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

Курсовая работа
“Математические модели систем с распределёнными параметрами”

Выполнил студент гр. 3530904/90102 Дергунов Н. С.
Преподаватель Воскобойников С. П.

Санкт-Петербург 2022

Оглавление

Постановка задачи

Вариант Q8.

$$-\left[\frac{1}{r}\frac{\partial}{\partial r}\left(rk_1(r,z)\frac{\partial u}{\partial r}\right)+\frac{\partial}{\partial z}\left(k_2(r,z)\frac{\partial u}{\partial z}\right)\ \right]=f(r,z)$$

$$0<c_{11}\le k_1(r,z)\le c_{12},\qquad 0<c_{21}\le k_2(r,z)\le c_{22},\qquad 0<R_0\le r\le R_1,\qquad 0\le z\le L$$

с граничными условиями, определяемыми вариантом задания. Метод решения системы алгебраических уравнений - метод сопряжённых градиентов

$$k_1\frac{\partial u}{\partial r}\bigg|_{r\,=\,R_0}=\chi_1 u|_{r=R_0}-g_1(z),\qquad \chi_1\ge 0,$$

$$-k_1\frac{\partial u}{\partial r}\bigg|_{r\,=\,R_1}=\chi_2 u|_{r=R_1}-g_2(z),\qquad \chi_2\ge 0,$$

$$u|_{z=0}=g_3(r),\qquad u|_{z=L}=g_4(r)$$

Требуется:

1. Используя интегро-интерполяционный метод, вывести разностную схему для уравнения и граничных условий на равномерной сетке.

2. Получить разложение невязки и главный член погрешности аппроксимации для уравнения и граничных условий на равномерной сетке.

3. Для случая $N_x=4, N_y=4$ сделать необходимые преобразования разностной схемы для применения метода решения алгебраической системы, указанного в варианте. Нарисовать структуру системы алгебраических уравнений, отметив * ненулевые элементы матрицы.

4. Написать тесты и программу реализующую данный метод, проверить поученные результаты.

Разностная схема

$$-\left[\frac{1}{r}\frac{\partial}{\partial r}\left(rk_1(r,z)\frac{\partial u}{\partial r}\right)+\frac{\partial}{\partial z}\left(k_2(r,z)\frac{\partial u}{\partial z}\right)\ \right]=f(r,z),\ \ r\in[R_0,R_1],\quad z\in[0,L],$$

$$0<c_{11}\le k_1(r,z)\le c_{12},\qquad 0<c_{21}\le k_2(r,z)\le c_{22},$$

Домножим на r

$$-\left[\frac{\partial}{\partial r}\left(rk_1(r,z)\frac{\partial u}{\partial r}\right)+\frac{\partial}{\partial z}\left(rk_2(r,z)\frac{\partial u}{\partial z}\right)\right]=rf(r,z)$$

$$N_r - \text{число разбиений } [R_0,R_1]$$

$$r_0<\ r_1<\dots<r_{N_z},\qquad r_i\in[R_0,R_1],\qquad r_0=R_0,\qquad r_{N_r}=R_1,\qquad h_r=\frac{R_1-R_0}{N_r}$$

$$r_{i-1/2}=\frac{r_i+r_{i-1}}{2},\qquad i=1,2,\dots,N_r$$

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

$$N_z - \text{число разбиений } [0,L]$$

$$z_0<\ z_1<\dots<z_{N_z},\qquad z_j\in[0,L],\qquad z_0=0,\qquad z_{N_z}=L,\qquad h_Z=\frac{L}{N_z}$$

$$z_{j-1/2}=\frac{z_j+z_{j-1}}{2},\qquad j=1,2,\dots,N_z$$

$$\hbar_j=\begin{cases}\dfrac{h_{j+1}}{2}, & j=0\\ \dfrac{h_j+h_{j+1}}{2}, & j=1,2,\dots,N_z-1\\ \dfrac{h_j}{2}, & j=N_z\end{cases}$$

Основная сетка для $i=1,2,\dots,N_r-1,\quad j=1,2,\dots,N_z-1$

$$-\left[\int\limits_{r_{i-1/2}}^{r_{i+1/2}}\int\limits_{z_{j-1/2}}^{z_{j+1/2}}\frac{\partial}{\partial r}\left(rk_1\frac{\partial u}{\partial r}\right)drdz+\int\limits_{r_{i-1/2}}^{r_{i+1/2}}\int\limits_{z_{j-1/2}}^{z_{j+1/2}}\frac{\partial}{\partial z}\left(rk_2\frac{\partial u}{\partial z}\right)drdz\right]=\int\limits_{r_{i-1/2}}^{r_{i+1/2}}\int\limits_{z_{j-1/2}}^{z_{j+1/2}}rfdrdz$$

$$-\left[\int_{z_{j-1/2}}^{z_{j+1/2}} rk_1(r_{i+1/2},z)\frac{\partial u}{\partial r}\bigg|_{r=r_{i+1/2}} dz - \int_{z_{j-1/2}}^{z_{j+1/2}} rk_1(r_{i-1/2},z)\frac{\partial u}{\partial r}\bigg|_{r=r_{i-1/2}} dz + \int_{r_{i-1/2}}^{r_{i+1/2}} rk_2(r,z_{j+1/2})\frac{\partial u}{\partial z}\bigg|_{z=z_{j+1/2}} dr - \int_{r_{i-1/2}}^{r_{i+1/2}} rk_2(r,z_{j-1/2})\frac{\partial u}{\partial z}\bigg|_{z=z_{j-1/2}} dr\right] = \int_{r_{i-1/2}}^{r_{i+1/2}}\int_{z_{j-1/2}}^{z_{j+1/2}} rf drdz$$

$$\int_{r_{i-1/2}}^{r_{i+1/2}} \varphi(r,z)dr \approx \hbar_i\varphi_i, \int_{z_{j-1/2}}^{z_{j+1/2}} \varphi(r,z)dz \approx \hbar_i\varphi_j,$$

$$\int_{r_{i-1/2}}^{r_{i+1/2}}\int_{z_{j-1/2}}^{z_{j+1/2}} \varphi drdz \approx \hbar_i\hbar_j\varphi_{i,j}$$

$$k_1(r_{i-1/2},z)\frac{\partial u}{\partial r}\bigg|_{\substack{r=r_{i-1/2}\\ z=z_j}} \approx k_1(r_{i-1/2},z_j)\frac{v_{i,j}-v_{i-1,j}}{h_r}$$

$$k_2(r_i,z_{j-1/2})\frac{\partial u}{\partial z}\bigg|_{\substack{z=z_{j-1/2}\\ r=r_i}} \approx k_2(r_i,z_{j-1/2})\frac{v_{i,j}-v_{i,j-1}}{h_z}$$

$$-\left[h_z r_{i+1/2}k_1(r_{i+1/2},z_j)\frac{v_{i+1,j}-v_{i,j}}{h_r} - h_z r_{i-1/2}k_1(r_{i-1/2},z_j)\frac{v_{i,j}-v_{i-1,j}}{h_r} + h_r r_i k_2(r_i,z_{j+1/2})\frac{v_{i,j+1}-v_{i,j}}{h_z} - h_r r_i k_2(r_i,z_{j-1/2})\frac{v_{i,j}-v_{i,j-1}}{h_z}\right] = h_r h_z r_i f_{i,j}$$

