

**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 ОДУ второго порядка с краевыми условиями II и III рода.

**Студент** Юмаев А. Р.

**Группа** ИУ7-65Б

**Оценка (баллы)**

**Преподаватель** Градов В.М.

# 1. Теоретический раздел

Задана математическая модель

Уравнение для функции (1)

Краевые условия

В обозначениях уравнения 14.1 лекции №14

Функция задана уравнением:

Константы c и d из условий ,

Разностная схема

Разностные аналоги краевых условий при (получены в Лекции №14)

Получим интегро-интерполяционным методом разносный аналог краевого условия при

Для этого обозначим и учтем то, что поток

Проинтегрировав уравнение (1) на отрезке получим:

Из полученных краевых условий находим коэффициенты .

Данная система решаемся методом итераций. Обозначим текущую итерацию s, а предыдущую .

Значения параметров

*,*

*,*

*,*

*,*

*,*

*,*

*,*

*,*

*,*

*,*

*,*

*,*

*,*

*,*

*,*

*,*

*.*

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 оте математическая модель описывает нестационарное температурное поле 𝑇(𝑥, 𝑡). Это поле в дальнейшем с течением времени меняться не будет и должно совпасть с температурным распределением 𝑇(𝑥) математическая модель описывает нестационарное температурное поле 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. 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 отслеживать поток.

