Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Санкт-Петербургский политехнический университет Петра Великого»
Институт компьютерных наук и технологий
Высшая школа программной инженерии

ОТЧЕТ ПО КУРСОВОЙ РАБОТЕ

Разработка программы для моделирования
стационарного двумерного распределения температуры

Вариант М1

по дисциплине «Математические модели систем с распределёнными параметрами»

Выполнил
студентка гр. 3530904/90102 Афанасьев Е.Д.

Руководитель
доцент Воскобойников С.П.

Санкт-Петербург
2022

Оглавление

Задание

Постановка задачи: Используя интегро-интерполяционный метод, разработать программу для моделирования распределения температуры в брусе, описываемого математической моделью

$$-\left[\frac{\partial}{\partial x}\left(k_1(x,y)\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left(k_2(x,y)\frac{\partial u}{\partial y}\right)\right]=f(x,y),$$

$$a \le x \le b, c \le y \le d, 0 < c_{11} \le k_1(x,y) \le c_{12} 0 < c_{21} \le k_2(x,y) \le c_{22}$$

с граничными условиями, определяемыми вариантом задания

$$u|_{x=a} = g_1(y), \qquad u|_{x=b} = g_2(y),$$

$$u|_{x=c} = g_3(y), \qquad u|_{y=d} = g_4(x)$$

Разностная схема

$$-\left[\frac{\partial}{\partial x}\left(k_1(x,y)\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left(k_2(x,y)\frac{\partial u}{\partial y}\right)+q(x,y)u\right]=f(x,y)$$

$$\textit{Краевые условия первого рода}\quad u\big|_{x=a}=\gamma_1,\qquad u\big|_{x=b}=\gamma_2$$

$$\textit{Краевые условия первого рода}\quad u\big|_{y=c}=\gamma_3,\qquad u\big|_{y=d}=\gamma_4$$

N_x – *число разбиений интервала* $[a,b]$

$$x_0<x_1<\ldots<x_{N_x},\quad x_i\in[a,b],\quad x_0=a,\ x_{N_x}=b$$

$$h_i=x_i-x_{i-1},\quad i=1,2,\ldots,N_x$$

$$x_{i-1/2}=\frac{x_i+x_{i-1}}{2},\quad i=1,2,\ldots,N_x$$

$$\hbar_i=\begin{cases}\frac{h_{i+1}}{2}, & i=0\\ \frac{h_i+h_{i+1}}{2}, & i=1,2,\ldots,N_x-1\\ \frac{h_i}{2}, & i=N_x\end{cases}$$

N_y – *число разбиений интервала* $[c,d]$

$$y_0<y_1<\ldots<y_{N_y},\quad y_j\in[c,d],\quad y_0=c,\ y_{N_y}=d$$

$$h_j=y_j-y_{j-1},\quad j=1,2,\ldots,N_y$$

$$y_{j-1/2}=\frac{y_j+y_{j-1}}{2},\quad j=1,2,\ldots,N_y$$

$$\hbar_j=\begin{cases}\frac{h_{j+1}}{2}, & j=0\\ \frac{h_j+h_{j+1}}{2}, & j=1,2,\ldots,N_y-1\\ \frac{h_j}{2}, & j=N_y\end{cases}$$

$$-\left[\hbar_j k_1(x_{i+1/2},y_j)\frac{v_{i+1,j}-v_{i,j}}{h_{i+1}}-\hbar_j k_1(x_{i-1/2},y_j)\frac{v_{i,j}-v_{i-1,j}}{h_i}+\right.$$
$$\left.+\hbar_i k_2(x_i,y_{j+1/2})\frac{v_{i,j+1}-v_{i,j}}{h_{j+1}}-\hbar_i k_2(x_i,y_{j-1/2})\frac{v_{i,j}-v_{i,j-1}}{h_j}+\hbar_i\hbar_j q_{ij}v_{ij}\right]=\hbar_i\hbar_j f_{ij}$$

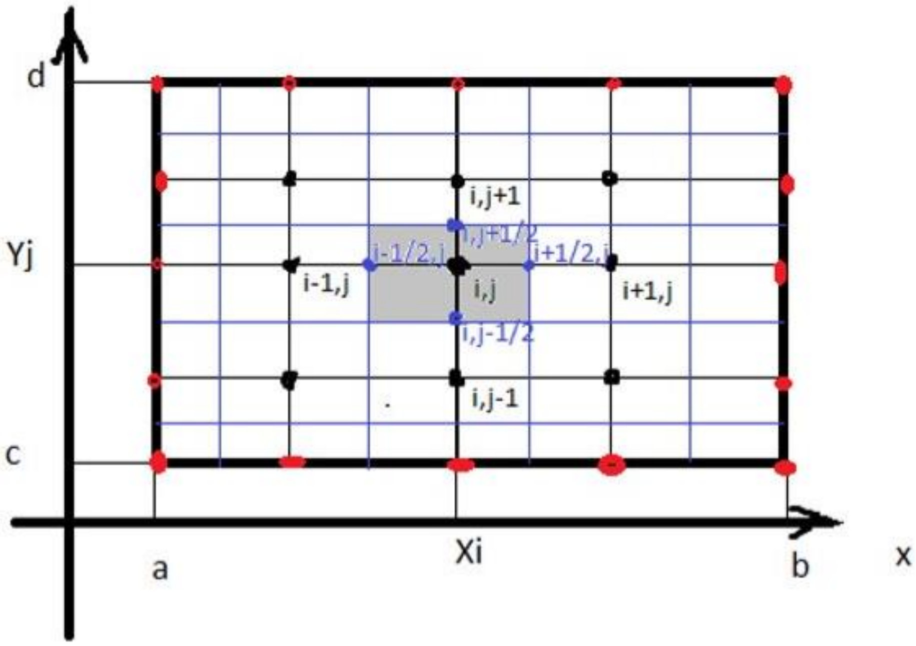
$$v_{i,j}=\gamma_1(y_j),\quad i=0,\quad j=0,1,\ldots,N_y\qquad\qquad v_{i,j}=\gamma_2(y_j),\quad i=N_x,\quad j=0,1,\ldots,N_y$$

$$v_{i,j}=\gamma_3(x_i),\quad i=0,1,\ldots,N_x,\quad j=0,\qquad\qquad v_{i,j}=\gamma_4(x_i),\quad i=0,1,\ldots,N_x,\quad j=N_y,$$

$$N=(N_x+1)(N_y+1)$$

$$x_i=a+ih_x,\quad i=0,1,\ldots,N_x\qquad h_i=h_x=\frac{b-a}{N_x},\qquad y_j=c+jh_y,\quad j=0,1,\ldots,N_y\qquad h_j=h_y=\frac{d-c}{N_y},$$