Граничные условия для $i = 0, \quad j = 1,2,\dots,N_z - 1$

$$k_1\frac{\partial u}{\partial r}\bigg|_{r=R_0} = \chi_1 u|_{r=R_0} - g_1(z), \qquad \chi_1 \geq 0$$

$$-\left[\int_{r_i}^{r_{i+1/2}}\int_{z_{j-1/2}}^{z_{j+1/2}} \frac{\partial}{\partial r}\left(rk_1\frac{\partial u}{\partial r}\right)drdz + \int_{r_i}^{r_{i+1/2}}\int_{z_{j-1/2}}^{z_{j+1/2}} \frac{\partial}{\partial z}\left(rk_2\frac{\partial u}{\partial z}\right)drdz\right] = \int_{r_i}^{r_{i+1/2}}\int_{z_{j-1/2}}^{z_{j+1/2}} rf drdz$$

$$-\left[\int_{z_{j-1/2}}^{z_{j+1/2}} rk_1(r_{i+1/2},z)\frac{\partial u}{\partial r}\bigg|_{r=r_{i+1/2}} dz - \int_{z_{j-1/2}}^{z_{j+1/2}} rk_1(r_{i-1/2},z)\frac{\partial u}{\partial r}\bigg|_{r=r_i} dz + \int_{r_i}^{r_{i+1/2}} rk_2(r,z_{j+1/2})\frac{\partial u}{\partial z}\bigg|_{z=z_{j+1/2}} dr - \int_{r_i}^{r_{i+1/2}} rk_2(r,z_{j-1/2})\frac{\partial u}{\partial z}\bigg|_{z=z_{j-1/2}} dr\right] = \int_{r_i}^{r_{i+1/2}}\int_{z_{j-1/2}}^{z_{j+1/2}} rf drdz$$

$$-\left[h_z r_{i+1/2}k_1\left(r_{i+\frac{1}{2}},z_j\right)\frac{v_{i+1,j}-v_{i,j}}{h_r} - h_z r_i\left(\chi_1 u|_{r=R_0} - g_1(z_j)\right) + h_r r_i k_2\left(r_i,z_{j+\frac{1}{2}}\right)\frac{v_{i,j+1}-v_{i,j}}{2h_z} - h_r r_i k_2\left(r_i,z_{j-\frac{1}{2}}\right)\frac{v_{i,j}-v_{i,j-1}}{2h_z}\right] = \frac{h_r h_z r_i f_{i,j}}{2}$$

Граничные условия для $i = N_r, \quad j = 1,2,\dots,N_z - 1$

$$-k_1\frac{\partial u}{\partial r}\bigg|_{r\,=\,R_1} = \chi_2 u|_{r=R_1} - g_2(z), \qquad \chi_2 \geq 0,$$

$$-\left[\int_{r_{i-1/2}}^{r_i}\int_{z_{j-1/2}}^{z_{j+1/2}} \frac{\partial}{\partial r}\left(rk_1\frac{\partial u}{\partial r}\right)drdz + \int_{r_{i-1/2}}^{r_i}\int_{z_{j-1/2}}^{z_{j+1/2}} \frac{\partial}{\partial z}\left(rk_2\frac{\partial u}{\partial z}\right)drdz\right] = \int_{r_{i-1/2}}^{r_i}\int_{z_{j-1/2}}^{z_{j+1/2}} rf drdz$$

$$-\left[\int_{z_{j-1/2}}^{z_{j+1/2}} rk_1(r_{i+1/2},z)\frac{\partial u}{\partial r}\bigg|_{r=r_i} dz - \int_{z_{j-1/2}}^{z_{j+1/2}} rk_1(r_{i-1/2},z)\frac{\partial u}{\partial r}\bigg|_{r=r_{i-1/2}} dz + \int_{r_{i-1/2}}^{r_i} rk_2(r,z_{j+1/2})\frac{\partial u}{\partial z}\bigg|_{z=z_{j+1/2}} dr - \int_{r_{i-1/2}}^{r_i} rk_2(r,z_{j-1/2})\frac{\partial u}{\partial z}\bigg|_{z=z_{j-1/2}} dr\right] = \int_{r_{i-1/2}}^{r_i}\int_{z_{j-1/2}}^{z_{j+1/2}} rf drdz$$

$$-\left[-h_z r_i\left(\chi_2 u|_{r=R_1} - g_2(z_i)\right) - h_z r_{i-1/2}k_1\left(r_{i-\frac{1}{2}},z_j\right)\frac{v_{i,j}-v_{i-1,j}}{h_r} + h_r r_i k_2\left(r_i,z_{j+\frac{1}{2}}\right)\frac{v_{i,j+1}-v_{i,j}}{2h_z} - h_r r_i k_2\left(r_i,z_{j-\frac{1}{2}}\right)\frac{v_{i,j}-v_{i,j-1}}{2h_z}\right] = \frac{h_r h_z r_i f_{i,j}}{2}$$

Граничный условия для $i = 1,2,\dots,N_r, \quad j = 0$

$$u|_{z=0} = g_3(r), \qquad v_{i,j} = g_3(r_i)$$

Граничные условия для $i = 1,2,\dots,N_r, \quad j = N_z$

$$u|_{z=L} = g_4(r), \qquad v_{i,j} = g_4(r_i)$$

Разложение невязки и получение главного члена погрешности аппроксимации

Порядок аппроксимации основной сетки

$$-\left[\frac{\partial}{\partial r}\left(rk_1(r,z)\frac{\partial u}{\partial r}\right)+\frac{\partial}{\partial z}\left(rk_2(r,z)\frac{\partial u}{\partial z}\right)\right]=rf(r,z)$$

$$\widetilde{k_1}(r,z) \;=\; rk_1(r,z),\qquad \widetilde{k_2}(r,z) \;=\; rk_2(r,z),\qquad \tilde{f}(r,z) \;=\; rf(r,z)$$

$$-\left[\frac{\partial}{\partial r}\left(\widetilde{k_1}(r,z)\frac{\partial u}{\partial r}\right)+\frac{\partial}{\partial z}\left(\widetilde{k_2}(r,z)\frac{\partial u}{\partial z}\right)\right]=\tilde{f}(r,z)$$

$$\widetilde{\xi_{\imath,\jmath}}=\frac{\xi_{i,j}}{h_r h_z}$$

$$\begin{aligned}\widetilde{\xi_{\imath,\jmath}}=\widetilde{f_{\imath,\jmath}}&+\widetilde{k_1}\left(r_{i+\frac{1}{2}},z_i\right)\frac{u_{i+1,j}-u_{i,j}}{h_r^2}-\widetilde{k_1}\left(r_{i-\frac{1}{2}},z_i\right)\frac{u_{i,j}-u_{i-1,j}}{h_r^2}+\widetilde{k_2}\left(r_i,z_{i+\frac{1}{2}}\right)\frac{u_{i,j+1}-u_{i,j}}{h_z^2}\\&-\widetilde{k_2}\left(r_i,z_{i-\frac{1}{2}}\right)\frac{u_{i,j}-u_{i,j-1}}{h_z^2}\\&=\left[\tilde{f}+\frac{\partial}{\partial r}\left(\widetilde{k_1}\frac{\partial u}{\partial r}\right)+\frac{\partial}{\partial z}\left(\widetilde{k_2}\frac{\partial u}{\partial z}\right)\right]_{i,j}\\&+h_r^2\left(\frac{1}{12}\widetilde{k_1}\frac{\partial^4 u}{\partial r^4}+\frac{1}{6}\frac{\partial\widetilde{k_1}}{\partial r}\frac{\partial^3 u}{\partial r^3}+\frac{1}{8}\frac{\partial^2\widetilde{k_1}}{\partial r^2}\frac{\partial^2 u}{\partial r^2}+\frac{1}{24}\frac{\partial u}{\partial r}\frac{\partial^3\widetilde{k_1}}{\partial r^3}\right)_{i,j}+O(h_r^3)\\&+h_z^2\left(\frac{1}{12}\widetilde{k_2}\frac{\partial^4 u}{\partial z^4}+\frac{1}{6}\frac{\partial\widetilde{k_2}}{\partial z}\frac{\partial^3 u}{\partial z^3}+\frac{1}{8}\frac{\partial^2\widetilde{k_2}}{\partial z^2}\frac{\partial^2 u}{\partial z^2}+\frac{1}{24}\frac{\partial u}{\partial z}\frac{\partial^3\widetilde{k_2}}{\partial z^3}\right)_{i,j}+O(h_z^3)\end{aligned}$$

Выполним обратную замену.

