МИНИСТЕРСТВО НАУКИ
И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

НГТУ НЭТИ | Факультет прикладной математики и информатики

Кафедра прикладной математики

Курсовая работа № 1

по дисциплине «Уравнения математической физики»

ПРИМЕНЕНИЕ МКЭ ДЛЯ НЕСТАЦИОНАРНЫХ ЗАДАЧ

Группа ПМ-05 ГРУШЕВ АНДРЕЙ

Вариант 6

Новосибирск, 2023

I. Условие задачи:

МКЭ для гиперболического уравнения в декартовой система координат. Неявная трехслойная схема для аппроксимации по времени. Базисные функции линейные на треугольниках.

II. Метод решения:

Конечноэлементная аппроксимация

Исходное уравнение имеет вид:

$$\chi \frac{\partial^2 \mathrm{u}}{\partial \mathrm{t}^2} + \sigma \frac{\partial \mathrm{u}}{\partial \mathrm{t}} - \ div(\lambda grad(u)) = f$$

Неявная трёхслойная схема аппроксимации по времени:

$$\chi \frac{u^j - 2u^{j-1} + u^{j-2}}{\Delta t^2} + \sigma \frac{u^j - u^{j-2}}{2\Delta t} - div(\lambda grad(u^j)) = f^j$$

Будем полагать, что ось времени t разбита на так называемые временные слои значениями t_{j}, а значения искомой функции u и параметров λ, χ, σ и f уравнения на j-ом временном слое будем обозначать через u^{j}, λ^{j}, χ^{j}, σ^{j}, f^{j}.

Помимо краевых условий, начально краевая-задача должна включать два начальных условия:

$$u|_{t\ =\ t_0} = u^0; u|_{t\ =\ t_1} = u^1$$

Рассмотрим процедуру построения неявной трёхслойной схемы для решения дифференциального уравнения гиперболического типа.

Представим искомое решение u на интервале () в следующем виде:

$$u\ =\ u^{j-2}\eta_2^j(t)\ + u^{j-1}\eta_1^j(t) +\ u^j\eta_0^j(t)$$

Данное соотношение определяет аппроксимацию функции u по времени как квадратичный интерполянт её значений на временных слоях $\mathrm{t} = \mathrm{t}_{\mathrm{j-2}}, \mathrm{t} = \mathrm{t}_{\mathrm{j-1}}$ и $\mathrm{t} = \mathrm{t}_{\mathrm{j}}$.

Функции $\eta_2^j(t), \eta_1^j(t), \eta_0^j(t)$ могут быть записаны в виде:

$$\eta_2^j(t) = \frac{1}{\Delta t_1 \Delta t}\left(t - t_{j-1}\right)\left(t - t_j\right),$$

$$\eta_1^j(t) = -\frac{1}{\Delta t_1 \Delta t_0}(t - t_{j-2})(t - t_j),$$

$$\eta_0^j(t) = \frac{1}{\Delta t \Delta t_0}(t - t_{j-2})(t - t_{j-1}),$$

где

$$\Delta t = t - t_{j-2}, \Delta t_1 = t_{j-1} - t_{j-2}, \Delta t_0 = t - t_{j-1}.$$

Применим данное представление для аппроксимации производной по времени гиперболического уравнения на временном слое $t = t_j$.

$$\chi \frac{\partial^2 (u^{j-2}\eta_2^j(t) \; + u^{j-1}\eta_1^j(t) + \; u^j \eta_0^j(t))}{\partial \mathrm{t}^2} + \sigma \frac{\partial (u^{j-2}\eta_2^j(t) \; + u^{j-1}\eta_1^j(t) + \; u^j \eta_0^j(t))}{\partial \mathrm{t}} - \; div(\lambda grad(u^{\mathrm{j}})) = f^{\mathrm{j}}$$

Вычислим производные по t.

$$\frac{d\eta_2^j(t)}{dt} = \frac{\Delta t_0}{\Delta t_1 \Delta t}, \frac{d\eta_1^j(t)}{dt} = -\frac{\Delta t}{\Delta t_1 \Delta t_0}, \frac{d\eta_0^j(t)}{dt} = \frac{\Delta t + \Delta t_0}{\Delta t_0 \Delta t}$$

$$\frac{d^2\eta_2^j(t)}{dt^2} = \frac{2}{\Delta t_1 \Delta t}, \frac{d^2\eta_1^j(t)}{dt^2} = -\frac{2}{\Delta t_1 \Delta t_0}, \frac{d^2\eta_0^j(t)}{dt^2} = \frac{2}{\Delta t_0 \Delta t}$$

С учётом полученных выражений гиперболическое уравнение может быть переписано в виде:

$$\chi \left(\frac{2}{\Delta t_1 \Delta t} u^{j-2} - \frac{2}{\Delta t_1 \Delta t_0} u^{j-1} + \frac{2}{\Delta t \Delta t_0} u^j \right) + \sigma \left(\frac{\Delta t_0}{\Delta t_1 \Delta t} u^{j-2} - \frac{\Delta t}{\Delta t_1 \Delta t_0} u^{j-1} + \frac{\Delta t + \Delta t_0}{\Delta t \Delta t_0} u^j \right) - div(\lambda grad(u^j)) = f^j$$

Выполняя конечноэлементную аппроксимацию, получим СЛАУ следующего вида:

$$
\left(M^\chi \frac{2}{\Delta t \Delta t_0} + M^\sigma \frac{\Delta t + \Delta t_0}{\Delta t \Delta t_0} - G\right) q^j
= b_j - q^{j-2} M^\chi \frac{2}{\Delta t_1 \Delta t} + q^{j-1} M^\chi \frac{2}{\Delta t_1 \Delta t_0} - q^{j-2} M^\sigma \frac{\Delta t_0}{\Delta t_1 \Delta t}
+ q^{j-1} M^\sigma \frac{\Delta t}{\Delta t_1 \Delta t_0}.
$$

Переход к локальным матрицам

Рассмотрим треугольник Ω_m с вершинами $(\hat{x}_1, \hat{y}_1)$, $(\hat{x}_2, \hat{y}_2)$ и $(\hat{x}_3, \hat{y}_3)$.

На каждом элементе Ω_m треугольной сетки определим три локальные базисные функции:

$$
\hat{\psi}_i(x, y) = \alpha_0^i + \alpha_1^i x + \alpha_2^i y, \quad i = 1 \dots 3
$$

Такие, что функция $\hat{\psi}_1$ равна единице в вершине $(\hat{x}_1, \hat{y}_1)$ и нулю в двух других, $\hat{\psi}_2$ равна единице в вершине $(\hat{x}_2, \hat{y}_2)$ и нулю в двух остальных, а $\hat{\psi}_3$ равна единице в третьей вершине $(\hat{x}_3, \hat{y}_3)$ и нулю в двух остальных.

