Министерство образования и науки РФ
Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа программной инженерии

Отчёт по лабораторной работе №2
по дисциплине «Математические модели систем с распределёнными параметрами»

Выполнил студент гр. 3530904/00104 Смирнов Е. А.

Руководитель С. П. Воскобойников

Санкт-Петербург
2022

Оглавление

Постановка задачи

Номер Варианта **СР-3**.

Задание: используя интегро-интерполяционный метод (метод баланса), разработать программу для моделирования нестационарного распределения температуры в полом цилиндре, описываемого математической моделью вида

$$\frac{\partial u}{\partial t} = \left[\frac{1}{r}\frac{\partial}{\partial r}\left(rk(r,t)\frac{\partial u}{\partial r}\right) - q(r,t)u\right] + f(r,t)$$

$r \in [R_L,\ R_R],\ t \in [0,T],\ 0 < C_1 \le k(r,t) \le C_2,\ 0 \le q(r)$
с начальным условием $u_{t=0} = \varphi(r)$
с параметром $\chi_2(t) \ge 0$
с граничными условиями:

$$1)\ u_{r=R_L} = \vartheta_1(t)$$

$$2)\ -k\frac{\partial u}{\partial r}\bigg|_{r=R_R} = \chi_2(t)u(t)|_{r=R_R} - \vartheta_2(t)$$

Построение полудискретной модели

1. Дискретизация

Введем число N – число разбиений

$$r_0 < r_1 < \dots < r_N, r_i \in [R_l, R_R], r_0 = R_l, r_N = R_R$$

2. Интегро-интерполяционный метод

Основная сетка

$$\boldsymbol{h_i} = r_i - r_{i-1}, i = 1,2,\dots N$$

Вспомогательная сетка

- Разбиваем *каждый получившийся интервал пополам*

$$r_{i-\frac{1}{2}} = \frac{r_i + r_{i-1}}{2}, i = 1,2,\dots N$$

- Записываем шаг вспомогательной сетки ($\hbar$)

$$\hbar_i = \begin{cases} \dfrac{h_{i+1}}{2}, i = 0 \\ \dfrac{h_i + h_{i+1}}{2}, i = 1,2\dots N-1 \\ \dfrac{h_i}{2}, i = N \end{cases}$$

3. Аппроксимация основного уравнения

Введем обозначения для упрощения последующих записей:
точное решение - $u(r_i, t) = u_i$
приближенное решение $v(r_i, t) = v_i$

1. Умножим обе части исходного уравнения на r (интегрируем с весом r)

$$\frac{\partial u}{\partial t} r = \left[\frac{\partial}{\partial r}\left(rk(r,t)\frac{\partial u}{\partial r}\right) - q(r,t)ru\right] + rf(r,t)$$

2. Интегрируем исходное уравнение по вспомогательной сетке

$$\int_{r_{i-\frac{1}{2}}}^{r_{i+\frac{1}{2}}} \frac{\partial u}{\partial t} r dr = \int_{r_{i-\frac{1}{2}}}^{r_{i+\frac{1}{2}}} \left[\frac{\partial}{\partial r}\left(rk(r,t)\frac{\partial u}{\partial r}\right) - q(r,t)ru\right] dr + \int_{r_{i-\frac{1}{2}}}^{r_{i+\frac{1}{2}}} rf(r,t)dr$$

$$i = 1,2,\dots,N-1$$

$$\int_{r_{i-\frac{1}{2}}}^{r_{i+\frac{1}{2}}} \frac{\partial u}{\partial t} r dr = \left[rk(r,t)\frac{\partial u}{\partial r}\bigg|_{r=r_{i+1/2}} - rk(r,t)\frac{\partial u}{\partial r}\bigg|_{r=r_{i-\frac{1}{2}}} - \int_{r_{i-\frac{1}{2}}}^{r_{i+\frac{1}{2}}} rq(r,t)u(r)dr \right] + \int_{r_{i-\frac{1}{2}}}^{r_{i+\frac{1}{2}}} rf(r,t)dr$$

$$i = 1,2,\dots,N-1$$

2. Аппроксимируем (формула центральных разностей)

$$k(r,t)\frac{du_i}{dr}\bigg|_{r=r_{i-\frac{1}{2}}} \approx k_{i-\frac{1}{2}}\frac{u_i - u_{i-1}}{2\frac{h_i}{2}} = k_{i-\frac{1}{2}}\frac{u_i - u_{i-1}}{h_i}$$

3. Аппроксимируем (формула средних прямоугольников)

$$\int_{r_{i-\frac{1}{2}}}^{r_{i+\frac{1}{2}}} r\varphi(r)dr \approx \hbar_i\, r_i\, \varphi_i$$