# 2. Листинг

|  |
| --- |
| # -\*- coding: utf-8 -\*-  import numpy  import matplotlib.pyplot as plt  from math import fabs  def k(T\_num):  return a1 \* (b1 + c1 \* (T\_num\*\*m1))  def c(T\_num):  return a2 + b2 \* T\_num\*\*m2 - c2/(T\_num\*\*2)  def alpha(x):  return c\_coef / (x - d)  def p(x):  return 2/R \* alpha(x)  def f(x):  return 2\*T0/R \* alpha(x)  def An(T\_num, h):  k\_minus = (k(T\_num - tao) + k(T\_num)) / 2  return tao/h \* k\_minus  def Bn(T\_num, Ai, Di, xi, h):  return Ai + Di + c(T\_num)\*h + p(xi)\*h\*tao  def Dn(T\_num, h):  k\_plus = (k(T\_num + tao) + k(T\_num)) / 2  return tao/h \* k\_plus  def Fn(T\_num, xi, h):  return f(xi) \* h \* tao + c(T\_num) \* T\_num \* h  def getK0(T\_start, h):  k\_plus = (k(T\_start + tao) + k(T\_start)) / 2  c0\_half = (c(T\_start) + c(T\_start + tao)) / 2  pn\_half = p(h/2)  return h/8 \* c0\_half + h/4 \* c(T\_start) + tao/h \* k\_plus + tao\*h/8 \* pn\_half + tao\*h/4 \* p(x0)  def getM0(T\_start, h):  k\_plus = (k(T\_start + tao) + k(T\_start)) / 2  c0\_half = (c(T\_start) + c(T\_start + tao)) / 2  pn\_half = p(h/2)  return h/8 \* c0\_half - tao/h \* k\_plus + tao \* h/8 \* pn\_half  def getP0(T\_start, h):  c0\_half = (c(T\_start) + c(T\_start + tao)) / 2  return h/8 \* c0\_half \* (T\_old[0] + T\_old[1]) + h/4 \* c(T\_start) \* T\_start \  + F0 \* tao \  + tao \* h/8 \* (3 \* f(x0) + f(h))  def getKN(T\_end, h):  k\_minus = (k(T\_end - tao) + k(T\_end)) / 2  cn\_half = (c(T\_end) + c(T\_end - tao)) / 2  pn\_half = p(l - h/2)  return h/4 \* c(T\_end) \  + h/8 \* cn\_half \  + tao\*h/4 \* p(l) \  + tao\*h/8 \* pn\_half \  + tao \* alphaN \  + tao/h \* k\_minus  def getMN(T\_end, h):  k\_minus = (k(T\_end - tao) + k(T\_end)) / 2  cn\_half = (c(T\_end) + c(T\_end - tao)) / 2  pn\_half = p(l - h/2)  return h/8 \* cn\_half \  + tao\*h/8 \* pn\_half \  - tao/h \* k\_minus  def getPN(T\_end, h):  cn\_half = (c(T\_end) + c(T\_end - tao)) / 2  fn\_half = f(l - h / 2)  return h/4 \* c(T\_end) \* T\_old[-1] \  + alphaN \* T0 \* tao \  + h/4 \* tao \* (f(l) + fn\_half) \  + h/8 \* cn\_half \* (T\_old[-1] + T\_old[-2])  def progonka(A, B, C, D, K0, M0, P0, KN, MN, PN):  xi = [0]  eta = [0]  xi.append(-M0 / K0)  eta.append(P0 / K0)  for i in range(1, len(A)):  xi.append(C[i] / (B[i] - A[i] \* xi[-1]))  eta.append((D[i] + A[i] \* eta[-1]) / (B[i] - A[i] \* xi[-2]))  y = [(PN - MN \* eta[-1]) / (KN + MN \* xi[-1])]  for i in range(len(A) - 2, -1, -1):  y.append(xi[i] \* y[-1] + eta[i])  y.reverse()  return y  def do\_plot(masx, masy, xlabel, ylabel):  plt.plot(masx, masy)  plt.xlabel(xlabel)  plt.ylabel(ylabel)  plt.grid(True)  def calc\_changes(a, b):  lb = len(b)  la = len(a)  if lb > la:  a, b = b, a  la = len(a)  diff = []  for i in range(la):  if i < lb:  diff.append(fabs(b[i] - a[i]))  else:  diff.append(a[i])    return diff  if \_\_name\_\_ == "\_\_main\_\_":  a1 = 0.0134  b1 = 1  c1 = 4.35 \* 1e-4  m1 = 1  a2 = 2.049  b2 = 0.563 \* 1e-3  c2 = 0.528 \* 1e+5  m2 = 1  alpha0 = 0.05  alphaN = 0.01  l = 10  T0 = 300  R = 0.5  F0 = 50  x0 = 0  h = 1e-2  tao = 1  d = alphaN \* l / (alphaN - alpha0)  c\_coef = - alpha0 \* d  T = [T0 for x in numpy.arange(x0, l + h, h)]  time = 0  mas\_x = [x for x in numpy.arange(x0, l + h, h)]  mas\_t = [0]  do\_plot(mas\_x[1:], T[1:], 'Length (cm)', 'Temperature (K)')  T\_global = [T[:]]  A = []  D = []  B = []  F = []  k\_i = 0  while True:  T\_old = T[:]  A.clear()  B.clear()  D.clear()  F.clear()  i = 0  for x in numpy.arange(x0, l + h, h):  Ai = An(T\_old[i], h)  Di = Dn(T\_old[i], h)  Bi = Bn(T\_old[i], Ai, Di, x, h)  Fi = Fn(T\_old[i], x, h)  A.append(Ai)  D.append(Di)  B.append(Bi)  F.append(Fi)  i += 1  K0 = getK0(T\_old[0], h)  M0 = getM0(T\_old[0], h)  P0 = getP0(T\_old[0], h)  KN = getKN(T\_old[-1], h)  MN = getMN(T\_old[-1], h)  PN = getPN(T\_old[-1], h)  T = progonka(A, B, D, F, K0, M0, P0, KN, MN, PN)  T[0] = T[1]  diff = calc\_changes(T, T\_old)  maxDiff = max(diff)  if fabs(maxDiff / T[diff.index(maxDiff)]) < 1e-4:  T\_global.append(T[:])  time += tao  mas\_t.append(time)  do\_plot(mas\_x[1:], T[1:], 'Length (cm)', 'Temperature (K)')  break  if k\_i % 10 == 0:  do\_plot(mas\_x[1:], T[1:], 'Length (cm)', 'Temperature (K)')  T\_global.append(T[:])  time += tao  mas\_t.append(time)  k\_i += 1  plt.show()  matrix = []  for i in range(len(T\_global[0])):  matrix.append([])  for i in range(len(T\_global)):  for j in range(len(T\_global[i])):  matrix[j].append(T\_global[i][j])  for i in range(len(matrix)):  if (i % 10 == 0):  do\_plot(mas\_t, matrix[i], 'Time (sec)', 'Temperature (K)')  plt.show() |