Определенные таким образом локальные базисные функции $\hat{\psi}_i$ на треугольнике Ω_m фактически являются $\mathcal{L}$ координатами этого треугольника, т.е. коэффициенты α функций $\hat{\psi}_i$ могут быть вычислены по формуле:

$$
\alpha = \begin{pmatrix} \alpha_0^1 & \alpha_1^1 & \alpha_2^1 \\ \alpha_0^2 & \alpha_1^2 & \alpha_2^2 \\ \alpha_0^3 & \alpha_1^3 & \alpha_2^3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ \hat{x}_1 & \hat{x}_2 & \hat{x}_3 \\ \hat{y}_1 & \hat{y}_2 & \hat{y}_3 \end{pmatrix}^{-1}
$$

Представив соотношение в матричном виде, с учетом коэффициентов α получим

$$
\begin{pmatrix} 1 & 1 & 1 \\ \hat{x}_1 & \hat{x}_2 & \hat{x}_3 \\ \hat{y}_1 & \hat{y}_2 & \hat{y}_3 \end{pmatrix} \begin{pmatrix} \mathcal{L}_1 \\ \mathcal{L}_2 \\ \mathcal{L}_3 \end{pmatrix} = \begin{pmatrix} 1 \\ x \\ y \end{pmatrix}
$$

Для удобного расчета интегралов от $\mathcal{L}$-координат будем использовать следующее соотношение:

$$
\int_{\Omega_k} (\mathcal{L}_1)^{v_1} (\mathcal{L}_2)^{v_2} (\mathcal{L}_3)^{v_3} dr dz = \frac{v_1! \, v_2! \, v_3!}{(v_1 + v_2 + v_3 + 2)!} |det\boldsymbol{D}|
$$

$$
detD = (\hat{x}_2 - \hat{x}_1) \cdot (\hat{y}_3 - \hat{y}_1) - (\hat{x}_3 - \hat{x}_1) \cdot (\hat{y}_2 - \hat{y}_1)
$$

Компоненты локальных матриц в декартовой системе координат имеют вид:

Компоненты локальной матрицы жесткости:

$$\hat{G}_{ij} = \int_{\Omega_k} \lambda \left(\frac{\partial \hat{\psi}_i}{\partial x} \frac{\partial \hat{\psi}_j}{\partial x} + \frac{\partial \hat{\psi}_i}{\partial y} \frac{\partial \hat{\psi}_j}{\partial y} \right) dxdy$$

Компоненты локальной матрицы массы:

$$\hat{M}_{ij} = \int_{\Omega_k} \gamma \, \hat{\psi}_i \hat{\psi}_j dxdy$$

Получим выражения для локальных матриц жесткости и массы конечного элемента Ω_m. Учитывая представление функций и заменяя параметр λ некоторым постоянным на конечном элементе Ω_m значением $\bar{\lambda}$, получим

$$\hat{G}_{ij} = \bar{\lambda} \tfrac{|det\ D|}{2} \left(\alpha_1^i \alpha_1^j + \alpha_2^i \alpha_2^j \right), \ i = 1 \dots 3, \ j = 1 \dots 3$$

$$\hat{M} = \frac{\bar{\gamma} |det\, D|}{24} \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}$$

Локальный вектор правой части:

$$\hat{b}_i = \int_{\Omega_k} f \, \hat{\psi}_i dxdy$$

Для вычисления локального вектора правой части $\hat{b}$, требуется разложение вектора f − правой части соответствующего дифференциального уравнения.

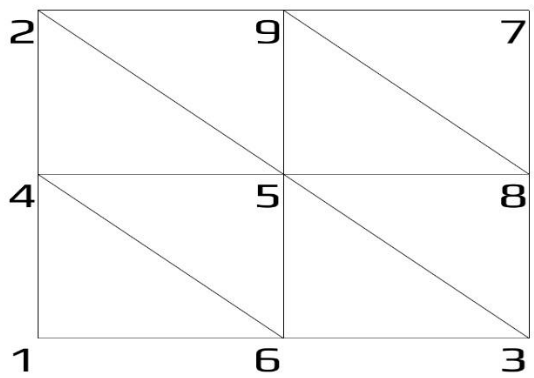
$$f = f_1 \hat{\psi}_1 + f_2 \hat{\psi}_2 + f_3 \hat{\psi}_3$$

В таком случае компоненты правой части вычисляются следующим образом:

$$\hat{b}_i = \int_{\Omega_k} \left(f_1 \hat{\psi}_1 + f_2 \hat{\psi}_2 + f_3 \hat{\psi}_3 \right) \cdot \hat{\psi}_i dxdy, \ \ i = \overline{1,3}$$

III. Тестирование

Для тестирования программы использовалась следующая сетка



X изменяется от 1 до 9 с шагом 4.

Y изменяется от 1 до 5 с шагом 2.

Первые краевые условия заданы на всех границах и имеют значение функции.

1. $u^{*} = t, \lambda = 1, \sigma = 2, \chi = 2\ , f = \ 2, t = [0, 1, 2, 3, 4]$

t = 2

x	y	u^{*}	$u^{\text{практ.}}$	$\lvert u^{*} - u^{\text{практ.}} \rvert$	$\frac{\lVert u^{*} - u^{\text{практ.}} \rVert}{\lVert u^{*} \rVert}$
1,000000E+000	1,000000E+000	2,000000E+000	2,000000E+000	0,000000E+000	
1,000000E+000	5,000000E+000	2,000000E+000	2,000000E+000	0,000000E+000	
9,000000E+000	1,000000E+000	2,000000E+000	2,000000E+000	0,000000E+000	
1,000000E+000	3,000000E+000	2,000000E+000	2,000000E+000	0,000000E+000	
5,000000E+000	3,000000E+000	2,000000E+000	2,000000E+000	4,440892E-016	7,401487E-017
5,000000E+000	1,000000E+000	2,000000E+000	2,000000E+000	0,000000E+000	
9,000000E+000	5,000000E+000	2,000000E+000	2,000000E+000	0,000000E+000	
9,000000E+000	3,000000E+000	2,000000E+000	2,000000E+000	0,000000E+000	
5,000000E+000	5,000000E+000	2,000000E+000	2,000000E+000	0,000000E+000	

t = 3

x	y	u^{*}	$u^{\text{практ.}}$	$\lvert u^{*} - u^{\text{практ.}} \rvert$	$\frac{\lVert u^{*} - u^{\text{практ.}} \rVert}{\lVert u^{*} \rVert}$
1,000000E+000	1,000000E+000	3,000000E+000	3,000000E+000	0,000000E+000	
1,000000E+000	5,000000E+000	3,000000E+000	3,000000E+000	0,000000E+000	
9,000000E+000	1,000000E+000	3,000000E+000	3,000000E+000	0,000000E+000	
1,000000E+000	3,000000E+000	3,000000E+000	3,000000E+000	0,000000E+000	
5,000000E+000	3,000000E+000	3,000000E+000	3,000000E+000	8,881784E-016	9,868649E-017
5,000000E+000	1,000000E+000	3,000000E+000	3,000000E+000	0,000000E+000	
9,000000E+000	5,000000E+000	3,000000E+000	3,000000E+000	0,000000E+000	
9,000000E+000	3,000000E+000	3,000000E+000	3,000000E+000	0,000000E+000	
5,000000E+000	5,000000E+000	3,000000E+000	3,000000E+000	0,000000E+000	

t = 4

x	y	u^{*}	$u^{\text{практ.}}$	$\lvert u^{*} - u^{\text{практ.}} \rvert$	$\frac{\lVert u^{*} - u^{\text{практ.}} \rVert}{\lVert u^{*} \rVert}$
1,000000E+000	1,000000E+000	4,000000E+000	4,000000E+000	0,000000E+000	
1,000000E+000	5,000000E+000	4,000000E+000	4,000000E+000	0,000000E+000	
9,000000E+000	1,000000E+000	4,000000E+000	4,000000E+000	0,000000E+000	
1,000000E+000	3,000000E+000	4,000000E+000	4,000000E+000	0,000000E+000	7,401487E-017
5,000000E+000	3,000000E+000	4,000000E+000	4,000000E+000	8,881784E-016	
5,000000E+000	1,000000E+000	4,000000E+000	4,000000E+000	0,000000E+000	
9,000000E+000	5,000000E+000	4,000000E+000	4,000000E+000	0,000000E+000	

9,000000E+000	3,000000E+000	4,000000E+000	4,000000E+000	0,000000E+000	
5,000000E+000	5,000000E+000	4,000000E+000	4,000000E+000	0,000000E+000	

