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»

Отчет по лабораторной работе №4 по дисциплине «Моделирование»

Тема Модели на основе ДУ в частных производных с краевыми условиями 2 и 3 рода

Студент Романов А.В.

Группа ИУ7-63Б

Оценка (баллы) __________

Преподаватель Градов В. М.

Москва — 2021 г.

Тема работы

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

Цель работы

Получение навыков разработки алгоритмов решения смешанной краевой задачи при реализации моделей, построенных на квазилинейном уравнении параболического типа.

Теоретические сведения

Задана математическая модель:

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

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:

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

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

Разностная схема с разностным краевым условие при $x = 0$:

$$(\frac{h}{8}\ \widehat{c_{\frac{1}{2}}} + \frac{h}{4}\ \widehat{c_0} + \widehat{X_{\frac{1}{2}}}\ \frac{\tau}{h} + \frac{\tau h}{8}p_{\frac{1}{2}} + \frac{\tau h}{4}p_0)\ \widehat{y_0} + (\frac{h}{8}\ \widehat{c_{\frac{1}{2}}} - \widehat{X_{\frac{1}{2}}}\ \frac{\tau}{h} + \frac{\tau h}{8}p_{\frac{1}{2}})\ \widehat{y_1} =$$
$$= \frac{h}{8}\ \widehat{c_{\frac{1}{2}}}\ (y_0 + y_1) + \frac{h}{4}\ \widehat{c_0}\ y_0 + \widehat{F}\ \tau + \frac{\tau h}{4}(\widehat{f_{\frac{1}{2}}} + \widehat{c_0}) \tag{5}$$

При получении разностного аналога краевого условия при $x = l$ учесть, что поток:

$$F_N = \alpha N(y_N - T_0), F_{N-\frac{1}{2}} = X_{N-\frac{1}{2}}\frac{y_{N-1} - y_N}{h} \tag{6}$$

Заданы начальные параметры:

- $k(T) = a_1(b_1 + c_1 T^{m_1})$, Вт/см К
- $c(T) = a_2 + b_2 T^{m_2} - \frac{c_2}{T^2}$, Дж/$cm^3$ К
- $a_1 = 0.0134, b_1 = 1, c_1 = 4.35 \cdot 10^{-4}, m_1 = 1$
- $a_2 = 2.049, b_2 = 0.563 \cdot 10^{-3}, c_2 = 0.528 \cdot 10^5, m_2 = 1$
- $\alpha(x) = \frac{c}{x-d}, \alpha_0 = 0.05$ Вт/cm^2 К, $\alpha_N = 0.01$ Вт/cm^2 К
- $l = 10$ см
- $T_0 = 300K$
- $R = 0.5$ см
- $F(t) = 50$ Вт/cm^2

