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

Лабораторная работа №2
по дисциплине «Уравнения математической физики»

РЕШЕНИЕ НЕЛИНЕЙНЫХ НАЧАЛЬНО-КРАЕВЫХ ЗАДАЧ

Группа ПМ-92 ИВАНОВ ВЛАДИСЛАВ

Вариант 5 КУТУЗОВ ИВАН

Преподаватели ПАТРУШЕВ ИЛЬЯ ИГОРЕВИЧ

ЗАДОРОЖНЫЙ АЛЕКСАНДР ГЕННАДЬЕВИЧ

Новосибирск, 2022

Содержание

1 Цель работы

Разработать программу решения нелинейной одномерной краевой задачи методом конечных элементов. Провести сравнение метода простой итерации и метода Ньютона для решения данной задачи.

2 Задание

1. Выполнить конечноэлементную аппроксимацию исходного уравнения в соответствии с заданием. Получить формулы для вычисления компонент матрицы A и вектора правой части b для метода простой итерации.

2. Реализовать программу решения нелинейной задачи методом простой итерации с учетом следующих требований:
   - язык программирования C++ или Фортран;
   - предусмотреть возможность задания неравномерных сеток по пространству и по времени, разрывность параметров уравнения по подобластям, учет краевых условий;
   - матрицу хранить в ленточном формате, для решения СЛАУ использовать метод LU-разложения;
   - предусмотреть возможность использования параметра релаксации.
3. Выполнить линеаризацию нелинейной системы алгебраических уравнений с использованием метода Ньютона. Получить формулы для вычисления компонент линеаризованных матрицы A^L и вектора правой части b^L.
4. Реализовать программу решения нелинейной задачи методом Ньютона.
5. Протестировать разработанные программы.
6. Исследовать реализованные методы на различных зависимостях коэффициента от решения (или производной решения) в соответствии с заданием. На одних и тех же задачах сравнить по количеству итераций метод простой итерации и метод Ньютона. Исследовать скорость сходимости от параметра релаксации.

Уравнение:

$$-div(\lambda(u)gradu) + \sigma\frac{du}{dt} = f$$

Базисные функции линейные.

3 Конечноэлементная аппроксимация

Временная аппроксимация по двуслойной неявной схеме:

$$-div(\lambda(u_s)gradu_s) + \frac{\sigma}{\Delta t_s}u_s = f + \frac{\sigma}{\Delta t_s}u_{s-1}$$

Для конечноэлементной аппроксимации имеем систему нелинейных уравнений:

$$A(q_s)q_s = b(q_s)$$

у которой компоненты матрицы $\mathbf{A(q_s)q_s}$ и вектора правой части $\mathbf{b(q_s)}$ вычисляются следующим образом:

$$A_{ij}(q_s) = \int_\Omega \lambda_s(u^h(q_s))grad\psi_i grad\psi_j d\Omega + \frac{1}{\Delta t_s}\int_\Omega \sigma_s(u^h(q_s))\psi_i\psi_j d\Omega + \int_{S_3} \beta_s(u^h(q_s))\psi_i\psi_j dS$$

$$\begin{aligned} b_i(q_s) = \int_\Omega f_s(u^h(q_s))\psi_i d\Omega + \frac{1}{\Delta t_s}\int_\Omega (u^h(q_s))(u^h(q_{s-1}))d\Omega + \\ + \int_{S_2} \Theta_s(u^h(q_s))\psi_i dS + \int_{S_2} \beta_s(u^h(q_s))u_{\beta,s}(u^h(q_s))\psi_i dS \end{aligned}$$

$$u^h(q_s) = \sum_k q_{k,s}\psi_k \qquad u^h(q_{s-1}) = \sum_k q_{k,s-1}\psi_k$$

3.1 Матрица жесткости

$$G_{i,j} = \int_\Omega \lambda(u) grad\psi_i grad\psi_j d\Omega$$

$$G_{0,0} = \sum_{k=0}^{1} \int_\Omega \lambda_k \psi_k grad\psi_0 grad\psi_0 d\Omega =$$

$$= \int_\Omega \lambda_0 \psi_0 grad\psi_0 grad\psi_0 d\Omega + \int_\Omega \lambda_1 \psi_1 grad\psi_0 grad\psi_0 d\Omega =$$

$$= \frac{1}{h}\Big[\lambda_0 \int_\Omega \psi_0 d\Omega + \lambda_1 \int_\Omega \psi_1 d\Omega\Big] = \frac{1}{h}\Big[\lambda_0 \int_0^1 \xi d\xi + \lambda_1 \int_0^1 (1-\xi) d\xi\Big] =$$

$$= \frac{1}{h}\Big[\lambda_0 \frac{\xi^2}{2}\Big|_0^1 + \lambda_1(\xi - \frac{\xi^2}{2})\Big|_0^1\Big] = \frac{\lambda_0 + \lambda_1}{2h} = G_{1,1}$$

$$G_{0,1} = \sum_{k=0}^{1} \int_\Omega \lambda_k \psi_k grad\psi_1 grad\psi_1 d\Omega =$$

$$= \int_\Omega \lambda_0 \psi_0 grad\psi_1 grad\psi_1 d\Omega + \int_\Omega \lambda_1 \psi_1 grad\psi_1 grad\psi_1 d\Omega =$$

$$= -\frac{1}{h}\Big[\lambda_0 \int_\Omega \psi_0 d\Omega + \lambda_1 \int_\Omega \psi_1 d\Omega\Big] = -\frac{1}{h}\Big[\lambda_0 \int_0^1 \xi d\xi + \lambda_1 \int_0^1 (1-\xi) d\xi\Big] =$$

$$= -\frac{1}{h}\Big[\lambda_0 \frac{\xi^2}{2}\Big|_0^1 + \lambda_1(\xi - \frac{\xi^2}{2})\Big|_0^1\Big] = -\frac{\lambda_0 + \lambda_1}{2h} = G_{1,0}$$

$$\mathbf{G} = \frac{\lambda_0 + \lambda_1}{2h} \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