2. $u^{*} = xt, \lambda = 1, \sigma = 2, \chi = 2\ , f\ =\ 2x, t\ =\ [0, 1, 2, 3, 4]$

t = 2

x	y	u^{*}	$u^{\text{практ.}}$	$\lvert u^{*} - u^{\text{практ.}} \rvert$	$\frac{\lVert u^{*} - u^{\text{практ.}} \rVert}{\lVert u^{*} \rVert}$
1,000000E+000	1,000000E+000	2,000000E+000	2,000000E+000	0,000000E+000	0,000000E+000
1,000000E+000	5,000000E+000	2,000000E+000	2,000000E+000	0,000000E+000	
9,000000E+000	1,000000E+000	1,800000E+001	1,800000E+001	0,000000E+000	
1,000000E+000	3,000000E+000	2,000000E+000	2,000000E+000	0,000000E+000	
5,000000E+000	3,000000E+000	1,000000E+001	1,000000E+001	0,000000E+000	
5,000000E+000	1,000000E+000	1,000000E+001	1,000000E+001	0,000000E+000	
9,000000E+000	5,000000E+000	1,800000E+001	1,800000E+001	0,000000E+000	
9,000000E+000	3,000000E+000	1,800000E+001	1,800000E+001	0,000000E+000	
5,000000E+000	5,000000E+000	1,000000E+001	1,000000E+001	0,000000E+000	

t = 3

x	y	u^{*}	$u^{\text{практ.}}$	$\lvert u^{*} - u^{\text{практ.}} \rvert$	$\frac{\lVert u^{*} - u^{\text{практ.}} \rVert}{\lVert u^{*} \rVert}$
1,000000E+000	1,000000E+000	3,000000E+000	3,000000E+000	0,000000E+000	3,304886E-017
1,000000E+000	5,000000E+000	3,000000E+000	3,000000E+000	0,000000E+000	
9,000000E+000	1,000000E+000	2,700000E+001	2,700000E+001	0,000000E+000	
1,000000E+000	3,000000E+000	3,000000E+000	3,000000E+000	0,000000E+000	
5,000000E+000	3,000000E+000	1,500000E+001	1,500000E+001	1,776357E-015	
5,000000E+000	1,000000E+000	1,500000E+001	1,500000E+001	0,000000E+000	
9,000000E+000	5,000000E+000	2,700000E+001	2,700000E+001	0,000000E+000	
9,000000E+000	3,000000E+000	2,700000E+001	2,700000E+001	0,000000E+000	
5,000000E+000	5,000000E+000	1,500000E+001	1,500000E+001	0,000000E+000	

t = 4

x	y	u^{*}	$u^{\text{практ.}}$	$\lvert u^{*} - u^{\text{практ.}} \rvert$	$\frac{\lVert u^{*} - u^{\text{практ.}} \rVert}{\lVert u^{*} \rVert}$
1,000000E+000	1,000000E+000	4,000000E+000	4,000000E+000	0,000000E+000	9,914657E-017
1,000000E+000	5,000000E+000	4,000000E+000	4,000000E+000	0,000000E+000	
9,000000E+000	1,000000E+000	3,600000E+001	3,600000E+001	0,000000E+000	
1,000000E+000	3,000000E+000	4,000000E+000	4,000000E+000	0,000000E+000	
5,000000E+000	3,000000E+000	2,000000E+001	2,000000E+001	7,105427E-015	
5,000000E+000	1,000000E+000	2,000000E+001	2,000000E+001	0,000000E+000	
9,000000E+000	5,000000E+000	3,600000E+001	3,600000E+001	0,000000E+000	
9,000000E+000	3,000000E+000	3,600000E+001	3,600000E+001	0,000000E+000	
5,000000E+000	5,000000E+000	2,000000E+001	2,000000E+001	0,000000E+000	

3. $u^{*} = xyt, \lambda = 1, \sigma = 2, \chi = 2\ , f\ =\ 2xy, t\ =\ [0, 1, 2, 3, 4]$

t = 2

x	y	u^{*}	$u^{\text{практ.}}$	$\lvert u^{*} - u^{\text{практ.}} \rvert$	$\frac{\lVert u^{*} - u^{\text{практ.}} \rVert}{\lVert u^{*} \rVert}$

1,000000E+000	1,000000E+000	2,000000E+000	2,000000E+000	0,000000E+000	
1,000000E+000	5,000000E+000	1,000000E+001	1,000000E+001	0,000000E+000	
9,000000E+000	1,000000E+000	1,800000E+001	1,800000E+001	0,000000E+000	
1,000000E+000	3,000000E+000	6,000000E+000	6,000000E+000	0,000000E+000	
5,000000E+000	3,000000E+000	3,000000E+001	3,000000E+001	7,105427E-015	5,805429E-017
5,000000E+000	1,000000E+000	1,000000E+001	1,000000E+001	0,000000E+000	
9,000000E+000	5,000000E+000	9,000000E+001	9,000000E+001	0,000000E+000	
9,000000E+000	3,000000E+000	5,400000E+001	5,400000E+001	0,000000E+000	
5,000000E+000	5,000000E+000	5,000000E+001	5,000000E+001	0,000000E+000	

t = 3

x	y	u^*	$u^{\text{практ.}}$	$\lvert u^* - u^{\text{практ.}} \rvert$	$\frac{\lVert u^* - u^{\text{практ.}} \rVert}{\lVert u^* \rVert}$
1,000000E+000	1,000000E+000	3,000000E+000	3,000000E+000	0,000000E+000	
1,000000E+000	5,000000E+000	1,500000E+001	1,500000E+001	0,000000E+000	
9,000000E+000	1,000000E+000	2,700000E+001	2,700000E+001	0,000000E+000	
1,000000E+000	3,000000E+000	9,000000E+000	9,000000E+000	0,000000E+000	
5,000000E+000	3,000000E+000	4,500000E+001	4,500000E+001	7,105427E-015	3,870286E-017
5,000000E+000	1,000000E+000	1,500000E+001	1,500000E+001	0,000000E+000	
9,000000E+000	5,000000E+000	1,350000E+002	1,350000E+002	0,000000E+000	
9,000000E+000	3,000000E+000	8,100000E+001	8,100000E+001	0,000000E+000	
5,000000E+000	5,000000E+000	7,500000E+001	7,500000E+001	0,000000E+000	

t = 4

x	y	u^*	$u^{\text{практ.}}$	$\lvert u^* - u^{\text{практ.}} \rvert$	$\frac{\lVert u^* - u^{\text{практ.}} \rVert}{\lVert u^* \rVert}$
1,000000E+000	1,000000E+000	4,000000E+000	4,000000E+000	0,000000E+000	
1,000000E+000	5,000000E+000	2,000000E+001	2,000000E+001	0,000000E+000	
9,000000E+000	1,000000E+000	3,600000E+001	3,600000E+001	0,000000E+000	
1,000000E+000	3,000000E+000	1,200000E+001	1,200000E+001	0,000000E+000	
5,000000E+000	3,000000E+000	6,000000E+001	6,000000E+001	3,552714E-014	1,451357E-016
5,000000E+000	1,000000E+000	2,000000E+001	2,000000E+001	0,000000E+000	
9,000000E+000	5,000000E+000	1,800000E+002	1,800000E+002	0,000000E+000	
9,000000E+000	3,000000E+000	1,080000E+002	1,080000E+002	0,000000E+000	
5,000000E+000	5,000000E+000	1,000000E+002	1,000000E+002	0,000000E+000	