$$-\left[h_y k_1(x_{i+1/2},y_j)\frac{v_{i+1,j}-v_{i,j}}{h_x}-h_y k_1(x_{i-1/2},y_j)\frac{v_{i,j}-v_{i-1,j}}{h_x}+\right.$$
$$\left.+h_x k_2(x_i,y_{j+1/2})\frac{v_{i,j+1}-v_{i,j}}{h_y}-h_x k_2(x_i,y_{j-1/2})\frac{v_{i,j}-v_{i,j-1}}{h_y}+h_x h_y q_{ij}v_{ij}\right]=h_x h_y f_{ij}$$



$$u(x_i, y_j) = u_{i,j} \qquad\qquad v(x_i, y_j) = v_{i,j} \qquad\qquad u_{i,j} \approx v_{i,j}$$

$$-\left[\int\limits_{x_{i-1/2}}^{x_{i+1/2}}\int\limits_{y_{j-1/2}}^{y_{j+1/2}} \frac{\partial}{\partial x}\left(k_1 \frac{\partial u}{\partial x}\right)dxdy + \int\limits_{x_{i-1/2}}^{x_{i+1/2}}\int\limits_{y_{j-1/2}}^{y_{j+1/2}} \frac{\partial}{\partial y}\left(k_2 \frac{\partial u}{\partial y}\right)dxdy + \int\limits_{x_{i-1/2}}^{x_{i+1/2}}\int\limits_{y_{j-1/2}}^{y_{j+1/2}} qudxdy\right] = \int\limits_{x_{i-1/2}}^{x_{i+1/2}}\int\limits_{y_{j-1/2}}^{y_{j+1/2}} f\,dxdy$$

$$i = 1,2,\ldots,N_x - 1 \qquad\qquad j = 1,2,\ldots,N_y - 1$$

$$-\left[\int\limits_{y_{j-1/2}}^{y_{j+1/2}} k_1(x_{i+1/2}, y)\frac{\partial u}{\partial x}\bigg|_{x=x_{i+1/2}} dy - \int\limits_{y_{j-1/2}}^{y_{j+1/2}} k_1(x_{i-1/2}, y)\frac{\partial u}{\partial x}\bigg|_{x=x_{i-1/2}} dy + \right.$$

$$\left. + \int\limits_{x_{i-1/2}}^{x_{i+1/2}} k_2(x, y_{j+1/2})\frac{\partial u}{\partial y}\bigg|_{y=y_{j+1/2}} dx - \int\limits_{x_{i-1/2}}^{x_{i+1/2}} k_2(x, y_{j-1/2})\frac{\partial u}{\partial y}\bigg|_{y=y_{j-1/2}} dx + \int\limits_{x_{i-1/2}}^{x_{i+1/2}}\int\limits_{y_{j-1/2}}^{y_{j+1/2}} qudxdy\right] = \int\limits_{x_{i-1/2}}^{x_{i+1/2}}\int\limits_{y_{j-1/2}}^{y_{j+1/2}} fdxdy$$

$$\int\limits_{x_{i-1/2}}^{x_{i+1/2}} \varphi(x,y)dx \approx \hbar_i \varphi(x_i, y) = \hbar_i \varphi_i \qquad \int\limits_{y_{j-1/2}}^{y_{j+1/2}} \varphi(x,y)dy \approx \hbar_j \varphi(x, y_j) = \hbar_j \varphi_j \qquad \int\limits_{x_{i-1/2}}^{x_{i+1/2}}\int\limits_{y_{j-1/2}}^{y_{j+1/2}} \varphi dxdy \approx \hbar_i \hbar_j \varphi_{i,j}$$

$$k_1(x_{i-1/2}, y_j)\frac{\partial u}{\partial x}\bigg|_{\substack{x=x_{i-1/2}\\ y=y_j}} \approx k_1(x_{i-1/2}, y_j)\frac{v_{i,j} - v_{i-1,j}}{h_i}, \qquad k_2(x_i, y_{j-1/2})\frac{\partial u}{\partial y}\bigg|_{\substack{x=x_i\\ y=y_{j-1/2}}} \approx k_2(x_i, y_{j-1/2})\frac{v_{i,j} - v_{i,j-1}}{h_j}$$

$$i = 1,2,\ldots,N_x - 1 \qquad j = 1,2,\ldots,N_y - 1$$

$$-\left[\hbar_j k_1\left(x_{i+1/2}, y_j\right)\frac{v_{i+1,j} - v_{i,j}}{h_{i+1}} - \hbar_j k_1\left(x_{i-1/2}, y_j\right)\frac{v_{i,j} - v_{i-1,j}}{h_i} + \right.$$
$$\left. + \hbar_i k_2\left(x_i, y_{j+1/2}\right)\frac{v_{i,j+1} - v_{i,j}}{h_{j+1}} - \hbar_i k_2\left(x_i, y_{j-1/2}\right)\frac{v_{i,j} - v_{i,j-1}}{h_j} + \hbar_i \hbar_j q_{i,j} v_{i,j}\right] = \hbar_i \hbar_j f_{i,j}$$

$$v_{i,j} = \gamma_1\left(y_j\right), \quad i = 0, \quad j = 0,1,\ldots,N_y \qquad v_{i,j} = \gamma_2\left(y_j\right), \quad i = N_x, \quad j = 0,1,\ldots,N_y$$

$$v_{i,j} = \gamma_3\left(x_i\right), \quad i = 0,1,\ldots,N_x, \quad j = 0, \qquad v_{i,j} = \gamma_4\left(x_i\right), \quad i = 0,1,\ldots,N_x, \quad j = N_y,$$

$$N = \left(N_x + 1\right)\left(N_y + 1\right)$$

Анализ порядка аппроксимации уравнения и граничных условий

$$-\left[\hbar_j k_1\left(x_{i+1/2}, y_j\right)\frac{v_{i+1,j} - v_{i,j}}{h_{i+1}} - \hbar_j k_1\left(x_{i-1/2}, y_j\right)\frac{v_{i,j} - v_{i-1,j}}{h_i} + \right.$$
$$\left. + \hbar_i k_2\left(x_i, y_{j+1/2}\right)\frac{v_{i,j+1} - v_{i,j}}{h_{j+1}} - \hbar_i k_2\left(x_i, y_{j-1/2}\right)\frac{v_{i,j} - v_{i,j-1}}{h_j} + \hbar_i \hbar_j q_{i,j} v_{i,j}\right] = \hbar_i \hbar_j f_{i,j}$$