4. Запишем получившуюся аппроксимацию уравнения для $i = 1,2,\dots,N-1$

$$\hbar_i \frac{d\boldsymbol{v_i}}{d\boldsymbol{t}}\boldsymbol{r_i} = \left[\boldsymbol{r}_{i+\frac{1}{2}}\boldsymbol{k}_{i+\frac{1}{2}}\frac{\boldsymbol{v_{i+1}} - \boldsymbol{v_i}}{\boldsymbol{h_{i+1}}} - \boldsymbol{r}_{i-\frac{1}{2}}\boldsymbol{k}_{i-\frac{1}{2}}\frac{\boldsymbol{v_i} - \boldsymbol{v_{i-1}}}{\boldsymbol{h_i}} - \hbar_i\,\boldsymbol{r_i}\,\boldsymbol{q_i}\boldsymbol{v_i}\right] + \hbar_i\,\boldsymbol{r_i}\,\boldsymbol{f_i},$$

$$i = 1,2,\dots,N-1$$

4. Аппроксимация граничных условий

1) условие 1. $u_{r=R_L} = \vartheta_1$

$$\mathrm{v_i} = \vartheta_1, \mathrm{i} = 0$$

2) условие 2. - $k(r,t)\frac{du}{dr}\Big|_{r=R_R} = \chi_2 u|_{r=R_r} - \vartheta_2$

Проинтегрируем основное уравнение в $i = N$ по вспомогательной сетке:

$$\int_{r_{i-\frac{1}{2}}}^{r_i} \frac{\partial u}{\partial t} r dr = \int_{r_{i-\frac{1}{2}}}^{r_i} \left[\frac{\partial}{\partial r}\left(rk(r,t)\frac{\partial u}{\partial r}\right) - q(r,t)ru\right] dr + \int_{r_{i-\frac{1}{2}}}^{r_i} rf(r,t)dr,\ i = N$$

$$\int_{r_{i-\frac{1}{2}}}^{r_i} \frac{\partial u}{\partial t} r dr = \left[rk(r,t)\frac{du}{dr}\bigg|_{r=r_i} - rk(r,t)\frac{du}{dr}\bigg|_{r=r_{i-\frac{1}{2}}} - \int_{r_{i-\frac{1}{2}}}^{r_i} rq(r,t)u(r)dr \right] + \int_{r_{i-\frac{1}{2}}}^{r_i} rf(r,t)dr,$$

Используя формулы центральных разностей и правых прямоугольников и, подставляя вместо производной - заданное граничное условие справа получаем:

$$\hbar_i \frac{\boldsymbol{dv_i}}{\boldsymbol{dt}} \boldsymbol{r_i} = \left[-\boldsymbol{r_i}(\boldsymbol{\chi_2 v_i} - \boldsymbol{\vartheta_2}) - \boldsymbol{r}_{\boldsymbol{i-\frac{1}{2}}} \boldsymbol{k}_{\boldsymbol{i-\frac{1}{2}}} \frac{\boldsymbol{v_i} - \boldsymbol{v_{i-1}}}{\boldsymbol{h_i}} - \hbar_{\boldsymbol{i}} \, \boldsymbol{r_i} \, \boldsymbol{q_i v_i} \right] + \hbar_i \, \boldsymbol{r_i} \, \boldsymbol{f_i}$$
$$i = N$$

5. Разностная схема

$$v_i = \vartheta_1 \qquad i = 0$$
$$\hbar_i \frac{dv_i}{dt} r_i = \left[r_{i+\frac{1}{2}} k_{i+\frac{1}{2}} \frac{v_{i+1} - v_i}{h_{i+1}} - r_{i-\frac{1}{2}} k_{i-\frac{1}{2}} \frac{v_i - v_{i-1}}{h_i} - \hbar_i \, r_i \, q_i v_i \right] + \hbar_i \, r_i \, f_i \qquad i = 1,2, \dots, N-1$$
$$\hbar_i \frac{dv_i}{dt} r_i = \left[-r_i(\chi_2 v_i - \vartheta_2) - r_{i-\frac{1}{2}} k_{i-\frac{1}{2}} \frac{v_i - v_{i-1}}{h_i} - \hbar_i \, r_i \, q_i v_i \right] + \hbar_i \, r_i \, f_i \,, \qquad i = N$$

Подставим граничное условие в первое уравнение.

