1 Теория

Решаемое уравнение в общем виде:

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

Решаемое уравнение в декартовой двумерной системе координат:

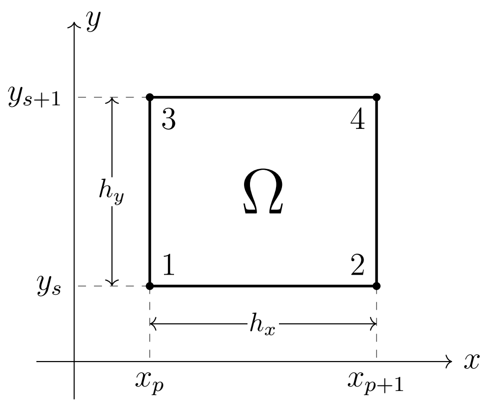
$$-\frac{\partial}{\partial x}\left(\lambda \frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(\lambda \frac{\partial u}{\partial y}\right) + \gamma u + \sigma \frac{\partial u}{\partial t} + \chi \frac{\partial^2 u}{\partial t^2} = f$$

Первые краевые условия:

$$u|_S = u_s$$

Формулы для билинейных базисных функций прямоугольных элементов:

$$\begin{aligned} X_1(x) &= \frac{x_{p+1} - x}{h_x} & h_x &= x_{p+1} - x_p \\ X_2(x) &= \frac{x - x_p}{h_x} & h_y &= y_{s+1} - x_s \\ Y_1(y) &= \frac{y_{s+1} - y}{h_y} & x &\in [x_p, x_{p+1}],\ y \in [y_s, y_{s+1}] \\ Y_2(y) &= \frac{y - y_s}{h_y} & \Omega_{ps} &= [x_p, x_{p+1}] \times [y_s, y_{s+1}] \end{aligned}$$



$$\psi_4(x, y) = X_2(x) Y_2(y)$$

$$\psi_3(x, y) = X_2(x) Y_1(y)$$

$$\psi_2(x, y) = X_1(x) Y_2(y)$$

$$\psi_1(x, y) = X_1(x) Y_1(y)$$

И значение конечно-элементной аппроксимации на этом конечном элементе равно:

$$u^*_{ps}(x, y) = \sum_{i=1}^{4} q_i \psi_i(x, y)$$

Аналитические выражения для вычисления элементов локальных матриц:

$$G_{ij} = \int\limits_{x_p}^{x_{p+1}} \int\limits_{y_s}^{y_{s+1}} \lambda \left(\frac{\partial \psi_i}{\partial x} \frac{\partial \psi_j}{\partial x} + \frac{\partial \psi_i}{\partial y} \frac{\partial \psi_j}{\partial y} \right) \mathrm{d}x\, \mathrm{d}y$$

$$M^{\gamma}_{ij} = \int\limits_{x_p}^{x_{p+1}} \int\limits_{y_s}^{y_{s+1}} \gamma \psi_i \psi_j \, \mathrm{d}x\, \mathrm{d}y, \qquad b_i = \int\limits_{x_p}^{x_{p+1}} \int\limits_{y_s}^{y_{s+1}} f \psi_i \, \mathrm{d}x\, \mathrm{d}y$$

Вычисленные матрицы для билинейных прямоугольных элементов:

$$\mathbf{G} = \frac{\bar{\lambda}}{6}\frac{h_y}{h_x}\begin{pmatrix} 2 & -2 & 1 & -1 \\ -2 & 2 & -1 & 1 \\ 1 & -1 & 2 & -2 \\ -1 & 1 & -2 & 2 \end{pmatrix} + \frac{\bar{\lambda}}{6}\frac{h_x}{h_y}\begin{pmatrix} 2 & 1 & -2 & -1 \\ 1 & 2 & -1 & -2 \\ -2 & -1 & 2 & 1 \\ -1 & -2 & 1 & 2 \end{pmatrix}$$

$$\mathbf{C} = \frac{h_x h_y}{36}\begin{pmatrix} 4 & 2 & 2 & 1 \\ 2 & 4 & 1 & 2 \\ 2 & 1 & 4 & 2 \\ 1 & 2 & 2 & 4 \end{pmatrix} \qquad \begin{gathered} \mathbf{M}^{\gamma} = \bar{\gamma}\mathbf{C} \\ \mathbf{f} = (f_1, f_2, f_3, f_4)^t \\ \mathbf{b} = \mathbf{C}\cdot\mathbf{f} \end{gathered}$$

Схема Кранка-Николсона:

$$\frac{\partial u}{\partial t} = \frac{u^j - u^{j-2}}{2\Delta t}, \qquad \frac{\partial^2 u}{\partial t^2} = \frac{u^j - 2u^{j-1} + u^{j-2}}{\Delta t^2}$$

$$u = \frac{u^j + u^{j-2}}{2}, \qquad f = \frac{f^j + f^{j-2}}{2}$$

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

Подставляя это в уравнение Галёркина, получаем СЛАУ из глобальных матриц:

$$\left(\frac{\mathbf{G}}{2} + \frac{\mathbf{M}^{\gamma}}{2} + \frac{\mathbf{M}^{\sigma}}{2\Delta t} + \frac{\mathbf{M}^{\chi}}{\Delta t^2}\right)\mathbf{q}^j = \frac{(\mathbf{b}^j + \mathbf{b}^{j-2})}{2} - \frac{\mathbf{G}\mathbf{q}^{j-2}}{2} - \frac{\mathbf{M}^{\gamma}\mathbf{q}^{j-2}}{2} + \frac{\mathbf{M}^{\sigma}\mathbf{q}^{j-2}}{2\Delta t} - \frac{\mathbf{M}^{\chi}\left(-2\mathbf{q}^{j-1} + \mathbf{q}^{j-2}\right)}{\Delta t^2}$$

