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» (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04 Программная инженерия**

О Т Ч Е Т

по лабораторной работе № 4

Название: Программно-алгоритмическая реализация моделей на основе дифференциальных уравнений в частных производных с краевыми условиями II и III рода

Дисциплина: Моделирование

Студент ИУ7-62Б _____ Е.В. Брянская
(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ В.М. Градов
(Подпись, дата) (И.О. Фамилия)

Москва, 2021

Задание

Тема. Программно-алгоритмическая реализация моделей на основе дифференциальных уравнений в частных производных с краевыми условиями II и III рода.

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 T(x, t)

$$c(T)\frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\big(k(T)\frac{\partial T}{\partial x}\big) - \frac{2}{R}\alpha(x)T + \frac{2T_0}{R}\alpha(x) \tag{1}$$

   Краевые условия:

$$\begin{cases} t = 0, T(x,0) = T_0, \\ x = 0, -k(T(0))\frac{\partial T}{\partial x} = F_0, \\ x = l, -k(T(l))\frac{\partial T}{\partial x} = \alpha_N(T(l) - T_0) \end{cases} \tag{2}$$

   В обозначениях уравнения лекции

$$p(x) = \frac{2}{R}\alpha(x) \tag{3}$$

$$f(u) = f(x) = \frac{2T_0}{R}\alpha(x) \tag{4}$$

2. Разностная схема с разностным краевым условием при $x = 0$ получена в Лекции и может быть использована в данной работе. Самостоятельно надо получить интегроинтерполяционным методом разностный аналог краевого условия при $x = l$, точно так же, как это сделано при $x = 0$. Для этого надо проинтегрировать на отрезке $[x_{N-1/2}, x_N]$ выписанное выше уравнение 1 и учесть, что поток $\hat{F}_N = \alpha_N(\hat{y}_N - T_0)$, а $\hat{F}_{N-1/2} = \hat{\chi}_{N-1/2}\frac{\hat{y}_{N-1} - \hat{y}_N}{h}$.

3. Значения параметров для отладки (все размерности согласованы)

$$k(T) = a_1(b_1 + c_1 T^{m_1})$$

$$c(T) = a_2 + b_2 T^{m_2} - \frac{c_2}{T^2}$$

$$a_1 = 0.0134, b_1 = 1, c_1 = 4.35 \cdot 10^{-4}, m_1 = 1$$

$$a_2 = 2.049, b_2 = 0.563 \cdot 10^{-3}, c_2 = 0.528 \cdot 10^5, m_2 = 1$$

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

$\alpha_0 = 0.05$ Вт/см2 K,
$\alpha_N = 0.01$ Вт/см2 K,
$l = 10$ см,
$T_0 = 300$ K,
$R = 0.5$ см,
$F(t) = 50$ Вт/см2 (для отладки принять постоянным).

Выполнение

Задача решается интегро-интерполяционным методом.

Выбирается шаблон и связанная с шаблоном ячейка. Далее проводится интегрирование уравнения (1). В результате нескольких преобразований получается следующее:

$$\hat{c}_n(\hat{t}_n - t_n)h = \tau(\hat{F}_{n-1/2} - \hat{F}_{n+1/2}) - p_n\hat{t}_n\tau h + \hat{f}_n\tau h \tag{5}$$

С учётом формул (3) и (4) получаются такие выражения:

$$\begin{cases} \hat{F}_{n+1/2} = \hat{\chi}_{n+1/2}\dfrac{\hat{t}_n - \hat{t}_{n+1}}{h} \\ \hat{F}_{n-1/2} = \hat{\chi}_{n-1/2}\dfrac{\hat{t}_{n-1} - \hat{t}_n}{h} \end{cases} \tag{6}$$

Следующий шаг для 1ого уравнения:

$$\hat{\chi}_{n+1/2} = \frac{\hat{k}_n + \hat{k}_{n+1}}{2} \tag{7}$$

Аналогично для 2ого.