3.2 Матрица масс

$$M_{i,j} = \frac{\sigma}{\Delta t_s} \int_\Omega \psi_i \psi_j d\Omega$$

$$M_{0,0} = \frac{\sigma}{\Delta t_s} \int_\Omega \psi_0 \psi_0 d\Omega = \frac{\sigma h}{\Delta t_s} \int_0^1 \xi^2 d\xi = \frac{\sigma h}{\Delta t_s} \frac{\xi^3}{3}\Big|_0^1 = \frac{\sigma h}{3\Delta t_s} = M_{1,1}$$

$$M_{0,1} = \frac{\sigma}{\Delta t_s} \int_\Omega \psi_0 \psi_1 d\Omega = \frac{\sigma h}{\Delta t_s} \int_0^1 \xi(1-\xi) d\xi = \frac{\sigma h}{\Delta t_s}(\frac{\xi^2}{2} - \frac{\xi^3}{3})\Big|_0^1 = \frac{\sigma h}{6\Delta t_s} = M_{1,0}$$

$$\mathbf{M} = \frac{\sigma h}{6\Delta t_s} \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$$

3.3 Вектор правой части

$$b_i = \int_\Omega f_s \psi_i d\Omega + \frac{1}{\Delta t_s} \int_\Omega \sigma u^h_{q-1} \psi_i d\Omega \qquad \Big[u^h_{q-1} = \sum_{k=0}^{1} q_{k,s-1} \psi_k \Big]$$

$$b_0 = \sum_{k=0}^{1} \int_\Omega f_k \psi_k \psi_0 d\Omega + \frac{\sigma}{\Delta t_s} \sum_{k=0}^{1} \int_\Omega q_{k,q-1} \psi_k \psi_0 d\Omega$$

$$= \Big[f_0 \int_\Omega \psi_0 \psi_0 d\Omega + f_1 \int_\Omega \psi_1 \psi_0 d\Omega \Big] + \frac{\sigma}{\Delta t_s} \Big[q_{0,s-1} \int_\Omega \psi_0 \psi_0 d\Omega + q_{1,s-1} \int_\Omega \psi_1 \psi_0 d\Omega \Big]$$

$$= h \Big[f_0 \int_0^1 \xi^2 d\xi + f_1 \int_0^1 (1-\xi)\xi d\xi \Big] + \frac{\sigma}{\Delta t_s} \Big[q_{0,s-1} \int_0^1 \xi^2 d\xi + q_{1,s-1} \int_0^1 (1-\xi)\xi d\xi \Big]$$

$$= h \Big[f_0 \frac{\xi^3}{3} \Big|_0^1 + f_1 (\frac{\xi^2}{2} - \frac{\xi^3}{3}) \Big|_0^1 \Big] + \frac{\sigma}{\Delta t_s} \Big[q_{0,s-1} \frac{\xi^3}{3} \Big|_0^1 + q_{1,s-1} (\frac{\xi^2}{2} - \frac{\xi^3}{3}) \Big|_0^1 \Big]$$

$$= h \Big[f_0 \frac{1}{3} + f_1 \frac{1}{6} \Big] + \frac{\sigma}{\Delta t_s} \Big[\frac{1}{3} q_{0,s-1} + \frac{1}{6} q_{1,s-1} \Big]$$

$$= \frac{h}{6} \Big[2f_0 + f_1 \Big] + \frac{\sigma}{6\Delta t_s} \Big[2q_{0,s-1} + q_{1,s-1} \Big]$$

$$b_1 = \sum_{k=0}^{1} \int_\Omega f_k \psi_k \psi_1 d\Omega + \frac{\sigma}{\Delta t_s} \sum_{k=0}^{1} \int_\Omega q_{k,q-1} \psi_0 \psi_1 d\Omega =$$

$$= \Big[f_0 \int_\Omega \psi_0 \psi_1 d\Omega + f_1 \int_\Omega \psi_1 \psi_1 d\Omega \Big] + \frac{\sigma}{\Delta t_s} \Big[q_{0,s-1} \int_\Omega \psi_0 \psi_1 d\Omega + q_{1,s-1} \int_\Omega \psi_1 \psi_1 d\Omega \Big] =$$

$$= h \Big[f_0 \int_0^1 \xi(1-\xi) d\xi + f_1 \int_0^1 (1-\xi)^2 d\xi \Big] + \frac{\sigma}{\Delta t_s} \Big[q_{0,s-1} \int_0^1 \xi(1-\xi) d\xi + q_{1,s-1} \int_0^1 (1-\xi)^2 d\xi \Big] =$$

$$= h \Big[f_0 (\frac{\xi^2}{2} - \frac{\xi^3}{3}) \Big|_0^1 + f_1 (1-\xi)^3 \Big|_0^1 \Big] + \frac{\sigma}{\Delta t_s} \Big[q_{0,s-1} (\frac{\xi^2}{2} - \frac{\xi^3}{3}) \Big|_0^1 + q_{1,s-1} (1-\xi)^3 \Big|_0^1 \Big] =$$

$$= \frac{h}{6} \Big[f_0 \frac{1}{6} + f_1 \frac{1}{3} \Big] + \frac{\sigma}{\Delta t_s} \Big[\frac{1}{6} q_{0,s-1} + \frac{1}{3} q_{1,s-1} \Big]$$

$$= \frac{h}{6} \Big[f_0 + 2f_1 \Big] + \frac{\sigma}{6\Delta t_s} \Big[q_{0,s-1} + 2q_{1,s-1} \Big]$$

$$\mathbf{b} = \frac{hx}{6} \begin{pmatrix} 2f_0 + f_1 \\ f_0 + 2f_1 \end{pmatrix} + \frac{\sigma}{6\Delta t_s} \begin{pmatrix} 2q_{0,s-1} + q_{1,s-1} \\ q_{0,s-1} + 2q_{1,s-1} \end{pmatrix}$$