В нашем случае, так как γ, σ, χ являются константами, можно записать:

$$\left(\frac{\mathbf{G}}{2} + \mathbf{C}\left(\frac{\gamma}{2} + \frac{\sigma}{2\Delta t} + \frac{\chi}{\Delta t^2}\right)\right)\mathbf{q}^j = \frac{(\mathbf{b}^j + \mathbf{b}^{j-2})}{2} - \frac{\mathbf{G}\mathbf{q}^{j-2}}{2} + \mathbf{C}\left(\mathbf{q}^{j-1}\frac{2\chi}{\Delta t^2} + \mathbf{q}^{j-2}\left(-\frac{\gamma}{2} + \frac{\sigma}{2\Delta t} - \frac{\chi}{\Delta t^2}\right)\right)$$

Для неравномерной же сетки по времени имеем только отличие в:

$$t_2 = t^{j-2}, \quad t_1 = t^{j-1}, \quad t_0 = t^j$$

$$\frac{\partial u}{\partial t} = \frac{u^j - u^{j-2}}{t_2 - t_1} = \frac{u^j - u^{j-2}}{d_1}$$

$$\frac{\partial^2 u}{\partial t^2} = 2\frac{u^j - u^{j-1}\dfrac{t_0 - t_2}{t_1 - t_2} + u^{j-2}\dfrac{t_0 - t_1}{t_1 - t_2}}{t_0\left(t_0 - t_1 - t_2\right) + t_1 t_2} = \frac{u^j - u^{j-1}m_1 + u^{j-2}m_2}{d_2}$$

Эти выражения были упрощены при помощи замен:

$$d_1 = t_0 - t_2, \quad d_2 = \frac{t_0\left(t_0 - t_1 - t_2\right) + t_1 t_2}{2}, \quad m_1 = \frac{t_0 - t_2}{t_1 - t_2}, \quad m_2 = \frac{t_0 - t_1}{t_1 - t_2}$$

И итоговый результат будет:

$$\left(\frac{\mathbf{G}}{2} + \mathbf{C}\left(\frac{\gamma}{2} + \frac{\sigma}{d_1} + \frac{\chi}{d_2}\right)\right)\mathbf{q}^j = \frac{(\mathbf{b}^j + \mathbf{b}^{j-2})}{2} - \frac{\mathbf{G}\mathbf{q}^{j-2}}{2} + \mathbf{C}\left(\mathbf{q}^{j-1}\frac{m_1\chi}{d_2} + \mathbf{q}^{j-2}\left(-\frac{\gamma}{2} + \frac{\sigma}{d_1} - \frac{m_2\chi}{d_2}\right)\right)$$

2 Структуры данных

Для задания сетки используется класс:

```
class grid_generator_t
{
public:
    grid_generator_t(double a, double b, int n, double t = 0);
    double operator()(int i) const;
    int size(void) const;
    double back(void) const;
private:
    double a, len, t, n1;
};
```

Для задания узла конечного элемента структура:

```
struct basic_elem_t
{
    int i; /// Номер узла

    double x, y; /// Координаты узла

    basic_elem_t *up, *down, *left, *right; /// Указатели на соседей узла

    /** Проверяет, является ли элемент граничным. Он таким является, если у него нет хотя бы одного соседа. */
    bool is_boundary(void) const;
};
```

Для задания конечного элемента используется структура:

```
struct elem_t
{
    int i; /// Номер конечного элемента
    basic_elem_t* e[4]; /// Указатели на все 4 элемента конечного узла, нумерация такая:
    /**
         Y
         ^  3 +-----+ 4
         |    |     |
         |  1 +-----+ 2
        -+---------------> X
         |
     */

    double get_hx(void) const; /// Ширина конечного элемента
    double get_hy(void) const; /// Высота конечного элемента

    /** Рассчитать значение внутри конечного элемента. q - вектор рассчитыванных весов. */
    double value(double x, double y, const vector_t& q) const;
};
```

Прямоугольная сетка задается и вычисляется с помощью класса:

```
class grid_t
{
public:
    vector<elem_t> es; /// Массив конечных элементов сетки
    vector<basic_elem_t> bes; /// Массив узлов сетки
    int n; /// Число узлов

    /** Рассчитать неравномерную сетку. */
    void calc(const grid_generator_t& gx, const grid_generator_t& gy);
};
```

Локальные матрицы формируются, получая на вход конечный элемент `elem_t`.

Для генерации разреженной матрицы используется класс с возможностью произвольного доступа к элементам:

```
class matrix_sparse_ra_t
{
public:
    matrix_sparse_ra_t(int n);

    /** Установить значение в позиции (i, j) */
    double& operator()(int i, int j);

    /** Получить значение в позиции (i, j). Если туда ещё не устанавливалось значение, вызывается
    ↪  исключение. */
    const double& operator()(int i, int j) const;

    /** Преобразует текущую матрицу к разреженной матрице. */
    matrix_sparse_t to_sparse(void) const;
private:
    int n;
    vector<double> dm;
    vector<map<int, double>> lm, um;
};
```

3 Исследования

Во всех исследованиях заданы следующие параметры $\lambda = \gamma = \sigma = \chi = 1$.

СЛАУ решается при помощи Локально-Оптимальной Схемы (ЛОС) с неполным LU предобуславливанием.

3.1 Таблицы

Далее в таблицах будут указаны две функции: $\mathrm{space}(x, y)$ и $\mathrm{time}(t)$, итоговая функция u будет формироваться из них: $u(x, y, t) = \mathrm{space}(x, y) + \mathrm{time}(t)$.

В таблицах для каждой функции указано три значения:

• Интеграл разности между истинной функцией и конечно-элементоной аппроксимацией.

• Норма разности векторов q для найденного решения и q, полученного из истинного значения функции.

• Время решения в миллисекундах.