$$\hbar_i \frac{dv_i}{dt} r_i = \left[r_{i+\frac{1}{2}} k_{i+\frac{1}{2}} \frac{v_{i+1} - v_i}{h_{i+1}} - r_{i-\frac{1}{2}} k_{i-\frac{1}{2}} \frac{v_i - \boldsymbol{\vartheta_1}}{h_i} - \hbar_i \, r_i \, q_i v_i \right] + \hbar_i \, r_i \, f_i \qquad i = 1$$
$$\hbar_i \frac{dv_i}{dt} r_i = \left[r_{i+\frac{1}{2}} k_{i+\frac{1}{2}} \frac{v_{i+1} - v_i}{h_{i+1}} - r_{i-\frac{1}{2}} k_{i-\frac{1}{2}} \frac{v_i - v_{i-1}}{h_i} - \hbar_i \, r_i \, q_i v_i \right] + \hbar_i \, r_i \, f_i \qquad i = 2,3 \; \dots, N-1$$
$$\hbar_i \frac{dv_i}{dt} r_i = \left[-r_i(\chi_2 v_i - \vartheta_2) - r_{i-\frac{1}{2}} k_{i-\frac{1}{2}} \frac{v_i - v_{i-1}}{h_i} - \hbar_i \, r_i \, q_i v_i \right] + \hbar_i \, r_i \, f_i \,, \qquad i = N$$

6. Матричное уравнение

Сгруппировав уравнения по производным одного порядка, систему можно записать в матричном виде. Введем следующие обозначения:

<u>Для $\mathrm{i} = 1$</u>

$$\boldsymbol{b_i} = \left(\frac{r_{i+\frac{1}{2}} k_{i+\frac{1}{2}}}{h_{i+1}} \right), \boldsymbol{g_i} = \vartheta_1 \frac{r_{i-\frac{1}{2}} k_{i-\frac{1}{2}}}{h_i} + \hbar_i \, r_i \, f_i, \boldsymbol{d_i} = \hbar_i r_i$$
$$\boldsymbol{c_i} = \left(-\frac{r_{i-\frac{1}{2}} k_{i-\frac{1}{2}}}{h_i} - \frac{r_{i+\frac{1}{2}} k_{i+\frac{1}{2}}}{h_{i+1}} - \hbar_i \, r_i \, q_i \right)$$

<u>Для $\mathrm{i} = 2,3 \; \dots, \mathrm{N} - 1$</u>

$$\boldsymbol{a_i} = \left(\frac{r_{i-\frac{1}{2}} k_{i-\frac{1}{2}}}{h_i} \right), \boldsymbol{c_i} = \left(-\frac{r_{i-\frac{1}{2}} k_{i-\frac{1}{2}}}{h_i} - \frac{r_{i+\frac{1}{2}} k_{i+\frac{1}{2}}}{h_{i+1}} - \hbar_i \, r_i \, q_i \right),$$
$$\boldsymbol{b_i} = \left(\frac{r_{i+\frac{1}{2}} k_{i+\frac{1}{2}}}{h_{i+1}} \right), \boldsymbol{g_i} = \hbar_i \, r_i \, f_i \,, \boldsymbol{d_i} = \hbar_i r_i$$

<u>Для $\mathrm{i} = \mathrm{N}$</u>

$$\boldsymbol{a_i} = \left(\frac{r_{i-\frac{1}{2}} k_{i-\frac{1}{2}}}{h_i} \right) \; \boldsymbol{c_i} = \left(-\frac{r_{i-\frac{1}{2}} k_{i-\frac{1}{2}}}{h_i} - \hbar_i \, r_i \, q_i - r_i \chi_2 \right), \boldsymbol{g_i} = \hbar_i \, r_i \, f_i + r_i \vartheta_2, \boldsymbol{d_i} = \hbar_i r_i$$

матрица $\boldsymbol{A} = \begin{bmatrix} c_1 & b_1 & & & & \\ a_2 & c_2 & b_2 & & & \\ & a_3 & c_3 & b_3 & & \\ & & & \dots & & \\ & & & & c_{N-1} & b_{N-1} \\ & & & & a_N & c_N \end{bmatrix}$

матрица $\boldsymbol{D} = \begin{vmatrix} \hbar_1 r_1 & & & & \\ & \hbar_2 r_2 & & & \\ & & \dots & & \\ & & & \hbar_{N-1} r_{N-1} & \\ & & & & \hbar_N r_N \end{vmatrix}$

векторы: $\boldsymbol{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \dots \\ v_{N-1} \\ v_N \end{bmatrix}$ и $\boldsymbol{g} = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ \dots \\ g_{N-1} \\ g_N \end{bmatrix}$ и $\boldsymbol{\varphi} = \begin{bmatrix} \varphi_1 \\ \varphi_2 \\ \varphi_3 \\ \dots \\ \varphi_{N-1} \\ \varphi_N \end{bmatrix}$

v, g $\in R^{(N+1)}$
A – (N) x (N)

$$\boldsymbol{D}\frac{\boldsymbol{dv}}{\boldsymbol{dt}} = \boldsymbol{Av} + \boldsymbol{g} \qquad \boldsymbol{v}|_{\boldsymbol{t=0}} = \boldsymbol{\varphi}$$

$$\frac{\boldsymbol{dv}}{\boldsymbol{dt}} = \boldsymbol{D^{-1}Av} + \boldsymbol{D^{-1}g} \qquad \boldsymbol{v}|_{\boldsymbol{t=0}} = \boldsymbol{\varphi}$$