$$i = 1,\ldots,N_x - 1; \quad j = 1,\ldots,N_y - 1;$$

$$-\left[h_y k_1\left(x_{i+1/2}, y_j\right)\frac{v_{i+1,j} - v_{i,j}}{h_x} - h_y k_1\left(x_{i-1/2}, y_j\right)\frac{v_{i,j} - v_{i-1,j}}{h_x} + \right.$$
$$\left. + h_x k_2\left(x_i, y_{j+1/2}\right)\frac{v_{i,j+1} - v_{i,j}}{h_y} - h_x k_2\left(x_i, y_{j-1/2}\right)\frac{v_{i,j} - v_{i,j-1}}{h_y} + h_x h_y q_{ij} v_{ij}\right] = h_x h_y f_{ij}$$

$$\xi_{i,j} = h_x h_y f_{i,j} + h_y k_1\left(x_{i+1/2}, y_j\right)\frac{u_{i+1,j} - u_{i,j}}{h_x} - h_y k_1\left(x_{i-1/2}, y_j\right)\frac{u_{i,j} - u_{i-1,j}}{h_x} +$$
$$+ h_x k_2\left(x_i, y_{j+1/2}\right)\frac{u_{i,j+1} - u_{i,j}}{h_y} - h_x k_2\left(x_i, y_{j-1/2}\right)\frac{u_{i,j} - u_{i,j-1}}{h_y} + h_x h_y q_{i,j} u_{i,j}$$

$$\xi_{i,j} = h_x h_y f_{i,j} + h_y k_1\left(x_{i+1/2}, y_j\right)\frac{u_{i+1,j} - u_{i,j}}{h_x} - h_y k_1\left(x_{i-1/2}, y_j\right)\frac{u_{i,j} - u_{i-1,j}}{h_x} +$$

$$+ h_x k_2\left(x_i, y_{j+1/2}\right)\frac{u_{i,j+1} - u_{i,j}}{h_y} - h_x k_2\left(x_i, y_{j-1/2}\right)\frac{u_{i,j} - u_{i,j-1}}{h_y} + h_x h_y q_{i,j} u_{i,j} =$$

$$= h_x h_y f_{i,j} + h_y\left[h_x\left(\frac{\partial}{\partial x}\left(k_1\frac{\partial u}{\partial x}\right)\right)_{i,j} + h_x^3\left(\frac{1}{12}k_1\frac{\partial^4 u}{\partial x^4} + \frac{1}{6}\frac{\partial k_1}{\partial x}\frac{\partial^3 u}{\partial x^3} + \frac{1}{8}\frac{\partial^2 k_1}{\partial x^2}\frac{\partial^2 u}{\partial x^2} + \frac{1}{24}\frac{\partial^3 k_1}{\partial x^3}\frac{\partial u}{\partial x}\right)_{i,j} + O\left(h_x^4\right)\right] +$$

$$+ h_x\left[h_y\left(\frac{\partial}{\partial y}\left(k_2\frac{\partial u}{\partial y}\right)\right)_{i,j} + h_y^3\left(\frac{1}{12}k_2\frac{\partial^4 u}{\partial y^4} + \frac{1}{6}\frac{\partial k_2}{\partial y}\frac{\partial^3 u}{\partial y^3} + \frac{1}{8}\frac{\partial^2 k_2}{\partial y^2}\frac{\partial^2 u}{\partial y^2} + \frac{1}{24}\frac{\partial^3 k_2}{\partial y^3}\frac{\partial u}{\partial y}\right)_{i,j} + O\left(h_y^4\right)\right] + h_x h_y q_{i,j} u_{i,j}$$

$$\xi_{i,j} = h_x h_y f_{i,j} + h_y k_1\left(x_{i+1/2}, y_j\right)\frac{u_{i+1,j} - u_{i,j}}{h_x} - h_y k_1\left(x_{i-1/2}, y_j\right)\frac{u_{i,j} - u_{i-1,j}}{h_x} +$$

$$+ h_x k_2\left(x_i, y_{j+1/2}\right)\frac{u_{i,j+1} - u_{i,j}}{h_y} - h_x k_2\left(x_i, y_{j-1/2}\right)\frac{u_{i,j} - u_{i,j-1}}{h_y} + h_x h_y q_{i,j} u_{i,j} =$$

$$= h_x h_y\left[f + \frac{\partial}{\partial x}\left(k_1\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(k_2\frac{\partial u}{\partial y}\right) + qu\right]_{i,j} + h_y\left[h_x^3\left(\frac{1}{12}k_1\frac{\partial^4 u}{\partial x^4} + \frac{1}{6}\frac{\partial k_1}{\partial x}\frac{\partial^3 u}{\partial x^3} + \frac{1}{8}\frac{\partial^2 k_1}{\partial x^2}\frac{\partial^2 u}{\partial x^2} + \frac{1}{24}\frac{\partial^3 k_1}{\partial x^3}\frac{\partial u}{\partial x}\right)_{i,j} + O\left(h_x^4\right)\right] +$$

$$+ h_x\left[h_y^3\left(\frac{1}{12}k_2\frac{\partial^4 u}{\partial y^4} + \frac{1}{6}\frac{\partial k_2}{\partial y}\frac{\partial^3 u}{\partial y^3} + \frac{1}{8}\frac{\partial^2 k_2}{\partial y^2}\frac{\partial^2 u}{\partial y^2} + \frac{1}{24}\frac{\partial^3 k_2}{\partial y^3}\frac{\partial u}{\partial y}\right)_{i,j} + O\left(h_y^4\right)\right]$$

$$\tilde{\xi}_{i,j} = \frac{\xi_{i,j}}{h_x h_y}$$

$$\tilde{\xi}_{i,j} = f_{i,j} + k_1\left(x_{i+1/2}, y_j\right)\frac{u_{i+1,j} - u_{i,j}}{h_x^2} - k_1\left(x_{i-1/2}, y_j\right)\frac{u_{i,j} - u_{i-1,j}}{h_x^2} +$$

$$+ k_2\left(x_i, y_{j+1/2}\right)\frac{u_{i,j+1} - u_{i,j}}{h_y^2} - k_2\left(x_i, y_{j-1/2}\right)\frac{u_{i,j} - u_{i,j-1}}{h_y^2} + q_{i,j} u_{i,j} =$$

$$= \left[f + \frac{\partial}{\partial x}\left(k_1\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(k_2\frac{\partial u}{\partial y}\right) + qu\right]_{i,j} + h_x^2\left(\frac{1}{12}k_1\frac{\partial^4 u}{\partial x^4} + \frac{1}{6}\frac{\partial k_1}{\partial x}\frac{\partial^3 u}{\partial x^3} + \frac{1}{8}\frac{\partial^2 k_1}{\partial x^2}\frac{\partial^2 u}{\partial x^2} + \frac{1}{24}\frac{\partial^3 k_1}{\partial x^3}\frac{\partial u}{\partial x}\right)_{i,j} + O\left(h_x^3\right) +$$

$$+ h_y^2\left(\frac{1}{12}k_2\frac{\partial^4 u}{\partial y^4} + \frac{1}{6}\frac{\partial k_2}{\partial y}\frac{\partial^3 u}{\partial y^3} + \frac{1}{8}\frac{\partial^2 k_2}{\partial y^2}\frac{\partial^2 u}{\partial y^2} + \frac{1}{24}\frac{\partial^3 k_2}{\partial y^3}\frac{\partial u}{\partial y}\right)_{i,j} + O\left(h_y^3\right)$$

