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.

Москва, 2020 г.

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):

$$\frac{d}{dx}\left(k(x)\frac{dT}{dx}\right) - \frac{2}{R}\ \alpha(x)T + 2T_0\alpha(x) = 0$$

2. Краевые условия: $\begin{cases} x = 0, -k(0)\frac{dT}{dx} = F_0, \\ x = \ l, -k(l)\frac{dt}{dx} = \ \alpha_N(T(l) - T_0) \end{cases}$

3. Функции $\alpha(x), k(x)$ заданы своими константами

$$\alpha(x) = \frac{c}{x - d}$$
$$k(x) = \frac{a}{x - b}$$

Физический смысл задачи

Сформулированная математическая модель описывает температурное поле $T(x)$ вдоль цилиндрического стержня радиуса R и длиной l, причем $R \ll l$ и температуру можно принять постоянной по радиусу цилиндра. Ось x направлена вдоль оси цилиндра и начало координат совпадает с левым торцем стержня. Слева при $x = 0$ цилиндр нагружается тепловым потоком F_0. Стержень обдувается воздухом, температура которого равна T_0. В результате происходит съем тепла с цилиндрической поверхности и поверхности правого торца при $x = l$. Функции $k(x), a(x)$ 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

$k_0 = 0.4 \frac{Вт}{см}$ К,

$k_N = 0.1 \frac{Вт}{см}$ К,

$a_0 = 0.05 \frac{Вт}{см^2}$ К,

$a_N = 0.01 \frac{Вт}{см^2}$ К,

$l = 10$ см,

$T_0 = 300$ К,

$R = 0.5$ см,

$F_0 = 50 \frac{Вт}{см^2}$.

