

**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 № *3*

**Дисциплина*:*** Моделирование

**Тема:** Программно- алгоритмическая реализация моделей на основе ОДУ второго порядка с краевыми условиями II и III рода.

**Студент** Мирзоян С. А.

**Группа** ИУ7-65Б

**Оценка (баллы)**

**Преподаватель** Градов В.М.

Москва. 2020 г.

**Цель работы*:*** Получение навыков разработки алгоритмов решения краевой задачи при реализации моделей, построенных на ОДУ второго порядка.

Исходные данные*.*

1. Уравнение для функции T(x):
2. Краевые условия:
3. Функции заданы своими константами.

### Физическое смысл задачи*.*

Сформулированная математическая модель описывает температурное поле  вдоль цилиндрического стержня радиуса  и длиной , причем  и температуру можно принять постоянной по радиусу цилиндра. Ось  направлена вдоль оси цилиндра и начало координат совпадает с левым торцем стержня. Слева при  цилиндр нагружается тепловым потоком . Стержень обдувается воздухом, температура которого равна . В результате происходит съем тепла с цилиндрической поверхности и поверхности правого торца при . Функции являются, соответственно, коэффициентами теплопроводности материала стержня и теплоотдачи при обдуве.

### Листинг.

1. **import** matplotlib.pyplot as plt
2. **import** numpy as np
4. **def** plot\_maker(masx, masy, xlabel, ylabel):
5. plt.plot(masx, masy, color **=** 'r')
6. plt.xlabel(xlabel)
7. plt.ylabel(ylabel)
8. # plt.legend((name1, name2))
9. plt.grid(True)
10. plt.show()
12. **def** k(x):
13. **return** a**/**(x **-** b)
14. **def** alpha(x):
15. **return** 3**\***x**/**(x **-** d)
16. **def** P(Ax):
17. **return** 2 **\*** Ax **/** R
18. **def** F(Ax):
19. **return** (2 **\*** T0 **\*** Ax)**/**R
21. **def** Xn\_formula(x, h, flag):
22. **if** flag **==** "+":
23. res **=** 2 **\*** k(x) **\*** k(x **+** h) **/** (k(x) **+** k(x **+** h))
24. **if** flag **==** "-":
25. res **=** 2 **\*** k(x) **\*** k(x **-** h) **/** (k(x) **+** k(x **-** h))
26. **return** res
28. **def** An(x, h):
29. res **=** 2 **\*** k(x) **\*** k(x **-** h) **/** (k(x) **+** k(x **-** h))
30. **return** res**/**h
31. **def** Bn(x, h, Ai, Ci):
32. **return** Ai **+** Ci **+** P(x) **\*** h
33. **def** Cn(x, h):
34. res **=** 2 **\*** k(x) **\*** k(x **+** h) **/** (k(x) **+** k(x **+** h))
35. **return** res**/**h
36. **def** Dn(x, h):
37. **return** F(x) **\*** h


41. **def** get\_K0(x0, h):
42. pn\_1\_div\_2 **=** (P(x0) **+** P(x0 **+** h))**/** 2
43. **return** Xn\_formula(x0, h, "+") **+** h**\*\***2 **\*** pn\_1\_div\_2 **/** 8 **+** h**\*\***2 **\***P(x0)**/**4
45. **def** get\_M0(x0, h):
46. pn\_1\_div\_2 **=** (P(x0) **+** P(x0 **+** h))**/** 2
47. **return** **-**Xn\_formula(x0, h, '+') **+** h**\*\***2 **\*** pn\_1\_div\_2 **/** 8
49. **def** get\_P0(x0, h):
50. fn\_1\_div\_2 **=** (F(x0) **+** F(x0 **+** h))**/** 2
51. **return** h **\*** F0 **+** h**\*\***2 **\*** (fn\_1\_div\_2 **+** F(x0)) **/** 4
53. **def** get\_KN(x, h):
54. res **=** 2 **\*** k(x) **\*** k(x **-** h) **/** (k(x) **+** k(x **-** h))
55. **return** **-**P(x)**\***h**/**4 **-** (P(x**-**h) **+** P(x))**\***h**/**16 **-** alpha(x) **-** res**/**h
56. **def** get\_MN(x, h):
57. res **=** 2 **\*** k(x) **\*** k(x **-** h) **/** (k(x) **+** k(x **-** h))
58. **return** res**/**h **-** (P(x**-**h) **+** P(x))**\***h**/**16
59. **def** get\_PN(xn, h):
60. **return** **-**alpha(xn) **\*** T0 **-**  h **\*** (3**\***F(xn) **+** F(xn **-** h))**/**8