Введем следующие обозначения
1. Ã = $D^{-1}A$
2. ġ = $D^{-1}g$

$$\frac{\boldsymbol{dv}}{\boldsymbol{dt}} = \tilde{\mathrm{A}}\boldsymbol{v} + \dot{\mathrm{g}} \qquad \boldsymbol{v}|_{\boldsymbol{t=0}} = \boldsymbol{\varphi}$$

Для нахождения решения воспользуемся явным и неявным методами ломаных Эйлера.

Введем **M** – число разбиений для переменной время(t):

$$t_0 < t_1 < \dots < t_M \,, t_k \in [0,T], t_0 = 0 \,, t_M = T$$

Основная сетка для t:

$$\boldsymbol{\tau_k} = t_k - t_{k-1} \,, k = 1,2, \dots M$$

Явный метод ломанных Эйлера

Аппроксимируем производную: $\left.\frac{dv}{dt}\right|_{t=t_k} = \frac{v_{k+1} - v_k}{\tau}$

$$\left.\frac{dv}{dt}\right|_{t=t_k} = [\tilde{A}(t)v + \dot{g}(t)]_{t=t_k}$$

$$\frac{v_{k+1} - v_k}{\tau} = \tilde{A}(t_k)v_k + \dot{g}(t_k)$$

$$v_{k+1} = v_k + \tau(\tilde{A}(t_k)v_k + \dot{g}(t_k))$$

$$\boldsymbol{v_{k+1}} = (\boldsymbol{E} + \boldsymbol{\tau}\tilde{A}(\boldsymbol{t_k}))\boldsymbol{v_k} + \boldsymbol{\tau}\dot{g}(\boldsymbol{t_k})$$

Решаем уравнение двумя циклами, один вложенный в другой, первый цикл - по первой переменной (k: [0, T]). В каждый момент времени(t_k) пробегаемся по второму циклу - по второй переменной (r: [$\mathrm{R_l}$, $\mathrm{R_R}$]).

Неявный метод ломанных Эйлера

Аппроксимируем производную: $\left.\frac{dv}{dt}\right|_{t=t_{k+1}} = \frac{v_{k+1} - v_k}{\tau}$

$$\left.\frac{dv}{dt}\right|_{t=t_{k+1}} = [\tilde{A}(t)v + \dot{g}(t)]_{t=t_{k+1}}$$

$$\frac{v_{k+1} - v_k}{\tau} = \tilde{A}(t_{k+1})v_{k+1} + \dot{g}(t_{k+1})$$

$$\frac{v_{k+1}}{\tau} - \tilde{A}(t_{k+1})v_{k+1} = \frac{v_k}{\tau} + \dot{g}(t_{k+1})$$

$$(\frac{\boldsymbol{1}}{\boldsymbol{\tau}}\boldsymbol{E} - \tilde{A}(\boldsymbol{t_{k+1}}))\boldsymbol{v_{k+1}} = \frac{\boldsymbol{v_k}}{\boldsymbol{\tau}} + \dot{g}(\boldsymbol{t_{k+1}})$$

Решаем получившуюся систему линейных алгебраических уравнений методом прогонки.

Тесты

Для всех тестов одинаково определены следующие параметры:
RL = 1 ; RR = 5; T = 5

Тест 1 Константный тест
$u = 1; k = 1; q = 1; \varphi = 1; \chi_2 = 2$

Находим:
$f = 1; \vartheta_1 = 1; \vartheta_2 = 2$

Явный метод

M N	1000	2000	4000	8000	10000
4	0,00000e+00	4,32987e-15	8,54872e-15	7,64552e-15	2,12053e-14
8	0,00000e+00	1,33227e-15	2,66454e-15	5,32907e-15	6,55032e-15
16	1,708748e+98	5,55112e-16	7,77156e-16	1,44329e-15	1,77636e-15
32	2,88979e+174	2,22045e-16	3,33067e-16	4,44089e-16	4,10783e-14

Неявный метод

M N	1000	2000	4000	8000	10000
4	1,11022e-16	0,00000e+00	0,00000e+00	2,53131e-14	1,11022e-16
8	2,10942e-15	7,77156e-16	1,11022e-16	2,64233e-14	6,99441e-15
16	4,44089e-16	1,66533e-15	9,99201e-16	2,59792e-14	3,10862e-15
32	1,36557e-14	1,22125e-15	3,33067e-16	1,11022e-15	9,99201e-16

Результаты теста показывают достаточно хороший результат. На данном тесте мы можем заметить сопоставимую погрешность. Однако в явном методе при N=32 можно увидеть, очень большую погрешность до e^+174; но при увеличении M погрешность снова становится небольшой.

