

**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** Юмаев А. Р.

**Группа** ИУ7-65Б

**Оценка (баллы)**

**Преподаватель** Градов В.М.

# 1. Теоретический раздел

**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

**Физический смысл задачи**

Сформулированная математическая модель описывает температурное поле вдоль цилиндрического стержня радиуса *R* и длиной *l*, причем и температуру можно принять постоянной по радиусу цилиндра. Ось *x* 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.

# 2. Листинг

|  |
| --- |
| import matplotlib.pyplot as plt  import numpy as np    def plotGrapth(x, y, xlabel, ylabel):  plt.grid(True)  plt.xlabel(xlabel)  plt.ylabel(ylabel)  plt.plot(x, y, 'g')  plt.show()  def k(x):  return a / (x - b)  def alpha(x):  return 3 \* x / (x - d)  def P(Ax):  return 2 \* Ax / R  def F(Ax):  return (2 \* T0 \* Ax) / R    def Xn\_formula(x, h, flag):  if flag == "+":  res = 2 \* k(x) \* k(x + h) / (k(x) + k(x + h))  if flag == "-":  res = 2 \* k(x) \* k(x - h) / (k(x) + k(x - h))  return res  def An(x, h):  res = 2 \* k(x) \* k(x - h) / (k(x) + k(x - h))  return res / h  def Bn(x, h, Ai, Ci):  return Ai + Ci + P(x) \* h  def Cn(x, h):  res = 2 \* k(x) \* k(x + h) / (k(x) + k(x + h))  return res / h  def Dn(x, h):  return F(x) \* h  def get\_K0(x0, h):  pn\_1\_div\_2 = (P(x0) + P(x0 + h)) / 2  return Xn\_formula(x0, h, "+") + (h \*\* 2) \* pn\_1\_div\_2 / 8 + (h \*\* 2) \* P(x0) / 4  def get\_M0(x0, h):  pn\_1\_div\_2 = (P(x0) + P(x0 + h)) / 2  return -Xn\_formula(x0, h, '+') + (h \*\* 2) \* pn\_1\_div\_2 / 8  def get\_P0(x0, h):  fn\_1\_div\_2 = (F(x0) + F(x0 + h)) / 2  return h \* F0 + (h \*\* 2) \* (fn\_1\_div\_2 + F(x0)) / 4  def get\_KN(x, h):  res = 2 \* k(x) \* k(x - h) / (k(x) + k(x - h))  return -P(x) \* h / 4 - (P(x - h) + P(x)) \* h / 16 - alpha(x) - res / h  def get\_MN(x, h):  res = 2 \* k(x) \* k(x - h) / (k(x) + k(x - h))  return res / h - (P(x - h) + P(x)) \* h / 16  def get\_PN(xn, h):  return -alpha(xn) \* T0 - h \* (3 \* F(xn) + F(xn - h)) / 8  def running(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  k0 = 0.4  kN = 0.1  alpha0 = 0.05  alphaN = 0.01  l = 30  T0 = 300  R = 0.5  F0 = 50  h = 1e-3  x0 = 0  b = kN / (kN - k0)  a = -k0 \* b  d = alphaN / (alphaN - alpha0)  c = -alpha0 \* d  A = []  B = []  C = []  D = []  x\_array = []  for x in np.arange(x0, h + 1, h):  x\_array.append(x)  Ai, Ci, Di = An(x, h), Cn(x, h), Dn(x, h)  Bi = Bn(x, h, Ai, Ci)  A.append(Ai)  B.append(Bi)  C.append(Ci)  D.append(Di)  K0 = get\_K0(x0, h)  P0 = get\_P0(x0, h)  M0 = get\_M0(x0, h)  KN = get\_KN(l, h)  PN = get\_PN(l, h)  MN = get\_MN(l, h) |

# 3. Результаты работы программы

3.1 Тестовые данные

*,*

*,*

,

,

*,*

*,*

*,*

*.*

3.2 Графики зависимых величин

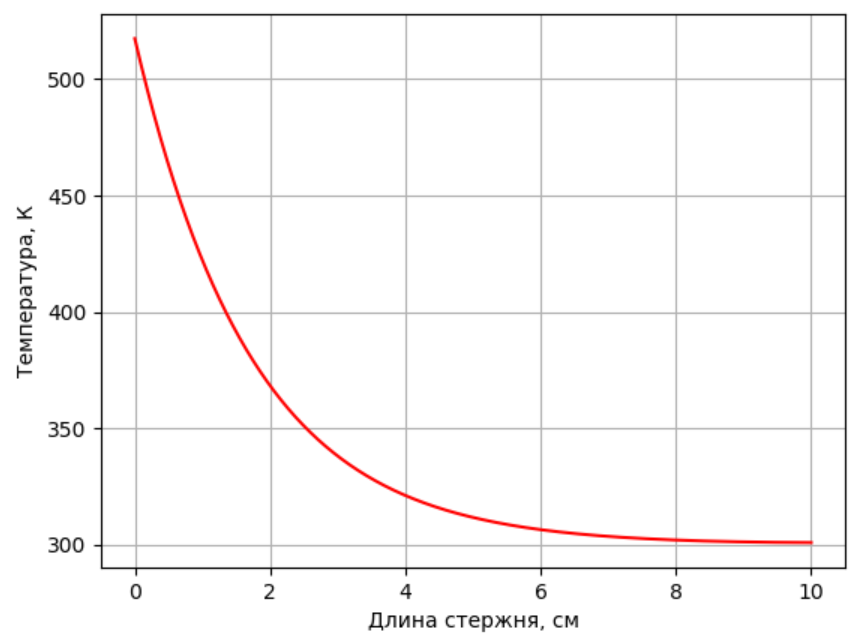
******

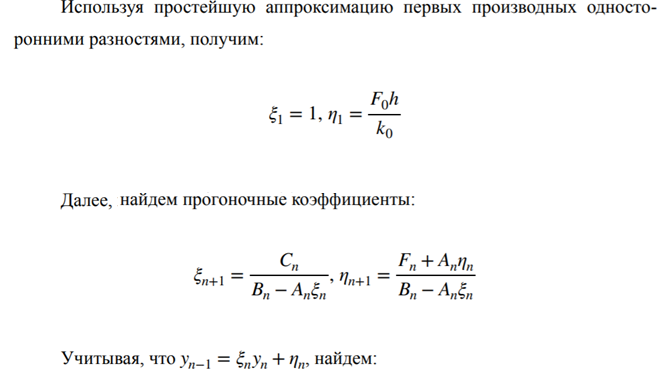
Рисунок 1. График зависимости температуры

|  |  |
| --- | --- |
|  |  |
| Рисунок 2. Значение при заданном выше F0 | Рисунок 3. Зависимость температуры при увеличенных втрое значения a(x) |
|  | https://sun9-46.userapi.com/c857032/v857032601/175bc1/Y_neRdY9IZU.jpg |
| *Рисунок 4. График зависимости T (x) при F0 = 0 Вт/см2.* | Рисунок 5. Гармонические колебания при R < 0 см и l = 30 см. |

При увеличении теплосъема и неизменном потоке уровень температур *T(x)* снижается, а градиент увеличивается (при сравнении рисунков 1 и 3). На рисунке 4 можно наблюдать, что, в отсутствии теплового нагружения, температура стержня равна окружающей температуре, погрешность определяется приближенным характером вычислений.

# 4. 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.**

*, ,*

*,*

Прямой ход (:

Обратный ход