$$\begin{aligned}\widetilde{\xi_{\imath,\jmath}}&=\left[rf+\frac{\partial}{\partial r}(\mathrm{r}k_1\frac{\partial u}{\partial r})+\frac{\partial}{\partial z}(\mathrm{r}k_2\frac{\partial u}{\partial z})\right]_{i,j}\\&+h_r^2\left(\frac{1}{12}rk_1\frac{\partial^4 u}{\partial r^4}+\frac{1}{6}\frac{\partial rk_1}{\partial r}\frac{\partial^3 u}{\partial r^3}+\frac{1}{8}\frac{\partial^2 rk_1}{\partial r^2}\frac{\partial^2 u}{\partial r^2}+\frac{1}{24}\frac{\partial u}{\partial r}\frac{\partial^3 rk_1}{\partial r^3}\right)_{i,j}+O(h_r^3)\\&+rh_z^2\left(\frac{1}{12}k_2\frac{\partial^4 u}{\partial z^4}+\frac{1}{6}\frac{\partial k_2}{\partial z}\frac{\partial^3 u}{\partial z^3}+\frac{1}{8}\frac{\partial^2 k_2}{\partial z^2}\frac{\partial^2 u}{\partial z^2}+\frac{1}{24}\frac{\partial u}{\partial z}\frac{\partial^3 k_2}{\partial z^3}\right)_{i,j}+O(h_z^3)\end{aligned}$$

$$\left[rf+\frac{\partial}{\partial r}(\mathrm{r}k_1\frac{\partial u}{\partial r})+\frac{\partial}{\partial z}(rk_2\frac{\partial u}{\partial z})\right]_{i,j}=0$$

$$p_r=2-0=2$$

$$\Phi_r=\left[\frac{1}{12}rk_1\frac{\partial^4 u}{\partial r^4}+\frac{1}{6}\frac{\partial rk_1}{\partial r}\frac{\partial^3 u}{\partial r^3}+\frac{1}{8}\frac{\partial^2 rk_1}{\partial r^2}\frac{\partial^2 u}{\partial r^2}+\frac{1}{24}\frac{\partial u}{\partial r}\frac{\partial^3 rk_1}{\partial r^3}\right]$$

$$p_z=2-0=2$$

$$\Phi_z=r\left[\frac{1}{12}k_2\frac{\partial^4 u}{\partial z^4}+\frac{1}{6}\frac{\partial k_2}{\partial z}\frac{\partial^3 u}{\partial z^3}+\frac{1}{8}\frac{\partial^2 k_2}{\partial z^2}\frac{\partial^2 u}{\partial z^2}+\frac{1}{24}\frac{\partial u}{\partial z}\frac{\partial^3 k_2}{\partial z^3}\right]$$

Порядок аппроксимации для $i=0, \quad j=1,2,\dots,N_z-1$

$$k_1\frac{\partial u}{\partial r}\bigg|_{r\,=\,R_0}=\chi_1 u|_{r=R_0}-g_1(z), \qquad \chi_1\ge 0$$

$$\widetilde{\xi_{\iota,J}}=r\left[-k_1\frac{\partial u}{\partial r}+\chi_1 u-g_1(z)\right]_{i,j}+\frac{h_z}{2}\left[rf+\frac{\partial}{\partial r}(rk_1\frac{\partial u}{\partial r})+\frac{\partial}{\partial z}(rk_2\frac{\partial u}{\partial z})\right]_{i,j}$$
$$-\ rh_z^2\left[\frac{1}{6}k_2\frac{\partial^3 u}{\partial z^3}+\frac{1}{4}\frac{\partial k_2}{\partial z}\frac{\partial^2 u}{\partial z^2}+\frac{1}{8}\frac{\partial u}{\partial z}\frac{\partial^2 k_2}{\partial z^2}\right]_{i,j}+O(h_z^3)$$
$$+\frac{h_z}{2}\left[h_r^2\left(\frac{1}{12}rk_1\frac{\partial^4 u}{\partial r^4}+\frac{1}{6}\frac{\partial rk_1}{\partial r}\frac{\partial^3 u}{\partial r^3}+\frac{1}{8}\frac{\partial^2 rk_1}{\partial r^2}\frac{\partial^2 u}{\partial r^2}+\frac{1}{24}\frac{\partial u}{\partial r}\frac{\partial^3 rk_1}{\partial r^3}\right)_{i,j}+O(h_r^3)\right]$$

$$\left[-k_1\frac{\partial u}{\partial r}+\chi_2 u-g_1(z)\right]\bigg|_{r=R_0}=0, \qquad f+\frac{\partial}{\partial r}(k_1\frac{\partial u}{\partial r})+\frac{\partial}{\partial z}(k_2\frac{\partial u}{\partial z})=0$$

$$p_r=2-0=2$$

$$\Omega_r=\left[\frac{1}{24}rk_1\frac{\partial^4 u}{\partial r^4}+\frac{1}{12}\frac{\partial rk_1}{\partial r}\frac{\partial^3 u}{\partial r^3}+\frac{1}{16}\frac{\partial^2 rk_1}{\partial r^2}\frac{\partial^2 u}{\partial r^2}+\frac{1}{48}\frac{\partial u}{\partial r}\frac{\partial^3 rk_1}{\partial r^3}\right]$$

$$p_z=2-0=2$$

$$\Omega_z=r\left[\frac{1}{6}k_2\frac{\partial^3 u}{\partial z^3}+\frac{1}{4}\frac{\partial k_2}{\partial z}\frac{\partial^2 u}{\partial z^2}+\frac{1}{8}\frac{\partial u}{\partial z}\frac{\partial^2 k_2}{\partial z^2}\right]$$

Порядок аппроксимации Для $i=N_r, \quad j=1,2,\dots,N_z-1$

$$-k_1\frac{\partial u}{\partial r}\bigg|_{r\,=\,R_1}=\chi_2 u|_{r=R_1}-g_2(z), \qquad \chi_2\ge 0$$

$$\widetilde{\xi_{\iota,J}}=r\left[k_1\frac{\partial u}{\partial r}+\chi_2 u-g_2(z)\right]_{i,j}+\frac{h_z}{2}\left[rf+\frac{\partial}{\partial r}(rk_1\frac{\partial u}{\partial r})+\frac{\partial}{\partial z}(\mathrm{r}k_2\frac{\partial u}{\partial z})\right]_{i,j}$$
$$-\ h_z^2\left[\frac{1}{6}k_2\frac{\partial^3 u}{\partial z^3}+\frac{1}{4}\frac{\partial k_2}{\partial z}\frac{\partial^2 u}{\partial z^2}+\frac{1}{8}\frac{\partial u}{\partial z}\frac{\partial^2 k_2}{\partial z^2}\right]_{i,j}+O(h_z^3)$$
$$+\frac{h_z}{2}\left[h_r^2\left(\frac{1}{12}rk_1\frac{\partial^4 u}{\partial r^4}+\frac{1}{6}\frac{\partial rk_1}{\partial r}\frac{\partial^3 u}{\partial r^3}+\frac{1}{8}\frac{\partial^2 rk_1}{\partial r^2}\frac{\partial^2 u}{\partial r^2}+\frac{1}{24}\frac{\partial u}{\partial r}\frac{\partial^3 rk_1}{\partial r^3}\right)_{i,j}+O(h_r^3)\right]$$

$$\left[k_1\frac{\partial u}{\partial r}+\chi_2 u-g_2(z)\right]\bigg|_{r=R_1}=0, \qquad \mathrm{r}f+\frac{\partial}{\partial r}(\mathrm{r}k_1\frac{\partial u}{\partial r})+\frac{\partial}{\partial z}(\mathrm{r}k_2\frac{\partial u}{\partial z})=0$$

$$p_r=2-0=2$$

$$\Omega_r=\left[\frac{1}{24}rk_1\frac{\partial^4 u}{\partial r^4}+\frac{1}{12}\frac{\partial rk_1}{\partial r}\frac{\partial^3 u}{\partial r^3}+\frac{1}{16}\frac{\partial^2 rk_1}{\partial r^2}\frac{\partial^2 u}{\partial r^2}+\frac{1}{48}\frac{\partial u}{\partial r}\frac{\partial^3 rk_1}{\partial r^3}\right]$$

$$p_z=2-0=2$$

$$\Omega_z=-r\left[\frac{1}{6}k_2\frac{\partial^3 u}{\partial z^3}+\frac{1}{4}\frac{\partial k_2}{\partial z}\frac{\partial^2 u}{\partial z^2}+\frac{1}{8}\frac{\partial u}{\partial z}\frac{\partial^2 k_2}{\partial z^2}\right]$$

Для остальных граничных условий имеем точное решение.

Преобразования разностной схемы для применения метода сопряженных градиентов.