Исходный код алгоритма

```
from collections import namedtuple
from math import pow, fabs

Params = namedtuple('Params', 'a1 b1 c1 m1 a2 b2 c2 m2 alpha0 alphaN l T0 R
    F0 h t eps')

params = Params(
  0.0134, 1, 4.35e-4, 1, 2.049, 0.563e-3, 0.528e5, 1, 0.05, 0.01, 10, 300,
     0.5, 50, 1e-3, 1, 1e-2
)


def approximation_plus(func, n, step):
  return (func(n) + func(n + step)) / 2


def approximation_minus(func, n, step):
  return (func(n) + func(n - step)) / 2


def k(T):
  return params.a1 * (params.b1 + params.c1 * pow(T, params.m1))


def c(T):
  return params.a2 + params.b2 * pow(T, params.m2) - (params.c2 / pow(T, 2))


def alpha(x):
```

```
    d = (params.alphaN * params.l) / (params.alphaN - params.alpha0)
    c = -params.alpha0 * d
    return c / (x - d)


def p(x) :
    return alpha(x) * 2 / params.R


def f(x):
    return alpha(x) * 2 * params.T0 / params.R


def A(T):
    return params.t / params.h * approximation_minus(k, T, params.t)


def D(T):
    return params.t / params.h * approximation_plus(k, T, params.t)


def B(x, T):
    return A(T) + D(T) + params.h * c(T) + params.h * params.t * p(x)


def F(x, T):
    return params.h * params.t * f(x) + T * params.h * c(T)


def get_left_conditions(T):
    c_plus = approximation_plus(c, T[0], params.t)
    k_plus = approximation_plus(k, T[0], params.t)

    K0 = params.h / 8 * c_plus + params.h / 4 * c(T[0]) + params.t / params.h
        * k_plus + \
        params.t * params.h / 8 * p(params.h / 2) + params.t * params.h / 4 * p
            (0)

    M0 = params.h / 8 * c_plus - params.t / params.h * k_plus + params.t *
        params.h / 8 * p(params.h / 2)

    P0 = params.h / 8 * c_plus * (T[0] + T[1]) + params.h / 4 * c(T[0]) * T[0]
        + \
        params.F0 * params.t + params.t * params.h / 8 * (3 * f(0) + f(params.h)
            )

    return K0, M0, P0


def get_right_conditions(T):
```

```
  c_minus = approximation_minus(c, T[-1], params.t)
  k_minus = approximation_minus(k, T[-1], params.t)

  KN = params.h / 8 * c_minus + params.h / 4 * c(T[-1]) + params.t / params.h * k_minus + \
    params.t * params.alphaN + params.t * params.h/ 8 * p(params.l - params.h / 2) + \
    params.t * params.h / 4 * p(params.l)

  MN = params.h / 8 * c_minus - params.t / params.h * k_minus + \
    params.t * params.h / 8 * p(params.l - params.h / 2)

  PN = params.h / 8 * c_minus * (T[-1] + T[-2]) + params.h / 4 * c(T[-1]) * T[-1] + \
    params.t * params.alphaN * params.T0 + params.t * params.h / 4 * (f(params.l) + f(params.l - params.h / 2))

  return KN, MN, PN


def get_new_T(T):
  K0, M0, P0 = get_left_conditions(T)
  KN, MN, PN = get_right_conditions(T)

  xi = [0, -M0 / K0]
  eta = [0, P0 / K0]

  x = params.h
  n = 1

  while x + params.h < params.l:
    Tn = T[n]
    denominator = (B(x, Tn) - A(Tn) * xi[n])

    next_xi = D(Tn) / denominator
    next_eta = (F(x, Tn) + A(Tn) * eta[n]) / denominator

    xi.append(next_xi)
    eta.append(next_eta)

    n += 1
    x += params.h

  T_new = [0 for _ in range(n + 1)]
  T_new[n] = (PN - MN * eta[n]) / (KN + MN * xi[n])

  for i in range(n - 1, -1, -1):
    T_new[i] = xi[i + 1] * T_new[i + 1] + eta[i + 1]

  return T_new
```

```
def simple_iteration_method():
  T = [params.T0 for _ in range(int(params.l / params.h) + 1)]
  T_new = [0 for _ in range(int(params.l / params.h) + 1)]

  result = [T]
  ti = 0

  epsilon_condition = True
  while epsilon_condition:
    T_prev = T
    current_max = 1

    while current_max >= 1:
      T_new = get_new_T(T_prev)
      current_max = fabs((T[0] - T_new[0]) / T_new[0])

      for T_i, Tnew_i in zip(T, T_new):
        d = fabs(T_i - Tnew_i) / Tnew_i
        if d > current_max:
          current_max = d

      T_prev = T_new

    result.append(T_new)
    ti += params.t

    epsilon_condition = False
    for T_i, Tnew_i in zip(T, T_new):
      if fabs((T_i - Tnew_i) / Tnew_i) > params.eps:
        epsilon_condition = True

    T = T_new

  return result, ti
```

Результаты работы программы

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

Проинтегрируем уравнение на отрезке $[X_{n-\frac{1}{2}}; x_n]$ (с учётом **(6)**). Примем $F = -k(u)\frac{\partial T}{\partial x}$.