3.2 Графики зависимых величин

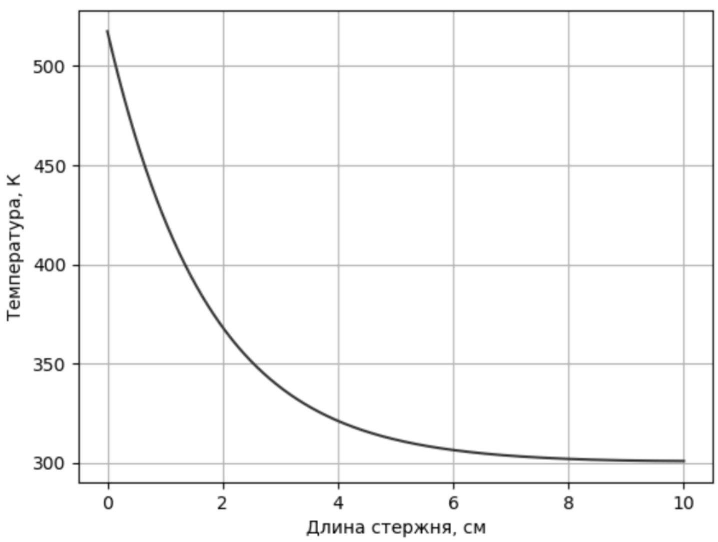

Рисунок 1. График зависимости температуры

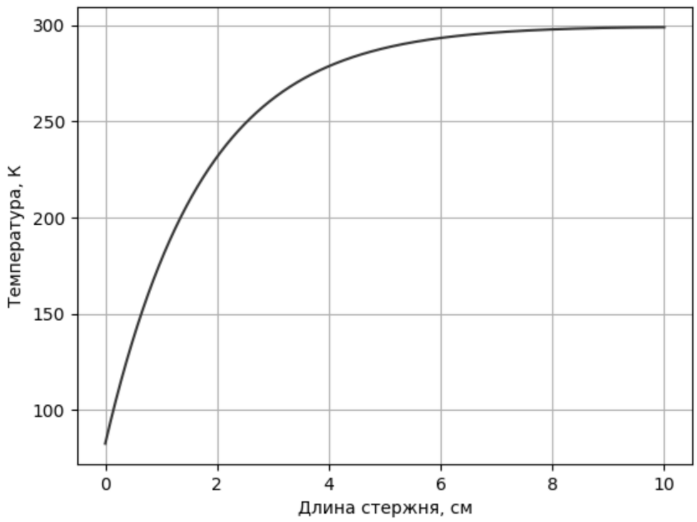

Рисунок 2. Значение при заданном выше F0

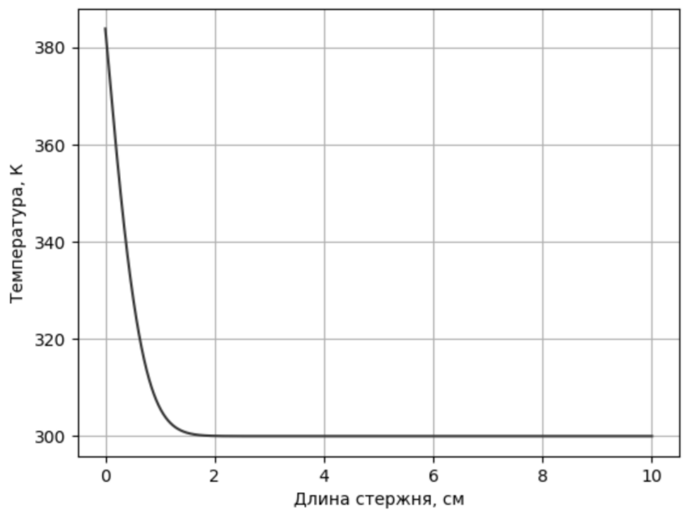

Рисунок 3. Зависимость температуры при увеличенных втрое значения $a(x)$

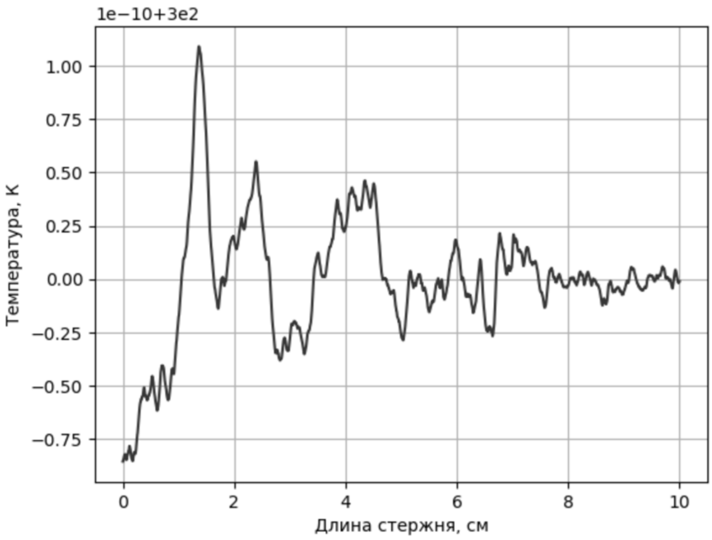

Рисунок 4. График зависимости T (x) при F0 = 0 Вт/см2.

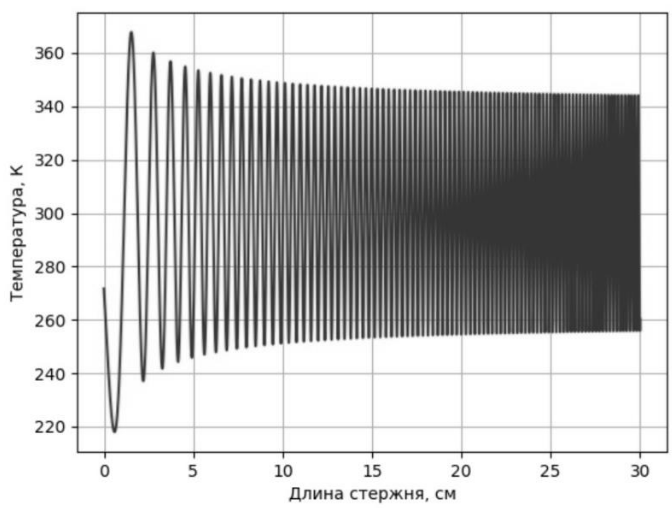

Рисунок 5. Гармонические колебания при R < 0 см и l = 30 см.

При увеличении теплосъема и неизменном потоке F_0 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?**
   a. При $F_0 = 0$ $T(x) = T_0 \pm \varepsilon$, где ε – погрешность
   b. Должна быть положительная производная функции $T(x)$ при $F_0 < 0$
   c. При отрицательном радиусе стержня $R<0$, должны наблюдаться гармонические колебания.
2. **Получите простейший разностный аналог нелинейного краевого условия при**
   $x = l,\quad -k(l)\dfrac{dT}{dx} = \alpha_N\,(T(l) - T_0) + \varphi(T):$
   Разностная аппроксимация краевого условия:
   $\frac{Y_{N-1}-Y_N}{h} k_N = \alpha_N(y_N - T_0) + \varphi(y_N)$

3. **Опишите алгоритм применения метода прогонки, если при $x = 0$ краевое условие линейное (как в настоящей работе), а при $x = l$, как в п.2**

Используя простейшую аппроксимацию первых производных односторонними разностями, получим:

$$\xi_1 = 1,\ \eta_1 = \frac{F_0 h}{k_0}$$

Далее, найдем прогоночные коэффициенты:

$$\xi_{n+1} = \frac{C_n}{B_n - A_n\xi_n},\ \eta_{n+1} = \frac{F_n + A_n\eta_n}{B_n - A_n\xi_n}$$

Учитывая, что $y_{n-1} = \xi_n y_n + \eta_n$, найдем:

$$y_N = \frac{k_N\eta_N + h\alpha\beta - h\varphi(y_N)}{k_N(1 - \xi_N) + h\alpha}$$

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

$\eta_1 = \frac{F_0}{B_0}, \eta_N = \frac{A_N}{B_N},$

$\xi_1 = \frac{C_0}{B_0}, \xi_N = \frac{F_N}{B_N},$

Прямой ход ($1 \leq i \leq p-1$):

$$\xi_{i+1} = \frac{C_i}{B_i - A_i \xi_i};$$

$$\eta_{i+1} = \frac{(F_i + A_i \eta_i)}{B_i - A_i \xi_i}$$

Обратный ход ($p \leq i \leq N-1$)

$$\widehat{\xi_i} = \frac{A_i}{B_i - C_i \xi_{i+1}}$$

$$\widehat{\eta_i} = \frac{F_i + C_i \widehat{\eta_{i+1}}}{B_i - C_i \widehat{\xi_{i+1}}}$$

$$y_{p-1} = \xi_p y_p + \eta_p$$

$$y_{p+1} = \hat{\xi}_p y_p + \hat{\eta}_p$$

$$A_p y_{p-1} + B_p y_p + C_p y_{p+1} = -P_p$$

$$=> y_p = \frac{F_p + A_p \eta_p + C_p \hat{\eta}_{p+1}}{B_p - A_p \xi_p - C_p \hat{\xi}_{p+1}}$$