4. $u^* = xt^2, \lambda = 1, \sigma = 2, \chi = 2\ , f\ =\ 2xt + 2x, t = [0, 1, 2, 3, 4]$

t = 2

x	y	u^*	$u^{\text{практ.}}$	$\lvert u^* - u^{\text{практ.}} \rvert$	$\frac{\lVert u^* - u^{\text{практ.}} \rVert}{\lVert u^* \rVert}$
1,000000E+000	1,000000E+000	4,000000E+000	4,000000E+000	0,000000E+000	
1,000000E+000	5,000000E+000	4,000000E+000	4,000000E+000	0,000000E+000	
9,000000E+000	1,000000E+000	3,600000E+001	3,600000E+001	0,000000E+000	
1,000000E+000	3,000000E+000	4,000000E+000	4,000000E+000	0,000000E+000	
5,000000E+000	3,000000E+000	2,000000E+001	4,000000E+000	1,600000E+001	2,232582E-001
5,000000E+000	1,000000E+000	2,000000E+001	2,000000E+001	0,000000E+000	
9,000000E+000	5,000000E+000	3,600000E+001	3,600000E+001	0,000000E+000	
9,000000E+000	3,000000E+000	3,600000E+001	3,600000E+001	0,000000E+000	
5,000000E+000	5,000000E+000	2,000000E+001	2,000000E+001	0,000000E+000	

t = 3

x	y	u^*	$u^{\text{практ.}}$	$\lvert u^* - u^{\text{практ.}} \rvert$	$\frac{\lVert u^* - u^{\text{практ.}} \rVert}{\lVert u^* \rVert}$
1,000000E+000	1,000000E+000	9,000000E+000	9,000000E+000	0,000000E+000	3,439830E-001
1,000000E+000	5,000000E+000	9,000000E+000	9,000000E+000	0,000000E+000	
9,000000E+000	1,000000E+000	8,100000E+001	8,100000E+001	0,000000E+000	
1,000000E+000	3,000000E+000	9,000000E+000	9,000000E+000	0,000000E+000	
5,000000E+000	3,000000E+000	4,500000E+001	-1,046667E+001	5,546667E+001	
5,000000E+000	1,000000E+000	4,500000E+001	4,500000E+001	0,000000E+000	
9,000000E+000	5,000000E+000	8,100000E+001	8,100000E+001	0,000000E+000	
9,000000E+000	3,000000E+000	8,100000E+001	8,100000E+001	0,000000E+000	
5,000000E+000	5,000000E+000	4,500000E+001	4,500000E+001	0,000000E+000	

t = 4

x	y	u^*	$u^{\text{практ.}}$	$\lvert u^* - u^{\text{практ.}} \rvert$	$\frac{\lVert u^* - u^{\text{практ.}} \rVert}{\lVert u^* \rVert}$
1,000000E+000	1,000000E+000	1,600000E+001	1,600000E+001	0,000000E+000	4,611523E-001
1,000000E+000	5,000000E+000	1,600000E+001	1,600000E+001	0,000000E+000	
9,000000E+000	1,000000E+000	1,440000E+002	1,440000E+002	0,000000E+000	
1,000000E+000	3,000000E+000	1,600000E+001	1,600000E+001	0,000000E+000	
5,000000E+000	3,000000E+000	8,000000E+001	-5,219556E+001	1,321956E+002	
5,000000E+000	1,000000E+000	8,000000E+001	8,000000E+001	0,000000E+000	
9,000000E+000	5,000000E+000	1,440000E+002	1,440000E+002	0,000000E+000	
9,000000E+000	3,000000E+000	1,440000E+002	1,440000E+002	0,000000E+000	
5,000000E+000	5,000000E+000	8,000000E+001	8,000000E+001	0,000000E+000	

5. $u^* = x^2t^2, \lambda = 1, \sigma = 2, \chi = 2\ , f = 2x^2t + 2x^2 - 1, t = [0, 1, 2, 3, 4]$

t = 2

x	y	u^*	$u^{\text{практ.}}$	$\lvert u^* - u^{\text{практ.}} \rvert$	$\frac{\lVert u^* - u^{\text{практ.}} \rVert}{\lVert u^* \rVert}$
1,000000E+000	1,000000E+000	4,000000E+000	4,000000E+000	0,000000E+000	1,734354E-001
1,000000E+000	5,000000E+000	4,000000E+000	4,000000E+000	0,000000E+000	
9,000000E+000	1,000000E+000	3,240000E+002	3,240000E+002	0,000000E+000	
1,000000E+000	3,000000E+000	4,000000E+000	4,000000E+000	0,000000E+000	
5,000000E+000	3,000000E+000	1,000000E+002	-1,866667E+000	1,018667E+002	
5,000000E+000	1,000000E+000	1,000000E+002	1,000000E+002	0,000000E+000	
9,000000E+000	5,000000E+000	3,240000E+002	3,240000E+002	0,000000E+000	
9,000000E+000	3,000000E+000	3,240000E+002	3,240000E+002	0,000000E+000	
5,000000E+000	5,000000E+000	1,000000E+002	1,000000E+002	0,000000E+000	

t = 3

x	y	u^*	$u^{\text{практ.}}$	$\lvert u^* - u^{\text{практ.}} \rvert$	$\frac{\lVert u^* - u^{\text{практ.}} \rVert}{\lVert u^* \rVert}$
1,000000E+000	1,000000E+000	9,000000E+000	9,000000E+000	0,000000E+000	2,700439E-001
1,000000E+000	5,000000E+000	9,000000E+000	9,000000E+000	0,000000E+000	
9,000000E+000	1,000000E+000	7,290000E+002	7,290000E+002	0,000000E+000	
1,000000E+000	3,000000E+000	9,000000E+000	9,000000E+000	0,000000E+000	
5,000000E+000	3,000000E+000	2,250000E+002	-1,318711E+002	3,568711E+002	

5,000000E+000	1,000000E+000	2,250000E+002	2,250000E+002	0,000000E+000	
9,000000E+000	5,000000E+000	7,290000E+002	7,290000E+002	0,000000E+000	
9,000000E+000	3,000000E+000	7,290000E+002	7,290000E+002	0,000000E+000	
5,000000E+000	5,000000E+000	2,250000E+002	2,250000E+002	0,000000E+000	

t = 4

x	y	u^*	$u^{\text{практ.}}$	$\lvert u^* - u^{\text{практ.}} \rvert$	$\frac{\lVert u^* - u^{\text{практ.}} \rVert}{\lVert u^* \rVert}$
1,000000E+000	1,000000E+000	1,600000E+001	1,600000E+001	0,000000E+000	3,657166E-001
1,000000E+000	5,000000E+000	1,600000E+001	1,600000E+001	0,000000E+000	
9,000000E+000	1,000000E+000	1,296000E+003	1,296000E+003	0,000000E+000	
1,000000E+000	3,000000E+000	1,600000E+001	1,600000E+001	0,000000E+000	
5,000000E+000	3,000000E+000	4,000000E+002	-4,592095E+002	8,592095E+002	
5,000000E+000	1,000000E+000	4,000000E+002	4,000000E+002	0,000000E+000	
9,000000E+000	5,000000E+000	1,296000E+003	1,296000E+003	0,000000E+000	
9,000000E+000	3,000000E+000	1,296000E+003	1,296000E+003	0,000000E+000	
5,000000E+000	5,000000E+000	4,000000E+002	4,000000E+002	0,000000E+000	