$$\int_{x_{N-\frac{1}{2}}}^{x_N} int_{t_m}^{t_{m+1}} c(t)\frac{\partial T}{\partial t}dt = -\int_{t_m}^{t_{m+1}} dt \int_{x_{N-\frac{1}{2}}}^{x_N} \frac{\partial F}{\partial x}dx - \int_{x_{N-\frac{1}{2}}}^{x_N} dx \int^{t_{m+1}} t_m p(x)T dt + \int_{x_{N-\frac{1}{2}}}^{x_N} dx \int^{t_{m+1}} t_m f(x) dt \tag{7}$$

Интегрируя аналогично разностному аналогу краевого условия при $x = 0$ (из лекции) получим, учтя **(6)**:

$$\frac{h}{4}(\widehat{c_N}\,(\widehat{y_N} - y_N) - \widehat{c_{N-\frac{1}{2}}}\,(\frac{\widehat{y_N} + \widehat{y_{N-1}}}{2} - \frac{y_N + y_{N+1}}{2})) =$$

$$= \tau(\alpha_N(\widehat{y_N} - T_0) - \widehat{X_N}\,\frac{\widehat{y_N} + \widehat{y_{N-1}}}{h}) - \tau\frac{h}{4}(p_N\,\widehat{y_N} - p_{n-\frac{1}{2}} - \frac{\widehat{y_N} + \widehat{y_{N-1}}}{2} + (\widehat{f_N} - \widehat{f_{N-\frac{1}{2}}})) \quad (8)$$

Приведем уравнение к виду $K_N\,\widehat{y_N} + M_N\,\widehat{y_{N-1}} = P_N$:

$$(\frac{h}{4}\,\widehat{c_N} + \frac{h}{8}\,\widehat{c_{N-\frac{1}{2}}} + \tau\alpha_N + \frac{\tau}{h}\,\widehat{X_{N-\frac{1}{2}}} + \frac{h}{4}\tau p_N + \frac{h}{8}\tau p_{N-\frac{1}{2}})\,\widehat{y_n} + (\frac{h}{8}\,\widehat{c_{N-\frac{1}{2}}} - \frac{\tau}{h}\,\widehat{X_{N-\frac{1}{2}}} + \frac{h}{8}\tau p_{N-\frac{1}{2}})\,\widehat{y_{N-1}} =$$

$$= \alpha_N\tau T_0 + \frac{h}{4}\,\widehat{C_N}\,y_N + \frac{h}{8}\,\widehat{c_{N-\frac{1}{2}}}\,(y_N + y_{N-1}) + \frac{h}{4}\tau(\widehat{f_N} + \widehat{f_{N-\frac{1}{2}}}) \quad (9)$$

Будеми использовать простую аппроксимацию:

$$p_{N-\frac{1}{2}} = \frac{p_{N-1} + p_N}{2} \quad (10)$$

Получим $K_0, M_0, P_0, K_N, M_N, P_N$:

$$\begin{cases} \widehat{K_0}\widehat{y_0} + \widehat{M_0}\widehat{y_1} = \widehat{p_0} \\ \widehat{A_n}\widehat{y_{n-1}} - \widehat{B_n}\widehat{y_n} + \widehat{D_n}\widehat{y_{n+1}} = -\widehat{F_n} \\ \widehat{K_n}\widehat{y_N} + \widehat{M_{N-1}}\widehat{y_{N-1}} = \widehat{P_N} \end{cases} \quad (11)$$

Систему **(11)** решим методом итераций (s - номер итерации):

$$A_n^{s-1}y_{n+1}^s - B_n^{s-1}y_n^s + D_n^{s-1}y_{n-1}^s = -F_n^{s-1} \quad (12)$$

2. График зависимости температуры $T(x, t_m)$ от координаты x при нескольких фкисированных значених времени t_m при заданных выше параметрах.