4 Исследования

Параметры, используемые во всех исследованиях:

- $\sigma : 1$
- Точность $(\varepsilon) : 1e - 7$
- Максимальное количество итераций $(maxiter) : 10000$
- Область $(\Omega) : [0, 1]$

- Отрезок времени : $[0, 1]$
- Количество узлов : 11

4.1 Исследование на различных зависимостях λ и u

	1	u	u^2	u^3	u^4	u^5	e^u	$cosu$
$2x+t$	2.62e-02	1.37e-02	5.75e-03	2.05e-03	4.23e-01	3.32e+00	5.82e-04	-nan(ind)
$2x^2+t$	2.62e-02	1.11e-02	7.76e-03	3.41e-01	2.14e+00	1.14e-01	1.51e-02	-nan(ind)
$2x^3+t$	2.10e-02	4.15e-03	3.75e-02	5.97e-01	2.34e+00	3.37e-01	4.16e-02	-nan(ind)
$2x^4+t$	1.58e-02	1.35e-02	9.60e-02	7.81e-01	9.26e-01	8.86e-01	7.68e-02	-nan(ind)
$2x+t^2$	5.25e-02	2.70e-02	1.33e-02	2.41e+00	1.67e+00	3.23e-01	4.43e-03	nan
$2x+t^3$	7.87e-02	3.99e-02	2.06e-02	1.54e-01	3.21e-01	8.23e-01	8.34e-03	-nan(ind)
$2x+t^4$	1.05e-01	5.25e-02	2.76e-02	7.98e-01	1.37e+00	1.30e+00	1.22e-02	-nan(ind)
$2x^2+t^2$	5.25e-02	2.39e-02	7.58e-03	2.10e+00	1.42e-01	7.37e+00	9.25e-03	-nan(ind)
$2x^3+t^3$	7.35e-02	7.68e-02	1.23e-02	2.29e-01	9.14e-01	6.70e-01	2.66e-02	-nan(ind)
e^x+t	2.55e-02	7.74e-03	8.97e-04	6.53e-03	1.29e-02	2.00e+01	6.44e-03	-nan(ind)
e^x+t^2	5.17e-02	1.75e-02	3.36e-03	4.98e-03	1.22e-02	7.04e-01	4.54e-03	-nan(ind)
$x+e^t$	7.13e-02	2.20e-02	6.86e-03	1.95e-03	1.93e-04	6.41e-04	2.51e-03	-nan(ind)
x^2+e^t	7.13e-02	2.27e-02	6.11e-03	6.75e-04	3.84e-03	6.40e-03	7.09e-04	-nan(ind)
e^x+e^t	7.06e-02	1.46e-02	1.04e-03	3.52e-03	6.19e-03	8.92e-03	7.44e-03	-nan(ind)
$2x+cos(t)$	2.21e-02	1.64e-02	1.93e-02	4.99e-02	1.08e-01	7.33e-02	9.18e-03	-nan(ind)
$2cos(x)+t$	2.58e-02	8.50e-03	1.86e-03	9.05e-04	2.18e-03	2.86e-03	2.02e-04	-nan(ind)

4.2 Исследование точности решения при дроблении сетки

Параметры, используемые в данном исследовании:

- Шаг для неравномерных сеток (k) : 1.1
- Функция $\lambda(u)$: u

4.2.1 $2x+t$

i	$nodes$	$iterations$	$norm$
0	11	12	1.37e-02
1	21	12	1.01e-02
2	41	12	7.35e-03
3	81	12	5.26e-03
4	161	12	3.74e-03

равн. пространство, равн. время

i	$nodes$	$iterations$	$norm$
0	11	9	1.37e-02
1	21	9	1.01e-02
2	41	9	7.35e-03
3	81	9	5.26e-03
4	161	9	3.74e-03

равн. пространство, неравн. время

i	$nodes$	$iterations$	$norm$
0	11	8	8.62e-02
1	21	8	1.59e-01
2	41	8	1.39e-01
3	81	8	1.17e-01
4	161	8	9.19e-02

неравн. пространство, равн. время

i	$nodes$	$iterations$	$norm$
0	11	8	8.62e-02
1	21	8	1.59e-01
2	41	8	1.39e-01
3	81	8	1.17e-01
4	161	8	9.19e-02

неравн. пространство, неравн. время

4.2.2 $2x^2 + t$

i	$nodes$	$iterations$	$norm$
0	11	16	1.11e-02
1	21	16	1.06e-02
2	41	16	8.59e-03
3	81	16	6.46e-03
4	161	16	4.71e-03

равн. пространство, равн. время

i	$nodes$	$iterations$	$norm$
0	11	13	1.11e-02
1	21	13	1.06e-02
2	41	13	8.59e-03
3	81	13	6.46e-03
4	161	13	4.71e-03

равн. пространство, неравн. время

i	$nodes$	$iterations$	$norm$
0	11	8	1.61e-01
1	21	8	2.13e-01
2	41	8	1.68e-01
3	81	8	1.30e-01
4	161	8	9.66e-02

неравн. пространство, равн. время

i	$nodes$	$iterations$	$norm$
0	11	8	1.61e-01
1	21	8	2.13e-01
2	41	8	1.68e-01
3	81	8	1.30e-01
4	161	8	9.66e-02

неравн. пространство, неравн. время

4.2.3 $2x^3 + t$

i	$nodes$	$iterations$	$norm$
0	11	19	4.15e-03
1	21	18	8.34e-03
2	41	18	8.40e-03
3	81	18	6.86e-03
4	161	18	5.18e-03

равн. пространство, равн. время

i	$nodes$	$iterations$	$norm$
0	11	16	4.15e-03
1	21	15	8.34e-03
2	41	15	8.40e-03
3	81	15	6.86e-03
4	161	15	5.18e-03