Затем полученные выражения подставляются в (5), которое далее приводится к виду:

$$\hat{A}_n\hat{t}_{n-1} - \hat{B}_n\hat{t}_n + \hat{D}_n\hat{t}_{n+1} = -F_n$$

И применяется метод простой прогонки.

Результаты работы.

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

   Проинтегрируем (1) на отрезке $[x_{N-1/2}; x_N]$:

$$\int_{x_{N-1/2}}^{x_N} dx \int_{t_m}^{t_{m+1}} c(T)\frac{\partial T}{\partial t}dt = -\int_{t_m}^{t_{m+1}} dt \int_{x_{N-1/2}}^{x_N} \frac{\partial F}{\partial x}dx - \\ -\int_{x_{N-1/2}}^{x_N} dx \int_{t_m}^{t_{m+1}} p(x)T dt + \int_{x_{N-1/2}}^{x_N} dx \int_{t_m}^{t_{m+1}} f(x)dt$$

   Пользуясь методами приближённого интегрирования получаем:

$$\frac{h}{4}\big(\hat{c}_N(\hat{T}_N - T_N) + \hat{c}_{N-1/2}(\hat{T}_{N-1/2} - T_{N-1/2})\big) = -\tau(\hat{F}_N - \hat{F}_{N-1/2}) - \\ -\tau\frac{h}{4}(p_N\hat{T}_N + p_{N-1/2}\hat{T}_{N-1/2}) + \tau\frac{h}{4}(\hat{f}_N + \hat{f}_{N-1/2})$$

   Учитывая правое краевое условие:

$$\frac{h}{4}\big(\hat{c}_N(\hat{T}_N - T_N) + \hat{c}_{N-1/2}(\hat{T}_{N-1/2} - T_{N-1/2})\big) = \\ -\tau(\alpha_N(\hat{T}_N - T_0) - \hat{\chi}_{N-1/2}\frac{\hat{T}_{N-1} - \hat{T}_N}{h}) - \\ -\tau\frac{h}{4}(p_N\hat{T}_N + p_{N-1/2}\hat{T}_{N-1/2}) + \tau\frac{h}{4}(\hat{f}_N + \hat{f}_{N-1/2})$$

   Приводим к форме $\hat{K}_N\hat{T}_N + \hat{M}_{N-1}\hat{T}_{N-1} = \hat{P}_N$.

$$\begin{cases} \hat{M}_N = \dfrac{h}{8}\big(\hat{c}_{N-1/2} + \tau p_{N-1/2}\big) - \tau\dfrac{\hat{\chi}_{N-1/2}}{h} \\ \hat{K}_N = \dfrac{h}{8}\big(2\hat{c}_N + \hat{c}_{N-1/2} + 2p_N\tau + \tau p_{N-1/2}\big) + \tau(\alpha_N - \dfrac{\hat{\chi}_{N-1/2}}{h}) \\ \hat{P}_N = \dfrac{h}{4}\big(\hat{c}_N T_N + \hat{c}_{N-1/2}\dfrac{T_N + T_{N-1}}{2} + \tau\hat{f}_N + \tau\hat{f}_{N-1/2}\big) + \tau\alpha_N T_0 \end{cases}$$

2. *График зависимости температуры $T(x, t_m)$ от координаты x при фиксированных значениях времени t_m при заданных выше параметрах.*