Тест 2 Нелинейный (с зависимостью от r)

$u = r^2; k = 2 * r; q = r + 1; \varphi = r^2; \chi_2 = 2$

Находим:

$f = r^3 + r^2 - 12 * r; \vartheta_1 = 1; \vartheta_2 = 150$

Явный метод

M N	1000	2000	4000	8000	10000
4	8,05841e-01	8,05841e-01	8,05841e-01	8,05841e-01	8,05841e-01
8	1,63436e-02	1,63436e-02	1,63436e-02	1,63436e-02	1,63436e-02
16	Infinity	Infinity	2,65441e-03	2,65441e-03	2,65441e-03
32	Infinity	Infinity	Infinity	Infinity	Infinity

Неявный метод

M N	1000	2000	4000	8000	10000
4	8,05841e-01	8,05841e-01	8,05841e-01	8,05841e-01	8,05841e-01
8	1,63436e-02	1,63436e-02	1,63436e-02	1,63436e-02	1,63436e-02
16	2,65441e-03	2,65441e-03	2,65441e-03	2,65441e-03	2,65441e-03
32	6,29582e-04	6,29582e-04	6,29582e-04	6,29582e-04	6,29582e-04

Точность обоих методов сопоставима. Явный метод показывает взрыв погрешности из-за неустойчивости шага при N=16 и 32. В первом случае, погрешность при увеличении M до 4000 и далее становится допустимой. Проверим результаты для второго случая, увеличив M у явного метода.

M N	20000	50000	100000
32	6,29582e-04	6,29582e-04	6,29582e-04

Данный тест подтверждает зависимость устойчивости у явного метода от шага по времени.

Тест 3 С зависимостью от t и от r

$u = r * e^{-t}; k = 10 * e^{-t}; q = 10 * r * e^{-t}; \varphi = r; \chi_2 = 2$

Находим:

$f = 10 * e^{-2t}(r^2 - \frac{1}{r}) - r * e^{-t};\ \vartheta_1 = e^{-t}; \vartheta_2 = 10 * e^{-2t} +\ 10 * e^{-t}$

Явный метод

M N	1000	2000	4000	8000	10000
4	9,57612e-01	9,62303e-01	9,64702e-01	9,65930e-01	9,66178e-01
8	5,47384e-01	5,49988e-01	5,51317e-01	5,51990e-01	5,52127e-01
16	9,12000e+91	1,42515e+33	3,35935e-01	3,35631e-01	3,35112e-01
32	Infinity	Infinity	Infinity	5,85655e+133	5,78388e+46

Неявный метод

M N	1000	2000	4000	8000	10000
4	9,77599e-01	9,72274e-01	9,69685e-01	9,68421e-01	9,68170e-01
8	5,58424e-01	5,55486e-01	5,54062e-01	5,53363e-01	5,53225e-01
16	3,37938e-01	3,37555e-01	3,36950e-01	3,36539e-01	3,35812e-01
32	2,25695e-01	2,26670e-01	2,27179e-01	2,27437e-01	2,27489e-01

Из результатов теста можно сделать вывод, что погрешность в *явном* методе немного, но меньше, чем в *неявном*, следовательно, явный метод более точный по времени.

Результаты теста показывают, что для явного метода нужно брать маленький шаг разбиения N, чтобы не было результатов, как при N=32. Также не стоит забывать про шаг по времени для устойчивости решения должен быть меньше, чем (2 / $|\lambda|max$).

Вывод

В ходе лабораторной работы с помощью интегро-интерполяционного метода была разработана программа для моделирования распределения температуры в полом цилиндре, описываемом математической моделью с заданными граничными условиями.

Сравнили два метода решения системы: явный и неявный методы Эйлера. Выяснили, что явный метод Эйлера имеет более точные результаты по времени, но они не намного отличаются от неявного метода (влияние ошибок округления). Порядок точности у них одинаковый. Однако явный метод Эйлера имеет недостаток, проявляющийся в ограничении на шаг. Неявный метод Эйлера с довольно хорошей точностью результатов и не имеющий ограничений на шаг - более удобен в решении такого вида задач.

Приложение

Программа написана на языке Kotlin.