3.1.1 10 на 10 на 10

Сетка по пространству: $(x, y) \in [0, 1] \times [0, 1]$, строк и столбцов 10. Сетка по времени: $t \in [0, 1]$, количество элементов сетки 10. Все сетки равномерные.

space(x,y) \ time(t)	0	t	t^2	t^3	t^4	e^t
1	$0.36 \cdot 10^{-15}$ $0.57 \cdot 10^{-14}$ 47	$0.36 \cdot 10^{-11}$ $0.45 \cdot 10^{-10}$ 46	$0.34 \cdot 10^{-11}$ $0.42 \cdot 10^{-10}$ 48	$0.76 \cdot 10^{-3}$ $1 \cdot 10^{-2}$ 38	$0.76 \cdot 10^{-3}$ $1 \cdot 10^{-2}$ 37	$0.61 \cdot 10^{-3}$ $0.8 \cdot 10^{-2}$ 51
$x + y$	$0.22 \cdot 10^{-11}$ $0.35 \cdot 10^{-10}$ 66	$0.35 \cdot 10^{-11}$ $0.46 \cdot 10^{-10}$ 74	$0.24 \cdot 10^{-11}$ $0.38 \cdot 10^{-10}$ 61	$0.76 \cdot 10^{-3}$ $1 \cdot 10^{-2}$ 52	$0.76 \cdot 10^{-3}$ $1 \cdot 10^{-2}$ 77	$0.61 \cdot 10^{-3}$ $0.8 \cdot 10^{-2}$ 84
$x^2 + y$	$0.28 \cdot 10^{-2}$ $0.24 \cdot 10^{-10}$ 81	$0.28 \cdot 10^{-2}$ $0.15 \cdot 10^{-10}$ 89	$0.28 \cdot 10^{-2}$ $0.24 \cdot 10^{-10}$ 120	$0.35 \cdot 10^{-2}$ $1 \cdot 10^{-2}$ 53	$0.35 \cdot 10^{-2}$ $1 \cdot 10^{-2}$ 63	$0.34 \cdot 10^{-2}$ $0.8 \cdot 10^{-2}$ 61
$x^2y + y^3$	$0.28 \cdot 10^{-2}$ $0.17 \cdot 10^{-10}$ 40	$0.28 \cdot 10^{-2}$ $0.23 \cdot 10^{-10}$ 113	$0.28 \cdot 10^{-2}$ $0.38 \cdot 10^{-10}$ 62	$0.35 \cdot 10^{-2}$ $1 \cdot 10^{-2}$ 60	$0.35 \cdot 10^{-2}$ $1 \cdot 10^{-2}$ 64	$0.34 \cdot 10^{-2}$ $0.8 \cdot 10^{-2}$ 82
xy^2	$0.69 \cdot 10^{-3}$ $0.93 \cdot 10^{-11}$ 71	$0.69 \cdot 10^{-3}$ $0.17 \cdot 10^{-10}$ 57	$0.69 \cdot 10^{-3}$ $0.13 \cdot 10^{-10}$ 60	$0.14 \cdot 10^{-2}$ $1 \cdot 10^{-2}$ 90	$0.14 \cdot 10^{-2}$ $1 \cdot 10^{-2}$ 56	$0.13 \cdot 10^{-2}$ $0.8 \cdot 10^{-2}$ 59
$x^4 + y^4$	$0.43 \cdot 10^{-2}$ 0.02 52	$0.43 \cdot 10^{-2}$ 0.02 60	$0.43 \cdot 10^{-2}$ 0.02 55	$0.48 \cdot 10^{-2}$ $1 \cdot 10^{-2}$ 50	$0.48 \cdot 10^{-2}$ $1 \cdot 10^{-2}$ 37	$0.47 \cdot 10^{-2}$ 0.012 52
e^{xy}	$0.68 \cdot 10^{-3}$ $0.18 \cdot 10^{-3}$ 52	$0.68 \cdot 10^{-3}$ $0.18 \cdot 10^{-3}$ 50	$0.68 \cdot 10^{-3}$ $0.18 \cdot 10^{-3}$ 50	$0.14 \cdot 10^{-2}$ $0.98 \cdot 10^{-2}$ 48	$0.14 \cdot 10^{-2}$ $0.98 \cdot 10^{-2}$ 53	$0.13 \cdot 10^{-2}$ $0.78 \cdot 10^{-2}$ 57

Вывод: полностью (на всей области конечных элементов, а не только в узлах) аппроксимируются только линейные функции по пространству и для степени t равной 0, 1 или 2.

Вывод: значения в узлах полностью аппроксимируются только до полиномов 3 степени включительно по пространству.

Вывод: порядок аппроксимации по пространству — 3, порядок аппроксимации по времени — 2.

Вывод: все функции считаются примерно за одинаковое время.

3.1.2 50 на 50 на 50

Сетки аналогичны предыдущему пункту, только число элементов по всем сеткам равно 50.