$\tau = 0.01$, $\varepsilon = 10^{-4}$

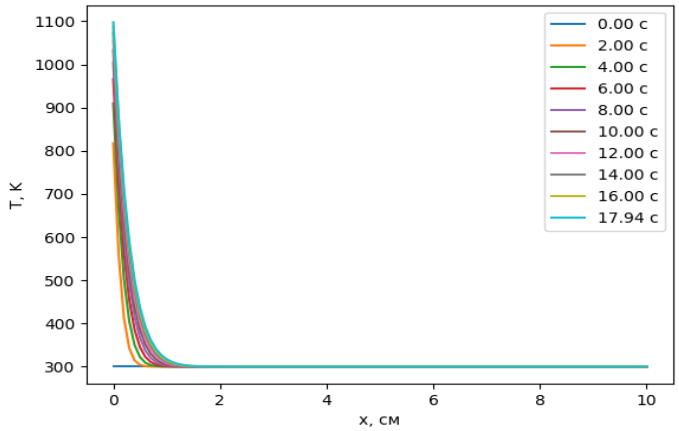

Рис. 1 — Задание 2

3. *График зависимости $T(x_n, t)$ при нескольких значениях координаты x_n*

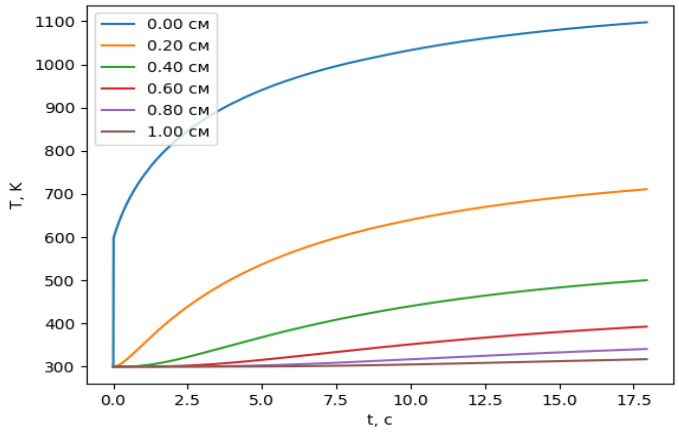

Рис. 2 — Задание 3

Вопросы при защите лабораторной работы

1. *Приведите результаты тестирования программы (графики, общие соображения, качественный анализ)*

   (a) $F_0(t) = 0$

   Из-за того, что тепловой поток равен нулю, температура равна T_0 для всех x.

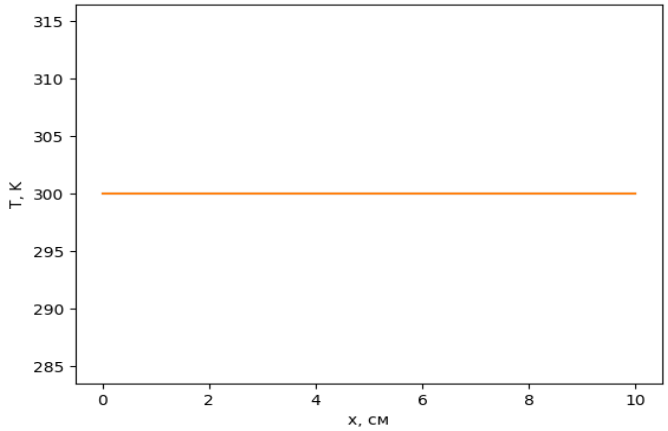

Рис. 3 — Тест 1

   (b) Изменение теплопроводности (увеличение)

   Увеличение теплопроводности должно вызвать более равномерное распределение тепла в температурном поле. По результатам видно, что значение температуры стало меньше, и графики стали более пологими.

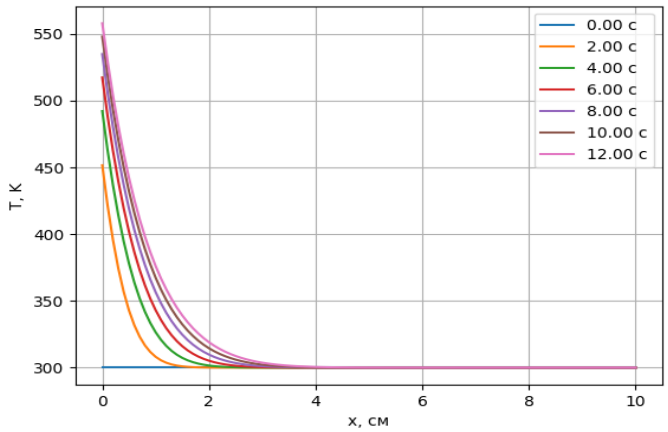

Рис. 4 — Тест 2

(c) Изменение теплоёмкости (увеличение)

Увеличение теплоёмкости должно вызвать уменьшение температуры, и как следствие, во всем температурном поле также должна ухудшиться теплопроводность. Поэтому распределение температуры в поле должно стать менее равномерным. Также должно увеличиться время нагрева.

