|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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\_\_**

|  |  |
| --- | --- |
| **Тема:** Программно- алгоритмическая реализация моделей на основе дифференциальных уравнений в частных производных с краевыми условиями II и III рода.  **Студент \_\_\_\_**Оберган Т.М.**\_\_\_\_\_\_**  **Группа \_\_\_\_**ИУ7-65Б\_\_\_\_\_\_\_\_\_\_\_  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель \_\_\_**Градов В.М.\_\_\_ |  |

Москва.

2020 г.

**Оглавление**

[Исходные данные 3](#_Toc42077915)

[Физическое содержание задачи 4](#_Toc42077916)

[Листинг 6](#_Toc42077917)

[Результаты работы 10](#_Toc42077918)

[Вопросы при защите лабораторной работы 14](#_Toc42077919)

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

# Исходные данные

1. Задана математическая модель.

Уравнение для функции

(1)

Краевые условия

 (2)

В обозначениях уравнения (14.1) лекции №14

(3)

2. Разностная схема с разностным краевым условием при

(4)

При получении разностного аналога краевого условия при необходимо учесть, что поток

(5)

3. Значения параметров для отладки (все размерности согласованы)

, 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).

# Физическое содержание задачи

Постановки задач в данной лабораторной работе и работе №3 во многом совпадают. Отличия заключаются в следующем:

1. Сформулированная в данной работе математическая модель описывает нестационарное температурное поле, зависящее от координаты x и меняющееся во времени.

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

3. При цилиндр нагружается тепловым потоком , 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 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 .

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

# Лабораторная по моделированию №4

# Реализация модели на основе ДУ в частных производных

# с краевыми условиями II и III рода.

**import** matplotlib.pyplot **as** plt

**import** numpy **as** np

**from** math **import** fabs

**from** mpl\_toolkits.mplot3d **import** Axes3D

# Исходные данные модели

**def** k(T):

**return** a1 \* (b1 + c1 \* T\*\*m1)

**def** c(T):

**return** a2 + b2 \* T\*\*m2 - (c2 / T\*\*2)

**def** alpha(x):

d = (alphaN\*l) / (alphaN-alpha0)

c = - alpha0 \* d

**return** c / (x-d)

**def** p(x) :

**return** (2/R) \* alpha(x)

**def** f(x):

**return** (2\*T0/R) \* alpha(x)

**def** A(T):

**return** t/h \* approc\_minus\_half(k, T, t)

**def** D(T):

**return** t/h \* approc\_plus\_half(k, T, t)

**def** B(x, T):

**return** A(T) + D(T) + h\*c(T) + h\*t\*p(x)

**def** F(x, T):

**return** h\*t\*f(x) + T\*h\*c(T)

# Простая аппроксимация

**def** approc\_plus\_half(func, n, step):

**return** (func(n) + func(n + step)) / 2

**def** approc\_minus\_half(func, n, step):

**return** (func(n) + func(n - step)) / 2

# Краевые условия

# При х = 0

**def** left\_boundary\_condition(T\_prev):

T\_prev\_0 = T\_prev[0]

c\_plus = approc\_plus\_half(c, T\_prev\_0, t)

k\_plus = approc\_plus\_half(k, T\_prev\_0, t)

c0 = c(T\_prev\_0)

K0 = h/8 \* c\_plus + h/4 \* c0 + t/h \* k\_plus + \

t \* h/8 \* p(h/2) + t \* h/4 \* p(0)

M0 = h/8 \* c\_plus - t/h \* k\_plus + t \* h/8 \* p(h/2)

P0 = h/8 \* c\_plus \* (T\_prev\_0 + T\_prev[1]) + \

h/4 \* c0 \* T\_prev\_0 + F0 \* t + t \* h/8 \* (3 \* f(0) + f(h))

**return** K0, M0, P0

# При х = N

**def** right\_boundary\_condition(T\_prev):

T\_prev\_N = T\_prev[-1]

c\_minus = approc\_minus\_half(c, T\_prev\_N, t)

k\_minus = approc\_minus\_half(k, T\_prev\_N, t)

cN = c(T\_prev\_N)

KN = h/8 \* c\_minus + h/4 \* cN + t/h \* k\_minus + t \* alphaN + \

t \* h/8 \* p(l - h/2) + t \* h/4 \* p(l)

