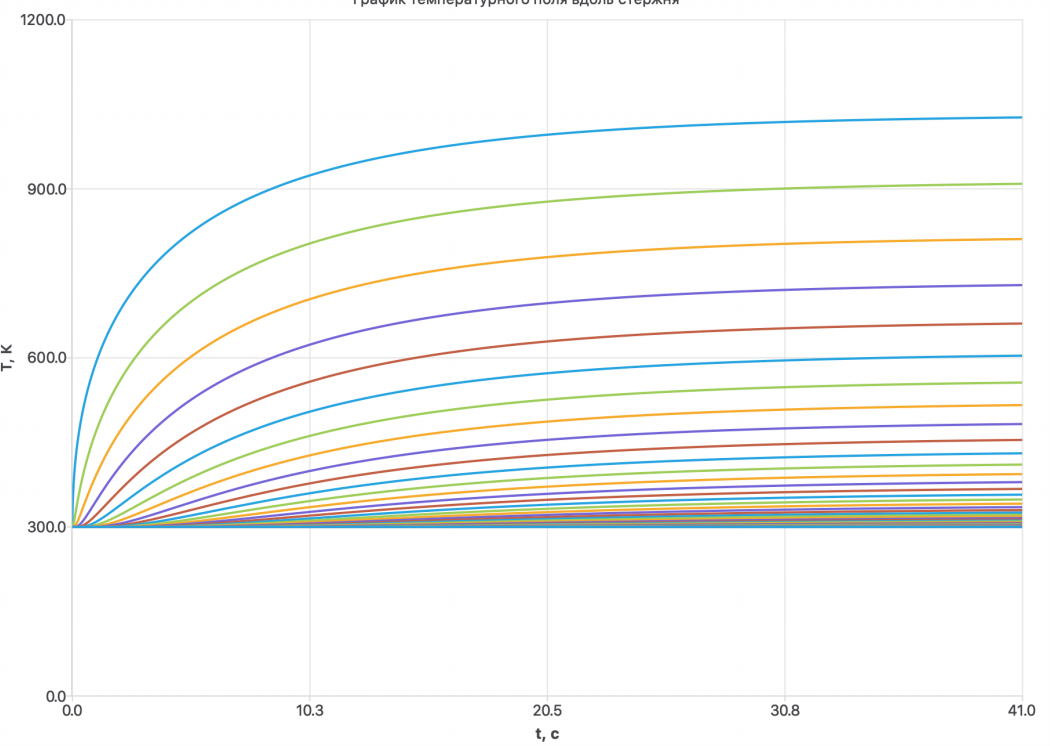


Рисунок 2. График зависимости температуры от времени.

**Ответы на вопросы:**

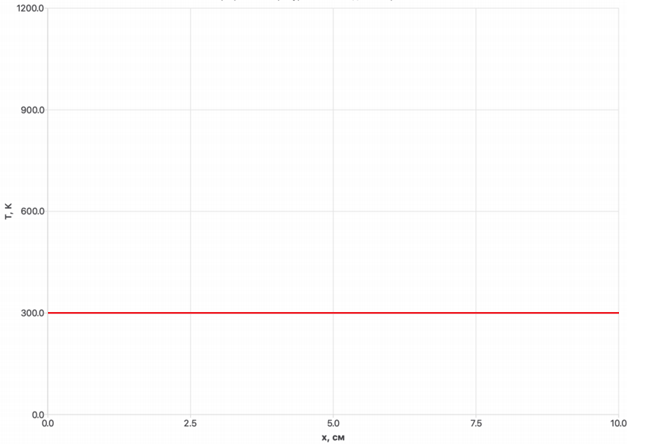
1. **Какие способы тестирования программы можно предложить?**
   1. При F0 = 0   
      

Рисунок 3. График при F = 0

* 1. Можно сравнить графики, получившиеся при выходе на стационарном режиме, с графиком из лабораторной работы №3. Для этого заменяем зависимость коэффициента теплопроводности от Т на зависимость только от координаты (то есть приводим к тому же условию, что и в предыдущей лабораторной