равн. пространство, неравн. время

i	$nodes$	$iterations$	$norm$
0	11	8	2.02e-01
1	21	8	2.33e-01
2	41	8	1.78e-01
3	81	8	1.34e-01
4	161	8	9.82e-02

неравн. пространство, равн. время

i	$nodes$	$iterations$	$norm$
0	11	8	2.02e-01
1	21	8	2.33e-01
2	41	8	1.78e-01
3	81	8	1.34e-01
4	161	8	9.82e-02

неравн. пространство, неравн. время

4.2.4 $2x^4 + t$

i	$nodes$	$iterations$	$norm$
0	11	21	1.35e-02
1	21	20	5.68e-03
2	41	20	7.65e-03
3	81	19	6.87e-03
4	161	19	5.38e-03

равн. пространство, равн. время

i	$nodes$	$iterations$	$norm$
0	11	18	1.35e-02
1	21	17	5.68e-03
2	41	17	7.65e-03
3	81	16	6.87e-03
4	161	16	5.38e-03

равн. пространство, неравн. время

i	$nodes$	$iterations$	$norm$
0	11	8	2.26e-01
1	21	8	2.43e-01
2	41	8	1.84e-01
3	81	8	1.36e-01
4	161	8	9.90e-02

неравн. пространство, равн. время

i	$nodes$	$iterations$	$norm$
0	11	8	2.26e-01
1	21	8	2.43e-01
2	41	8	1.84e-01
3	81	8	1.36e-01
4	161	8	9.90e-02

неравн. пространство, неравн. время

4.2.5 $2x + t^2$

i	$nodes$	$iterations$	$norm$
0	11	12	2.70e-02
1	21	12	2.00e-02
2	41	12	1.45e-02
3	81	12	1.04e-02
4	161	12	7.38e-03

равн. пространство, равн. время

i	$nodes$	$iterations$	$norm$
0	11	9	2.70e-02
1	21	9	2.00e-02
2	41	9	1.45e-02
3	81	9	1.04e-02
4	161	9	7.38e-03

равн. пространство, неравн. время

i	$nodes$	$iterations$	$norm$
0	11	8	9.12e-02
1	21	8	1.59e-01
2	41	9	1.39e-01
3	81	9	1.17e-01
4	161	9	9.19e-02

неравн. пространство, равн. время

i	$nodes$	$iterations$	$norm$
0	11	8	9.12e-02
1	21	9	1.59e-01
2	41	9	1.39e-01
3	81	9	1.17e-01
4	161	9	9.19e-02

неравн. пространство, неравн. время

4.2.6 $2x + t^3$

i	$nodes$	$iterations$	$norm$
0	11	12	3.99e-02
1	21	12	2.96e-02
2	41	12	2.14e-02
3	81	12	1.53e-02
4	161	12	1.09e-02

равн. пространство, равн. время

i	$nodes$	$iterations$	$norm$
0	11	8	3.99e-02
1	21	9	2.96e-02
2	41	9	2.14e-02
3	81	9	1.53e-02
4	161	9	1.09e-02

равн. пространство, неравн. время

i	$nodes$	$iterations$	$norm$
0	11	8	9.61e-02
1	21	9	1.59e-01
2	41	9	1.39e-01
3	81	9	1.17e-01
4	161	9	9.19e-02

неравн. пространство, равн. время

i	$nodes$	$iterations$	$norm$
0	11	8	9.61e-02
1	21	9	1.59e-01
2	41	9	1.39e-01
3	81	9	1.17e-01
4	161	9	9.19e-02

неравн. пространство, неравн. время

4.2.7 $2x + t^4$

i	$nodes$	$iterations$	$norm$
0	11	12	5.25e-02
1	21	12	3.89e-02
2	41	12	2.82e-02
3	81	12	2.02e-02
4	161	12	1.43e-02

равн. пространство, равн. время

i	$nodes$	$iterations$	$norm$
0	11	9	5.25e-02
1	21	9	3.89e-02
2	41	9	2.82e-02
3	81	9	2.02e-02
4	161	9	1.43e-02

равн. пространство, неравн. время

i	$nodes$	$iterations$	$norm$
0	11	8	1.01e-01
1	21	9	1.59e-01
2	41	9	1.39e-01
3	81	9	1.17e-01
4	161	9	9.19e-02

неравн. пространство, равн. время

i	$nodes$	$iterations$	$norm$
0	11	9	1.01e-01
1	21	9	1.59e-01
2	41	9	1.39e-01
3	81	9	1.17e-01
4	161	9	9.19e-02

неравн. пространство, неравн. время

4.2.8 $2x^2 + t^2$

i	$nodes$	$iterations$	$norm$
0	11	16	2.81e-02
1	21	16	2.32e-02
2	41	16	1.76e-02
3	81	16	1.29e-02
4	161	16	9.29e-03

равн. пространство, равн. время

i	$nodes$	$iterations$	$norm$
0	11	13	2.81e-02
1	21	13	2.32e-02
2	41	13	1.76e-02
3	81	13	1.29e-02
4	161	13	9.29e-03

равн. пространство, неравн. время

i	$nodes$	$iterations$	$norm$
0	11	8	1.67e-01
1	21	8	2.13e-01
2	41	9	1.68e-01
3	81	9	1.30e-01
4	161	9	9.66e-02

неравн. пространство, равн. время

i	$nodes$	$iterations$	$norm$
0	11	8	1.67e-01
1	21	9	2.13e-01
2	41	9	1.68e-01
3	81	9	1.30e-01
4	161	9	9.66e-02

неравн. пространство, неравн. время

4.2.9 $2x^3 + t^3$

i	$nodes$	$iterations$	$norm$
0	11	18	2.19e-02
1	21	18	3.16e-02
2	41	18	2.25e-02
3	81	18	1.35e-02
4	161	18	1.52e-02