63. **def** progon(A, B, C, D, K0, M0, P0, KN, MN, PN):
64. xi **=** [0]
65. eta **=** [0]
66. xi.append(**-**M0**/**K0)
67. eta.append(P0**/**K0)
68. **for** i **in** range(1, len(A)):
69. xi.append(C[i]**/**(B[i] **-** A[i]**\***xi[**-**1]))
70. eta.append((D[i] **+** A[i] **\*** eta[**-**1])**/**(B[i] **-** A[i]**\***xi[**-**2]))
71. y **=** [(PN **-** MN**\***eta[**-**1]) **/** (KN **+** MN **\*** xi[**-**1])]
72. **for** i **in** range(len(A) **-** 2, **-**1, **-**1):
73. y.append(xi[i] **\*** y[**-**1] **+** eta[i])
74. y.reverse()
75. **return** y

78. k0 **=** 0.4
79. kN **=** 0.1
80. alpha0 **=** 0.05
81. alphaN **=** 0.01
82. l **=** 30
83. T0 **=** 300
84. R **=** 0.5
85. F0 **=** 50
86. h **=** 1e**-**3
87. x0 **=** 0
89. ##k0 = float(input("k0: "))
90. ##kN = float(input("kN: "))
91. ##alpha0 = float(input("alpha0: "))
92. ##alphaN = float(input("alphaN: "))
93. ##l = float(input("l: "))
94. ##T0 = float(input("T0: "))
95. ##R = float(input("R: "))
96. ##F0 = float(input("F0: "))
97. ##x0 = float(input("x0: "))
98. ##h = float(input("h: "))

101. b **=** kN **\*** l **/** (kN **-** k0)
102. a **=** **-** k0 **\*** b
103. d **=** alphaN **\*** l **/** (alphaN **-** alpha0)
104. c **=** **-** alpha0 **\*** d
106. A **=** []
107. B **=** []
108. C **=** []
109. D **=** []
110. xmas **=** []
112. **for** x **in** np.arange(x0, l **+** h, h):
113. xmas.append(x)
114. Ai, Ci, Di **=** An(x, h), Cn(x, h), Dn(x, h)
115. Bi **=** Bn(x, h, Ai, Ci)
117. A.append(Ai)
118. B.append(Bi)
119. C.append(Ci)
120. D.append(Di)
122. K0 **=** get\_K0(x0, h)
123. P0 **=** get\_P0(x0, h)
124. M0 **=** get\_M0(x0, h)
126. KN **=** get\_KN(l, h)
127. PN **=** get\_PN(l, h)
128. MN **=** get\_MN(l, h)
130. dots **=** progon(A, B, C, D, K0, M0, P0, KN, MN, PN)
131. plot\_maker(xmas[1:], dots[1:], 'Длина стержня, см', 'Температура, К')