time(t) / space(x, y)	0	t	t^2	t^3	t^4	e^t
1	$0.16 \cdot 10^{-13}$ $0.99 \cdot 10^{-12}$ 11457	$0.47 \cdot 10^{-12}$ $0.27 \cdot 10^{-10}$ 7489	$0.34 \cdot 10^{-11}$ $0.25 \cdot 10^{-9}$ 5392	$0.32 \cdot 10^{-4}$ $0.19 \cdot 10^{-2}$ 9174	$0.32 \cdot 10^{-4}$ $0.19 \cdot 10^{-2}$ 10206	$0.27 \cdot 10^{-4}$ $0.16 \cdot 10^{-2}$ 8092
$x + y$	$0.22 \cdot 10^{-11}$ $0.14 \cdot 10^{-9}$ 7857	$0.43 \cdot 10^{-11}$ $0.26 \cdot 10^{-9}$ 6748	$0.19 \cdot 10^{-11}$ $0.12 \cdot 10^{-9}$ 9419	$0.32 \cdot 10^{-4}$ $0.19 \cdot 10^{-2}$ 4634	$0.32 \cdot 10^{-4}$ $0.19 \cdot 10^{-2}$ 5804	$0.27 \cdot 10^{-4}$ $0.16 \cdot 10^{-2}$ 8486
$x^2 + y$	$0.13 \cdot 10^{-3}$ $0.15 \cdot 10^{-10}$ 10410	$0.13 \cdot 10^{-3}$ $0.23 \cdot 10^{-10}$ 10355	$0.13 \cdot 10^{-3}$ $0.68 \cdot 10^{-10}$ 7923	$0.16 \cdot 10^{-3}$ $0.19 \cdot 10^{-2}$ 10917	$0.16 \cdot 10^{-3}$ $0.19 \cdot 10^{-2}$ 5032	$0.16 \cdot 10^{-3}$ $0.16 \cdot 10^{-2}$ 7798
$x^2y + y^3$	$0.13 \cdot 10^{-3}$ $0.22 \cdot 10^{-10}$ 7839	$0.13 \cdot 10^{-3}$ $0.31 \cdot 10^{-10}$ 7179	$0.13 \cdot 10^{-3}$ $0.53 \cdot 10^{-10}$ 7845	$0.16 \cdot 10^{-3}$ $0.19 \cdot 10^{-2}$ 9060	$0.16 \cdot 10^{-3}$ $0.19 \cdot 10^{-2}$ 6514	$0.16 \cdot 10^{-3}$ $0.16 \cdot 10^{-2}$ 5366
xy^2	$0.32 \cdot 10^{-4}$ $0.58 \cdot 10^{-11}$ 6304	$0.32 \cdot 10^{-4}$ $0.11 \cdot 10^{-10}$ 7394	$0.32 \cdot 10^{-4}$ $0.13 \cdot 10^{-10}$ 7865	$0.64 \cdot 10^{-4}$ $0.19 \cdot 10^{-2}$ 7112	$0.64 \cdot 10^{-4}$ $0.19 \cdot 10^{-2}$ 5152	$0.59 \cdot 10^{-4}$ $0.16 \cdot 10^{-2}$ 5503
$x^4 + y^4$	$0.2 \cdot 10^{-3}$ $0.37 \cdot 10^{-2}$ 6207	$0.2 \cdot 10^{-3}$ $0.37 \cdot 10^{-2}$ 6130	$0.2 \cdot 10^{-3}$ $0.37 \cdot 10^{-2}$ 7456	$0.23 \cdot 10^{-3}$ $0.19 \cdot 10^{-2}$ 6644	$0.23 \cdot 10^{-3}$ $0.19 \cdot 10^{-2}$ 6714	$0.22 \cdot 10^{-3}$ $0.21 \cdot 10^{-2}$ 6149
e^{xy}	$0.31 \cdot 10^{-4}$ $0.34 \cdot 10^{-4}$ 6485	$0.31 \cdot 10^{-4}$ $0.34 \cdot 10^{-4}$ 6127	$0.31 \cdot 10^{-4}$ $0.34 \cdot 10^{-4}$ 4679	$0.63 \cdot 10^{-4}$ $0.18 \cdot 10^{-2}$ 5148	$0.63 \cdot 10^{-4}$ $0.18 \cdot 10^{-2}$ 5606	$0.59 \cdot 10^{-4}$ $0.16 \cdot 10^{-2}$ 4890

Вывод: предыдущие выводы не опровеглись.
Вывод: время вычислений выросло примерно в 110 раз.

3.2 Неравномерные сетки

3.2.1 Функции нелинейной сетки

В ходе выполнения лабораторной работы была обнаружена функция, позволяющая легко задавать неравномерную сетку, сгущающуюся к одному из концов.

Если у нас задано начало — a и конец сетки — b, а количество элементов n, тогда сетку можно задать следующим образом:

$$x_i = a + m\left(\frac{i}{n}\right) \cdot (b - a), i = \overline{0, n}$$

где $m(x)$ — некоторая функция, задающая неравномерную сетку. При этом x обязан принадлежать области $[0, 1]$, а функция m возвращать значения из той же области, и при этом быть монотонной на этом участке. Тогда гарантируется условие монотонности сетки, то есть что при $j \leqslant i \Rightarrow x_j \leqslant x_i$.

Пример: при $m(x) = x$, сетка становится равномерной.

Найденная функция зависят от параметра неравномерности t:

$$m_t(x) = \frac{1 - (1 - |t|)^{x \operatorname{sign} t}}{1 - (1 - |t|)^{\operatorname{sign} t}}$$

Эта функции вырождается в x при $t = 0$; при $t = -1$, она вырождается в сетку, полностью находящуюся в 0; а при $t = 1$ она полностью сгущается к 1.

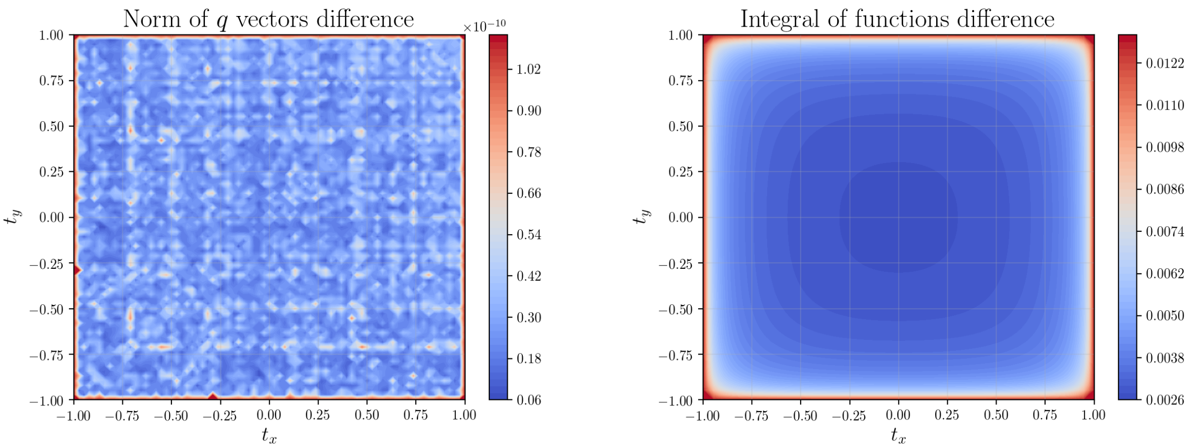
Таким образом, можно исследовать различные неравномерные сетки, изменяя параметр от -1 до 1, где точка $t = 0$ будет являться результатом на равномерной сетке.