равн. пространство, равн. время

i	$nodes$	$iterations$	$norm$
0	11	15	4.04e-02
1	21	15	3.60e-02
2	41	15	2.82e-02
3	81	15	2.10e-02
4	161	15	1.52e-02

равн. пространство, неравн. время

i	$nodes$	$iterations$	$norm$
0	11	8	2.12e-01
1	21	9	2.33e-01
2	41	9	1.78e-01
3	81	9	1.34e-01
4	161	9	9.82e-02

неравн. пространство, равн. время

i	$nodes$	$iterations$	$norm$
0	11	9	2.12e-01
1	21	9	2.33e-01
2	41	9	1.78e-01
3	81	9	1.34e-01
4	161	9	9.82e-02

неравн. пространство, неравн. время

4.2.10 $e^x + t$

i	$nodes$	$iterations$	$norm$
0	11	11	7.74e-03
1	21	11	6.54e-03
2	41	11	5.03e-03
3	81	11	3.71e-03
4	161	11	2.67e-03

равн. пространство, равн. время

i	$nodes$	$iterations$	$norm$
0	11	9	7.74e-03
1	21	9	6.54e-03
2	41	9	5.03e-03
3	81	9	3.71e-03
4	161	9	2.67e-03

равн. пространство, неравн. время

i	$nodes$	$iterations$	$norm$
0	11	7	9.14e-02
1	21	6	1.46e-01
2	41	6	1.22e-01
3	81	6	1.00e-01
4	161	6	7.69e-02

неравн. пространство, равн. время

i	$nodes$	$iterations$	$norm$
0	11	7	9.14e-02
1	21	6	1.46e-01
2	41	6	1.22e-01
3	81	6	1.00e-01
4	161	6	7.69e-02

неравн. пространство, неравн. время

4.2.11 $e^x + t^2$

i	$nodes$	$iterations$	$norm$
0	11	11	1.75e-02
1	21	11	1.38e-02
2	41	11	1.03e-02
3	81	11	7.46e-03
4	161	11	5.35e-03

равн. пространство, равн. время

i	$nodes$	$iterations$	$norm$
0	11	10	1.75e-02
1	21	10	1.38e-02
2	41	10	1.03e-02
3	81	10	7.46e-03
4	161	10	5.35e-03

равн. пространство, неравн. время

i	$nodes$	$iterations$	$norm$
0	11	7	9.50e-02
1	21	7	1.46e-01
2	41	7	1.22e-01
3	81	7	1.00e-01
4	161	7	7.69e-02

неравн. пространство, равн. время

i	$nodes$	$iterations$	$norm$
0	11	7	9.50e-02
1	21	7	1.46e-01
2	41	7	1.22e-01
3	81	7	1.00e-01
4	161	7	7.69e-02

неравн. пространство, неравн. время

4.2.12 $x + e^t$

i	$nodes$	$iterations$	$norm$
0	11	10	2.20e-02
1	21	10	1.63e-02
2	41	10	1.18e-02
3	81	10	8.44e-03
4	161	10	6.00e-03

равн. пространство, равн. время

i	$nodes$	$iterations$	$norm$
0	11	9	2.20e-02
1	21	9	1.63e-02
2	41	9	1.18e-02
3	81	9	8.44e-03
4	161	9	6.00e-03

равн. пространство, неравн. время

i	$nodes$	$iterations$	$norm$
0	11	7	3.82e-02
1	21	5	6.72e-02
2	41	5	6.05e-02
3	81	5	5.28e-02
4	161	5	4.20e-02

неравн. пространство, равн. время

i	$nodes$	$iterations$	$norm$
0	11	6	3.82e-02
1	21	5	6.72e-02
2	41	5	6.05e-02
3	81	5	5.28e-02
4	161	5	4.20e-02

неравн. пространство, неравн. время

4.2.13 $x^2 + e^t$

i	$nodes$	$iterations$	$norm$
0	11	11	2.27e-02
1	21	11	1.71e-02
2	41	11	1.25e-02
3	81	11	8.99e-03
4	161	11	6.41e-03

равн. пространство, равн. время

i	$nodes$	$iterations$	$norm$
0	11	10	2.27e-02
1	21	10	1.71e-02
2	41	10	1.25e-02
3	81	10	8.99e-03
4	161	10	6.41e-03

равн. пространство, неравн. время

i	$nodes$	$iterations$	$norm$
0	11	7	7.10e-02
1	21	5	9.48e-02
2	41	5	7.59e-02
3	81	5	5.94e-02
4	161	5	4.45e-02

неравн. пространство, равн. время

i	$nodes$	$iterations$	$norm$
0	11	7	7.10e-02
1	21	5	9.48e-02
2	41	5	7.59e-02
3	81	5	5.94e-02
4	161	5	4.45e-02

неравн. пространство, неравн. время

4.2.14 $e^x + e^t$

i	$nodes$	$iterations$	$norm$
0	11	9	1.46e-02
1	21	9	1.14e-02
2	41	9	8.49e-03
3	81	9	6.15e-03
4	161	9	4.41e-03

равн. пространство, равн. время

i	$nodes$	$iterations$	$norm$
0	11	8	1.46e-02
1	21	8	1.14e-02
2	41	8	8.49e-03
3	81	8	6.15e-03
4	161	8	4.41e-03

равн. пространство, неравн. время

i	$nodes$	$iterations$	$norm$
0	11	6	8.67e-02
1	21	5	1.39e-01
2	41	5	1.17e-01
3	81	5	9.67e-02
4	161	5	7.47e-02

неравн. пространство, равн. время

i	$nodes$	$iterations$	$norm$
0	11	6	8.67e-02
1	21	6	1.39e-01
2	41	6	1.17e-01
3	81	6	9.67e-02
4	161	6	7.47e-02