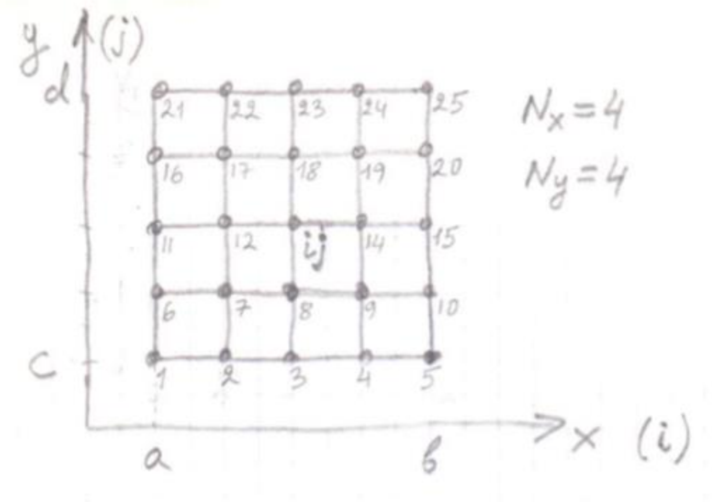
$$\left[f + \frac{\partial}{\partial x}\left(k_1\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(k_2\frac{\partial u}{\partial y}\right) + qu\right]_{i,j} = 0$$

$$p_x = 2 - 0 = 2 \qquad\qquad p_y = 2 - 0 = 2$$

$$\Phi_x = \frac{1}{12}k_1\frac{\partial^4 u}{\partial x^4} + \frac{1}{6}\frac{\partial k_1}{\partial x}\frac{\partial^3 u}{\partial x^3} + \frac{1}{8}\frac{\partial^2 k_1}{\partial x^2}\frac{\partial^2 u}{\partial x^2} + \frac{1}{24}\frac{\partial^3 k_1}{\partial x^3}\frac{\partial u}{\partial x}, \qquad \Phi_y = \frac{1}{12}k_2\frac{\partial^4 u}{\partial y^4} + \frac{1}{6}\frac{\partial k_2}{\partial y}\frac{\partial^3 u}{\partial y^3} + \frac{1}{8}\frac{\partial^2 k_2}{\partial y^2}\frac{\partial^2 u}{\partial y^2} + \frac{1}{24}\frac{\partial^3 k_2}{\partial y^3}\frac{\partial u}{\partial y}$$

Вид коэффициентов матрицы, структура матричной системы уравнений

$$
\begin{aligned}
&-\frac{h_x}{h_y}k_2\left(x_i, y_{j-1/2}\right)v_{i,j-1} - \frac{h_y}{h_x}k_1\left(x_{i-1/2}, y_j\right)v_{i-1,j} + \\
&+\left[\frac{h_x}{h_y}k_2\left(x_i, y_{j-1/2}\right) + \frac{h_x}{h_y}k_2\left(x_i, y_{j+1/2}\right) + \frac{h_y}{h_x}k_1\left(x_{i-1/2}, y_j\right) + \frac{h_y}{h_x}k_1\left(x_{i+1/2}, y_j\right) + h_x h_y q_{ij} v_{ij}\right]v_{i,j} - \\
&-\frac{h_y}{h_x}k_1\left(x_{i+1/2}, y_j\right)v_{i+1,j} - \frac{h_x}{h_y}k_2\left(x_i, y_{j+1/2}\right)v_{i,j+1} = h_x h_y f_{ij}
\end{aligned}
$$



$i = 1,\ldots,N_x - 1; \quad j = 1,\ldots,N_y - 1;$

Приведённый индекс. Переход к одному индексу

$$j = 0,1,\dots,N_y;$$

$$i = 0,1,\dots,N_x;$$

$$m = jL + i + 1, \qquad L = N_x + 1$$

$$v_{i,j-1} \to w_{m-L}$$

$$v_{i-1,j} \to w_{m-1}$$

$$v_{i,j} \to w_m$$

$$v_{i+1,j} \to w_{m+1}$$

$$v_{i,j+1} \to w_{m+L}$$

$$a_m = -\frac{h_x}{h_y} k_2\left(x_i, y_{j-1/2}\right), \qquad b_m = -\frac{h_y}{h_x} k_1\left(x_{i-1/2}, y_j\right)$$

$$c_m = \frac{h_x}{h_y} k_2\left(x_i, y_{j-1/2}\right) + \frac{h_x}{h_y} k_2\left(x_i, y_{j+1/2}\right) + \frac{h_y}{h_x} k_1\left(x_{i-1/2}, y_j\right) + \frac{h_y}{h_x} k_1\left(x_{i+1/2}, y_j\right) + h_x h_y q_{ij}$$

$$d_m = -\frac{h_y}{h_x} k_1\left(x_{i+1/2}, y_j\right) \qquad e_m = -\frac{h_x}{h_y} k_2\left(x_i, y_{j+1/2}\right)$$

$$g_m = h_x h_y f_{ij}$$

$$i = 0; \quad j = 0,1,\dots,N_y; \qquad m = jL + i + 1, \qquad c_m w_m = g_m, \qquad c_m = 1, \quad g_m = \gamma_1\left(y_j\right)$$

$$i = N_x; \quad j = 0,1,\dots,N_y; \qquad c_m w_m = g_m, \qquad c_m = 1, \quad g_m = \gamma_2\left(y_j\right)$$

$$i = 0,1,\dots,N_x; \quad j = 0; \qquad c_m w_m = g_m, \qquad c_m = 1, \quad g_m = \gamma_3\left(x_i\right)$$

$$i = 0,1,\dots,N_x; \quad j = N_y \qquad c_m w_m = g_m, \qquad c_m = 1, \quad g_m = \gamma_4\left(x_i\right)$$

$$Aw = g \qquad A \in R^{N\times N}, \quad w, g \in R^N, \quad N = (N_x + 1)(N_y + 1)$$