Код программы

```
class Solver (var N: Int, val M:Int, val RL: Double, val RR: Double, T:
Double, val Xi2: Double) {
    val a: Array<Double> = Array(N) { 0.0 }
    val b: Array<Double> = Array(N) { 0.0 }
    val c: Array<Double> = Array(N) { 0.0 }
    var d: Array<Double> = Array(N) { 0.0 }

    var v: Array<Array<Double>> = Array(N) { Array(M+1) {0.0} }
    var U: Array<Array<Double>> = Array(N) { Array(M+1) {0.0} }
    var g: Array<Double> = Array(N) { 0.0 }

    private val h:Double = (RR - RL)/N
    private val h_h:Double = h/2
    val tay: Double = T / M

    private val t: Array<Double> = Array(M+1){0.0}
    val r: Array<Double> = Array(N+1){0.0}
    private val r_h: Array<Double> = Array(N+1){0.0}

    lateinit var u: ((Double, Double) -> Double)
    lateinit var k: ((Double, Double) -> Double)
    lateinit var q: ((Double, Double) -> Double)
    lateinit var fi: ((Double) -> Double)

    lateinit var f: ((Double, Double) -> Double)
    lateinit var nu1: ((Double) -> Double)
    lateinit var nu2: ((Double) -> Double)

    /**
     * Заполняет значения векторов r и h, t для основной сетки и
вспомогательной.
     *
     * Массивы r и h нужны для ИИМ (интегро-интерполяционного метода)
     */
    private fun initNet() {
        // основная сетка h
        for (i in 0 .. N)
            r[i] = RL + i*h

        // вспомогательная сетка
        r_h[0] = 0.0
        for (i in 1 .. N)
            r_h[i] = (r[i] + r[i - 1]) / 2.0

        // основная сетка t
        for (i in 0..M)
            t[i] = 0.0 + i*tay
    }

    /**
     * Заполняет матрицы A и D и вектор g.
     */
    private fun initMatrix(time: Double, method: Int) {
        b[0] = ((r_h[2] * k(r_h[2], time)) / h)
        c[0] = - (r_h[1] * k(r_h[1], time)) / h - (r_h[2] * k(r_h[2], time))
/ h - h_h*r[1]*q(r[1], time)
```

```
        d[0] = 1.0 / (h_h * r[1])
        g[0] = nu1(time) * (r_h[1] * k(r_h[1], time)) / h + h_h * r[1] *
f(r[1], time)

        // основная часть СЛАУ
        for (i in 1 until N - 1) {
            a[i] = ((r_h[i+1] * k(r_h[i+1], time)) / h)
            c[i] = ( -(r_h[i+1] * k(r_h[i+1], time))/h - (r_h[i + 2] *
k(r_h[i + 2], time))/h - h_h * r[i+1] * q(r[i+1], time))
            b[i] = ((r_h[i + 2] * k(r_h[i + 2], time)) / h)
            g[i] = h_h * r[i+1] * f(r[i+ 1], time)
            d[i] = 1.0 / (h_h * r[i+1])
        }

        // граничное условие справа - последняя строчка СЛАУ
        a[N - 1] = ((r_h[N] * k(r[N], time)) / h)
        c[N - 1] = (-(r_h[N] * k(r[N], time)) / h - (h / 2.0) * r[N] *
q(r[N], time) - r[N] * Xi2)
        g[N - 1] = (h / 2.0) * r[N] * f(r[N], time) + r[N] * nu2(time)
        d[N - 1] = 2.0 / ((h) * r[N])

        if (method == 1) {
            // Приводим векторы a, b, c, g к форме записи c Ãv + ġ
            for (i in 0 until N) {
                if (i != 0) a[i] = a[i] * d[i]
                c[i] = c[i] * d[i]
                b[i] = b[i] * d[i]
                g[i] = g[i] * d[i]

                if (i != 0) a[i] *= tay
                c[i] = c[i] * tay + 1.0
                b[i] *= tay
                g[i] *= tay
            }
        }
        if (method == 2) {
            // Приводим векторы a, b, c, g к форме записи c Ãv + ġ
            for (i in 0 until N) {
                if (i != 0) a[i] = a[i] * d[i]
                c[i] = c[i] * d[i]
                b[i] = b[i] * d[i]