3.2.2 По пространству

Сетка по пространству: $(x, y) \in [0, 1] \times [0, 1]$, строк и столбцов 10. Сетка по времени: $t \in [0, 1]$, количество элементов сетки 10.

3.2.2.1 Функция 1

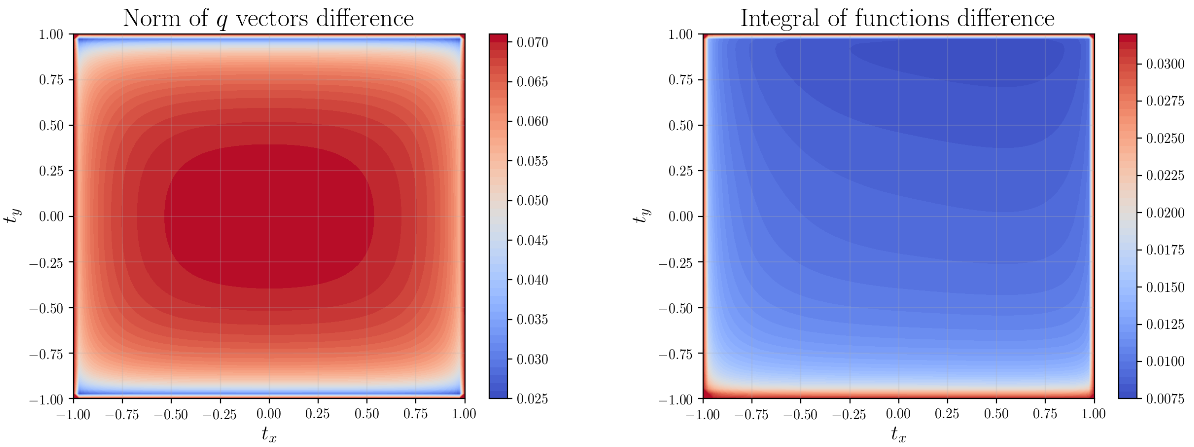
$$u = x^2 + y^2 + t^2$$



Вывод: так как эта функция полностью аппроксимируется данным методом в узлах, то не имеет значения насколько сетка неравномерна, примерно во всех элементах она имеет одинаковую невязку, согласно левому графику. Разве что в сильно неравномерных сетках, где элементы сильно сгущены к одному из концов, точностью страдает на несколько порядков.
Вывод: а по интегральной норме лучшей сеткой является раномерная сетка согласно правому графику.

3.2.2.2 Функция 2

$$u = x^4 + y^3 x + t^4$$

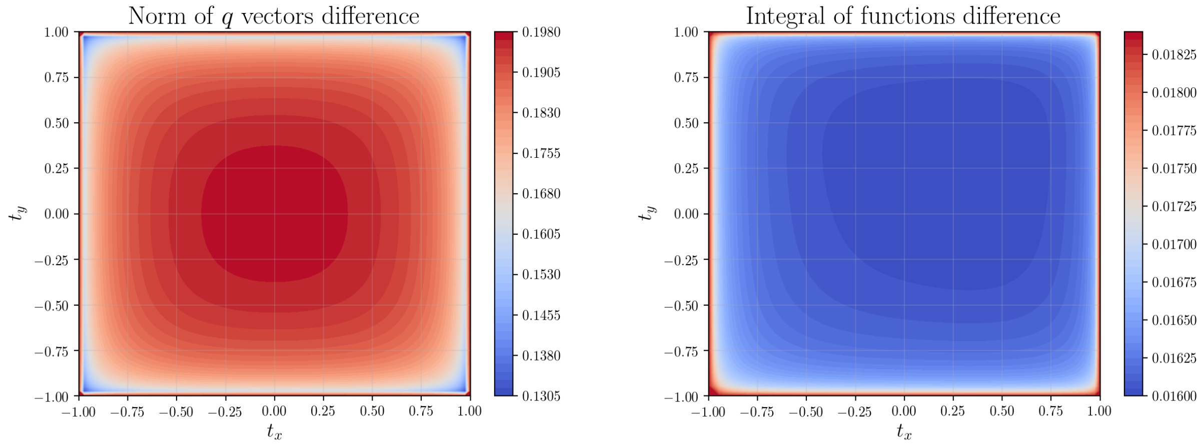


Вывод: согласно левому графику норма в узлах лучше всего аппроксимируется при сгущении по y в одну или другую сторону. По x же неравномерность сетки практически ни на что не влияет.
Вывод: лучшая точность, даваемая неравномерной сетки примерно на полпорядка лучше, чем при равномерной.

Вывод: по интегральной же норме существует некоторая комбинация параметров, при которых сетка получается оптимальной. Но различия от неравномерной сетки ничтожны.

3.2.2.3 Функция 3

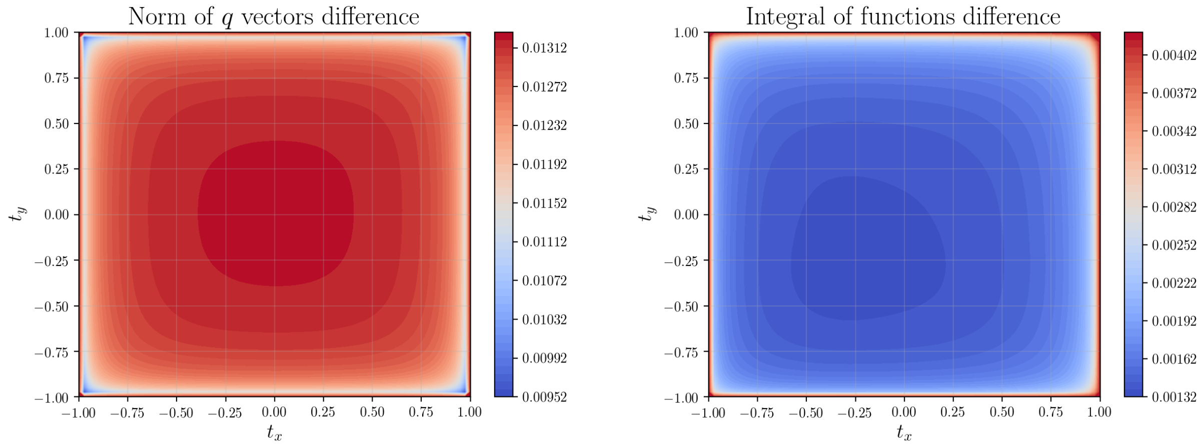
$$u = e^{xy} + e^{t^2}$$



Вывод: согласно левому графику аппроксимация в узлах тоже имеет некоторые оптимальлные значения, причем точность увеличивается на порядок.
Вывод: для интегральной же нормы различия же от равномерной сетки ничтожны при любых параметрах сетки.