6. $u^* = x^2y^2t^2, \lambda = 1, \sigma = 2, \chi = 2\ , f\ =\ 2x^2y^2t\ +\ 2x^2y^2\ -\ x^2\ -\ y^2, t = [0, 1, 2, 3, 4]$

t = 2

x	y	u^*	$u^{\text{практ.}}$	$\lvert u^* - u^{\text{практ.}} \rvert$	$\frac{\lVert u^* - u^{\text{практ.}} \rVert}{\lVert u^* \rVert}$
1,000000E+000	1,000000E+000	4,000000E+000	4,000000E+000	0,000000E+000	1,026056E-001
1,000000E+000	5,000000E+000	1,000000E+002	1,000000E+002	0,000000E+000	
9,000000E+000	1,000000E+000	3,240000E+002	3,240000E+002	0,000000E+000	
1,000000E+000	3,000000E+000	3,600000E+001	3,600000E+001	0,000000E+000	
5,000000E+000	3,000000E+000	9,000000E+002	-2,515556E+001	9,251556E+002	
5,000000E+000	1,000000E+000	1,000000E+002	1,000000E+002	0,000000E+000	
9,000000E+000	5,000000E+000	8,100000E+003	8,100000E+003	0,000000E+000	
9,000000E+000	3,000000E+000	2,916000E+003	2,916000E+003	0,000000E+000	
5,000000E+000	5,000000E+000	2,500000E+003	2,500000E+003	0,000000E+000	

t = 3

x	y	u^*	$u^{\text{практ.}}$	$\lvert u^* - u^{\text{практ.}} \rvert$	$\frac{\lVert u^* - u^{\text{практ.}} \rVert}{\lVert u^* \rVert}$
1,000000E+000	1,000000E+000	9,000000E+000	9,000000E+000	0,000000E+000	1,642870E-001
1,000000E+000	5,000000E+000	2,250000E+002	2,250000E+002	0,000000E+000	
9,000000E+000	1,000000E+000	7,290000E+002	7,290000E+002	0,000000E+000	
1,000000E+000	3,000000E+000	8,100000E+001	8,100000E+001	0,000000E+000	
5,000000E+000	3,000000E+000	2,025000E+003	-1,307954E+003	3,332954E+003	
5,000000E+000	1,000000E+000	2,250000E+002	2,250000E+002	0,000000E+000	
9,000000E+000	5,000000E+000	1,822500E+004	1,822500E+004	0,000000E+000	
9,000000E+000	3,000000E+000	6,561000E+003	6,561000E+003	0,000000E+000	
5,000000E+000	5,000000E+000	5,625000E+003	5,625000E+003	0,000000E+000	

t = 4

x	y	u^*	$u^{\text{практ.}}$	$\lvert u^* - u^{\text{практ.}} \rvert$	$\frac{\lVert u^* - u^{\text{практ.}} \rVert}{\lVert u^* \rVert}$
1,000000E+000	1,000000E+000	1,600000E+001	1,600000E+001	0,000000E+000	2,283599E-001

1,000000E+000	5,000000E+000	4,000000E+002	4,000000E+002	0,000000E+000	
9,000000E+000	1,000000E+000	1,296000E+003	1,296000E+003	0,000000E+000	
1,000000E+000	3,000000E+000	1,440000E+002	1,440000E+002	0,000000E+000	
5,000000E+000	3,000000E+000	3,600000E+003	-4,636133E+003	8,236133E+003	
5,000000E+000	1,000000E+000	4,000000E+002	4,000000E+002	0,000000E+000	
9,000000E+000	5,000000E+000	3,240000E+004	3,240000E+004	0,000000E+000	
9,000000E+000	3,000000E+000	1,166400E+004	1,166400E+004	0,000000E+000	
5,000000E+000	5,000000E+000	1,000000E+004	1,000000E+004	0,000000E+000	

Из полученных результатов можно установить, что порядок аппроксимации для трёхслойной неявной схемы по времени равен 2

IV. Текст программы

Cell.cs

```
namespace Kursovaya
{
    // Класс конечного элемента расчётной области
    internal class Cell
    {
        public int[] v = new int[3]; // Вершины элемента
        public double[,] alpha = new double[3, 3]; // Коэффициенты alpha
        public double detD = 0; // Определитель матрицы D на данном элементе
        public int area = 0;

        // Конструктор класса
        public Cell(int v0, int v1, int v2, int area)
        {
            v[0] = v0;
            v[1] = v1;
            v[2] = v2;
            this.area = area;
        }
    }
}
```

Node.cs

```
namespace Kursovaya
{
    // Класс узла расчётной области
    internal class Node
    {
        public double x;
        public double y;
        public int condition1 = 0;
        // Конструктор класса
        public Node(double x, double y)
        {
            this.x = x;
            this.y = y;
        }
    }
}
```

Data.cs

```
namespace Kursovaya
{
    // Класс содержащий данные о СЛАУ задачи
    internal class Data
    {
        public int nodes;    // Кол-во узлов
        public int cells;    // Кол-во элементов
        public int maxIter;  // Максимальное кол-во итерации для решателя СЛАУ
        public double eps;   // Точность решения

        public int[] ig;     // Массив ig разреженой матрицы (кол-во элементов в
строке-столбце)
        public int[] jg;     // Массив jg разреженой матрицы (номера столбцов-строк
элементов матрицы)

        public double[] di;  // Массив di разреженой матрицы (диагональ)
        public double[] ggl; // Массив ggl разреженой матрицы (нижний треугольник)
        public double[] ggu; // Массив ggu разреженой матрицы (верхний треугольник)

        public double[] d;   // Массив d LU-разложения матрицы (диагональ)
        public double[] l;   // Массив l LU-разложения матрицы (нижний треугольник)
        public double[] u;   // Массив u LU-разложения матрицы (верхний треугольник)

        public double[] r;   // Массив r используемый в ЛОС
        public double[] z;   // Массив z используемый в ЛОС
        public double[] p;   // Массив p используемый в ЛОС

        public double[] b;   // Массив-вектор правой части
        public double[] x;   // Массив-вектор решения

        public double[,] G; // Глобальная матрица жесткости
        public double[,] MHi; // Глобальная матрица массы компоненты хи
        public double[,] MSigma; // Глобальная матрица массы компоненты сигма
        public double[,] global; // Глобальная матрица А

        public double[] temp1, temp2;   // Вспомогательные массивы

        // Конструктор класса данных
        public Data(int nodes, int cells, int maxIter, double eps)
        {
            this.nodes = nodes;
            this.cells = cells;
            this.maxIter = maxIter;
            this.eps = eps;

            int arrSize = (nodes * (nodes - 1)) / 2;

            ig = new int[nodes + 1];
            jg = new int[arrSize];

            di = new double[nodes];
            ggl = new double[arrSize];
            ggu = new double[arrSize];

            d = new double[nodes];
            l = new double[arrSize];
            u = new double[arrSize];

            b = new double[nodes];
            x = new double[nodes];

            G = new double[nodes, nodes];
            MHi = new double[nodes, nodes];
            MSigma = new double[nodes, nodes];
            global = new double[nodes, nodes];
```

```
            temp1 = new double[nodes];
            temp2 = new double[nodes];

            r = new double[nodes];
            z = new double[nodes];
            p = new double[nodes];
        }
    }
}
```