MN = h/8 \* c\_minus - t/h \* k\_minus + t \* h/8 \* p(l - h/2)

PN = h/8 \* c\_minus \* (T\_prev\_N + T\_prev[-2]) + \

h/4 \* cN \* T\_prev\_N + t \* alphaN \* T0 + t \* h/4 \* (f(l) + f(l - h/2))

**return** KN, MN, PN

**def** get\_T\_new(T\_prev):

K0, M0, P0 = left\_boundary\_condition(T\_prev)

KN, MN, PN = right\_boundary\_condition(T\_prev)

eps = [0, -M0 / K0]

eta = [0, P0 / K0]

x = h

n = 1

**while** (x + h < l):

T\_prev\_n = T\_prev[n]

denominator = (B(x, T\_prev\_n) - A(T\_prev\_n) \* eps[n])

next\_eps = D(T\_prev\_n) / denominator

next\_eta = (F(x, T\_prev\_n) + A(T\_prev\_n) \* eta[n]) / denominator

eps.append(next\_eps)

eta.append(next\_eta)

n += 1

x += h

T\_new = [0] \* (n + 1)

T\_new[n] = (PN - MN\*eta[n]) / (KN + MN\*eps[n])

**for** i **in** **range**(n - 1, -1, -1):

T\_new[i] = eps[i+1] \* T\_new[i+1] + eta[i+1]

**return** T\_new

# Метод простых итераций

**def** simple\_iter():

step1 = **int**(l / h)

T = [T0] \* (step1 + 1)

T\_new = [0] \* (step1 + 1)

ti = 0

res = []

res.append(T)

lent = **len**(T)

**while** True:

T\_prev = T

**while** True:

T\_new = get\_T\_new(T\_prev)

cur\_max = fabs((T[0] - T\_new[0]) / T\_new[0])

**for** i **in** **range**(lent):

d = fabs(T[i] - T\_new[i]) / T\_new[i]

**if** d > cur\_max:

cur\_max = d

**if** cur\_max < 1:

**break**

T\_prev = T\_new

res.append(T\_new)

ti += t

flag\_eps\_ok = True

**for** i **in** **range**(lent):

**if** fabs((T[i] - T\_new[i]) / T\_new[i]) > 1e-2:

flag\_eps\_ok = False

**if** flag\_eps\_ok:

**break**

T = T\_new

**return** res, ti

**if** \_\_name\_\_ == "\_\_main\_\_":

# Исходные данные

a1 = 0.0134

b1 = 1

c1 = 4.35e-4

m1 = 1

a2 = 2.049

b2 = 0.563e-3

c2 = 0.528e5

m2 = 1

alpha0 = 0.05

alphaN = 0.01

l = 10

T0 = 300

R = 0.5

F0 = 50

h = 1e-3

t = 1

# Расчеты

res, ti = simple\_iter()

# Построение графиков

lenres = **len**(res)

last = **int**(**len**(res[0]) / 7) # оставим только седьмую часть графика

res\_cutted = [i[0:last:] **for** i **in** res] # обрезаем неинтересную часть графика

# Трехмерный график

x, y = np.mgrid[0:lenres:1, 0:last:1]

z = np.array([np.array(i) **for** i **in** res\_cutted])

fig\_3d = plt.figure()

xyz = fig\_3d.add\_subplot(111, projection='3d')

xyz.plot\_surface(x, y, z, cmap='inferno')

fig\_3d.show()

# Проекции

fig, (first\_graph, second\_graph) = plt.subplots(

nrows=1, ncols=2,

figsize=(8, 4))

# Первая cm - K

x = **list**(np.arange(0, l, h))

x\_cutted = x[:last:]

step = 3

**for** i **in** res\_cutted[::step]:

first\_graph.plot(x\_cutted, i)

first\_graph.plot(x\_cutted, res\_cutted[-1])

first\_graph.set\_xlabel("x, cm")

first\_graph.set\_ylabel("T, K")

first\_graph.grid()

# Вторая sec - K

te = **list**(**range**(0, ti, t))

**for** i **in** np.arange(0, l/3, 0.2):

line = [j[**int**(i/h)] **for** j **in** res]

second\_graph.plot(te, line[:-1])

second\_graph.set\_xlabel("t, sec")

second\_graph.set\_ylabel("T, K")

second\_graph.grid()

fig.show()

# Результаты работы

1. Представить разностный аналог краевого условия при x=l и его краткий вывод интегро-интерполяционным методом.