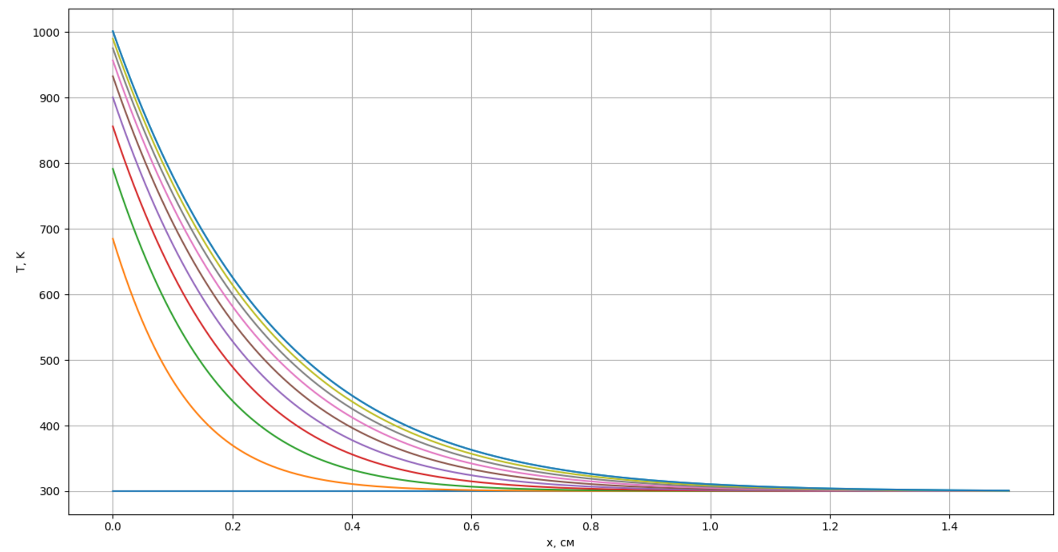

Рис. 1: График зависимости $T(x_n, t)$ от координаты x

3. График зависимости $T(x_n, t)$ при нескольких фиксированных значения координаты x_n

На рисунке 2 верхний график соответствует случаю $x = 0$, нижний случаю $x = l$.

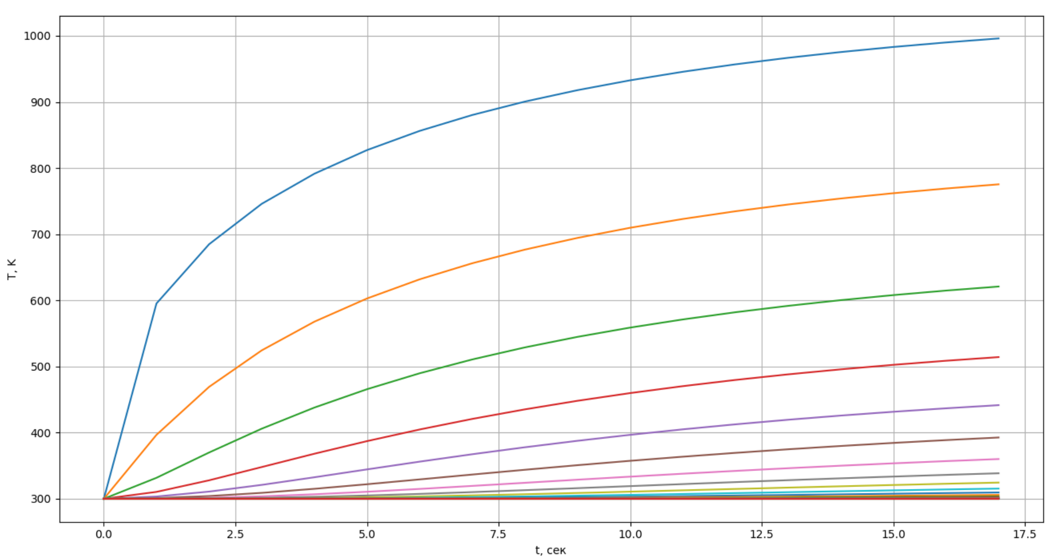

Рис. 2: График зависимости $T(x_n, t)$ при нескольких фиксированных значениях координаты x_n

Ответы на вопросы

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

При отрицательном тепловом потоке слева идет съем тепла (рисунки 3 - 4).