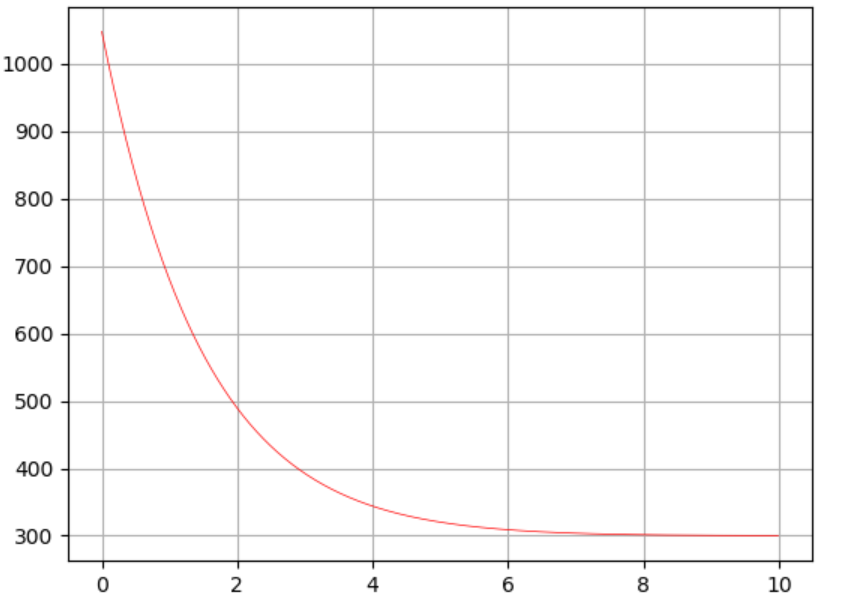
****

Рисунок 4. График из лабораторной работы 4

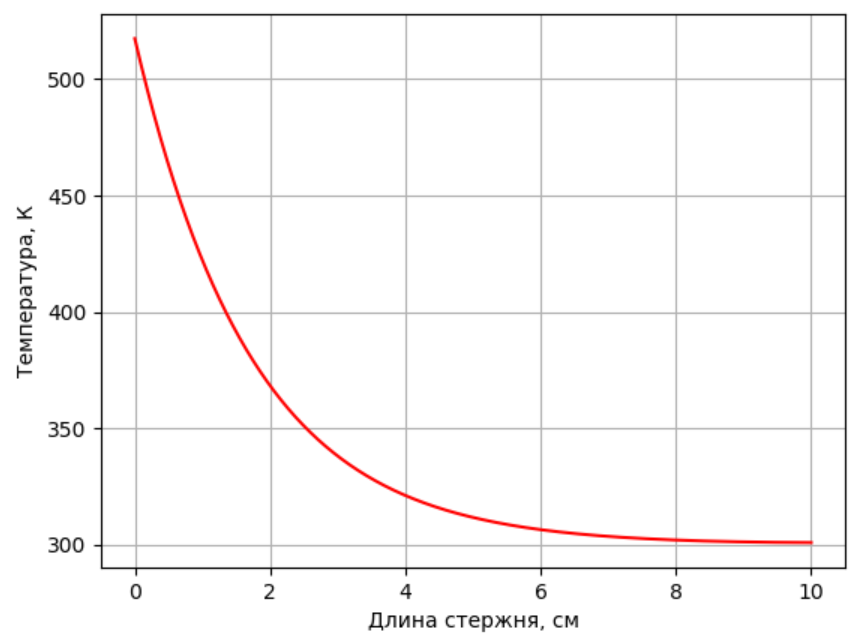
******

Рисунок 5. График из лабораторной работы 3.

* 1. При отрицательном радиусе стержня R<0, должны наблюдаться гармонические колебания.

1. Выполните линеаризацию уравнения (14.8) по Ньютону, полагая для простоты, что все коэффициенты зависят только от одной переменной . Приведите линеаризованный вариант уравнения и опишите алгоритм его решения. Воспользуйтесь процедурой вывода, описанной в лекции №8.  
   