Разностный аналог краевого условия при x = l получается путем интегрирования (1) на отрезке , с учетом (5).

Введем следующее обозначение:

(6)

Проинтегрируем (1):

(7)

Интегрируя аналогично разностному аналогу краевого условия при (из лекции 14) получим, учтя (5):

(

Приведем уравнение к виду

(9)

Будет принята простая аппроксимация (для функций c, X, p):

(10)

Из (4) можно получить :

(11)

Систему (11) можно решить методом итераций.

Пусть s – номер итерации.

(12)

2. График зависимости температуры от координаты x при нескольких фиксированных значениях времени при заданных выше параметрах.

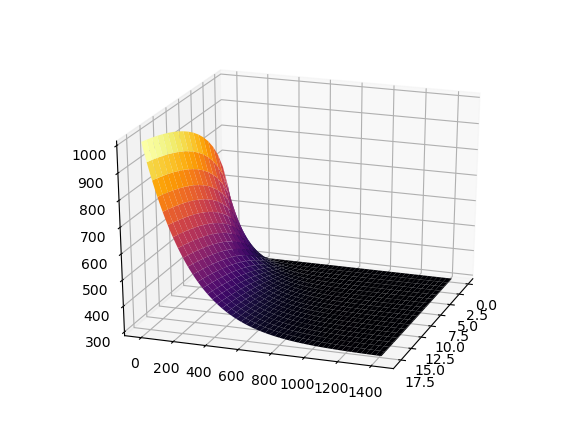


Рис. 1 – трехмерный график изменения температуры по времени и длине

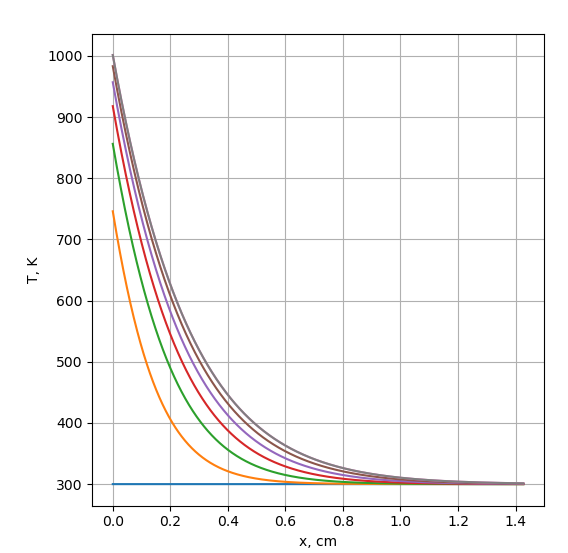


Рис. 2 – проекции трехмерного графика; зависимость температуры и длины в разные моменты времени. Синий график – в начальный момент времени

3. График зависимости при нескольких фиксированных значениях координаты . Обязательно представить случай n=0, т.е. .