### Результат работы программы*.*

### Тестовые данные для тестирования:

### 0.4 Вт/см К,

### 0.1 Вт/см К,

### 0.05 Вт/см2 К,

### 0.01 Вт/см2 К,

### 10 см,

### 300К,

### 0.5 см,

### 50 Вт/см2.

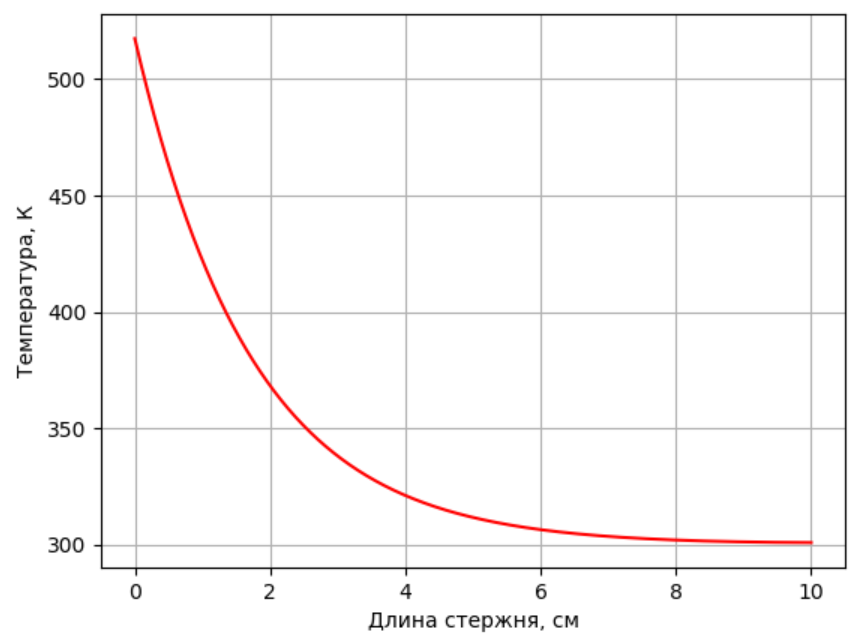
******

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

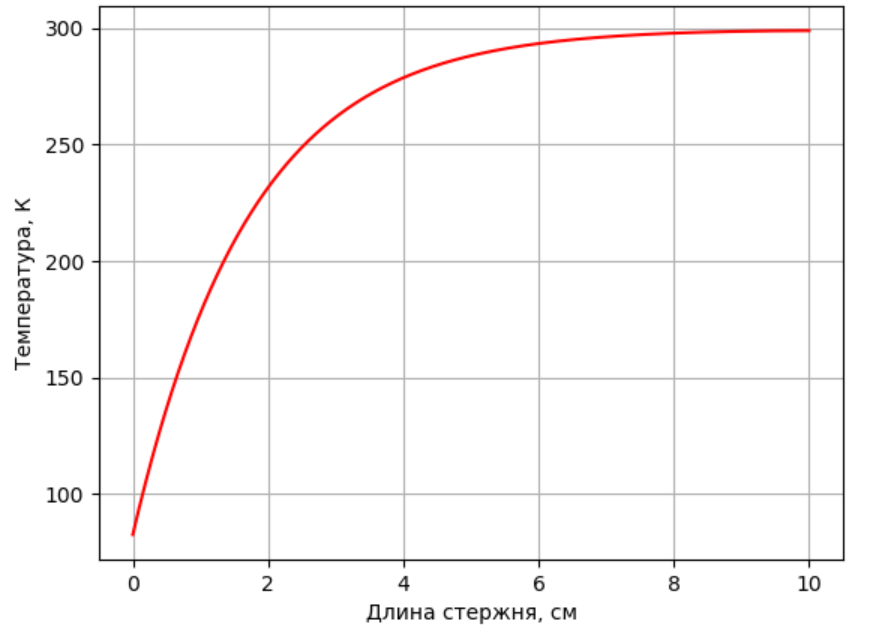


Рисунок 2. при *F*0 = -50 Вт/см2.