Program.cs

```
namespace Kursovaya
{
    internal class Program
    {
        // Функция f правой части уравнения
        public static double Target(double x, double y, double t, int area)
        {
            double result = 0;

            switch (area)
            {
                case 1:
                    result = 2;
                    break;
                case 2:
                    result = 2 * x;
                    break;
                case 3:
                    result = 2 * x * y;
                    break;
                case 4:
                    result = 2 * x * t + 2 * x;
                    break;
                case 5:
                    result = 2 * x * x * t + 2 * x * x - 1;
                    break;
                case 6:
                    result = 2 * x * x * y * y * t + 2 * x * x * y * y - x * x - y *
y;
                    break;
                default:
                    break;
            }

            return result;
        }

        // Параметр лямбда
        public static double Lambda(int area)
        {
            double result = 0;

            switch (area)
            {
                case 1:
                    result = 1;
                    break;
                case 2:
                    result = 1;
                    break;
                case 3:
                    result = 1;
                    break;
```

```
            case 4:
                result = 1;
                break;
            case 5:
                result = 1;
                break;
            case 6:
                result = 1;
                break;
            default:
                break;
        }

        return result;
    }

    // Параметр гамма
    public static double Gamma(int area)
    {
        double result = 0;

        switch (area)
        {
            case 1:
                result = 3;
                break;
            case 2:
                result = 2;
                break;
            case 3:
                result = 2;
                break;
            case 4:
                result = 2;
                break;
            default:
                break;
        }

        return result;
    }

    // Параметр хи
    public static double Hi(int area)
    {
        double result = 0;

        switch (area)
        {
            case 1:
                result = 2;
                break;
            case 2:
                result = 2;
                break;
            case 3:
                result = 2;
                break;
            case 4:
                result = 2;
                break;
            case 5:
                result = 2;
                break;
            case 6:
```

```
                result = 2;
                break;
            default:
                break;
        }

        return result;
    }

    // Параметр сигма
    public static double Sigma(int area)
    {
        double result = 0;

        switch (area)
        {
            case 1:
                result = 2;
                break;
            case 2:
                result = 2;
                break;
            case 3:
                result = 2;
                break;
            case 4:
                result = 2;
                break;
            case 5:
                result = 2;
                break;
            case 6:
                result = 2;
                break;
            default:
                break;
        }

        return result;
    }

    // Функция u истинная
    public static double Actual(double x, double y, double t, int type)
    {
        double result = 0;

        switch(type)
        {
            case 1:
                result = t;
                break;
            case 2:
                result = x * t;
                break;
            case 3:
                result = x * y * t;
                break;
            case 4:
                result = x * t * t;
                break;
            case 5:
                result = x * x * t * t;
                break;
            case 6:
                result = x * x * y * y * t * t;
```

```
                    break;
                default:
                    break;
            }

            return result;
        }

        // Рассчёт detD для конечного элемента (5.74)
        public static void CalcDetD(Cell cell, List<Node> nodes)
        {
            cell.detD += (nodes[cell.v[1]].x - nodes[cell.v[0]].x) *
(nodes[cell.v[2]].y - nodes[cell.v[0]].y);
            cell.detD -= (nodes[cell.v[2]].x - nodes[cell.v[0]].x) *
(nodes[cell.v[1]].y - nodes[cell.v[0]].y);
        }

// Рассчёт коэффициентов alpha для конечного элемента (5.69, 5.75)
public static void CalcAlphas(Cell cell, List<Node> nodes)
        {
            if (cell.detD != 0)
            {
                cell.alpha[0, 0] = (nodes[cell.v[1]].x * nodes[cell.v[2]].y -
nodes[cell.v[2]].x * nodes[cell.v[1]].y) / cell.detD;
                cell.alpha[0, 1] = (nodes[cell.v[1]].y - nodes[cell.v[2]].y) /
cell.detD;
                cell.alpha[0, 2] = (nodes[cell.v[2]].x - nodes[cell.v[1]].x) /
cell.detD;

                cell.alpha[1, 0] = (nodes[cell.v[2]].x * nodes[cell.v[0]].y -
nodes[cell.v[0]].x * nodes[cell.v[2]].y) / cell.detD;
                cell.alpha[1, 1] = (nodes[cell.v[2]].y - nodes[cell.v[0]].y) /
cell.detD;
                cell.alpha[1, 2] = (nodes[cell.v[0]].x - nodes[cell.v[2]].x) /
cell.detD;

                cell.alpha[2, 0] = (nodes[cell.v[0]].x * nodes[cell.v[1]].y -
nodes[cell.v[1]].x * nodes[cell.v[0]].y) / cell.detD;
                cell.alpha[2, 1] = (nodes[cell.v[0]].y - nodes[cell.v[1]].y) /
cell.detD;
                cell.alpha[2, 2] = (nodes[cell.v[1]].x - nodes[cell.v[0]].x) /
cell.detD;
            }
        }

        // Рассчёт локальной матрицы жёсткости
        public static void CalcG(Cell cell, List<Node> nodes, double[,] G)
        {
            double lambda = Lambda(cell.area);

            // Рассчёт коэффициентов alpha для конечного элемента (5.69, 5.75)
            cell.alpha[0, 0] = (nodes[cell.v[1]].x * nodes[cell.v[2]].y -
nodes[cell.v[2]].x * nodes[cell.v[1]].y) / cell.detD;
            cell.alpha[0, 1] = (nodes[cell.v[1]].y - nodes[cell.v[2]].y) / cell.detD;
            cell.alpha[0, 2] = (nodes[cell.v[2]].x - nodes[cell.v[1]].x) / cell.detD;

            cell.alpha[1, 0] = (nodes[cell.v[2]].x * nodes[cell.v[0]].y -
nodes[cell.v[0]].x * nodes[cell.v[2]].y) / cell.detD;
            cell.alpha[1, 1] = (nodes[cell.v[2]].y - nodes[cell.v[0]].y) / cell.detD;
            cell.alpha[1, 2] = (nodes[cell.v[0]].x - nodes[cell.v[2]].x) / cell.detD;

            cell.alpha[2, 0] = (nodes[cell.v[0]].x * nodes[cell.v[1]].y -
nodes[cell.v[1]].x * nodes[cell.v[0]].y) / cell.detD;
            cell.alpha[2, 1] = (nodes[cell.v[0]].y - nodes[cell.v[1]].y) / cell.detD;
            cell.alpha[2, 2] = (nodes[cell.v[1]].x - nodes[cell.v[0]].x) / cell.detD;
```

```
            double detD = Math.Abs(cell.detD);

            // Рассчёт значений компонент матрицы жесткости (5.80)
            for (int i = 0; i < 3; i++)
            {
                for (int j = 0; j < 3; j++)
                {
                    G[i, j] = lambda * detD * (cell.alpha[i, 1] * cell.alpha[j, 1] +
cell.alpha[i, 2] * cell.alpha[j, 2]) / 2;
                }
            }
        }

        // Рассчёт локальной матрицы массы
        public static void CalcM(Cell cell, double[,] M)
        {
            double gamma = Gamma(cell.area);

            double detD = Math.Abs(cell.detD);

            double[,] values = new double[3, 3]
            {
                { 2, 1, 1 },
                { 1, 2, 1 },
                { 1, 1, 2 }
            };

            // Рассчёт значений компонент матрицы масс (5.81)
            for (int i = 0; i < 3; i++)
            {
                for (int j = 0; j < 3; j++)
                {
                    M[i, j] = gamma * detD * values[i, j] / 24;
                }
            }
        }

        public static void CalcMHi(Cell cell, double[,] M)
        {
            double hi = Hi(cell.area);

            double detD = Math.Abs(cell.detD);

            double[,] values = new double[3, 3]
            {
                { 2, 1, 1 },
                { 1, 2, 1 },
                { 1, 1, 2 }
            };

            // Рассчёт значений компонент матрицы масс (5.81)
            for (int i = 0; i < 3; i++)
            {
                for (int j = 0; j < 3; j++)
                {
                    M[i, j] = hi * detD * values[i, j] / 24;
                }
            }
        }

        public static void CalcMSigma(Cell cell, double[,] M)
        {
            double sigma = Sigma(cell.area);
```

```
        double detD = Math.Abs(cell.detD);

        double[,] values = new double[3, 3]
        {
            { 2, 1, 1 },
            { 1, 2, 1 },
            { 1, 1, 2 }
        };

        // Рассчёт значений компонент матрицы масс (5.81)
        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < 3; j++)
            {
                M[i, j] = sigma * detD * values[i, j] / 24;
            }
        }
    }

    // Рассчёт локального вектора правой части
    public static void CalcB(Cell cell, List<Node> nodes, double[] b, double t)
    {
        double detD = Math.Abs(cell.detD);
        double[] f = new double[3];
        for (int i = 0; i < 3; i++)
            f[i] = Target(nodes[cell.v[i]].x, nodes[cell.v[i]].y, t, cell.area);

        b[0] = detD * (f[0] / 12 + f[1] / 24 + f[2] / 24);
        b[1] = detD * (f[0] / 24 + f[1] / 12 + f[2] / 24);
        b[2] = detD * (f[0] / 24 + f[1] / 24 + f[2] / 12);