Разностная схема с приведенными подобными членами:

Основная сетка для $i = 1, 2, \dots, N_r - 1, \quad j = 1, 2, \dots, N_z - 1$:

$$
\begin{aligned}
-\frac{h_z}{h_r} r_{i+1/2} k_1\left(r_{i+\frac{1}{2}}, z_j\right) v_{i+1,j} \\
&+ \left[\frac{h_z}{h_r} r_{i+1/2} k_1(r_{i+1/2}, z_j) + \frac{h_z}{h_r} r_{i-1/2} k_1(r_{i-1/2}, z_j) + \frac{h_r}{h_z} r_i k_2(r_i, z_{j+1/2})\right. \\
&\left.+ \frac{h_r}{h_z} r_i k_2(r_i, z_{j-1/2})\right] v_{i,j} - \frac{h_z}{h_r} r_{i-1/2} k_1(r_{i-1/2}, z_j) v_{i-1,j} \\
&- \frac{h_r}{h_z} r_i k_2(r_i, z_{j+1/2}) v_{i,j+1} - \frac{h_r}{h_z} r_i k_2(r_i, z_{j-1/2}) v_{i,j-1} = h_r h_z r_i f_{i,j}
\end{aligned}
$$

Для $i = 0, \quad j = 1, 2, \dots, N_z - 1$:

$$
\begin{aligned}
-\frac{h_z}{h_r} r_{1/2} k_1\left(r_{\frac{1}{2}}, z_j\right) v_{1,j} \\
&+ \left[\frac{h_z}{h_r} r_{1/2} k_1\left(r_{\frac{1}{2}}, z_j\right) + h_z r_0 \chi_1 + \frac{h_r}{2h_z} r_0 k_2\left(r_0, z_{j+\frac{1}{2}}\right) + \frac{h_r}{2h_z} r_0 k_2\left(r_0, z_{j-\frac{1}{2}}\right)\right] v_{0,j} \\
&- \frac{h_r}{2h_z} r_0 k_2\left(r_0, z_{j+\frac{1}{2}}\right) v_{0,j+1} - \frac{h_r}{2h_z} r_0 k_2\left(r_0, z_{j-\frac{1}{2}}\right) v_{0,j-1} \\
&= \frac{h_r h_z r_0 f_{0,j}}{2} + h_z r_0 g_1(z_j)
\end{aligned}
$$

Для $i = N_r, \quad j = 1, 2, \dots, N_z - 1$:

$$
\begin{aligned}
\left[h_z r_i \chi_2 + \frac{h_z}{h_r} r_{N_r-1/2} k_1(r_{N_r-1/2}, z_j) + \frac{h_r}{2h_z} r_{N_r} k_2(r_{N_r}, z_{j+1/2}) + \frac{h_r}{2h_z} r_{N_r} k_2(r_{N_r}, z_{j-1/2})\right] v_{N_r,j} \\
- \frac{h_z}{h_r} r_{N_r-1/2} k_1\left(r_{N_r-\frac{1}{2}}, z_j\right) v_{N_r-1,j} - \frac{h_r}{2h_z} r_{N_r} k_2\left(r_{N_r}, z_{j+\frac{1}{2}}\right) v_{N_r,j+1} \\
- \frac{h_r}{2h_z} r_{N_r} k_2\left(r_{N_r}, z_{j-\frac{1}{2}}\right) v_{N_r,j-1} = \frac{h_r h_z r_{N_r} f_{N_r,j}}{2} + h_z r_{N_r} g_2(z_j)
\end{aligned}
$$

Для $i = 1, 2, \dots, N_r, \quad j = 0$:

$$v_{i,0} = g_3(r_i)$$

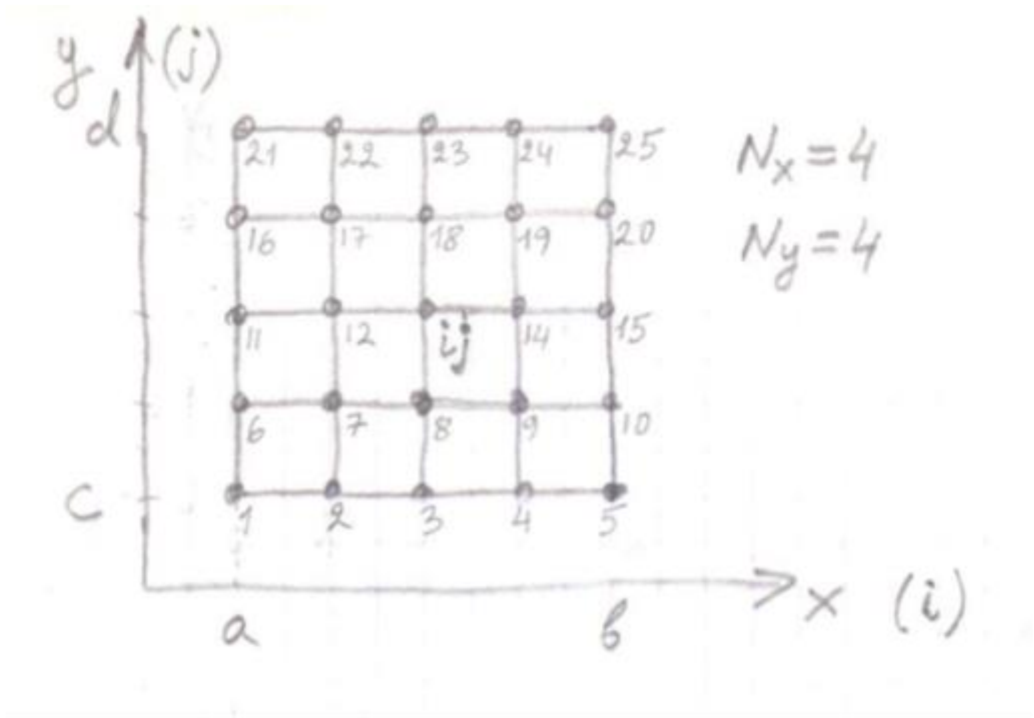
Для $i = 1, 2, \dots, N_r, \quad j = N_z$:

$$v_{i,N_z} = g_4(r_i)$$

Понижение размерности матрицы методом исключения неизвестных

Пронумеруем узлы матрицы следующим образом

Будим принимать, что $N_x = N_r = 4$, $N_y = N_z = 4$



Перейдем к одному индексу $i = 0,1, \ldots, N_r$, $j = 0,1, \ldots, N_Z$

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

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

Основная сетка $i = 1,2, \ldots, N_r - 1$, $j = 1,2, \ldots, N_Z - 1$, $m = jL + i + 1$

$$a_m = -\frac{h_r}{h_z} r_i k_2(r_i, z_{j-1/2}), \qquad b_m = -\frac{h_z}{h_r} r_{i-1/2} k_1(r_{i-1/2}, z_j),$$

$$c_m = \frac{h_z}{h_r} r_{i+1/2} k_1(r_{i+1/2}, z_j) + \frac{h_z}{h_r} r_{i-1/2} k_1(r_{i-1/2}, z_j) + \frac{h_r}{h_z} r_i k_2(r_i, z_{j+1/2}) + \frac{h_r}{h_z} r_i k_2(r_i, z_{j-1/2})$$

$$d_m = -\frac{h_z}{h_r} r_{i+1/2} k_1(r_{i+1/2}, z_j), \qquad e_m = -\frac{h_r}{h_z} r_i k_2(r_i, z_{j+1/2}), \qquad g_m = h_r h_z r_i f_{i,j}$$

$$a_m w_{m-L} + b_m w_{m-1} + c_m w_m + d_m w_{m+1} + e_m w_{m-L} = g_m$$

Для $i = 0$, $j = 1, 2, \ldots, N_z - 1$, $m = jL + 1$

$$a_m = -\frac{h_r}{2h_z} r_0 k_2(r_0, z_{j-1/2}), \qquad b_m = 0,$$

$$c_m = \frac{h_z}{h_r} r_{1/2} k_1(r_{1/2}, z_j) + h_z r_0 \chi_1 + \frac{h_r}{2h_z} r_0 k_2(r_0, z_{j+1/2}) + \frac{h_r}{2h_z} r_0 k_2(r_0, z_{j-1/2})$$

$$d_m = -\frac{h_z}{h_r} r_{\frac{1}{2}} k_1\left(r_{\frac{1}{2}}, z_j\right), \qquad e_m = -\frac{h_r}{2h_z} r_0 k_2\left(r_0, z_{j+\frac{1}{2}}\right), \qquad g_m = \frac{h_r h_z r_0 f_{0,j}}{2} + h_z r_0 g_1(z_j)$$

$$a_m w_{m-L} + 0 w_{m-1} + c_m w_m + d_m w_{m+1} + e_m w_{m-L} = g_m$$

$$a_m w_{m-L} + c_m w_m + d_m w_{m+1} + e_m w_{m-L} = g_m$$

Для $i = N_r, \quad j = 1, 2, \dots, N_z - 1, m = jL + 1 + N_r$

$$a_m = -\frac{h_r}{2h_z} r_{N_r} k_2(r_{N_r}, z_{j-1/2}), \qquad b_m = -\frac{h_z}{h_r} r_{N_r-1/2} k_1(r_{N_r-1/2}, z_j),$$