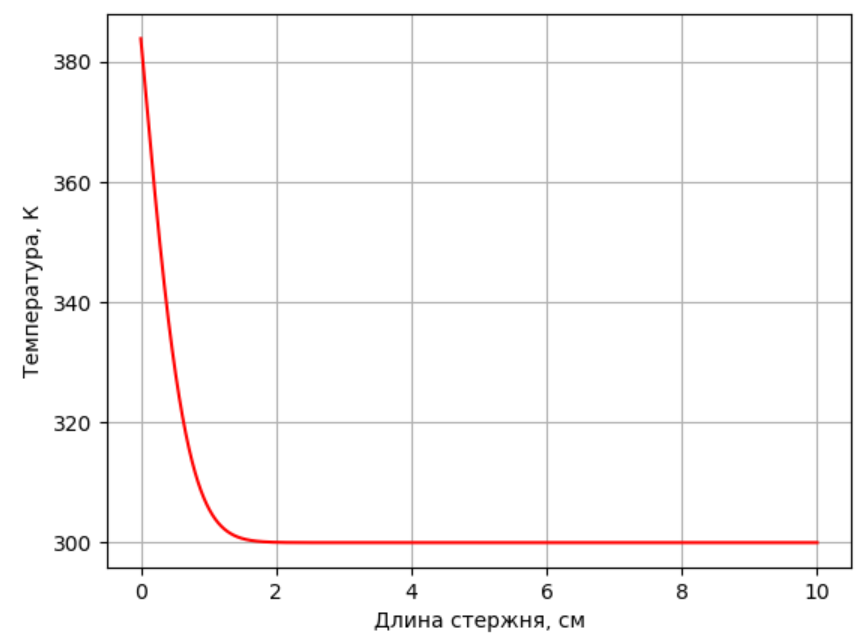


Рисунок 3. График зависимости *T* (*x*) при увеличенных значениях *α*(*x*) в 3 раза.

При увеличении теплосъема и неизменном потоке *F*0 уровень температур

*T* (*x*) снижается, а градиент увеличивается (при сравнении рисунков 1 и 3).

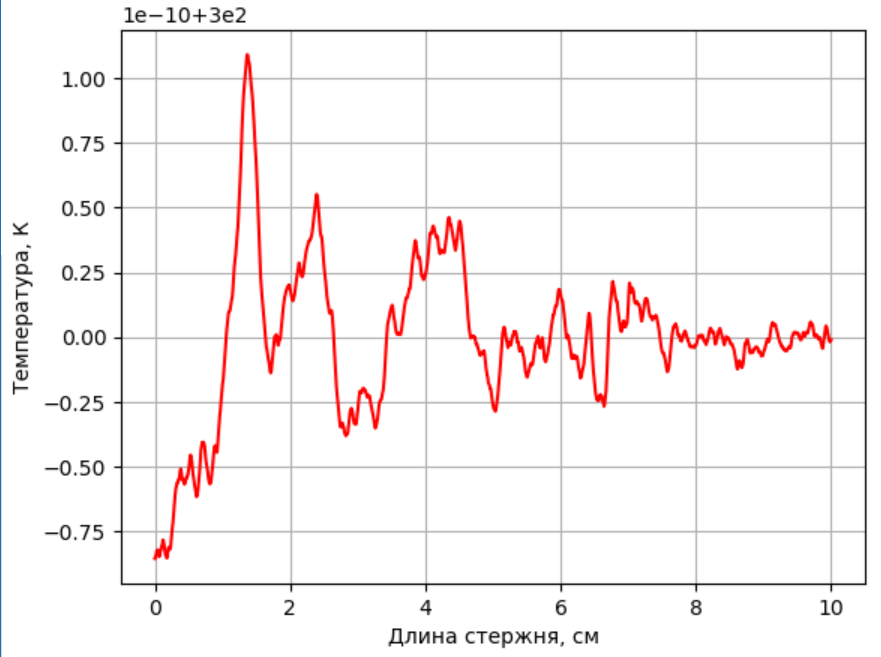


Рисунок 4. График зависимости *T* (*x*) при *F*0 = 0 Вт/см2.