                g[i] = g[i] * d[i]

                if (i != 0) a[i] *= -tay
                c[i] = 1.0 - c[i] * tay
                b[i] *= -tay
                g[i] *= tay
            }
        }
    }

    /**
     * Находит решение матричного уравнения Av=g методом 'прогонки' (прямой и
обратный ход)
     *
     * Решение ищется в виде v(i)=α(i+1)*v(i+1)+β(i+1)
     * @param A матрица размера N*N
     * @param g вектор размера N
     *
     * @return v вектор решений
     */
    fun progonka(aV: Array<Double>, cV: Array<Double>, bV: Array<Double>, gV:
Array<Double>, size: Int): Array<Double> {
        // прогоночные коэффициенты alpha и beta
```

```
        val alpha: Array<Double> = Array(size+1) { 0.0 }
        val beta: Array<Double> = Array(size+1) { 0.0 }
        val res: Array<Double> = Array(size+1) { 0.0 }

        // прямой ход - считаем коэффициенты alpha и beta
        alpha[1] = -bV[0] / cV[0]
        beta[1] = gV[0] / cV[0]

        for (i in 1 until size) {
            alpha[i + 1] = -(bV[i]) / (alpha[i] * aV[i] + cV[i])
            beta[i + 1] = (gV[i] - aV[i] * beta[i]) / (aV[i] * alpha[i] +
cV[i])
        }

        // обратный ход - находим веткор решений v
        res[size] = ((gV[size] - aV[size] * beta[size]) / (aV[size] *
alpha[size] + cV[size]))
        for (i in size-1 downTo 0 step 1) {
            res[i] = alpha[i + 1] * res[i + 1] + beta[i + 1]
        }

        return res
    }

    /** явный метод Эйлера **/
    fun explicitEuler() {
        var method = 1
        for (i in 0 until N) v[i][0] = fi(r[i])
        for (k in 1..M) {
            initMatrix(t[k-1], method)

            v[0][k] = c[0] * v[0][k - 1] + b[0] * v[1][k - 1] + g[0]
            for (i in 1 until N-1)
                v[i][k] = a[i] * v[i - 1][k - 1] + c[i] * v[i][k - 1] + b[i]
* v[i + 1][k - 1] + g[i]
            v[N-1][k] = a[N-1] * v[N-2][k - 1] + c[N-1] * v[N-1][k - 1] +
g[N-1]
        }
    }

    /** неявный метод Эйлера **/
    fun implicitEuler() {
        val method = 2
        for (i in 0 until N) v[i][0] = fi(r[i])
        for (k in 1 .. M) {
            initMatrix(t[k], method)

            val G = Array(N){0.0}
            for (i in 0 until N) G[i] = v[i][k-1] + g[i]

            val tmp = progonka(a, c, b, G, N-1)

            for (i in 0 until N) v[i][k] = tmp[i]
        }
    }

    fun init(u_: (Double, Double) -> Double, k_: (Double, Double) -> Double,
q_: (Double, Double) -> Double, fi_: (Double) -> Double,
             f_: (Double, Double) -> Double, nu1_: (Double) -> Double, nu2_ :
(Double) -> Double ) {
        u = u_
        k = k_
        q = q_
        fi = fi_
```

```
        f = f_
        nu1 = nu1_
        nu2 = nu2_

        initNet()
    }

    /**
     * Считает невязку [u-v] для каждой точки.
     * Находит максимальное отклонение по модулю, что и считается за
погрешность всего решения
     */
    fun printEps() {
        for (i in 1 until N) for (k in 1..M) U[i][k] = u(r[i], t[k])

        var maxEps = -1.0
        for(i in 1 until N) for (k in 1..M)
            maxEps = kotlin.math.max(maxEps, kotlin.math.abs(U[i][k] -
v[i][k]))
        println(String.format("eps = %.5e", maxEps))
    }
    companion object {
        fun printVector(v: Array<Double>, ch: Char) {
            print("$ch = ")
            v.forEach { i -> print(String.format("%2.6f; ", i)) }
            println()
        }
    }
}
```