# 3. Результаты работы программы

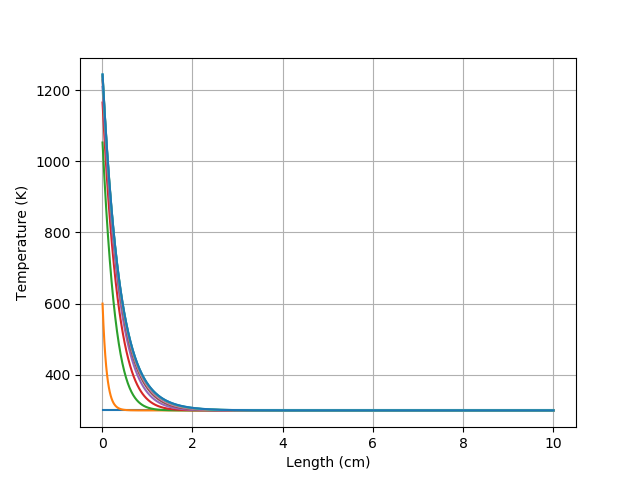


Рисунок 1. График зависимости T(x, t\_m) от координаты x при нескольких фиксированных значениях времени t\_m (верхний график построен при установившемся распределении T(x, t)).

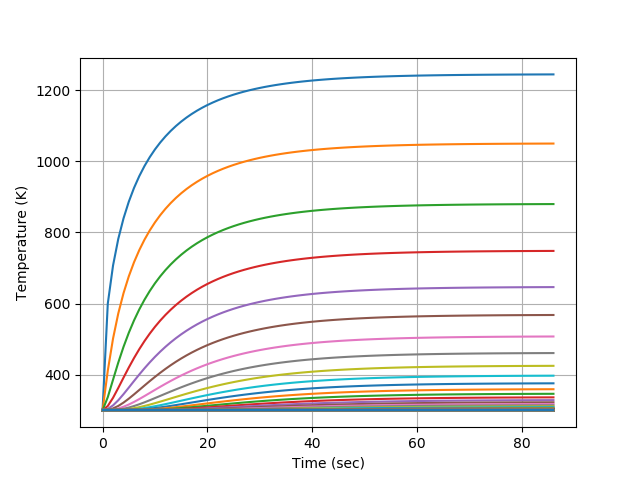


Рисунок 2. График зависимости T(x\_n, t) при нескольких фиксированных значениях координаты x\_n.

# 4. Ответы на вопросы

1. **Результаты тестирования программы**

Если из полученных разностных аналогов краевых условий обнулить 𝑐(𝑢), принять 𝜏 равным 1 и заменить зависимость коэффициента теплопроводности от 𝑇 на зависимость от координаты на стержне 𝑥, то можно получить график (Рис. 3), соответствующий полученному графику в 3 лабораторной работе (Рис. 4).