неравн. пространство, неравн. время

4.2.15 $2x + cos(t)$

i	$nodes$	$iterations$	$norm$
0	11	15	1.64e-02
1	21	14	1.22e-02
2	41	14	8.81e-03
3	81	14	6.31e-03
4	161	14	4.49e-03

равн. пространство, равн. время

i	$nodes$	$iterations$	$norm$
0	11	10	1.64e-02
1	21	10	1.22e-02
2	41	10	8.81e-03
3	81	11	6.31e-03
4	161	11	4.49e-03

равн. пространство, неравн. время

i	$nodes$	$iterations$	$norm$
0	11	9	8.56e-02
1	21	10	1.69e-01
2	41	10	1.46e-01
3	81	10	1.22e-01
4	161	10	9.51e-02

неравн. пространство, равн. время

i	$nodes$	$iterations$	$norm$
0	11	9	8.56e-02
1	21	10	1.69e-01
2	41	10	1.46e-01
3	81	11	1.22e-01
4	161	11	9.51e-02

неравн. пространство, неравн. время

4.2.16 $2cos(x) + t$

i	$nodes$	$iterations$	$norm$
0	11	10	8.50e-03
1	21	10	6.72e-03
2	41	10	5.02e-03
3	81	10	3.65e-03
4	161	10	2.61e-03

равн. пространство, равн. время

i	$nodes$	$iterations$	$norm$
0	11	8	8.50e-03
1	21	9	6.72e-03
2	41	9	5.02e-03
3	81	9	3.65e-03
4	161	9	2.61e-03

равн. пространство, неравн. время

i	$nodes$	$iterations$	$norm$
0	11	7	3.80e-02
1	21	5	7.44e-02
2	41	5	6.13e-02
3	81	5	4.90e-02
4	161	5	3.71e-02

неравн. пространство, равн. время

i	$nodes$	$iterations$	$norm$
0	11	6	3.80e-02
1	21	5	7.44e-02
2	41	5	6.13e-02
3	81	5	4.90e-02
4	161	5	3.71e-02

неравн. пространство, неравн. время

5 Выводы

При исследовании на различных параметрах λ метод хорошо сходится, когда степень полинома не превышает двух. На полиномах высших степеней нормальная сходимость достигается лишь на экспоненциальных целевых функциях. Так же хорошо метод сходится и при экспоненциальной функции λ. На гармонических функциях λ метод не сходятся.
Порядок сходимости определяется путем дробления сетки вдвое и является степенью увеличения точности с каждым дроблением. В ходе исследования было выявлено, что порядок сходимости равен 0.75.

6 Код программы

6.1 common.h

```
#pragma once
#define _CRT_SECURE_NO_WARNINGS

#include <fstream>
#include <vector>
#include <functional>
#include <cmath>
#include <string>
#include <iostream>
#include <iomanip>

using namespace std;

typedef function <double(double)> func1D;
typedef function <double(double, double)> func2D;
typedef vector <double> vect;
typedef vector <vector <double>> mat;

int gridDivCoef, timeDivCoef;
int i, maxiter, gridWidth, nodesCount, finiteElementsCount, tNum,
 ↪  answer;
double sum, tmp, x1, x2, hx, nx, kx, ht, nt, kt, dx, dt, t1, t2,
 ↪  lambda0, lambda1, E, delta, sigma, t;
bool gridUniformFlag, timeUniformFlag;
string fp;
vect result, di, al, au, b, bLocal, q, qPrev, multVectByA();
func1D lambda, duBydt, duBydx;
func2D u, f, lambdaGrad;
mat A, localG, localM, localA;
const double h = 0.00001;

void factorization();
void forwardGauss();
void backwardGauss();
void inputGrid();
void makeGridSpace();
void inputTime();
void makeGridTime();
void makeLocalA(int elemNum);
void makeLocalG(int elemNum);
void makeLocalM(int elemNum);
void makeLocalb(int elemNum);
void makeGlobalA(double _dt);
void makeGlobalb();
void config(const func2D& _u, const func2D& _f, const func1D& _lambda,
 ↪  double _sigma);
```

```cpp
bool exitCheck(int i);
pair<int, double> solve();

double operator * (const vect& a, const vect& b) {
    sum = 0;
    for (i = 0; i < a.size(); i++)
        sum += a[i] * b[i];
    return sum;
}
vect operator - (const vect& a, const vect& b) {
    result = a;
    for (i = 0; i < b.size(); i++)
        result[i] -= b[i];
    return result;
}

struct NODE {
    int borderNum;
    int type = -99;
    double x;
    bool firstNodeFlag = false;

    void setNodes(double _x, int _i, int _type, double _coef) {
        x = _x;
        i = _i;
        type = _type;
        if (i % int(pow(2, _coef)) == 0)
            firstNodeFlag = true;
    }
};

vector <NODE> nodes;
vect times;

double normInNodes(const vect& x) {
    tmp = 0;
    for (size_t i = 0; i < x.size(); i++)
        tmp += pow((x[i] - u(nodes[i].x, t)), 2);
    return sqrt(tmp) / nodes.size();
}

func1D derivative(const func1D& f) {
    return [f](double x) -> double {
        return (-f(x + 2 * h) + 8 * f(x + h) - 8 * f(x - h) + f(x - 2 *
        ↪  h)) / (12 * h);
    };
}

func2D rightPart(const func1D& lambda, const func2D& u, double sigma) {
```

```
    return [=](double x, double t) -> double {
        using namespace placeholders;
        duBydt = derivative(bind(u, x, _1));
        duBydx = derivative(bind(u, _1, t));
        lambdaGrad = [=](double x, double t) -> double {
            return lambda(u(x, t)) * duBydx(x);
        };
        auto div = derivative(bind(lambdaGrad, _1, t));
        return -div(x) + sigma * duBydt(t);
    };
}

double normE(const vect& x) {
    return sqrt(x * x);
}
```