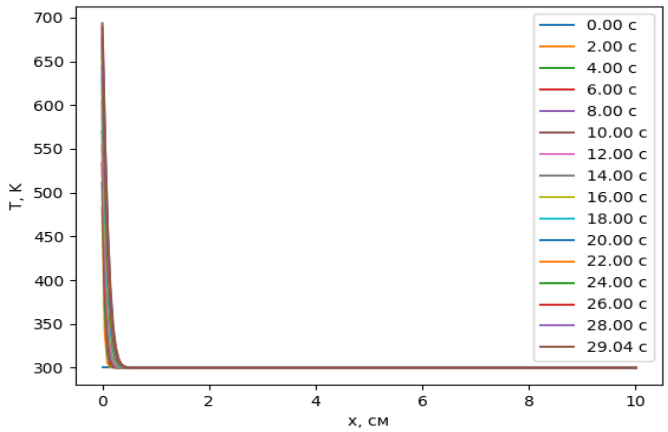

Рис. 5 — Тест 3

(d) $F_0(t) < 0$ ($F_0 = -15$)

В такой ситуации происходит съём тепла.

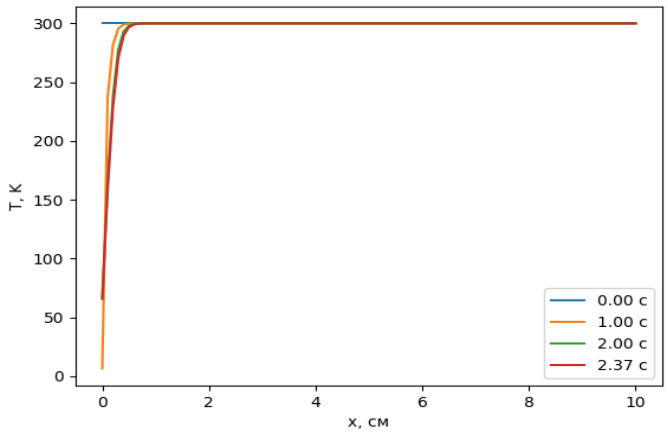

Рис. 6 — Тест 4

Код программы

Листинг 1 — Лабораторная работа №3

```
import matplotlib.pyplot as plt

a1 = 0.0134
b1 = 1
c1 = 4.35e-4
m1 = 1
a2 = 2.049
b2 = 0.563e-3
c2 = 0.528e5
m2 = 1
alpha_0 = 0.05
alpha_N = 0.01
l = 10
T0 = 300
R = 0.5c
F = 50

tau = 0.01
step = 0.1
N = round(l / step) + 1
eps = 1e-5

cur_temp = [T0] * N
x = [0] * N
for i in range(N):
  x[i] = i * step

def k(t):
  return a1 * (b1 + c1 * t**m1)

def c(t):
  return a2 + b2 * t**m2 - c2 / (t**2)

def alpha(x):
  d = alpha_N * l / (alpha_N - alpha_0)
  c = -(alpha_N * alpha_0 * l) / (alpha_N - alpha_0)
  return c / (x - d)
```

```
38
39 def p(x):
40   return 2 / R * alpha(x)
41
42 def f(x):
43   return 2 * T0 / R * alpha(x)
44
45 def hee_minus(n):
46   return (k(cur_temp[n]) + k(cur_temp[n - 1])) / 2
47
48 def hee_plus(n):
49   return (k(cur_temp[n]) + k(cur_temp[n + 1])) / 2
50
51 def A(n):
52   return hee_minus(n) * tau / step
53
54 def D(n):
55   return hee_plus(n) * tau / step
56
57 def B(n):
58   return A(n) + D(n) + c(cur_temp[n]) * step + p(x[n]) * step * tau
59
60 def F(n, t):
61   return f(x[n]) * step * tau + c(cur_temp[n]) * t * step
62
63 def M0():
64   c_12 = c((cur_temp[0] + cur_temp[1]) / 2)
65   p_12 = p(x[0] + step/2)
66   hee_12 = hee_plus(0)
67   return step / 8 * c_12 - hee_12 * tau / step + tau * step / 8 * p_12
68
69 def K0():
70   p0 = p(x[0])
71   p_12 = p(x[0] + step / 2)
72   c0 = c(cur_temp[0])
73   c_12 = c((cur_temp[0] + cur_temp[1]) / 2)
74   hee_12 = hee_plus(0)
75   return step / 8 * (c_12 + 2 * c0) + hee_12 * tau / step + tau * step /
       8 * (p_12 + 2 * p0)
76
```

```
77 def P0():
78   c0 = c(cur_temp[0])
79   c_12 = c((cur_temp[0] + cur_temp[1]) / 2)
80   f0 = f(x[0])
81   f_12 = f(x[0] + step/2)
82   return step/8 * c_12 * (cur_temp[0] + cur_temp[1]) + \
83        step/4 * c0 * cur_temp[0] + cF*tau + tau*step/4 * (f_12 + f0)
84
85 def MN():
86   c_N = c(cur_temp[N - 1])