Структура матрицы системы уравнений

$$AV = F, \quad A \in R^{N\times N}, \quad N = (N_x+1)\times(N_y+1), \; V, F \in R^N$$

$$A = \begin{bmatrix} C_0 & B_0 & & & & & \\ A_1 & C_1 & B_1 & & & & \\ & . & . & . & & & \\ & & A_j & C_j & B_j & & \\ & & & . & . & . & \\ & & & & A_{N_y-1} & C_{N_y-1} & B_{N_y-1} \\ & & & & & A_{N_y} & C_{N_y} \end{bmatrix}, \quad V = \begin{bmatrix} V_0 \\ V_1 \\ . \\ V_j \\ . \\ V_{N_y-1} \\ V_{N_y} \end{bmatrix}, \quad F = \begin{bmatrix} F_0 \\ F_1 \\ . \\ F_j \\ . \\ F_{N_y-1} \\ F_{N_y} \end{bmatrix},$$

$$A_j = \begin{bmatrix} * & & & & & & \\ & * & & & & & \\ & & . & & & & \\ & & & . & & & \\ & & & & . & & \\ & & & & & * & \\ & & & & & & * \end{bmatrix}, \quad C_j = \begin{bmatrix} * & * & & & & & \\ * & * & * & & & & \\ & . & . & . & & & \\ & & . & . & . & & \\ & & & . & . & . & \\ & & & & * & * & * \\ & & & & & * & * \end{bmatrix}, \quad B_j = \begin{bmatrix} * & & & & & & \\ & * & & & & & \\ & & . & & & & \\ & & & . & & & \\ & & & & . & & \\ & & & & & * & \\ & & & & & & * \end{bmatrix},$$

$$A_j \in R^{(N_x+1)\times(N_x+1)}, \quad C_j \in R^{(N_x+1)\times(N_x+1)}, \quad B_j \in R^{(N_x+1)\times(N_x+1)},$$

$$V_j = \begin{bmatrix} v_{0,j} \\ v_{1,j} \\ . \\ . \\ . \\ v_{N_x-1,j} \\ v_{N_x,j} \end{bmatrix}, \quad V_j \in R^{N_x+1}, \quad F_j = \begin{bmatrix} f_{0,j} \\ f_{1,j} \\ . \\ . \\ . \\ f_{N_x-1,j} \\ f_{N_x,j} \end{bmatrix}, \quad F_j \in R^{N_x+1},$$

Метод матричной прогонки

$$\begin{cases} C_j V_j + B_j V_{j+1} = F_j, & j = 0 \\ A_j V_{j-1} + C_j V_j + B_j V_{j+1} = F_j, & j = 1,2, \quad , N_y - 1 \\ A_j V_{j-1} + C_j V_j = F_j, & j = N_y \end{cases}$$

$$V_j = \alpha_{j+1} V_{j+1} + \beta_{j+1}, \qquad j = 0,1,2,\dots, N_y - 1$$

α_j *и* β_j *–свободные параметры (прогоночные коэффициенты),*

$$\alpha_j \in R^{(N_x+1)\times(N_x+1)}, \qquad \beta_j \in R^{N_x+1},$$

$$V_j = -C_j^{-1} B_j V_{j+1} + C_j^{-1} F_j, \quad j = 0$$

$$\alpha_{j+1} = -C_j^{-1} B_j, \quad \beta_{i+1} = C_j^{-1} F_j, \quad j = 0$$

$$V_{j-1} = \alpha_j V_j + \beta_j, \qquad j = 1,2,\dots, N_y - 1$$

$$A_j(\alpha_j V_j + \beta_j) + C_j V_j + B_j V_{j+1} = F_j, \quad (A_j \alpha_j + C_j) V_j = -B_j V_{j+1} + F_j - A_j \beta_j,$$

$$V_j = -(A_j \alpha_j + C_j)^{-1} B_j V_{j+1} + (A_j \alpha_j + C_j)^{-1} (F_j - A_j \beta_j),$$

$$\alpha_{j+1} = -(A_j \alpha_j + C_j)^{-1} B_j, \qquad \beta_{j+1} = (A_j \alpha_j + C_j)^{-1} (F_j - A_j \beta_j),$$

$$j = 1,2,\dots, N_y - 1$$

$$\begin{cases} V_{j-1} - \alpha_j V_j = \beta_j, & j = N_y - 1 \\ A_j V_{j-1} + C_j V_j = F_j, & j = N_y \end{cases}$$

$$V_j = (A_j \alpha_j + C_j)^{-1} (F_j - A_j \beta_j), \qquad j = N_y$$

$$j = N_y - 1, N_y - 2, \dots, 1, 0,$$

$$V_j = \alpha_{j+1} V_{j+1} + \beta_{j+1},$$

$$\alpha_1 = -C_0^{-1} B_0, \quad \beta_1 = C_0^{-1} F_0,$$

$$j = 1,2,\ldots,N_y - 1$$

$$\alpha_{j+1} = -\left(A_j \alpha_j + C_j\right)^{-1} B_j,$$

$$\beta_{j+1} = \left(A_j \alpha_j + C_j\right)^{-1} \left(F_j - A_j \beta_j\right),$$

$$\sim N_x^3 N_y$$

$$V_{N_y} = \left(A_{N_y} \alpha_{N_y} + C_{N_y}\right)^{-1} \left(F_{N_y} - A_{N_y} \beta_{N_y}\right),$$

$$j = N_y - 1, N_y - 2, \ldots, 1, 0,$$

$$V_j = \alpha_{j+1} V_{j+1} + \beta_{j+1},$$

$\det C_j \neq 0, \quad j = 0,1,2,\ldots,N_y$ $\quad \left\| C_0^{-1} B_0 \right\| \le 1,$ $\quad \left\| C_{N_y}^{-1} B_{N_y} \right\| \le 1,$ $\quad \left\| C_j^{-1} A_j \right\| + \left\| C_j^{-1} B_j \right\| \le 1,$

$\left\| \alpha_i \right\| \le 1$ $\quad \det\left(A_j \alpha_j + C_j\right) \neq 0$