    }

    // Генерация портрета матрицы
    public static void GenerateSparseGlobal(Data data)
    {
        int size = -1;

        for (int i = 0; i < data.nodes; i++)
        {
            for (int j = 0; j <= i; j++)
            {
                if (i == j)
                    data.di[i] = data.global[i, j];
                else
                {
                    if (data.global[i, j] != 0)
                    {
                        size++;
                        data.ggl[size] = data.global[i, j];
                        data.ggu[size] = data.global[i, j];
                        data.ig[i + 1] = size + 1;
                        data.jg[size] = j;
                    }
                    else
                    {
                        data.ig[i + 1] = size + 1;
                    }
                }
            }
        }
    }

    // Учёт первых краевых условий
    public static void Consider1(List<Node> nodes, Data data, double t)
    {
```

```
            for (int i = 0; i < data.nodes; i++)
            {
                // Если для узла задано первое краевое условие
                if (nodes[i].condition1 > 0)
                {
                    for (int k = 0; k < data.nodes; k++)
                        if (k == i)
                            data.global[k, k] = 1;
                        else
                            data.global[i, k] = 0;

                    // Ставим на i-ом (глобальном) элементе диагонали единицу
                    data.di[i] = 1;

                    // Обнуляем внедиагональные элементы i-ой строки в ggl
                    for (int j = data.ig[i]; j < data.ig[i + 1]; j++)
                        data.ggl[j] = 0;

                    // Обнуляем внедиагональные элементы i-ой строки в ggu
                    for (int j = 0; j < data.ig[data.nodes]; j++)
                        if (data.jg[j] == i)
                            data.ggu[j] = 0;

                    // В правой части замещаем значение на значение функции первого
краевого условия
                    data.b[i] = Actual(nodes[i].x, nodes[i].y, t, nodes[i].condi-
tion1);
                }
            }
        }

        public static double[] MatrixVector(double[,] matrix, double[] vector)
        {
            int n = vector.Length;
            double[] result = new double[n];

            for (int i = 0; i < n; i++)
                for (int k = 0; k < n; k++)
                    result[i] += matrix[i, k] * vector[k];

            return result;
        }



       static void Main(string[] args)
        {
            int test = 6;

            List<Node> nodes = new(); // Узлы сетки
            List<Cell> cells = new(); // Конечные элементы
            List<double[]> solutions = new(); // Решения по времени
            List<double> times = new(); // Временные точки

            double deltaT, deltaT1, deltaT0;
            double actual_value, t, error, d;

            string path = @"C:\Users\User\Documents\kurs_test";

            string[] files = { "nodes.txt", "cells.txt", "condition1.txt", "t.txt" };

            string? s;
            StreamReader reader;
            StreamWriter writer;

            // Заполнение списка узлов расчётной обалсти
```

```
            using (reader = new(Path.Combine(path, files[0])))
            {
                while ((s = reader.ReadLine()) != null)
                {
                    string[] coords = s.Split('\t');

                    nodes.Add(new Node(
                        double.Parse(coords[0]), // x
                        double.Parse(coords[1])  // y
                        ));
                }
                reader.Close();
            }

            // Заполнение списка конечных элементов
            using (reader = new(Path.Combine(path, files[1])))
            {
                while ((s = reader.ReadLine()) != null)
                {
                    string[] values = s.Split('\t');

                    cells.Add(new Cell(
                        int.Parse(values[0]) - 1, // 1 вершина
                        int.Parse(values[1]) - 1, // 2 вершина
                        int.Parse(values[2]) - 1, // 3 вершина
                        /*int.Parse(values[3])*/ test      // Область
                        ));
                }
                reader.Close();
            }

            // Считывание данных о первых краевых условиях
            using (reader = new(Path.Combine(path, files[2])))
            {
                int i = 0;
                while ((s = reader.ReadLine()) != null)
                {
                    string[] values = s.Split('\t');
                    i = int.Parse(values[0]) - 1;
                    //nodes[i].condition1 = int.Parse(values[1]);
                    nodes[i].condition1 = test;
                }
                reader.Close();
            }

            // Считывание данных о временных промежутках
            using (reader = new(Path.Combine(path, files[3])))
            {
                while ((s = reader.ReadLine()) != null)
                    times.Add(double.Parse(s));
                reader.Close();
            }

            // Инициализация класса данных
            Data data = new(nodes.Count, cells.Count, 10000, 1e-30);

            double[] temp1 = new double[nodes.Count];
            double[] temp2 = new double[nodes.Count];
            double[] temp3 = new double[nodes.Count];
            double[] temp4 = new double[nodes.Count];

            for (int i = 0; i < 2; i++)
            {
                solutions.Add(new double[nodes.Count]);
                for (int j = 0; j < data.nodes; j++)
```

```
            {
                solutions[i][j] = Actual(nodes[j].x, nodes[j].y, times[i], test);
            }
        }

        double[,] G = new double[3, 3]; // Локальная масса жёсткости
        double[,] MHi = new double[3, 3]; // Локальная масса масс хи
        double[,] MSigma = new double[3, 3]; // Локальная масса масс сигма
        double[] b = new double[3];     // Локальный вектор правой части

        foreach (Cell cell in cells)
        {
            CalcDetD(cell, nodes);

            CalcG(cell, nodes, G);
            CalcMHi(cell, MHi);
            CalcMSigma(cell, MSigma);

            for (int i = 0; i < 3; i++)
            {
                for (int j = 0; j < 3; j++)
                {
                    data.G[cell.v[i], cell.v[j]] += G[i, j];
                    data.MHi[cell.v[i], cell.v[j]] += MHi[i, j];
                    data.MSigma[cell.v[i], cell.v[j]] += MSigma[i, j];
                }
            }
        }

        for (int k = 2; k < times.Count; k++)
        {
            error = 0;
            d = 0;
            solutions.Add(new double[nodes.Count]);

            deltaT = times[k] - times[k - 2];
            deltaT1 = times[k - 1] - times[k - 2];
            deltaT0 = times[k] - times[k - 1];

            temp1 = MatrixVector(data.MHi, solutions[k - 2]);
            temp2 = MatrixVector(data.MHi, solutions[k - 1]);
            temp3 = MatrixVector(data.MSigma, solutions[k - 2]);
            temp4 = MatrixVector(data.MSigma, solutions[k - 1]);

            // Сборка глобальной матрицы
            foreach (Cell cell in cells)
            {
                CalcB(cell, nodes, b, times[k]);

                for (int i = 0; i < 3; i++)
                {
                    data.b[cell.v[i]] += b[i];
                }
            }

            for (int i = 0; i < data.nodes; i++)
            {
                data.b[i] -= temp1[i] * 2 / (deltaT1 * deltaT);
                data.b[i] += temp2[i] * 2 / (deltaT1 * deltaT0);

                data.b[i] -= temp3[i] * deltaT0 / (deltaT1 * deltaT);
                data.b[i] += temp4[i] * deltaT / (deltaT1 * deltaT0);

                for (int j = 0; j < data.nodes; j++)
                    data.global[i, j] +=
```

```
                            data.MHi[i, j] * 2 / (deltaT * deltaT0) +
                            data.MSigma[i, j] * (deltaT + deltaT0) / (deltaT * del-
taT0) -
                            data.G[i, j];
                }

                GenerateSparseGlobal(data);
                Consider1(nodes, data, times[k]);

                SLAESolver solver = new();

                solver.LOS_LUsq(data);

                Console.WriteLine("{0}", times[k]);
                Console.WriteLine();
                for (int i = 0; i < data.nodes; i++)
                {
                    Console.WriteLine(data.x[i]);
                    solutions[k][i] = data.x[i];
                }
                Console.WriteLine();

                using (writer = new(Path.Combine(path, test.ToString(),
times[k].ToString() + ".csv")))
                {
                    for (int i = 0; i < data.x.Length; i++)
                    {
                        actual_value = Actual(nodes[i].x, nodes[i].y, times[k],
cells[0].area);

                        t = Math.Abs(actual_value - data.x[i]);

                        error += t * t;
                        d += actual_value * actual_value;

                        writer.WriteLine("{0:E}; {1:E}; {2:E}; {3:E}; {4:E}",
                            nodes[i].x, nodes[i].y, actual_value, data.x[i], t);

                    }
                    writer.WriteLine("Error = {0:E}", Math.Sqrt(error / d));
                }

                for (int i = 0; i < data.nodes; i++)
                {
                    data.b[i] = 0;

                    for (int j = 0; j < data.nodes; j++)
                        data.global[i, j] = 0;
                }
            }
        }
    }
}
```