3.2.2.4 Функция 4

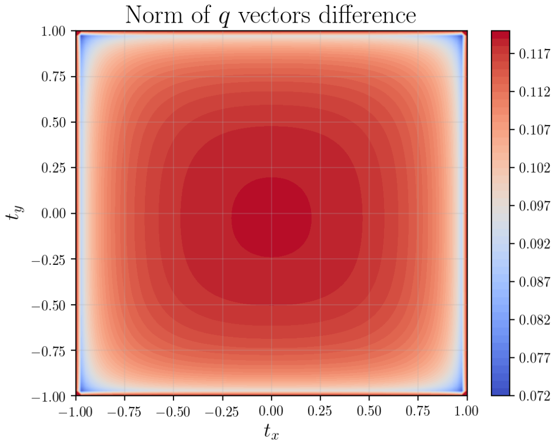
$$u = e^{(1-x)(1-y)} + e^{(1-t)^2}$$

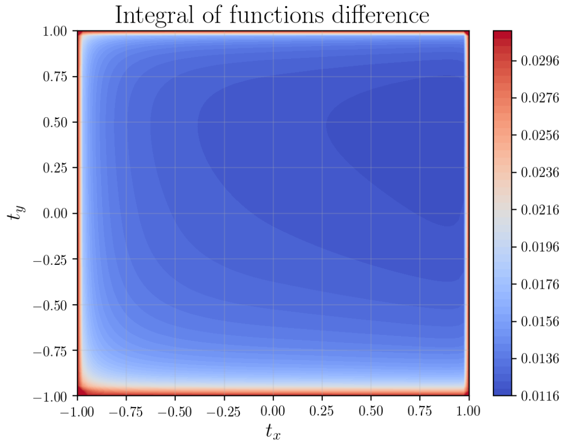


Вывод: эта функция отличается от предыдущей, что для неё инвертировано положение x и y, график по интегральной норме соответственно изменился.

3.2.2.5 Функция 5

$$u = x^3 + y^4 x^2 t + t^2 e^t$$





Вывод: всё аналогично предыдущим выводам и функциям.

3.2.2.6 Общие выводы

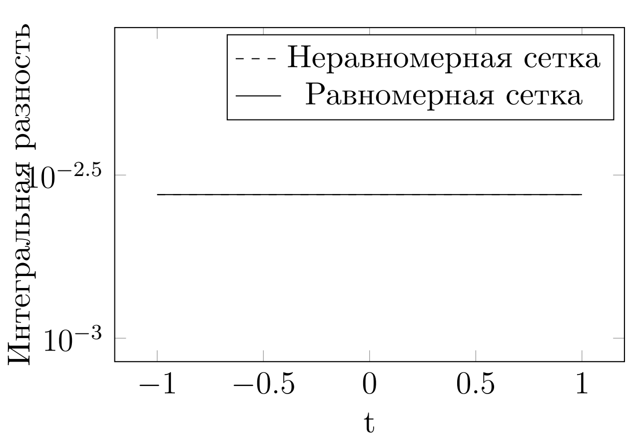
Вывод: хорошая аппроксимация в узлах $\neq$ хорошая аппроксимация по интегральной норме.
Вывод: согласно интегральной норме для неполиномиальных функций существует некоторый набор параметров t_x и t_y, при которых нелинейная сетка оптимальным образом аппроксимирует функцию.
Вывод: согласно норме в узлах для неполиномиальных функций оптимальными являются параметры в окрестности ± 1.

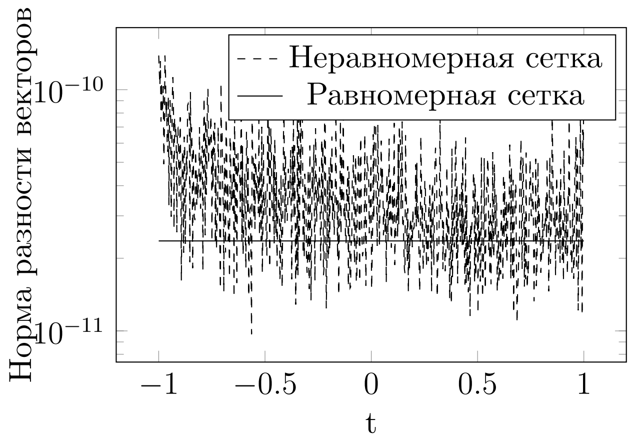
3.2.3 По времени

Сетка по пространству: $(x, y) \in [0, 1] \times [0, 1]$, строк и столбцов 10. Сетка по времени: $t \in [0, 1]$, количество элементов сетки 10.