Все блоки одинаковы (в нашем случае) и трудоёмкость метода матричной прогонки ~M^4
Тестируем подпрограмму на следующих данных

```
double[][] matr = {
        {3, 1,  0,  0, 1, 0,  0,  0, 0,  0,  0,  0, 0, 0, 0, 0},
        {6, 11, 2,  0, 0, 1,  0,  0, 0,  0,  0,  0, 0, 0, 0, 0},
        {0, 5,  11, 3, 0, 0,  1,  0, 0,  0,  0,  0, 0, 0, 0, 0},
        {0, 0,  4,  7, 0, 0,  0,  1, 0,  0,  0,  0, 0, 0, 0, 0},
        {1, 0,  0,  0, 7, 4,  0,  0, 1,  0,  0,  0, 0, 0, 0, 0},
        {0, 1,  0,  0, 3, 13, 5,  0, 0,  1,  0,  0, 0, 0, 0, 0},
        {0, 0,  1,  0, 0, 2,  13, 6, 0,  0,  1,  0, 0, 0, 0, 0},
        {0, 0,  0,  1, 0, 0,  1,  7, 0,  0,  0,  1, 0, 0, 0, 0},
        {0, 0,  0,  0, 1, 0,  0,  0, 11, 6,  0,  0, 1, 0, 0, 0},
        {0, 0,  0,  0, 0, 1,  0,  0, 1,  11, 5,  0, 0, 1, 0, 0},
        {0, 0,  0,  0, 0, 0,  1,  0, 0,  2,  11, 4, 0, 0, 1, 0},
        {0, 0,  0,  0, 0, 0,  0,  1, 0,  0,  3,  7, 0, 0, 0, 1},
        {0, 0,  0,  0, 0, 0,  0,  0, 1,  0,  0,  0, 7, 3, 0, 0},
        {0, 0,  0,  0, 0, 0,  0,  0, 0,  1,  0,  0, 4, 9, 2, 0},
        {0, 0,  0,  0, 0, 0,  0,  0, 0,  0,  1,  0, 0, 5, 11, 1},
        {0, 0,  0,  0, 0, 0,  0,  0, 0,  0,  0,  1, 0, 0, 6, 9},
};

double[] v = {6, 36, 58, 44, 17, 48, 73, 39, 25, 42, 59, 45, 14, 30, 50, 58};
```

Ответ:

Ожидание

1	2	3	4
1	2	3	4
1	2	3	4
1	2	3	4

Реальность

0.9999999999999998	2.0	3.000000000000001	4.0
1.0000000000000004	1.9999999999999998	2.999999999999999	4.000000000000001
1.0000000000000004	2.000000000000001	2.9999999999999987	4.000000000000001
1.0	1.9999999999999987	2.9999999999999987	4.000000000000001

Тесты

Для тестирования полученной схемы были использованы три теста. Для всех тестов a=1, b=2, c=1, d=2. Интервалы $x \in [1,2]$ $y \in [1,2]$

1. Константный тест без погрешности аппроксимации

$$k_1 = 1$$
$$k_2 = 1$$
$$U = 1$$

Вычисляем

$$g_1 = g_2 = g_3 = g_4 = 1$$
$$f = 0$$

N	Ei = max {|Ui-Vi|}	E(i-1) / Ei
4	2.220446049250313E-16	-
8	2.220446049250313E-16	1
16	1.1102230246251565E-15	0.2
32	5.10702591327572E-15	0.22
64	2.3203661214665772E-14	0.22
128	8.848477506262498E-14	0.26
256	1.1391998455678731E-12	0.08

2. Линейный тест

$$k_1 = x + y + 1$$
$$k_2 = x + y + 1$$
$$U = x + y$$

Вычисляем

$$g_1 = y + 1$$
$$g_2 = y + 2$$
$$g_3 = x + 1$$
$$g_4 = x + 2$$
$$f = -2$$

N	Ei = max {|Ui-Vi|}	E(i-1) / Ei
4	8.881784197001252E-16	-
8	1.7763568394002505E-15	0.5
16	6.661338147750939E-15	0.27
32	9.769962616701378E-15	0.68
64	3.4638958368304884E-14	0.28
128	3.526068326209497E-13	0.1
256	3.284483796051063E-12	0.11

3. Нелинейный тест

$$k_1 = x^2 + y + 1$$
$$k_2 = x + y^2 + 1$$
$$U = xy$$

Вычисляем

$$g_1 = y$$
$$g_2 = 2y$$
$$g_3 = x$$

$$g_4 = 2x$$
$$f = -4xy$$

N	Ei = max {|Ui-Vi|}	E(i-1) / Ei
4	4.440892098500626E-16	-
8	2.6645352591003757E-15	0.17
16	4.440892098500626E-15	0.6
32	7.105427357601002E-15	0.63
64	4.085620730620576E-14	0.17
128	2.6645352591003757E-13	0.15
256	2.7498003873915877E-12	0.1

4. Нелинейный тест 2

$$k_1 = x^2 + y + 1$$
$$k_2 = x + y^2 + 1$$
$$U = x^2 y^2$$

Вычисляем

$$g_1 = y^2$$
$$g_2 = 4y^2$$
$$g_3 = x^2$$
$$g_4 = 4x^2$$
$$f = -12x^2y^2 - 2y^3 - 2y^2 - 2x^3 - 2x^2$$

N	Ei = max {|Ui-Vi|}	E(i-1) / Ei
4	0.002064899235090678	-
8	5.349548410240601E-4	3.86
16	1.349936125665252E-4	3.96
32	3.382831077658466E-5	3.99
64	8.462095730799035E-6	3.997
128	2.116556609976783E-6	3.998
256	5.291581004485124E-7	3.99985

Выводы

В первых трех тестах с увеличением числа разбиений наблюдается возрастание погрешности. Причиной является рост числа обусловленности разностной схемы.

В четвертом тесте погрешность аппроксимации ненулевая, поэтому, кроме ошибок вычисления к погрешности добавляется еще и погрешность аппроксимации, что видно из представленных таблиц. С ростом числа разбиений шаг прохода по сетке уменьшается, и, следовательно, уменьшается и общая погрешность решения.