$$c_m = h_z r_i \chi_2 + \frac{h_z}{h_r} r_{N_r-1/2} k_1(r_{N_r-1/2}, z_j) + \frac{h_r}{2h_z} r_{N_r} k_2(r_{N_r}, z_{j+1/2}) + \frac{h_r}{2h_z} r_{N_r} k_2(r_{N_r}, z_{j-1/2})$$

$$d_m = 0, \qquad e_m = -\frac{h_r}{2h_z} r_{N_r} k_2\left(r_{N_r}, z_{j+\frac{1}{2}}\right), \qquad g_m = \frac{h_r h_z r_{N_r} f_{N_r,j}}{2} + h_z r_{N_r} g_2(z_j)$$

$$a_m w_{m-L} + b_m w_{m-1} + c_m w_m + 0 w_{m+1} + e_m w_{m+L} = g_m$$

$$a_m w_{m-L} + b_m w_{m-1} + c_m w_m + e_m w_{m+L} = g_m$$

Для $i = 1, 2, \dots, N_r, \quad j = 0, \; m = 1 + i$

$$c_m = 1, \qquad g_m = g_3(r_i), \qquad c_m w_m = g_m$$

Для $i = 1, 2, \dots, N_r, \quad j = N_z, \; m = N_z L + i + 1$

$$c_m = 1, \qquad g_m = g_4(r_i), \qquad c_m w_m = g_m$$

j	i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0	0	1																								
	1		1																							
	2			1																						
	3				1																					
	4					1																				
1	0	*					*	*				*														
	1		*				*	*	*				*													
	2			*				*	*	*				*												
	3				*				*	*	*				*											
	4					*				*	*					*										
2	0						*					*	*				*									
	1							*				*	*	*				*								
	2								*				*	*	*				*							
	3									*				*	*	*				*						
	4										*				*	*					*					
3	0											*					*	*				*				
	1												*				*	*	*				*			
	2													*				*	*	*				*		
	3														*				*	*	*				*	
	4															*				*	*					*
4	0																					1				
	1																						1			
	2																							1		
	3																								1	
	4																									1

Также мы можем сделать матрицу симметричной, для этого исключаем элементы путем домножение 1 на соответствующие помеченные элементы и с помощью вычитания избавляемся от них. В итоге мы получили СЛАУ:

$$\mathrm{A}w = g, \qquad A = A^T, \qquad (Ay, y) > 0, \qquad y \neq 0$$

где A-матрица, w-вектор неизвестных, g-вектор правой части. Решение алгебраической системы проводится метод сопряженных градиентов, для которого необходимо, чтобы матрица A была симметрична и положительно определена.

Пусть $w^{(0)}$ – произвольное начальное приближение, тогда $Aw - Aw^{(0)} = g - Aw^{(0)}$, что даст нам невязку $r^{(0)} = A(w - w^{(0)})$, предполагается, что у нас есть система из $s^{(i)}, i = 1, 2, \dots, n$, линейно-независимых векторов, тогда можем разложит по базису этих векторов с соответствующими коэффициентами $w - w^{(0)} = \sum_{i=1}^{n} a_i s^{(i)}$, найти коэффициенты можем с помощью СЛАУ $\sum_{i=1}^{n} a_i A s^{(i)} = r^{(0)}$, решение системы сильно упростится, если $\left(As^{(i)}, s^{(j)}\right)$ при $i \neq j$, а при $i = j$, скалярное произведение равнялось не 0 значению, в таком случае мы говорим об аортагональности. Из этого мы можем выразить коэффициенты $a_i = \frac{(r^{(0)}, s^{(i)})}{(As^{(i)}, s^{(i)})}$, и выразить решение $\mathrm{w} = w^{(0)} + \sum_{i=1}^{n} a_i s^{(i)}$.

Рассмотрим частичную сумму $w^{(n)} = w$, $w^{(n)} = w^{(0)} + \sum_{i=1}^{n} a_i s^{(i)}$, $w^{(k)} = w^{(0)} + \sum_{i=1}^{k} a_i s^{(i)}$,

$w^{(k)} = w^{(k-1)} + a_k s^{(k)}$, $w^{(k)} = w^{(k-1)} + a_k s^{(k)}$, для невязки получим рекуррентное соотношение $r^{(k)} = r^{(k-1)} - a_k A s^{(k)}$.

$$w^{(0)}, \qquad r^{(0)} = \mathrm{g} - Aw^{(0)}, \qquad s^{(1)} = ?$$

$$k = 1, 2, \dots, n, \qquad a_k = \frac{\left(r^{(0)}, s^{(k)}\right)}{(As^{(k)}, s^{(k)})}$$

$$w^{(k)} = w^{(k-1)} + a_k s^{(k)}, \qquad r^{(k)} = r^{(k-1)} - a_k A s^{(k)}$$

$$s^{(k+1)} = ?$$

При явном методе сопряженных градиентов $s^{(1)}$ берут равным $r^{(0)}$, а $s^{(k+1)} = r^{(k)} + \beta_k s^{(k)}$, с вводом дополнительного коэффициента $\beta_k = \frac{(r^{(k)},\ r^{(k)})}{(r^{(k-1)},\ r^{(k-1)})}$, при $\sqrt{(r^{(k)},\ r^{(k)})} < \gamma\varepsilon$, явный метод обладает тем свойством что при отсутствии ошибок округления мы можем получить точное решение не позднее чем на n-ом шаге, но возникает двойственность, из-за ошибок округления происходит разрушение аортогональности последовательности s и в результате к неточности, и метод становится итерационным.

Неявный метод

$$\mathrm{A}w = b, \qquad A = A^T, \qquad (Ay, y) > 0, \qquad y \neq 0$$

$$x^{(0)} - \text{произвольное начальное приблидение}$$

$$r^{(0)} = b - Ax^{(0)}, \qquad Bw^{(0)} = r^{(0)}, \qquad s^{(1)} = w^{(0)}, \qquad Bg = b, \qquad \gamma = \sqrt{(g, b)}$$

$$k = 1,2, \dots, K_{max}$$

$$a_k = \frac{\left(w^{(k-1)}, r^{(k-1)}\right)}{\left(As^{(k-1)}, s^{(k-1)}\right)}$$

$$x^{(k)} = x^{(k-1)} + a_k s^{(k)}, \qquad r^{(k)} = r^{(k-1)} + a_k A s^{(k-1)}$$

$$Bw^{(k)} = r^{(k)}, \sqrt{\left(w^{(k)}, r^{(k)}\right)} < \gamma\varepsilon$$

$$\beta_k = \frac{\left(w^{(k)}, r^{(k)}\right)}{\left(w^{(k-1)}, r^{(k-1)}\right)}, \qquad s^{(k+1)} = w^{(k)} + \beta_k s^{(k)}$$

О выборе матрицы предобусловливания

$$\mathrm{A}w = b, \qquad A = A^T, \qquad (Ay, y) > 0, \qquad y \neq 0$$

$$B = B^T, \qquad (By, y) > 0, \qquad y \neq 0$$

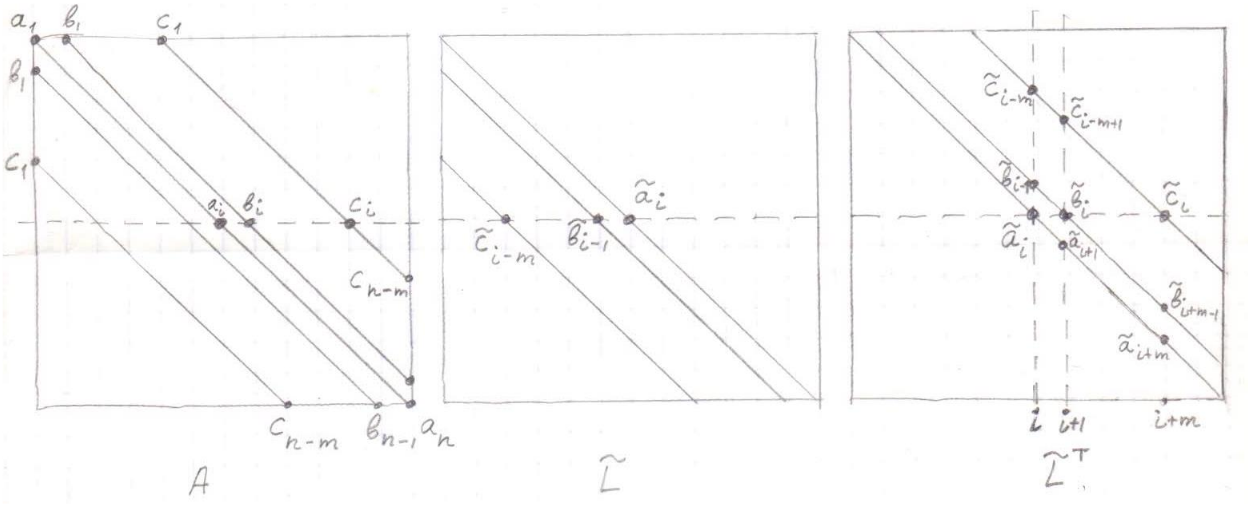
$$B = D, \qquad D = \begin{bmatrix} a_{11} & \_ & \_ \\ \_ & \cdots & \_ \\ \_ & \_ & a_{nn} \end{bmatrix}, \qquad B = \tilde{L}\,\tilde{L}^T$$