6.2 main.cpp

```
#include "common.h"

void main()
{
    double sigma = 1;

    u = { [](double x, double t) -> double { return 2*x + t; } };

    vector <func1D> lambda(8);
    lambda[0] = { [](double u) -> double {return 1; } };
    lambda[1] = { [](double u) -> double {return u; } };
    lambda[2] = { [](double u) -> double {return u * u; } };
    lambda[3] = { [](double u) -> double {return u * u * u; } };
    lambda[4] = { [](double u) -> double {return u * u * u * u; } };
    lambda[5] = { [](double u) -> double {return u * u * u * u * u; }
     ↪ };
    lambda[6] = { [](double u) -> double {return exp(u); } };
    lambda[7] = { [](double u) -> double {return cos(u); } };

    f = rightPart(lambda[1], u, sigma);

    cout << "Is the grid uniform? (1/0)\n";
    cin >> answer;
    if (answer == 1)
        gridUniformFlag = true;
    else
        gridUniformFlag = false;
    cout << "Is the time uniform? (1/0)\n";
    cin >> answer;
```

```
    if (answer == 1)
        timeUniformFlag = true;
    else
        timeUniformFlag = false;
    cout << "\n";
    cout << "i nodes iter norm\n";
    cout << scientific << setprecision(2);

    for (gridDivCoef = 0; gridDivCoef < 5; gridDivCoef++)
    {
        config(u, f, lambda[1], sigma);
        inputGrid();
        makeGridSpace();
        inputTime();
        makeGridTime();
        auto result = solve();
        cout << gridDivCoef << " " << nodesCount << " " << result.first << " " << result.second << endl;
    }
    cout << "\n";

    if (timeUniformFlag && gridUniformFlag) {

        gridDivCoef = 0;
        for (size_t i = 0; i < lambda.size(); i++)
        {
            f = rightPart(lambda[i], u, sigma);
            config(u, f, lambda[i], sigma);
            inputGrid();
            makeGridSpace();
            inputTime();
            makeGridTime();
            cout << "lambda(u)_" << i + 1 << ": " << solve().second << "\n";
        }
    }
}

void makeLocalG(int elemNum)
{
    lambda0 = lambda(q[elemNum]);
    lambda1 = lambda(q[elemNum + 1]);
    localG[0][0] = localG[1][1] = (lambda0 + lambda1) / (2 * hx);
    localG[0][1] = localG[1][0] = -(lambda0 + lambda1) / (2 * hx);
}

void makeLocalM(int elemNum)
{
    localM[0][0] = localM[1][1] = 2 * (sigma * hx) / (6 * dt);
```

```
    localM[0][1] = localM[1][0] = (sigma * hx) / (6 * dt);
}

void makeLocalA(int elemNum)
{
    localA = localG = localM = { {0,0}, {0,0} };
    makeLocalG(elemNum);
    makeLocalM(elemNum);
    for (size_t i = 0; i < 2; i++)
        for (size_t j = 0; j < 2; j++)
            localA[i][j] = localG[i][j] + localM[i][j];
}

void makeLocalb(int elemNum)
{
    bLocal = { 0, 0 };
    bLocal[0] = hx * (2 * f(nodes[elemNum].x, t) + f(nodes[elemNum + 1].x, t)) / 6
        + sigma * hx * (2 * qPrev[elemNum] + qPrev[elemNum + 1]) / (6 * dt);
    bLocal[1] = hx * (f(nodes[elemNum + 1].x, t) + 2 * f(nodes[elemNum + 1].x, t)) / 6
        + sigma * hx * (qPrev[elemNum] + 2 * qPrev[elemNum + 1]) / (6 * dt);
}

void makeGlobalA(double _dt)
{
    dt = _dt;
    A.clear();
    di.clear();
    au.clear();
    al.clear();
    A.resize(nodesCount);
    for (size_t i = 0; i < nodesCount; i++)
        A[i].resize(nodesCount, 0);
    di.resize(nodesCount, 0);
    al.resize(nodesCount - 1, 0);
    au.resize(nodesCount - 1, 0);

    for (size_t elemNum = 0; elemNum < finiteElementsCount; elemNum++)
    {
        makeLocalA(elemNum);
        di[elemNum] += localA[0][0];
        di[elemNum + 1] += localA[1][1];
        au[elemNum] += localA[0][1];
        al[elemNum] += localA[1][0];
        A[elemNum][elemNum] += localA[0][0];
        A[elemNum][elemNum + 1] += localA[0][1];
```

```
        A[elemNum + 1][elemNum] += localA[1][0];
        A[elemNum + 1][elemNum + 1] += localA[1][1];
    }

    A[0][0] = 1; A[0][1] = 0;
    A[nodesCount - 1][nodesCount - 1] = 1; A[nodesCount - 1][nodesCount
     ↪  - 2] = 0;
    di[0] = 1;
    au[0] = 0;
    di[nodesCount - 1] = 1;
    al[al.size() - 1] = 0;
}

void makeGlobalb()
{
    b.clear();
    b.resize(nodesCount, 0);

    for (size_t elemNum = 0; elemNum < finiteElementsCount; elemNum++)
    {
        makeLocalb(elemNum);
        b[elemNum] += bLocal[0];
        b[elemNum + 1] += bLocal[1];
    }

    b[0] = u(nodes[0].x, t);
    b[nodesCount - 1] = u(nodes[nodesCount - 1].x, t);
}

pair<int, double> solve()
{
    q.resize(nodesCount, 0);
    qPrev.resize(nodesCount, 0);
    vect qExact(nodesCount);
    for (size_t i = 0; i < nodesCount; i++)
        qExact[i] = u(nodes[i].x, times[0]);
    qPrev = qExact;

    int count = 0;
    for (size_t i = 1; i < times.size(); i++)
    {
        dt = times[i] - times[i - 1];
        t = times[i];