Код программы

```
public class Test1 {
    static int r = 8;
    static int Nx = r;
    static int Ny = r;
    static int type = 2;

    static double aX = 1.0;
    static double bX = 2.0;
    static double cY = 1.0;
    static double dY = 2.0;

    public static double a[][][] = new double[Nx + 1][Ny+ 1][Ny+ 1];
    public static double b[][][] = new double[Nx+ 1][Ny+ 1][Ny+ 1];
    public static double c[][][] = new double[Nx+ 1][Ny+ 1][Ny+ 1];
    public static double v[][] = new double[Nx+ 1][Ny+ 1];
    public static double f[][] = new double[Nx+ 1][Ny+ 1];
    public static double x[] = new double[Nx + 1];

    public static double y[] = new double[Ny + 1];
    public static double helpX[] = new double[Nx];
    public static double helpY[] = new double[Ny];
    public static double hx;
    public static double hy;

    public static void init(int r) {
        Test1.r = r;
        Nx = r;
        Ny  = r;

        a = new double[Nx + 1][Ny+ 1][Ny+ 1];
        b = new double[Nx+ 1][Ny+ 1][Ny+ 1];
        c = new double[Nx+ 1][Ny+ 1][Ny+ 1];
        v = new double[Nx+ 1][Ny+ 1];
        f = new double[Nx+ 1][Ny+ 1];
        x = new double[Nx + 1];

        y = new double[Ny + 1];
        helpX = new double[Nx];
        helpY = new double[Ny];
    }

    static double k1(double x, double y, int type) {
        if (type == 0) {
            return 1.0;
        }
        if (type == 1) {
            return x + y + 1;
        }
```

```
        if (type == 2) {
            return x * x + y + 1;
        }
        return 0;
    }

    static double k2(double x, double y, int type) {
        if (type == 0) {
            return 1.0;
        }
        if (type == 1) {
            return x + y + 1;
        }
        if (type == 2) {
            return x + y * y + 1;
        }
        return 0;
    }

/*    static double g1(double y, int type) {
        if (type == 0) {
            return 1.0;
        }
        if (type == 1) {
            return y + aX;
        }
        if (type == 2) {
            return y * aX;
        }
        return 0;
    }

    static double g2(double y, int type) {
        if (type == 0) {
            return 1.0;
        }
        if (type == 1) {
            return y + bX;
        }
        if (type == 2) {
            return bX * y;
        }
        return 0;
    }

    static double g3(double x, int type) {
        if (type == 0) {
            return 1.0;
        }
        if (type == 1) {
            return cY + x;
        }
        if (type == 2) {
            return cY * x;
        }
        return 0;
    }

    static double g4(double x, int type) {
        if (type == 0) {
            return 1.0;
        }
        if (type == 1) {
            return dY + x;
```

```
    }
    if (type == 2) {
        return dY * x;
    }
    return 0;
}

static double F(double x, double y, int type) {
    if (type == 0) {
        return 0.0;
    }
    if (type == 1) {
        return -2.0;
    }
    if (type == 2) {
        return -4 * x * y;
    }
    return 0;
}

static double U(double x, double y, int type) {
    if (type == 0) {
        return 1.0;
    }
    if (type == 1) {
        return x + y;
    }
    if (type == 2) {
        return x*y;
    }
    return 0;
}*/
static double g1(double y, int type) {
    if (type == 0) {
        return 1.0;
    }
    if (type == 1) {
        return y + aX;
    }
    if (type == 2) {
        return y * y * aX * aX;
    }
    return 0;
}

static double g2(double y, int type) {
    if (type == 0) {
        return 1.0;
    }
    if (type == 1) {
        return y + bX;
    }
    if (type == 2) {
        return bX * bX * y * y;
    }
    return 0;
}

static double g3(double x, int type) {
    if (type == 0) {
        return 1.0;
    }
    if (type == 1) {
        return cY + x;
```

```
        }
        if (type == 2) {
            return cY * cY * x * x;
        }
        return 0;
    }

    static double g4(double x, int type) {
        if (type == 0) {
            return 1.0;
        }
        if (type == 1) {
            return dY + x;
        }
        if (type == 2) {
            return dY * dY * x * x;
        }
        return 0;
    }

    static double F(double x, double y, int type) {
        if (type == 0) {
            return 0.0;
        }
        if (type == 1) {
            return -2.0;
        }
        if (type == 2) {
            return -12 * x * x * y * y - 2 * y * y * (y + 1) - 2 * x * x * (x
+ 1);
        }
        return 0;
    }

    static double U(double x, double y, int type) {
        if (type == 0) {
            return 1.0;
        }
        if (type == 1) {
            return x + y;
        }
        if (type == 2) {
            return x*x*y*y;
        }
        return 0;
    }

    public static double[][] inversion(double[][] A, int N) {
        double temp;
        double[][] E = new double[N][N];
        for (int i = 0; i < N; i++)
            for (int j = 0; j < N; j++) {
                E[i][j] = 0f;

                if (i == j)
                    E[i][j] = 1f;
            }
        for (int k = 0; k < N; k++) {
            temp = A[k][k];
            for (int j = 0; j < N; j++) {
                A[k][j] /= temp;
                E[k][j] /= temp;
            }
            for (int i = k + 1; i < N; i++) {
```

```java
                temp = A[i][k];
                for (int j = 0; j < N; j++) {
                    A[i][j] -= A[k][j] * temp;
                    E[i][j] -= E[k][j] * temp;
                }
            }
        }
        for (int k = N - 1; k > 0; k--) {
            for (int i = k - 1; i >= 0; i--) {
                temp = A[i][k];
                for (int j = 0; j < N; j++) {
                    A[i][j] -= A[k][j] * temp;
                    E[i][j] -= E[k][j] * temp;
                }
            }
        }
        for (int i = 0; i < N; i++)
            System.arraycopy(E[i], 0, A[i], 0, N);
        return A;
    }

    public static double[][] multMatrix(double[][] first, double[][] second)
{
        double[][] res = new double[first.length][first.length];
        for (int i = 0; i < first.length; i++) {
            for (int j = 0; j < first.length; j++) {
                for (int k = 0; k < first.length; k++) {
                    res[i][j] += first[i][k] * second[k][j];
                }
            }
        }
        return res;
    }

    public static double[] multMatrixAndVect(double[][] m, double[] vect) {
        double[] res = new double[m.length];
        for (int i = 0; i < m.length; ++i) {
            for (int j = 0; j < vect.length; ++j)
                res[i] += m[i][j] * vect[j];
        }
        return res;
    }

    public static double[][] minusOne(double[][] m) {
        double[][] res = new double[m.length][m[0].length];
        for (int i = 0; i < m.length; ++i) {
            for (int j = 0; j < m[0].length; ++j)
                res[i][j] = -1 * m[i][j];