$$\tilde{l}_{ij} = 0, \quad i < j$$

$$Bw^{(0)} = r^{(0)}, \qquad \tilde{L}y_0 = r_0, \qquad \tilde{L}^T w_0 = y_0,$$

$$Bw^{(k)} = r^{(k)}, \qquad \tilde{L}y_k = r_k, \qquad \tilde{L}^T w_k = y_k$$

Неполное разложение Холевского

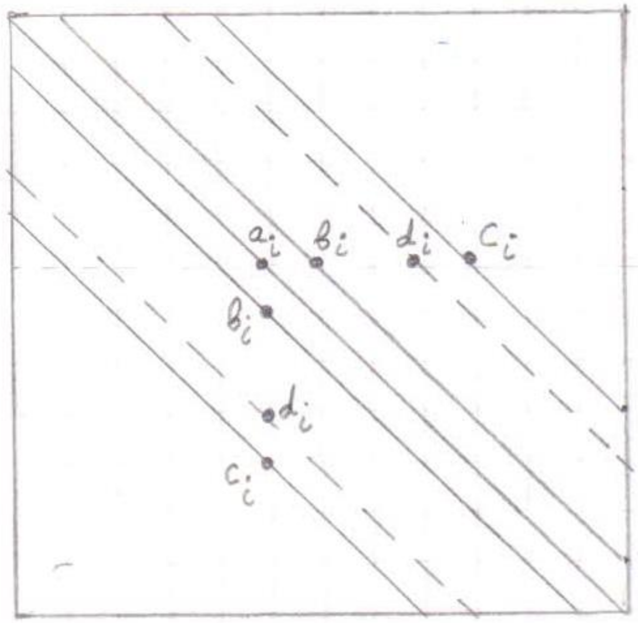


$$a_i = \tilde{a}_i^2 + \tilde{b}_{i-1}^2 + \tilde{c}_{i-m}^2, \qquad b_i = \tilde{a}_i \tilde{b}_i, \qquad c_i = \tilde{a}_i \tilde{c}_i,$$

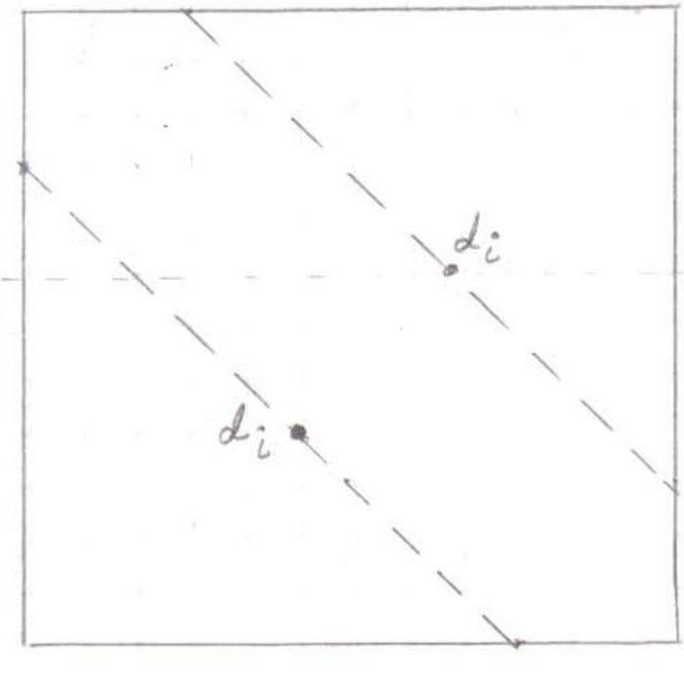
$$\tilde{a}_i = \sqrt{a_i - \tilde{b}_{i-1}^2 - \tilde{c}_{i-m}^2}, \quad i = 1,2,\ldots,n, \quad \tilde{b}_0 = 0, \quad \tilde{c}_{i-m} = 0, \quad i = 1,2,\ldots,m$$

$$\tilde{b}_i = \frac{b_i}{\tilde{a}_i}, \qquad \tilde{c}_i = \frac{c_i}{\tilde{a}_i},$$

$\tilde{L}\tilde{L}^T =$



$A - \tilde{L}\tilde{L}^T =$



Тесты

Параметр $\varepsilon = 10^{-8}$

$$-\left[\frac{1}{r}\frac{\partial}{\partial r}\left(rk_1(r,z)\frac{\partial u}{\partial r}\right)+\frac{\partial}{\partial z}\left(k_2(r,z)\frac{\partial u}{\partial z}\right)\ \right]=f(r,z)$$

$$0 < c_{11} \le k_1(r,z) \le c_{12}, \qquad 0 < c_{21} \le k_2(r,z) \le c_{22}, \qquad 0 < R_0 \le r \le R_1, \qquad 0 \le z \le L$$

$$k_1\frac{\partial u}{\partial r}\bigg|_{r\,=\,R_0} = \chi_1 u|_{r=R_0} - g_1(z), \qquad \chi_1 \ge 0,$$

$$-k_1\frac{\partial u}{\partial r}\bigg|_{r\,=\,R_1} = \chi_2 u|_{r=R_1} - g_2(z), \qquad \chi_2 \ge 0,$$

$$u|_{z=0} = g_3(r), \qquad u|_{z=L} = g_4(r)$$

$$R_0 = 1, \qquad R_1 = 2, \qquad L = 1, \qquad \chi_1 = 1, \qquad \chi_2 = 1$$

Константный случай

$$u = \ k1 = k2 = g_1 = g_2 = g_3 = g_4 = 1, \qquad f = 0$$

N	Итерации	Ошибка	Отношение к предыдущей
4	8	8.881784197001252E-16	0
8	13	3.0195148603695543E-9	2.941460667596855E-7
16	23	1.0806025185772228E-8	0.27942881942799874
32	42	2.3599347809266646E-8	0.45789507714823696
64	73	2.6286663179497793E-8	0.8977688665966889
128	141	6.191928259013935E-8	0.42453113278938326

Линейный случай

$$u = r + z, \qquad k1 = k2 = \ 1,$$

$$g_1 = R_0 + z - 1 = z, \qquad g_2 = R_1 + z - 1 = 1 + z,$$

$$g_3 = r, \qquad g_4 = r + L = r + 1,$$

$$f = -\frac{1}{r}$$

N	Итерации	Ошибка	Отношение к предыдущей
4	9	1.9539925233402755E-14	0
8	14	1.5312066947714698E-8	1.2761128396397888E-6
16	25	3.259752423545592E-8	0.46973097825202176
32	44	5.624021048866723E-8	0.579612415249131
64	86	1.2731810761223983E-7	0.4417298650083025
128	165	2.902959801431848E-7	0.43858033290520176

Нелинейный случай

$$u = k1 = k2 = r^2 + z^2,$$

$$g_1 = R_0^2 + z^2 - 2R_0(R_0^2 + z^2) = -z^2 - 1\ ,$$

$$g_2 = R_1^2 + z^2 + 2R_1(R_1^2 + z^2) = 5z^2 + 20,$$

$$g_3 = r^2, \qquad g_4 = r^2 + L^2 = r^2 + 1,$$

$$f = -10r^2 - 10z^2$$

N	Итерации	Ошибка	Отношение к предыдущей
4	9	0.03381672021202764	0
8	15	0.008919964249462176	3.791127325882119
16	27	0.0022537397644877277	3.9578501431329505
32	50	5.655931737542641E-4	3.984736501552829
64	95	1.414473268566141E-4	3.998613380142623
128	186	3.4983830765389357E-5	4.043220075159766

Вывод

В линейном и константном случаях погрешность аппроксимации отсутствует, ее небольшой рост с увеличением количества разбиений связано с накоплением ошибки округления. А в нелинейном случае наблюдается уменьшение ошибки в 4 раза при увеличении в 2 раза разбиений по оси r и z (то есть шаг сетки уменьшается), небольшие отклонения связаны так же с ошибками округления.