87   c_12 = c((cur_temp[N - 1] + cur_temp[N - 2]) / 2)
88   p_N = p(l)
89   p_12 = p(l - step/2)
90   hee_12 = hee_minus(N - 1)
91   return step / 8 * (2*c_N + c_12) + tau*step/8 * (2*p_N + p_12) + tau *
       (alpha_N + hee_12/step)
92
93 def KN():
94   hee_12 = hee_minus(N - 1)
95   c_12 = c((cur_temp[N-1] + cur_temp[N-2]) / 2)
96   p_12 = p(l - step/2)
97   return c_12*step/8 + p_12*tau*step/8 - tau*hee_12/step
98
99 def PN():
100  c_N = c(cur_temp[N - 1])
101  c_12 = c((cur_temp[N - 1] + cur_temp[N - 2]) / 2)
102  f_N = f(l)
103  f_12 = f(l - step/2)
104  return step/4 * (c_N*cur_temp[N-1] + c_12/2 * (cur_temp[N-1] +
       cur_temp[N-2])) + step*tau/4 * (f_N + f_12) + tau*alpha_N*T0
105
106 def calc_iter(prev_temp):
107  m0 = M0()
108  k0 = K0()
109  p0 = P0()
110  eps = [0] * N
111  eta = [0] * N
112  eps[1] = -m0 / k0
113  eta[1] = p0 / k0
114
```

```
115   for i in range(1, N - 1):
116     eps[i + 1] = D(i) / (B(i) - A(i) * eps[i])
117     eta[i + 1] = (F(i, prev_temp[i]) + A(i) * eta[i]) / (B(i) - A(i) *
      eps[i])
118
119   mN = MN()
120   kN = KN()
121   pN = PN()
122   new_temp = [0] * N
123   new_temp[-1] = (pN - kN * eta[-1]) / (mN + kN * eps[-1])
124   for i in range(N - 1, 0, -1):
125   new_temp[i - 1] = eps[i] * new_temp[i] + eta[i]
126   return new_temp
127
128 def temperature_check(old_t, new_t):
129   for i in range(N):
130     if abs((new_t[i] - old_t[i]) / new_t[i]) > eps:
131       return False
132   return True
133
134 def main():
135   global cur_temp
136
137   t_arr = [0]
138   temp_arr = [cur_temp.copy()]
139
140   while True:
141     prev_temp = cur_temp
142     while True:
143       new_temp = calc_iter(prev_temp)
144
145       if temperature_check(cur_temp, new_temp):
146         break
147       cur_temp = new_temp
148
149     t_arr.append(t_arr[-1] + tau)
150     temp_arr.append(new_temp)
151
152     if temperature_check(prev_temp, new_temp):
153       break
```

```
154         cur_temp = new_temp
155
156     cur_temp = new_temp
157
158     # Отрисовка графиков
159
160 if __name__ == '__main__':
161     main()
```