        }
        return res;
    }

    public static double[][] plusMatrix(double[][] m1, double[][] m2) {
        double[][] res = new double[m1.length][m1[0].length];
        for (int i = 0; i < m1.length; ++i) {
            for (int j = 0; j < m1[0].length; ++j)
                res[i][j] = m1[i][j] + m2[i][j];
        }
        return res;
    }

    public static double[] minusVect(double[] m1, double[] m2) {
        double[] res = new double[m1.length];
        for (int i = 0; i < m1.length; ++i) {
```

```
            res[i] = m1[i] - m2[i];
        }
        return res;
    }

    public static double[] plusVect(double[] m1, double[] m2) {
        double[] res = new double[m1.length];
        for (int i = 0; i < m1.length; ++i) {
            res[i] = m1[i] + m2[i];
        }
        return res;
    }

    public static void printMatrix(double[][] m1) {
        for (double[] doubles : m1) {
            for (int j = 0; j < m1[0].length; ++j) {
                System.out.print(doubles[j] + " ");
            }
            System.out.println();
        }
    }

    public static double[][] solveSystem(double[][][] massA, double[][][]
massB, double[][][] massC, double[][] func, int x, int y) {
        double[][][] alpha = new double[y][x][y];
        double[][] beta = new double[x][y];
        double[][] res = new double[x][y];
        double[][] tmpBeta;
        double[][] tmpRes;

        tmpRes = inversion(massC[0], x);
        double[][] tmpRes2 = minusOne(tmpRes);
        alpha[0] = multMatrix(tmpRes2, massB[0]);

        tmpBeta = inversion(massC[0], x);
        beta[0] = multMatrixAndVect(tmpBeta, func[0]);
        for (int i = 1; i < y; i++) {
            double[][] ttt = multMatrix(massA[i - 1], alpha[i - 1]);
            double[][] tmpResCom = plusMatrix(ttt, massC[i - 1]);
            double[][] tmpResCom1 = inversion(tmpResCom, x);
            double[][] tmpRes1 = minusOne(tmpResCom1);
            alpha[i] = multMatrix(tmpRes1, massB[i - 1]);

            double[][] tmpResCom3 = plusMatrix(multMatrix(massA[i - 1],
alpha[i - 1]), massC[i - 1]);
            double[][] tmpResCom4 = inversion(tmpResCom3, x);
            double[] tmpBeta1 = minusVect(func[i - 1],
multMatrixAndVect(massA[i - 1], beta[i - 1]));
            beta[i] = multMatrixAndVect(tmpResCom4, tmpBeta1);
        }

        double[][] tmpResCom = plusMatrix(multMatrix(massA[y - 1], alpha[y -
1]), massC[y - 1]);
        double[][] tmpResCom1 = inversion(tmpResCom, x);
        double[] tmpBeta1 = minusVect(func[y - 1], multMatrixAndVect(massA[y
- 1], beta[y - 1]));
        res[y - 1] = multMatrixAndVect(tmpResCom1, tmpBeta1);

        for (int i = y - 2; i >= 0; i--) {
            res[i] = plusVect(multMatrixAndVect(alpha[i + 1], res[i + 1]),
beta[i + 1]);
        }

        return res;
```

```
    }

    public static void main(String[] args) {
        //solve(true);
        int s = 0, size = 7;
        double eps[] = new double[size];

        for (int p = 4; p <= 256; p *= 2) {
            init(p);
            eps[s] = solve(false);
            s++;
        }

        for (int p = 1; p < size; p++) {
            System.out.println(eps[p - 1] / eps[p]);
        }
    }


    public static double solve(boolean debug) {
        //setka
        double stepX = (bX - aX) / Nx;
        double stepY = (dY - cY) / Ny;
        for (int i = 0; i < Nx + 1; i++) {
            x[i] = aX + stepX * i;
        }
        helpX[0] = aX + stepX / 2;
        for (int i = 1; i < Nx; i++) {
            helpX[i] = x[i] + stepX / 2;
        }

        for (int i = 0; i < Ny + 1; i++) {
            y[i] = cY + stepY * i;
        }
        helpY[0] = cY + stepY / 2;
        for (int i = 1; i < Ny; i++) {
            helpY[i] = y[i] + stepY / 2;
        }

        hx = stepX;
        hy = stepY;

        // set zeros
        for (int j = 0; j < Ny + 1; j++) {
            for (int i = 0; i < Nx + 1; i++) {
                for (int k = 0; k < Nx + 1; k++) {
                    a[j][i][k] = 0;
                    b[j][i][k] = 0;
                    c[j][i][k] = 0;
                }
            }
        }

        // make matrices
        int j = 0;

        //main part
        for (j = 1; j < Ny; j++) {
            a[j][0][0] = 0;
            b[j][0][0] = 0;
            c[j][0][0] = 1;
            f[j][0] = g1(y[j], type);
            for (int i = 1; i < Nx; i++) {
                a[j][i][i] = -(hx * k2(x[i], helpY[j - 1], type)) / hy;
```

```
// coefff i j - 1
                c[j][i][i - 1] = -(hy * k1(helpX[i - 1], y[j], type)) / hx;
// i-1, j  bm
                c[j][i][i] = hy * (k1(helpX[i], y[j], type) / hx + k1(helpX[i
- 1], y[j], type) / hx) +
                        hx * (k2(x[i], helpY[j], type) / hy + k2(x[i],
helpY[j - 1], type) / hy); // i j
                c[j][i][i + 1] = -(hy * k1(helpX[i], y[j], type)) / hx; //
i+1, j  dm
                b[j][i][i] = -(hx * k2(x[i], helpY[j], type)) / hy; // i, j+1
em

                f[j][i] = hx * hy * F(x[i], y[j], type);
            }
        }

        // upper border
        j = Ny;
        for (int i = 1; i < Nx; i++) {
            a[j][i][i] = 0;
            b[j][i][i] = 0;
            c[j][i][i] = 1;
            f[j][i] = g4(x[i], type);
        }

        // down border
        j = 0;
        for (int i = 0; i < Nx; i++) {
            a[j][i][i] = 0;
            b[j][i][i] = 0;
            c[j][i][i] = 1;
            f[j][i] = g1(y[i], type);
        }

        // right border
        int i = Nx;
        for (j = 0; j < Nx + 1; j++) {
            a[j][i][i] = 0;
            b[j][i][i] = 0;
            c[j][i][i] = 1;
            f[j][i] = g2(y[j], type);
        }

        // left border
        i = 0;
        for (j = 1; j < Nx + 1; j++) {
            a[j][i][i] = 0;
            b[j][i][i] = 0;
            c[j][i][i] = 1;
            f[j][i] = g3(x[j], type);
        }

        double[][] answer = solveSystem(a, b, c, f, Nx + 1, Ny + 1);

        if (debug) {
            for (i = 0; i < Nx + 1; i++) {
                for (int k = 0; k < Ny + 1; k++) {
                    System.out.print(answer[i][k] + " ");
                }
                System.out.println();
            }
            System.out.println("*******************************");
        }
```

```java
        double max = Math.abs(U(x[0], y[0], type) - answer[0][0]);
        for (i = 0; i < Nx + 1; i++) {
            for (int k = 0; k < Ny + 1; k++) {
                double tmp = Math.abs(U(x[i], y[k], type) - answer[i][k]);
                if (tmp > max) {
                    max = tmp;
                }
                if (debug)
                    System.out.print(tmp + " ");
            }
            if (debug)
                System.out.println();
        }
        if (debug)
            System.out.println("*******************************");
        System.out.println(max);

        return max;
    }
}
```