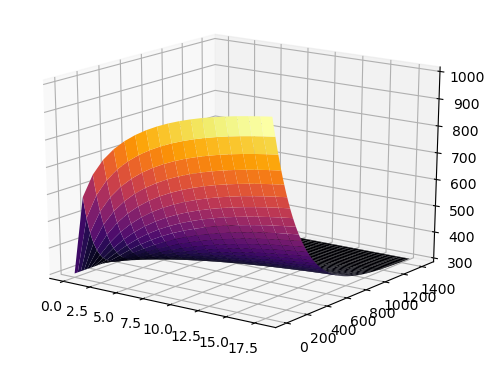


Рис. 3 – трехмерный график изменения температуры по времени и длине

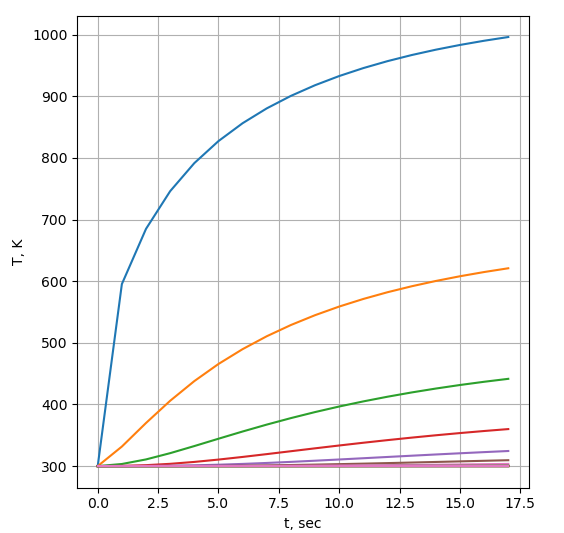


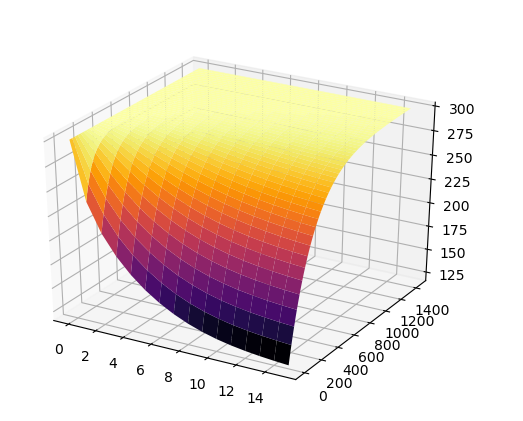
Рис. 4 – проекции трехмерного графика; зависимость температуры по времени в разных частях стержня. Синий график – при x = 0

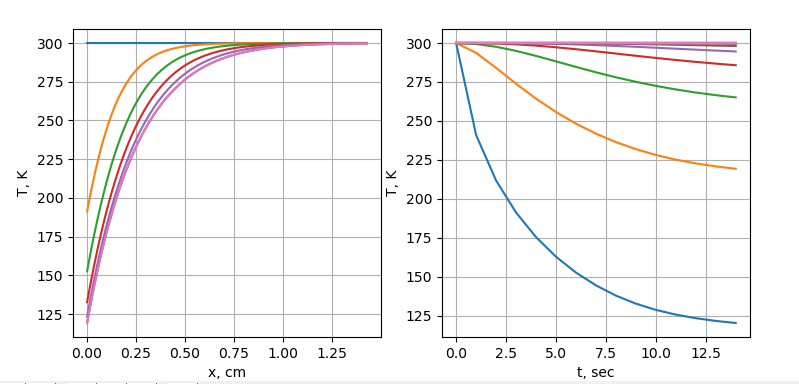
# Вопросы при защите лабораторной работы

1. Приведите результаты тестирования программы. Учесть опыт выполнения лабораторной работы №3.

Если принять F0 = -10

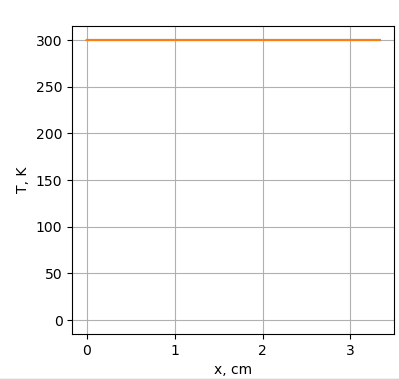
При отрицательном тепловом потоке слева идет съем тепла.



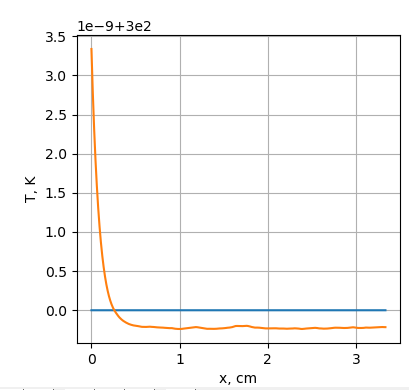


Если принять F0 = 0.

Тепловое нагружение отсутствует, причин для нагрева нет, температура стержня должна быть равна температуре окружающей среды T0.



На следующем графике видна погрешность:



\*Синий график – в начальный момент времени

2. 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.

Канонический вид системы квазилинейных разностных уравнений (14.8):

(13)

В условии дано:

=; =; =; =

(14)

Выполняя линеаризацию (13) по Ньютону, зная (14), получим:

(15)

Преобразуем к:

(16)

Уравнение (16) решается методом прогонки. В результате находятся все , после чего определяются значения искомой функции в узлах на s-итерации:

(17)

Итерационный процесс завершится при достижении условия:

(18)