Приложение 1

Код программ был написан на языке программирования Java 16.

```
import java.util.Arrays;
import java.util.HashMap;
import java.util.function.Function;

public class Main {

    private final static double EPS = 1e-8;
    private static int N = 5;
    private static final double X = 1.0;
    private static final double R0 = 1;
    private static final double R1 = 2;
    private static final double L = 1;
    private static final double Chi1 = 1;
    private static final double Chi2 = 1;


    private enum SystemParameters {
        DIAGONAL_A,
        DIAGONAL_B,
        DIAGONAL_C,
        VECTOR_G
    }

    @FunctionalInterface
    public interface FunctionTwoArgs<A, B, R> {
        R apply(A a, B b);
    }

    public static void main(String[] args) {
        System.out.println("Константный случай");
        test((r, z) -> 1.0,
                (r, z) -> 1.0,
                (r, z) -> 0.0,
                (z) -> 1.0,
                (z) -> 1.0,
                (r) -> 1.0,
                (r) -> 1.0,
                (r, z) -> 1.0);

        System.out.println("Линейный случай");
        test((r, z) -> 1.0,
                (r, z) -> 1.0,
                (r, z) -> -1 / r,
                (z) -> R0 + z - 1,
                (z) -> R1 + z + 1,
                (r) -> r,
                (r) -> r + L,
                Double::sum);

        System.out.println("Нелинейный случай");
        test((r, z) -> r * r + z * z,
                (r, z) -> r * r + z * z,
                (r, z) -> -10 * r * r - 10 * z * z,
                (z) -> R0 * R0 + z * z - 2 * R0 * R0 * R0 - 2 * R0 * z * z,
                (z) -> R1 * R1 + z * z + 2 * R1 * R1 * R1 + 2 * R1 * z * z,
                (r) -> r * r,
                (r) -> r * r + L * L,
```

```
                (r, z) -> r * r + z * z);
    }

    private static void test(FunctionTwoArgs<Double, Double, Double> k1,
                             FunctionTwoArgs<Double, Double, Double> k2,
                             FunctionTwoArgs<Double, Double, Double> f,
                             Function<Double, Double> g1,
                             Function<Double, Double> g2,
                             Function<Double, Double> g3,
                             Function<Double, Double> g4,
                             FunctionTwoArgs<Double, Double, Double> u) {
        HashMap<SystemParameters, double[]> system;
        N = 5;
        double hR = (R1 - R0) / (N - 1);
        double hZ = L / (N - 1);
        double r;
        double z = 0;
        double[] result = new double[N * N];
        system = getSystem(k1, k2, f, g1, g2, g3, g4);
        for (int i = 0; i < N; ++i) {
            r = R0;
            for (int j = 0; j < N; ++j) {
                result[i * N + j] = u.apply(r, z);
                r += hR;
            }
            z += hZ;
        }
        System.out.println("Отклонения от точного решения\n" +
Arrays.toString(sub(multiply(system, result),
                system.get(SystemParameters.VECTOR_G))));
        System.out.println("Ошибка");
        double prevError = 0;
        double nowError;
        N = 5;
        for (int i = 2; i <= 8; ++i) {
            N = (int) Math.round(Math.pow(2, i)) + 1;
            system = getSystem(k1, k2, f, g1, g2, g3, g4);
            result = leastGradientMethod(system,
system.get(SystemParameters.VECTOR_G), getEMatrix());
            nowError = getMaxError(result, u);
            System.out.println("N = " + (N - 1) + " Error = " + nowError + "
Ratio = " + prevError / nowError);
            prevError = nowError;
        }
    }


    private static double[] leastGradientMethod(HashMap<SystemParameters,
double[]> system, double[] first,
                                                HashMap<SystemParameters,
double[]> bMatrix) {
        double[] result = Arrays.copyOf(first, first.length);
        double[] r = sub(system.get(SystemParameters.VECTOR_G),
multiply(system, first));
        double[] p = solveB(bMatrix, r);
        double[] b = solveB(bMatrix, system.get(SystemParameters.VECTOR_G));
        double[] s = Arrays.copyOf(p, p.length);
        double alpha;
        double beta;
        double[] newR;
        double[] newP;
        int k;
        for (k = 1; k <= 10000; k++) {
```

```
            alpha = multiply(p, r) / multiply(multiply(system, s), s);
            result = addition(result, multiply(alpha, s));
            newR = sub(r, multiply(alpha, multiply(system, s)));
            newP = solveB(bMatrix, newR);
            double check = Math.sqrt(multiply(newP, newR) / multiply(b,
system.get(SystemParameters.VECTOR_G)));
            if (check < EPS) {
                ++k;
                break;
            }
            beta = multiply(newP, newR) / multiply(p, r);
            s = addition(newP, multiply(beta, s));
            r = newR;
            p = newP;
        }
        System.out.println("K = " + k);
        return result;
    }


    private static double[] getADiag(FunctionTwoArgs<Double, Double, Double>
k2) {
        double hR = (R1 - R0) / (N - 1);
        double hZ = L / (N - 1);
        double scale = hR / hZ;
        double[] result = new double[N * N];
        double z = hZ;
        double r;
        for (int j = 1; j < N - 1; j++) {
            r = R0;
            result[j * N] = -(scale / 2) * r * k2.apply(r, z - hZ / 2);
            r += hR;
            for (int i = 1; i < N - 1; i++) {
                result[j * N + i] = -(scale) * r * k2.apply(r, z - hZ / 2);
                r += hR;
            }
            result[j * N + N - 1] = -(scale / 2) * r * k2.apply(r, z - hZ /
2);
            z += hZ;
        }
        return result;
    }

    private static double[] getCDiag(FunctionTwoArgs<Double, Double, Double>
k1,
                                     FunctionTwoArgs<Double, Double, Double>
k2) {
        double hR = (R1 - R0) / (N - 1);
        double hZ = L / (N - 1);
        double scale = hZ / hR;
        double z = hZ;
        double r;
        double[] result = new double[N * N];
        for (int i = 0; i < N; i++) {
            result[i] = 1;
        }
        for (int j = 1; j < N - 1; j++) {
            r = R0;
            result[j * N] = scale * (r + hR / 2) * k1.apply(r + hR / 2, z)
                    + hZ * r * Chi1
                    + (1 / scale / 2) * r * k2.apply(r, z + hZ / 2)
                    + (1 / scale / 2) * r * k2.apply(r, z - hZ / 2);
            r += hR;
```

```
            for (int i = 1; i < N - 1; i++) {
                result[j * N + i] = scale * (r + hR / 2) * k1.apply(r + hR /
2, z)
                        + scale * (r - hR / 2) * k1.apply(r - hR / 2, z)
                        + (1 / scale) * r * k2.apply(r, z + hZ / 2)
                        + (1 / scale) * r * k2.apply(r, z - hZ / 2);
                r += hR;
            }
            result[j * N + N - 1] = hZ * r * Chi2
                    + scale * (r - hR / 2) * k1.apply(r - hR / 2, z)
                    + (1 / scale / 2) * r * k2.apply(r, z + hZ / 2)
                    + (1 / scale / 2) * r * k2.apply(r, z - hZ / 2);
            z += hZ;
        }
        for (int i = 0; i < N; i++) {
            result[N * (N - 1) + i] = 1;
        }
        return result;
    }

    private static double[] getDDiag(FunctionTwoArgs<Double, Double, Double>
k1) {
        double hR = (R1 - R0) / (N - 1);
        double hZ = L / (N - 1);
        double scale = hZ / hR;
        double z = hZ;
        double r;
        double[] result = new double[N * N];
        for (int j = 1; j < N - 1; j++) {
            r = R0;
            for (int i = 0; i < N - 1; i++) {
                result[j * N + i] = -scale * (r + hR / 2) * k1.apply(r + hR /
2, z);
                r += hR;
            }
            z += hZ;
        }
        return result;
    }

    private static double[] getEDiag(FunctionTwoArgs<Double, Double, Double>
k2) {
        double hR = (R1 - R0) / (N - 1);
        double hZ = L / (N - 1);
        double scale = hR / hZ;
        double[] result = new double[N * N];
        double z = hZ;
        double r;
        for (int j = 1; j < N - 1; j++) {
            r = R0;