Код тестирования

```
import java.lang.Math.exp
import java.lang.Math.pow
import java.util.*
import kotlin.math.pow

fun testConstant() {
    val RL = 1.0
    val RR = 5.0
    val T = 5.0
    val Xi2 = 2.0

    val u: (Double, Double) -> Double = { r, t -> 1.0 }
    val k: (Double, Double) -> Double = { r, t -> 1.0 }
    val q: (Double, Double) -> Double = { r, t -> 1.0 }
    val fi: (Double) -> Double = { r -> 1.0 }

    val f: (Double, Double) -> Double = { r, t -> 1.0 }
    val nu1: (Double) -> Double = { t -> 0.0 }
    val nu2: (Double) -> Double = { t -> 2.0 }

    val N = listOf(4, 8, 16, 32)
    val M = listOf(1000, 2000, 4000, 8000, 10000)
    println("Явный метод")
    for (n in N) {
        for (m in M) {
            val solver = Solver(n, m, RL, RR, T, Xi2)
            solver.init(u, k, q, fi, f, nu1, nu2)

            solver.explicitEuler()
```

```
            print("($n;$m) "); solver.printEps()
        }
    }
    println("\nНеявный метод")
    for (n in N) {
        for (m in M) {
            val solver = Solver(n, m, RL, RR, T, Xi2)
            solver.init(u, k, q, fi, f, nu1, nu2)

            solver.implicitEuler()
            print("($n;$m) "); solver.printEps()
        }
    }
}

fun testR() {
    val RL = 1.0
    val RR = 5.0
    val T = 5.0
    val Xi2 = 2.0

    val u: (Double, Double) -> Double = { r, t -> r * r }
    val k: (Double, Double) -> Double = { r, t -> 2.0 * r }
    val q: (Double, Double) -> Double = { r, t -> r + 1.0 }
    val fi: (Double) -> Double = { r -> r * r }

    val f: (Double, Double) -> Double = { r, t -> r * r * r + r * r - 12.0 *
r }
    val nu1: (Double) -> Double = { t -> 1.0 }
    val nu2: (Double) -> Double = { t -> 150.0 }

    val N = listOf(4, 8, 16, 32)
    val M = listOf(1000, 2000, 4000, 8000, 10000)
    println("Явный метод")
    for (n in N) {
        for (m in M) {
            val solver = Solver(n, m, RL, RR, T, Xi2)
            solver.init(u, k, q, fi, f, nu1, nu2)

            solver.explicitEuler()
            print("$n : $m "); solver.printEps()
        }
    }
    println("\nНеявный метод")
    for (n in N) {
        for (m in M) {
            val solver = Solver(n, m, RL, RR, T, Xi2)
            solver.init(u, k, q, fi, f, nu1, nu2)

            solver.implicitEuler()
            print("$n : $m "); solver.printEps()
        }
    }
}

fun testT() {
    val RL = 1.0
    val RR = 5.0
    val T = 5.0
    val Xi2 = 2.0

    val u: (Double, Double) -> Double = { r, t -> Math.E.pow(-t) }
    val k: (Double, Double) -> Double = { r, t -> Math.E.pow(-t) }
    val q: (Double, Double) -> Double = { r, t -> Math.E.pow(-t) }
```

```
    val fi: (Double) -> Double = { r -> 1.0 }

    val f: (Double, Double) -> Double = { r, t -> Math.E.pow(-t*2.0) -
Math.E.pow(-t)}
    val nu1: (Double) -> Double = { t -> Math.E.pow(-t) }
    val nu2: (Double) -> Double = { t -> 2.0*Math.E.pow(-t) }

    val N = listOf(4, 8, 16, 32)
    val M = listOf(1000, 2000, 4000, 8000, 10000)
    println("Явный метод")
    for (n in N) {
        for (m in M) {
            val solver = Solver(n, m, RL, RR, T, Xi2)
            solver.init(u, k, q, fi, f, nu1, nu2)

            solver.explicitEuler()
            print("$n : $m "); solver.printEps()
        }
    }
    println("\nНеявный метод")
    for (n in N) {
        for (m in M) {
            val solver = Solver(n, m, RL, RR, T, Xi2)
            solver.init(u, k, q, fi, f, nu1, nu2)

            solver.implicitEuler()
            print("$n : $m "); solver.printEps()
        }
    }
}

fun testRT() {
    val RL = 1.0
    val RR = 5.0
    val T = 5.0
    val Xi2 = 2.0

    val u: (Double, Double) -> Double = { r, t -> r*Math.E.pow(-t) }
    val k: (Double, Double) -> Double = { r, t -> 10.0*Math.E.pow(-t) }
    val q: (Double, Double) -> Double = { r, t -> 10.0*r*Math.E.pow(-t) }
    val fi: (Double) -> Double = { r -> r }

    val f: (Double, Double) -> Double = { r, t -> 10.0*Math.E.pow(-
t*2.0)*(r*r - 1/r) - r*Math.E.pow(-t)  }
    val nu1: (Double) -> Double = { t -> Math.E.pow(-t) }
    val nu2: (Double) -> Double = { t -> 10.0*Math.E.pow(-t*2.0) +
10.0*Math.E.pow(-t) }

    val N = listOf(4, 8, 16, 32)
    val M = listOf(1000, 2000, 4000, 8000, 10000)
    println("Явный метод")
    for (n in N) {
        for (m in M) {
            val solver = Solver(n, m, RL, RR, T, Xi2)
            solver.init(u, k, q, fi, f, nu1, nu2)

            solver.explicitEuler()
            print("$n : $m "); solver.printEps()
        }
    }
    println("\nНеявный метод")
    for (n in N) {
        for (m in M) {
            val solver = Solver(n, m, RL, RR, T, Xi2)
```

```
            solver.init(u, k, q, fi, f, nu1, nu2)

            solver.implicitEuler()
            print("$n : $m "); solver.printEps()
        }
    }
}

fun main(){
    println("Константный тест")
    testConstant()
    println("Нелинейный тест по r")
    testR()
    println("Тест по t")
    testT()
    println("Тест с зависимость по r и t")
    testRT()
}
```