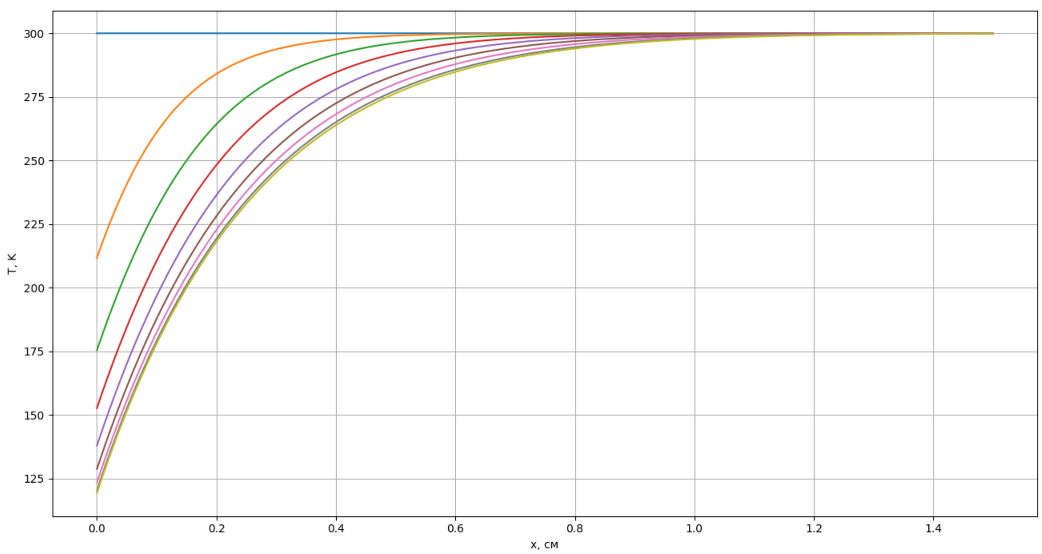

Рис. 3: График зависимости $T(x_n, t)$ от координаты x при $F_0 = -10$

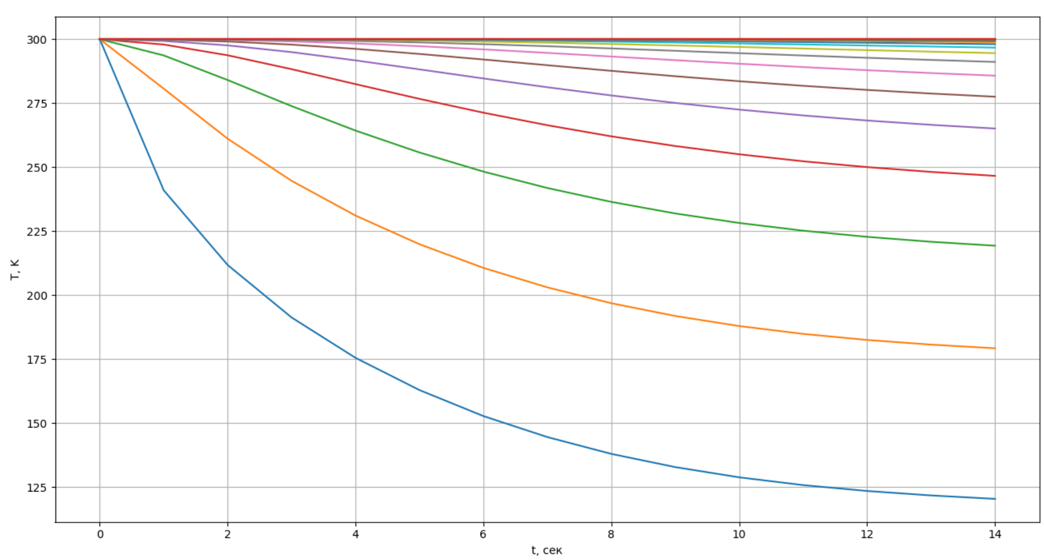

Рис. 4: График зависимости $T(x_n, t)$ при нескольких фиксированных значения координаты x_n и $F_0 = -10$

Если обнулить поток $F_0(T)$, то на выходе должны получить график температуры, установившейся в соответствии с температурой окружающей среды (рисунок 5).