            result[j * N] = -scale * r * k2.apply(r, z + hZ / 2) / 2;
            r += hR;
            for (int i = 1; i < N - 1; i++) {
                result[j * N + i] = -scale * r * k2.apply(r, z + hZ / 2);
                r += hR;
            }
            result[j * N + N - 1] = -scale * r * k2.apply(r, z + hZ / 2) / 2;
            z += hZ;
        }
        return result;
    }

    private static double[] getVectorG(FunctionTwoArgs<Double, Double,
```

```
Double> f,
                                               Function<Double, Double> g1,
                                               Function<Double, Double> g2,
                                               Function<Double, Double> g3,
                                               Function<Double, Double> g4) {
        double hR = (R1 - R0) / (N - 1);
        double hZ = L / (N - 1);
        double[] result = new double[N * N];
        double z = hZ;
        double r = R0;
        for (int i = 0; i < N; i++) {
            result[i] = g3.apply(r);
            r += hR;
        }
        for (int j = 1; j < N - 1; j++) {
            r = R0;
            result[j * N] = hR * hZ * r * f.apply(r, z) / 2
                    + hZ * r * g1.apply(z);
            r += hR;
            for (int i = 1; i < N - 1; i++) {
                result[j * N + i] = hR * hZ * r * f.apply(r, z);
                r += hR;
            }
            result[j * N + N - 1] = hR * hZ * r * f.apply(r, z) / 2
                    + hZ * r * g2.apply(z);
            z += hZ;
        }
        r = R0;
        for (int i = 0; i < N; i++) {
            result[N * (N - 1) + i] = g4.apply(r);
            r += hR;
        }
        return result;
    }

    private static HashMap<SystemParameters, double[]>
getSystem(FunctionTwoArgs<Double, Double, Double> k1,

FunctionTwoArgs<Double, Double, Double> k2,

FunctionTwoArgs<Double, Double, Double> f,

Function<Double, Double> g1,

Function<Double, Double> g2,

Function<Double, Double> g3,

Function<Double, Double> g4) {
        double[] a = getADiag(k2);
        double[] c = getCDiag(k1, k2);
        double[] d = getDDiag(k1);
        double[] e = getEDiag(k2);
        double[] g = getVectorG(f, g1, g2, g3, g4);

        for (int i = 0; i < N; i++) {
            g[N + i] -= g[i] * a[N + i];
            a[N + i] = 0;
            g[N * (N - 2) + i] -= g[N * (N - 1) + i] * e[N * (N - 2) + i];
            e[N * (N - 2) + i] = 0;
        }

        HashMap<SystemParameters, double[]> system = new HashMap<>();
```

```
        system.put(SystemParameters.DIAGONAL_A, c);
        system.put(SystemParameters.DIAGONAL_B, d);
        system.put(SystemParameters.DIAGONAL_C, e);
        system.put(SystemParameters.VECTOR_G, g);
        return system;
    }

    private static double getMaxError(double[] solve, FunctionTwoArgs<Double,
Double, Double> u) {
        double hR = (R1 - R0) / (N - 1);
        double hZ = L / (N - 1);
        double z = 0;
        double r;
        double maxError = 0;
        double nowError;
        for (int j = 0; j < N; j++) {
            r = R0;
            for (int i = 0; i < N; i++) {
                nowError = Math.abs(u.apply(r, z) - solve[j * N + i]);
                if (nowError > maxError) {
                    maxError = nowError;
                }
                r += hR;
            }
            z += hZ;
        }
        return maxError;
    }

    private static HashMap<SystemParameters, double[]>
getBMatrix(HashMap<SystemParameters, double[]> system) {
        HashMap<SystemParameters, double[]> result = new HashMap<>();
        int squareN = N * N;
        double[] a = new double[squareN];
        double[] b = new double[squareN];
        double[] c = new double[squareN];
        result.put(SystemParameters.DIAGONAL_A, a);
        result.put(SystemParameters.DIAGONAL_B, b);
        result.put(SystemParameters.DIAGONAL_C, c);
        a[0] = Math.sqrt(system.get(SystemParameters.DIAGONAL_A)[0]);
        for (int i = 1; i < N; i++) {
            b[i - 1] = system.get(SystemParameters.DIAGONAL_B)[i - 1] / a[i -
1];
            a[i] = Math.sqrt(system.get(SystemParameters.DIAGONAL_A)[i] -
Math.pow(b[i - 1], 2));
        }
        for (int i = N; i < squareN; i++) {
            c[i - N] = system.get(SystemParameters.DIAGONAL_C)[i - N];
            b[i - 1] = system.get(SystemParameters.DIAGONAL_B)[i - 1] / a[i -
1];
            a[i] = Math.sqrt(system.get(SystemParameters.DIAGONAL_A)[i] -
Math.pow(b[i - 1], 2) - Math.pow(c[i - N], 2));
        }
        return result;
    }

    private static double[] solveB(HashMap<SystemParameters, double[]>
bMatrix, double[] g) {
        int squareN = N * N;

        double[] y = new double[squareN];
        double[] a = bMatrix.get(SystemParameters.DIAGONAL_A);
        double[] b = bMatrix.get(SystemParameters.DIAGONAL_B);
```

```
        double[] c = bMatrix.get(SystemParameters.DIAGONAL_C);
        y[0] = g[0] / a[0];
        for (int i = 1; i < N; i++) {
            y[i] = (g[i] - b[i - 1] * y[i - 1]) / a[i];
        }
        for (int i = N; i < squareN; i++) {
            y[i] = (g[i] - b[i - 1] * y[i - 1] - c[i - N] * y[i - N]) / a[i];
        }

        double[] result = new double[squareN];
        result[squareN - 1] = y[squareN - 1] / a[squareN - 1];
        for (int i = squareN - 2; i >= N * (N - 1); i--) {
            result[i] = (y[i] - b[i] * result[i + 1]) / a[i];
        }
        for (int i = N * (N - 1) - 1; i >= 0; i--) {
            result[i] = (y[i] - b[i] * result[i + 1] - c[i] * result[i + N])
/ a[i];
        }
        return result;
    }

    private static HashMap<SystemParameters, double[]> getEMatrix() {
        HashMap<SystemParameters, double[]> e = new HashMap<>();
        int squareN = N * N;
        double[] a = new double[squareN];
        for (int j = 0; j < squareN; j++) {
            a[j] = 1;
        }
        e.put(SystemParameters.DIAGONAL_A, a);
        e.put(SystemParameters.DIAGONAL_B, new double[squareN]);
        e.put(SystemParameters.DIAGONAL_C, new double[squareN]);
        return e;
    }

    private static double multiply(double[] leftVector, double[] rightVector)
{
        double result = 0;
        for (int i = 0; i < leftVector.length; i++) {
            result += leftVector[i] * rightVector[i];
        }
        return result;
    }

    private static double[] multiply(HashMap<SystemParameters, double[]>
system, double[] vector) {
        double[] result = new double[vector.length];
        double[] diagA = system.get(SystemParameters.DIAGONAL_A);
        double[] diagB = system.get(SystemParameters.DIAGONAL_B);
        double[] diagC = system.get(SystemParameters.DIAGONAL_C);
        for (int i = 0; i < vector.length; i++) {
            result[i] = diagA[i] * vector[i];
        }
        for (int i = 0; i < vector.length - 1; i++) {
            result[i] += diagB[i] * vector[i + 1];
        }
        for (int i = 0; i < vector.length - N; i++) {
            result[i] += diagC[i] * vector[i + N];
        }
        for (int i = 1; i < vector.length; i++) {
            result[i] += diagB[i - 1] * vector[i - 1];
        }
        for (int i = N; i < vector.length; i++) {
            result[i] += diagC[i - N] * vector[i - N];
```

```
        }
        return result;
    }

    private static double[] multiply(double number, double[] vector) {
        double[] result = new double[vector.length];
        for (int i = 0; i < vector.length; i++) {
            result[i] = vector[i] * number;
        }
        return result;
    }

    private static double[] addition(double[] leftVector, double[]
rightVector) {
        double[] result = new double[leftVector.length];
        for (int i = 0; i < leftVector.length; i++) {
            result[i] = leftVector[i] + rightVector[i];
        }
        return result;
    }

    private static double[] sub(double[] leftVector, double[] rightVector) {
        double[] result = new double[leftVector.length];
        for (int i = 0; i < leftVector.length; i++) {
            result[i] = leftVector[i] - rightVector[i];
        }
        return result;
    }
}
```