3.2.3.1 Функция 1

$$u = x^2 + y^2 + t^2$$

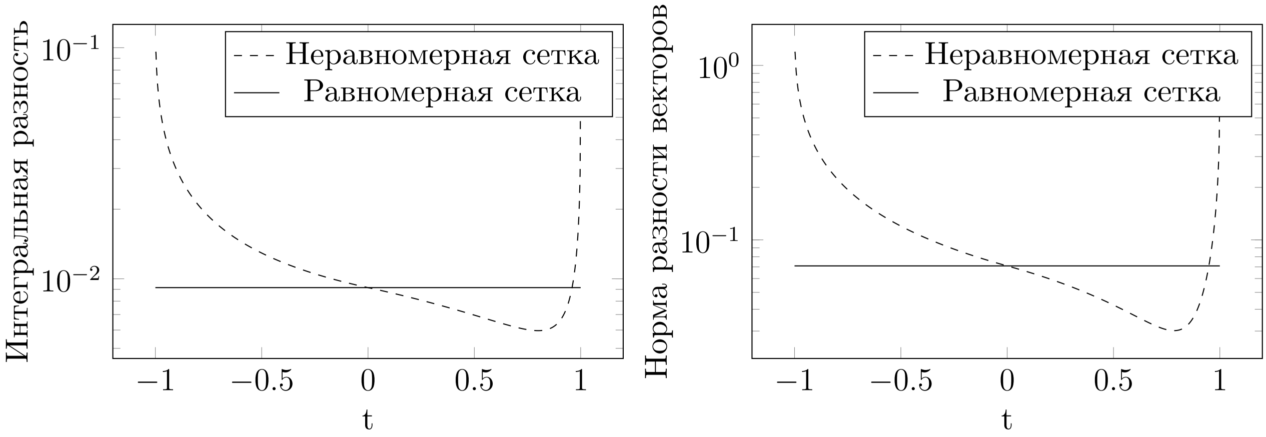




Вывод: так как по времени эта функция аппроксимируется точно, то неравномерность сетки никак не влияет на точность. Правый график колеблется в пределах максимальной точности, левый же абсолютно не меняется.

3.2.3.2 Функция 2

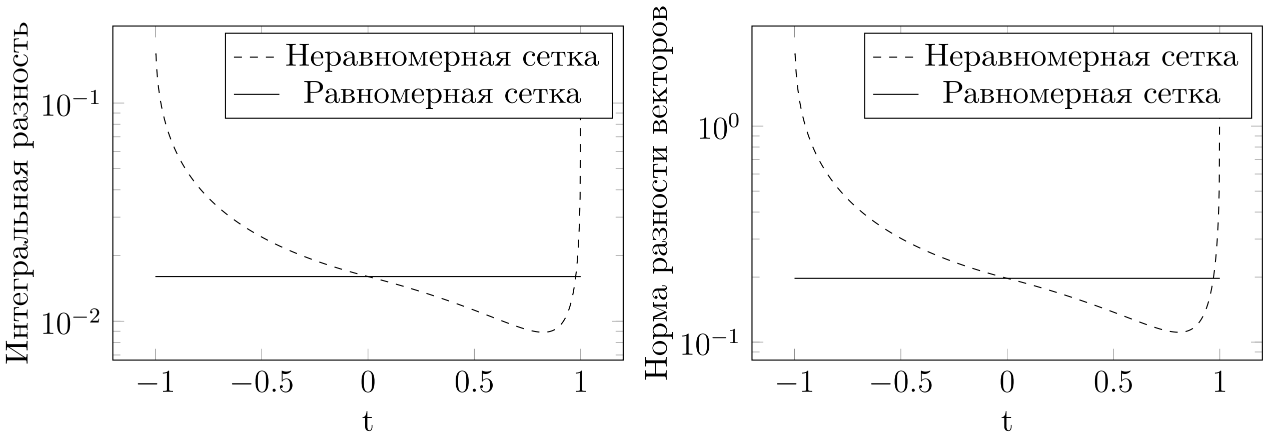
$$u = x^4 + y^3 x + t^4$$



Вывод: для данной функции в сетки есть выраженный минимум в окрестности $t = 0.7$, но улучшение точности на нем примерно полпорядка.

3.2.3.3 Функция 3

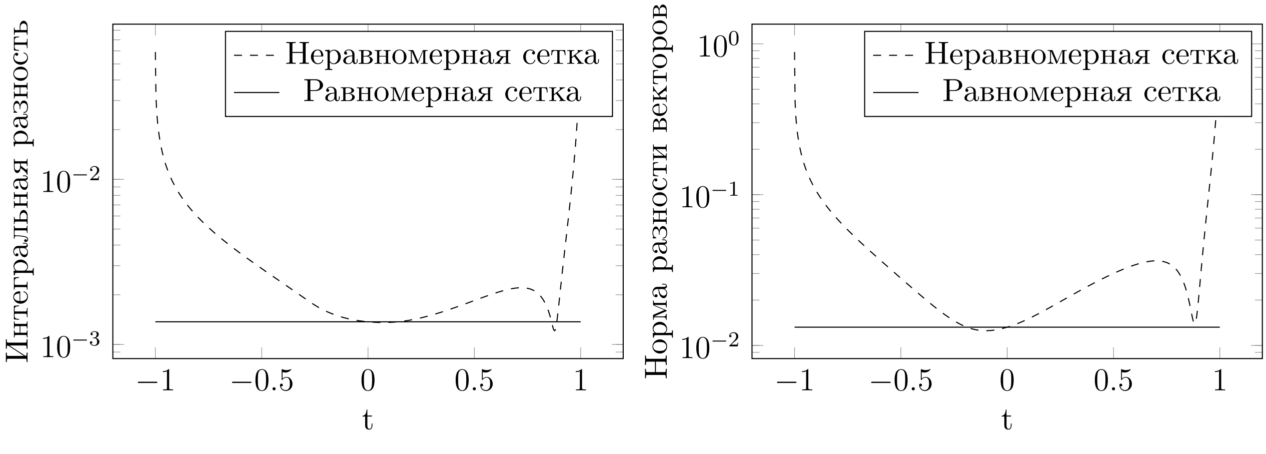
$$u = e^{xy} + e^{t^2}$$



Вывод: всё аналогично предыдущему.