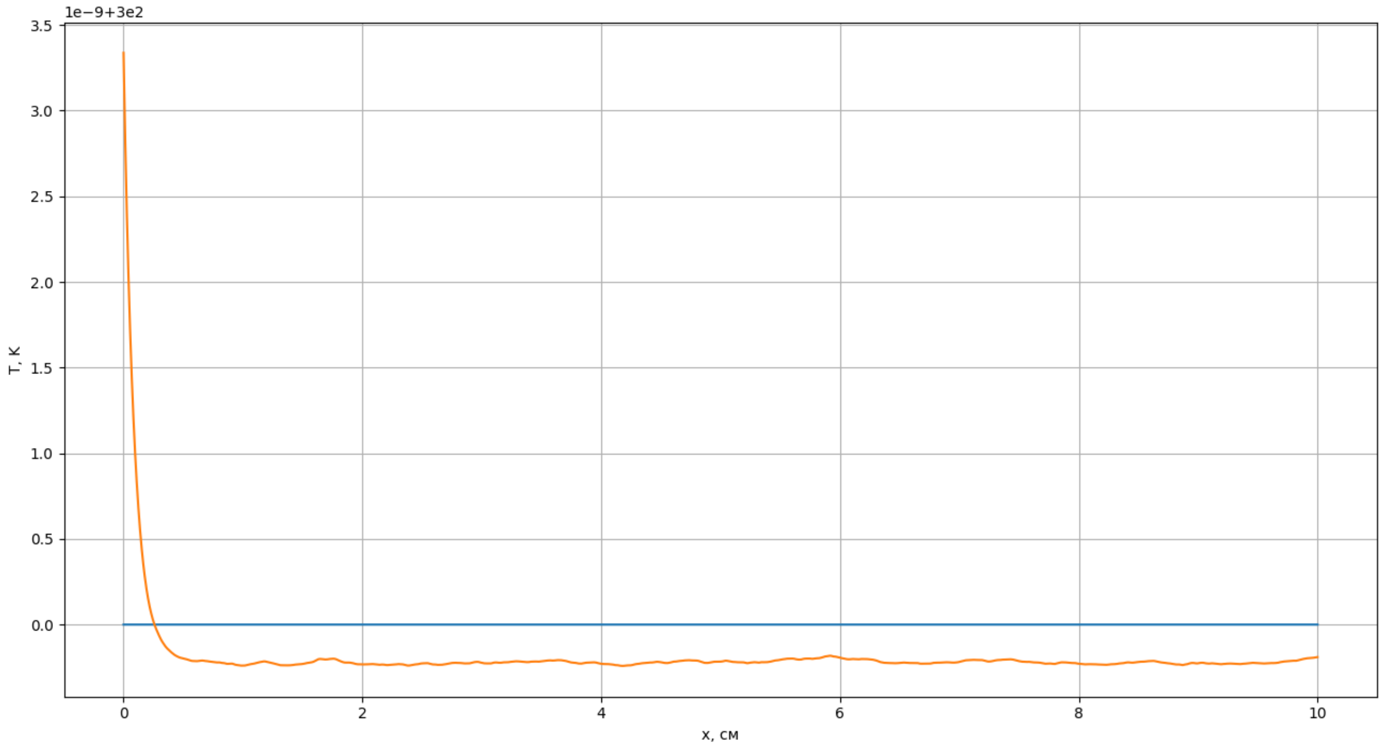

Рис. 5: График зависимости $T(x_n, t)$ при нулевом потоке.