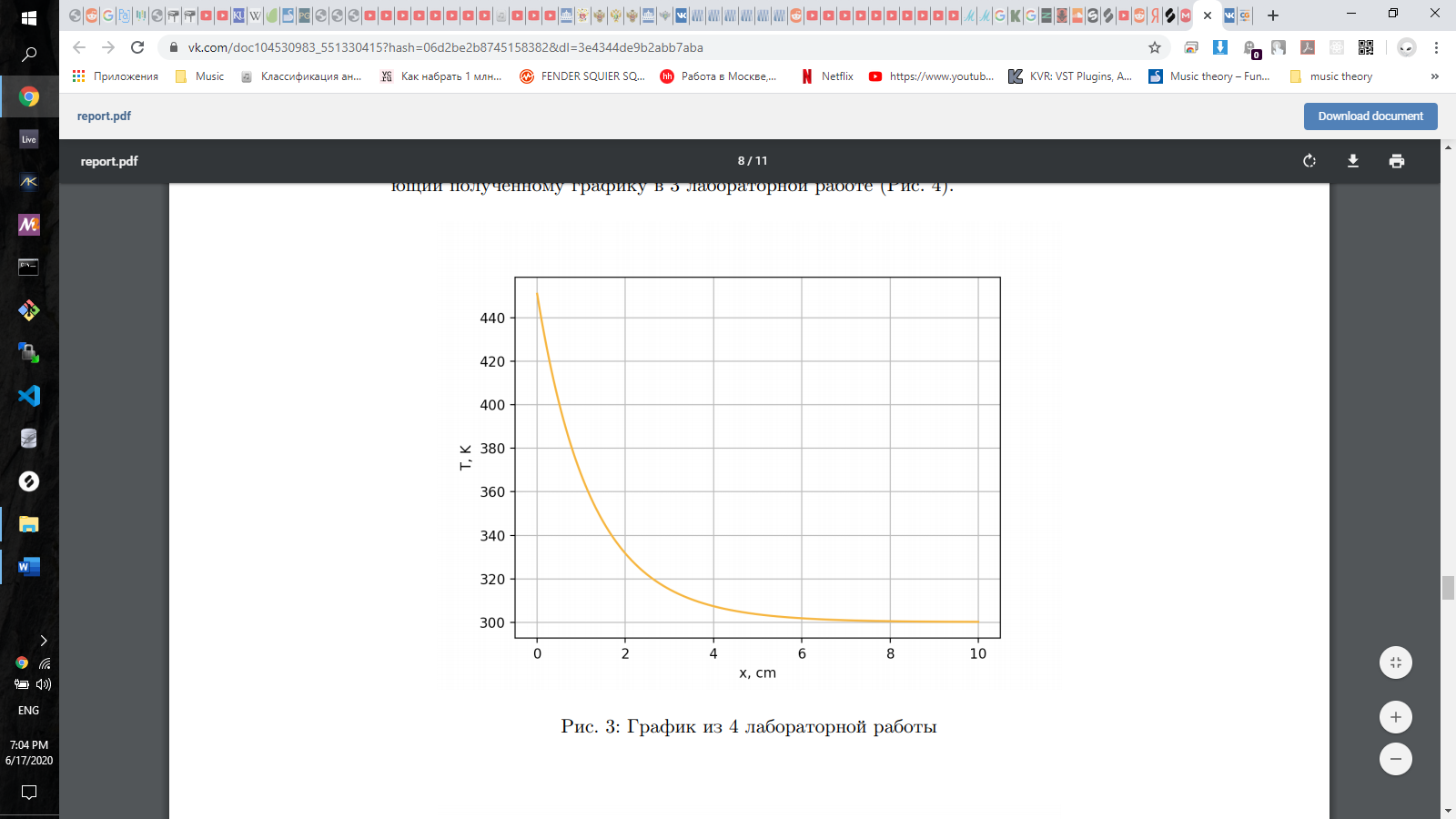


Рисунок 3. График зависимости T от x

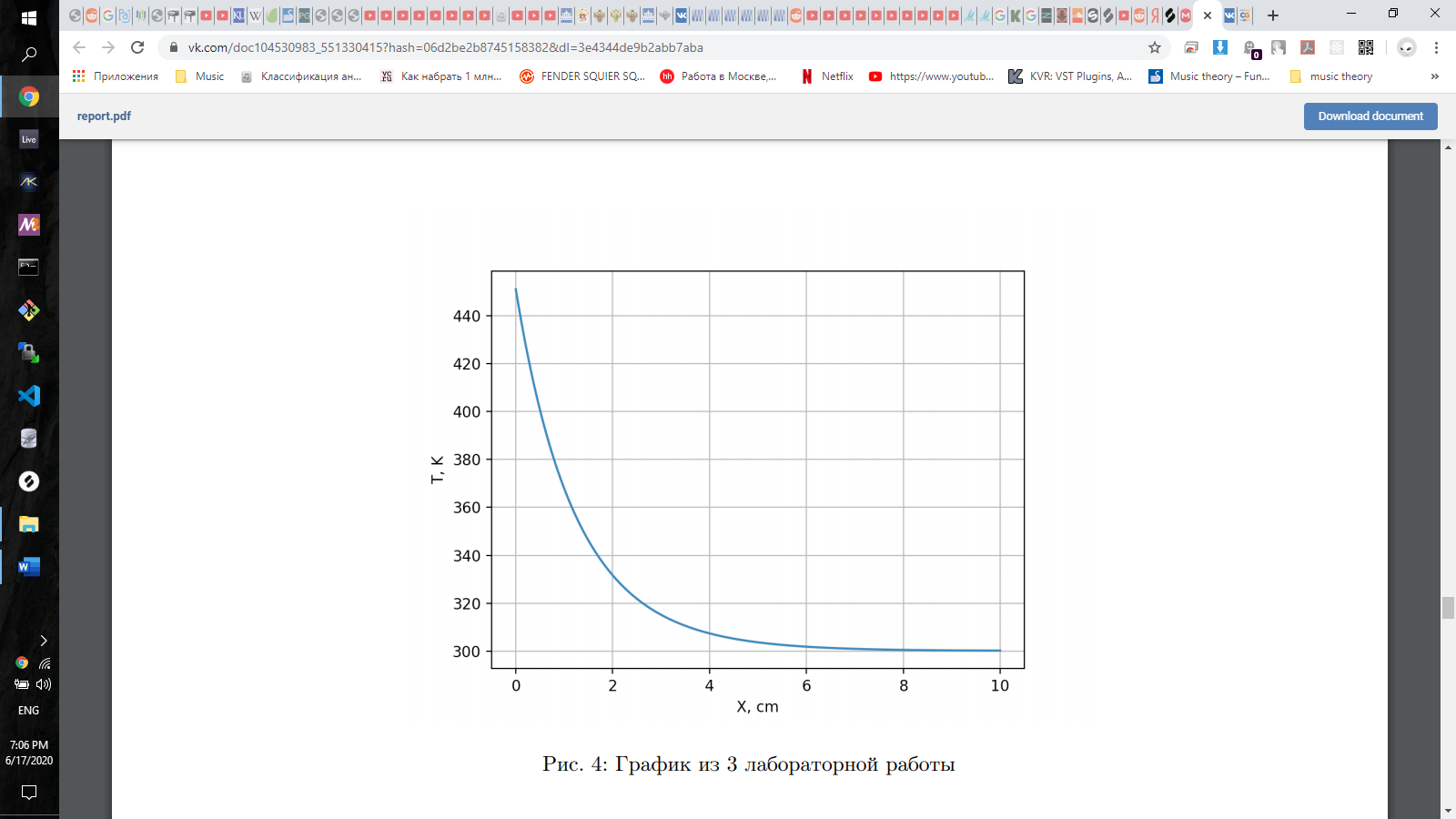


Рисунок 4. График зависимости T от x

1. Выполните линеаризацию уравнения (14.8) по Ньютону, полагая для простоты, что все коэффициенты зависят только от одной переменной .