На рисунке 4 можно наблюдать, что, в отсутствии теплового нагружения, температура стержня равна окружающей температуре, погрешность определяется приближенным характером вычислений.

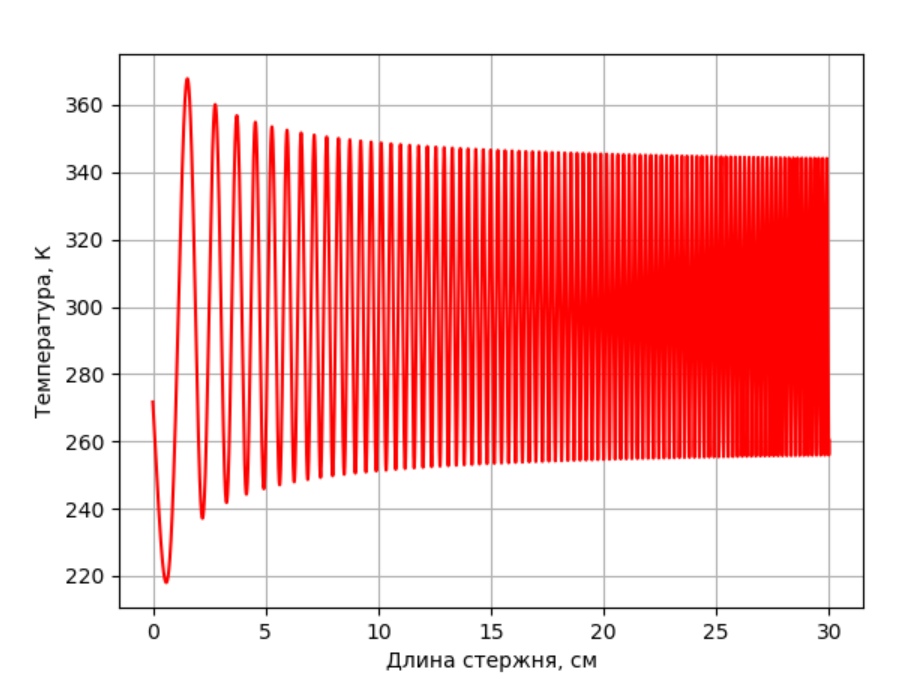
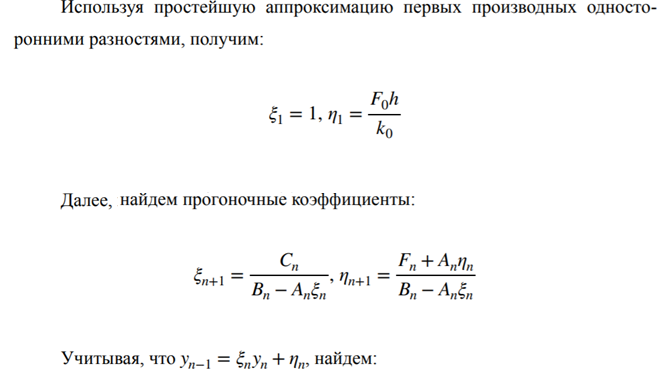


Рисунок 5. Гармонические колебания при R < 0 см\ и l = 30 см.

**Ответы на вопросы:**

1. **Какие способы тестирования программы можно предложить?**
   1. При F0 = 0 T(x) = Т0 , где – погрешность
   2. Должна быть положительная производная функции Т(х) при F0 < 0
   3. При отрицательном радиусе стержня R<0, должны наблюдаться гармонические колебания.
2. **Получите простейший разностный аналог нелинейного краевого условия при** :Разностная аппроксимация краевого условия:
3. **Опишите алгоритм применения метода прогонки, если при  краевое условие линейное (как в настоящей работе), а при , как в п.2  
   **

****

1. **Опишите алгоритм определения единственного значения сеточной функции  в одной заданной точке . Использовать встречную прогонку, т.е. комбинацию правой и левой прогонок (лекция №8). Краевые условия линейные.**