SLAESolver.cs

```
namespace Kursovaya
{
    // Класс-решатель СЛАУ
    internal class SLAESolver
    {
        // Скалярное произведение векторов (x, y)
        public static double ScalarMultiply(double[] x, double[] y)
        {
```

```
        double result = 0;
        for (int i = 0; i < x.Length; i++)
        {
            result += x[i] * y[i];
        }
        return result;
    }

    // Умножение разреженой матрицы на вектор
    public static double[] VectorMultiply(Data data, double[] x)
    {
        double[] y = new double[data.nodes];

        for (int i = 0; i < data.nodes; i++)
        {
            y[i] = x[i] * data.di[i];

            for (int j = data.ig[i]; j < data.ig[i + 1]; j++)
            {
                y[i] += data.ggl[j] * x[data.jg[j]];
                y[data.jg[j]] += data.ggu[j] * x[i];
            }
        }

        return y;
    }

    // Вычисление невязки
    public double CalcDiscrepancy(Data data)
    {
        double sum1 = 0, sum2 = 0;

        data.temp1 = VectorMultiply(data, data.x);

        for (int i = 0; i < data.nodes; i++)
        {
            sum1 += (data.b[i] - data.temp1[i]) * (data.b[i] - data.temp1[i]);
            sum2 += data.b[i] * data.b[i];
        }

        return Math.Sqrt(sum1 / sum2);
    }

    // Разложение LUsq
    public static void LU_sq(Data data)
    {
        for (int i = 0; i < data.l.Length; i++)
        {
            data.l[i] = data.ggl[i];
            data.u[i] = data.ggu[i];
        }

        for (int i = 0; i < data.d.Length; i++)
            data.d[i] = data.di[i];

        for (int i = 0; i < data.d.Length; i++)
        {
            double sumd = 0;
            int i0 = data.ig[i];
            int i1 = data.ig[i + 1];
            for (int k = i0; k < i1; k++)
            {
                int j = data.jg[k];
                double sl = 0, su = 0;
                int j0 = data.ig[j];
```

```
                int j1 = data.ig[j + 1];
                int ki = i0;
                int kj = j0;
                for (; ki < k && kj < j1;)
                {
                    int jl = data.jg[ki];
                    int ju = data.jg[kj];
                    if (jl == ju)
                    {
                        sl += data.u[kj] * data.l[ki];
                        su += data.l[kj] * data.u[ki];
                        ki++; kj++;
                    }
                    else if (jl < ju) ki++;
                    else kj++;
                }
                data.u[k] = (data.u[k] - su) / data.d[j];
                data.l[k] = (data.l[k] - sl) / data.d[j];
                sumd += data.u[k] * data.l[k];
            }
            data.d[i] = Math.Sqrt(Math.Abs(data.d[i] - sumd));
        }
    }

    // Прямой ход (стр. 875)
    public static void Straight(Data data, double[] a, double[] c)
    {
        for (int i = 0; i < a.Length; i++)
        {
            double sum = 0;
            int i0 = data.ig[i];
            int i1 = data.ig[i + 1];
            for (int k = i0; k < i1; k++)
            {
                int j = data.jg[k];
                sum += a[j] * data.l[k];
            }
            a[i] = (c[i] - sum) / data.d[i];
        }
    }

    // Обратный ход (стр. 876)
    public static void Reverse(Data data, double[] a, double[] c)
    {
        int n = a.Length;
        for (int i = 0; i < n; i++)
            a[i] = c[i];
        for (int i = n - 1; i >= 0; i--)
        {
            int i0 = data.ig[i];
            int i1 = data.ig[i + 1];
            a[i] /= data.d[i];
            for (int k = i1 - 1; k >= i0; k--)
            {
                int j = data.jg[k];
                a[j] -= a[i] * data.u[k];
            }
        }
    }

    // Локально-оптимальная схема с неполной факторизацией
    public void LOS_LUsq(Data data)
    {
        int n = data.di.Length;
        double scalar1 = 0;
```

```
        double scalar2 = 0;

        int iters = 0;

        LU_sq(data);

        data.temp1 = VectorMultiply(data, data.x);
        for (int i = 0; i < n; i++)
        {
            data.temp2[i] = data.b[i] - data.temp1[i];
        }

        Straight(data, data.r, data.temp2);

        Reverse(data, data.z, data.r);

        data.temp1 = VectorMultiply(data, data.z);

        Straight(data, data.p, data.temp1);

        double nev = ScalarMultiply(data.r, data.r);
        for (int k = 0; k < data.maxIter && nev > data.eps; k++)
        {
            iters++;
            scalar1 = ScalarMultiply(data.p, data.r);
            scalar2 = ScalarMultiply(data.p, data.p);
            double alpha = scalar1 / scalar2;
            for (int i = 0; i < n; i++)
            {
                data.x[i] += alpha * data.z[i];
                data.r[i] -= alpha * data.p[i];
            }

            Reverse(data, data.temp1, data.r);

            data.temp2 = VectorMultiply(data, data.temp1);

            Straight(data,data.temp1, data.temp2);

            scalar1 = ScalarMultiply(data.p, data.temp1);

            double beta = -scalar1 / scalar2;

            Reverse(data, data.temp2, data.r);

            for (int i = 0; i < n; i++)
            {
                data.z[i] = data.temp2[i] + beta * data.z[i];
                data.p[i] = data.temp1[i] + beta * data.p[i];
            }
            nev = ScalarMultiply(data.r, data.r);
        }
    }

    // Локально-оптимальная схема
    public void LOS(Data data)
    {
        int N = data.nodes;

        for (int i = 0; i < N; i++)
        {
            data.x[i] = 0; // Начальное приближение
        }

        double alpha, beta, nev;
```

```
            data.temp1 = VectorMultiply(data, data.x);

            for (int i = 0; i < data.nodes; i++)
            {
                data.r[i] = data.b[i] - data.temp1[i];
                data.z[i] = data.r[i];
            }
            data.p = VectorMultiply(data, data.r);

            nev = ScalarMultiply(data.r, data.r);

            for (int i = 0; i < data.maxIter && Math.Abs(nev) > data.eps; i++)
            {

                alpha = ScalarMultiply(data.p, data.r)
                    / //------------------------------
                        ScalarMultiply(data.p, data.p);

                for (int j = 0; j < data.nodes; j++)
                {
                    data.x[j] += alpha * data.z[j];
                    data.r[j] -= alpha * data.p[j];
                }

                data.temp1 = VectorMultiply(data, data.r);

                beta = (-1) * ScalarMultiply(data.p, data.temp1)
                   / //----------------------------------------
                       ScalarMultiply(data.p, data.p);

                for (int j = 0; j < data.nodes; j++)
                {
                    data.z[j] = data.r[j] + beta * data.z[j];
                    data.p[j] = data.temp1[j] + beta * data.p[j];
                }

                nev = ScalarMultiply(data.r, data.r);
            }
        }
    }
}
```