        do {
            qPrev = q;
            makeGlobalA(dt);
            makeGlobalb();
            factorization();
            forwardGauss();
```

```
            backwardGauss();
            count++;
        } while (exitCheck(count));
    }
    return make_pair(count, normInNodes(q));
}

void factorization()
{
    int lIndex = di.size();
    for (size_t i = 1; i < lIndex; i++)
    {
        au[i - 1] = au[i - 1] / di[i - 1];
        di[i] = di[i] - al[i - 1] * au[i - 1];
    }
}

void forwardGauss()
{
    q[0] = b[0] / di[0];
    for (size_t i = 1; i < di.size(); i++)
        q[i] = (b[i] - al[i - 1] * q[i - 1]) / di[i];
    b = q;
}

void backwardGauss()
{
    int lIndex = di.size() - 1;
    q[lIndex] = b[lIndex];
    for (int i = lIndex - 1; i >= 0; i--)
        q[i] = (b[i] - q[i + 1] * au[i]);
}

void inputGrid()
{
    if (gridUniformFlag)
        fp = "uniGrid.txt";
    else
        fp = "irrGrid.txt";
    ifstream fin(fp);
    fin >> x1 >> x2;
    fin >> gridWidth;
    if (!gridUniformFlag) {
        fin >> kx;
        nx = gridWidth - 1;
    }
    fin.close();
}
```

```
void inputTime()
{
    if (timeUniformFlag)
        fp = "uniTime.txt";
    else
        fp = "irrTime.txt";
    ifstream fin(fp);
    fin >> t1 >> t2;
    fin >> tNum;
    if (!timeUniformFlag) {
        fin >> kt;
        nt = tNum - 1;
    }
    fin.close();
}

void makeGridSpace()
{
    double x;
    size_t i, elem;

    if (gridUniformFlag) {
        hx = ((x2 - x1) / double(gridWidth - 1)) / pow(2, gridDivCoef);
        if (gridDivCoef != 0)
            gridWidth = (gridWidth - 1) * pow(2, gridDivCoef) + 1;
    }
    else {
        if (gridDivCoef != 0) {
            gridWidth = (gridWidth - 1) * pow(2, gridDivCoef) + 1;
            nx *= pow(2, gridDivCoef);
            kx *= pow(kx, 1.0 / gridDivCoef);
        }
        hx = (x2 - x1) * (1 - kx) / (1 - pow(kx, nx));
    }

    nodesCount = gridWidth;
    finiteElementsCount = nodesCount - 1;
    nodes.resize(gridWidth);

    if (gridUniformFlag) {
        i = 1;
        nodes[0].setNodes(x1, 0, 1, gridDivCoef);
        for (elem = 1; elem < nodesCount - 1; elem++, i++)
        {
            x = x1 + hx * i;
            nodes[elem].setNodes(x, i, 0, gridDivCoef);
            nodes[elem].borderNum = 0;
        }
        nodes[nodesCount - 1].setNodes(x2, gridWidth, 1, gridDivCoef);
```

```
    }
    else {
        i = 1;
        dx = hx * kx;
        x = x1 + hx;
        nodes[0].setNodes(x1, 0, 1, gridDivCoef);
        for (elem = 1; elem < gridWidth; elem++, i++, dx *= kx)
        {
            nodes[elem].setNodes(x, i, 0, gridDivCoef);
            nodes[elem].borderNum = 0;
            x += dx;
        }
        nodes[nodesCount - 1].setNodes(x2, gridWidth, 1, gridDivCoef);
    }
}

void makeGridTime()
{
    size_t i, elem;
    double t;
    times.resize(tNum);

    if (timeUniformFlag) {
        i = 1;
        ht = ((t2 - t1) / double(tNum - 1)) / pow(2, timeDivCoef);

        if (timeDivCoef != 0)
            gridWidth = (gridWidth - 1) * pow(2, timeDivCoef) + 1;

        times[0] = t1;
        for (elem = 1; elem < tNum; elem++, i++)
            times[elem] = t1 + ht * i;
        times[tNum - 1] = t2;
    }
    else {
        if (timeDivCoef != 0) {
            gridWidth = (gridWidth - 1) * pow(2, timeDivCoef) + 1;
            nt *= pow(2, timeDivCoef);
            kt *= pow(kt, 1.0 / timeDivCoef);
        }

        i = 1;
        ht = (t2 - t1) * (1 - kt) / (1 - pow(kt, nt));
        dt = ht * kt;
        t = t1 + ht;

        times[0] = t1;
        for (elem = 1; elem < tNum; elem++, i++, dt *= kt)
        {
```

```
            times[elem] = t;
            t += dt;
        }
        times[tNum - 1] = t2;
    }
}

bool exitCheck(int i) // by maxiter, step and relative discrepancy
{
    if (i > maxiter) return false;
    if (normE(q - qPrev) / normE(q) < delta) return false;
    if (normE(multVectByA() - b) / normE(b) < E) return false;
    return true;
}

vect multVectByA() // A*q
{
    vect tmp;
    tmp.resize(di.size());
    if (di.size() >= 2)
        tmp[0] = di[0] * q[0] + au[0] * q[1];
    if (di.size() >= 3)
        for (size_t i = 1; i < di.size() - 1; i++)
            tmp[i] = al[i - 1] * q[i - 1] + di[i] * q[i] + au[i] * q[i
             ↪  + 1];
    int lIndex = di.size() - 1;
    tmp[lIndex] = al[lIndex - 1] * q[lIndex - 1] + di[lIndex] *
     ↪  q[lIndex];
    return tmp;
}

void config(const func2D& _u, const func2D& _f, const func1D& _lambda,
 ↪  double _sigma)
{
    u = _u;
    f = _f;
    lambda = _lambda;
    sigma = _sigma;

    ifstream fin("parametersSys.txt");
    fin >> E >> delta >> maxiter;
    fin.close();
}
```