3.2.3.4 Функция 4

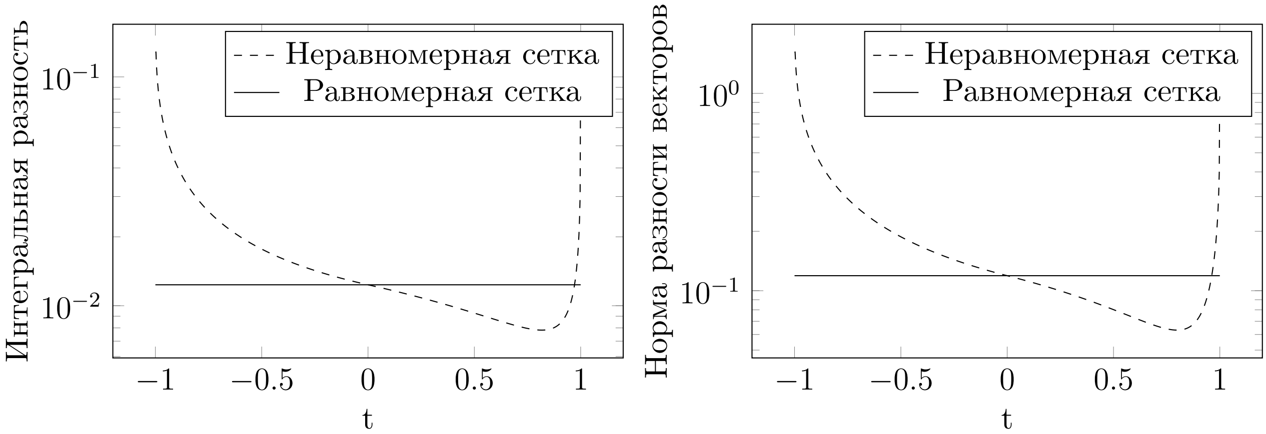
$$u = e^{(1-x)(1-y)} + e^{(1-t)^2}$$



Вывод: эта функция является перевернутой версией предыдущей, но график аналогчно не переверлся, а наблюдается более сложная зависимость. Для данной функции неравномерная сетка по времени практичеки везде дает отрицательный эффект по сравнению с равномерной сеткой.

3.2.3.5 Функция 5

$$u = x^3 + y^4x^2t + t^2e^t$$



3.2.3.6 Общие выводы

Вывод: у множества функций наблюдалось схожее поведение на неравномерной сетке по времени, с наличием ярко выраженного минимума, и использование сетки с данным оптимальным параметром может улучшить точность решения на полпорядка.

4 Код

main.cpp

```cpp
#include <iostream>
#include <cmath>
#include <string>
#include <fstream>
#include <thread>
#include <future>
#include "lib.h"
#include "fem.h"

using namespace std;
using namespace placeholders;

//-----------------------------------------------------------------------------
struct fem_result_t
{
    double integral_residual;
    double norm_residual;
    double time;
};

//-----------------------------------------------------------------------------
fem_result_t calc_fem_residual(
    const function_3d_t& u,
    const grid_generator_t& x_grid,
    const grid_generator_t& y_grid,
    const grid_generator_t& time_grid,
    const constants_t& c = {1, 1, 1, 1}
) {
    fem_result_t res;
    res.time = calc_time_microseconds([&](){
        auto f = calc_right_part(u, c);
```

```
        boundary_setter_t set_boundary_conditions = bind(write_first_boundary_conditions, _1, _2, _3, _4, u);

        grid_t grid;
        grid.calc(x_grid, y_grid);

        vector_t q0 = calc_true_approx(bind(u, _1, _2, time_grid(0)), grid.bes);
        vector_t q1 = calc_true_approx(bind(u, _1, _2, time_grid(1)), grid.bes);
        vector_t q = calc_true_approx(bind(u, _1, _2, time_grid.back()), grid.bes);

        auto steps = solve_differential_equation(f, set_boundary_conditions, q0, q1, c, grid, time_grid);

        res.integral_residual = calc_integral_norm(bind(u, _1, _2, time_grid.back()), grid.es, steps.back());
        res.norm_residual = (q-steps.back()).norm();
    });
    return res;
}

//-------------------------------------------------------------------------------
//-------------------------------------------------------------------------------
//-------------------------------------------------------------------------------

//-------------------------------------------------------------------------------
template<class Ret, class Key>
class async_performer_t
{
public:
    void add(const function<Ret(void)>& f, const Key& key) {
        mf[key] = async(f);
    }

    void finish(void) {
        int counter = 0;
        for (auto i = mf.rbegin(); i != mf.rend(); ++i) {
            if (counter % (mf.size()/10000 + 1) == 0)
                write_percent(double(counter)/mf.size());
            auto value = i->second.get();
            m[i->first] = value;
            counter++;
        }
        cout << "\r       \r";
    }

    auto begin(void) { return m.begin(); }
    auto end(void) { return m.end(); }

    Ret& operator[](const Key& key) { return m[key]; }
    const Ret& operator[](const Key& key) const { return m[key]; }
private:
    map<Key, future<Ret>> mf;
    map<Key, Ret> m;
};

//-------------------------------------------------------------------------------
//-------------------------------------------------------------------------------
//-------------------------------------------------------------------------------

//-------------------------------------------------------------------------------
template<class ForwardIt, class GetValue>
double max_element_ignore_nan(ForwardIt first, ForwardIt last, GetValue get) {
    return get(*max_element(first, last, [get] (auto& a, auto& b) -> bool {
        if (isnan(get(a)))
            return true;
        else
            return get(a) < get(b);
    }));
}

//-------------------------------------------------------------------------------
void investigate_t_changing(
    int n,
    const string& filename,
    const function<fem_result_t(double)>& ft
) {
    auto uniform_value = ft(0);

    async_performer_t<fem_result_t, int> performer;

    grid_generator_t grid(-1, 1, n);
    for (int i = 0; i < grid.size(); i++) {
        performer.add([i, ft, grid] () -> fem_result_t {
            return ft(grid(i));
        }, i);
    }

    performer.finish();
```

```

    int counter = 0;
    auto integral_residual_max = max_element_ignore_nan(performer.begin(), performer.end(), [] (auto& a) -> double { return a.second.integral_residual; });
    auto norm_residual_max = max_element_ignore_nan(performer.begin(), performer.end(), [] (auto& a) -> double { return a.second.norm_residual; });

    ofstream fout(filename + ".txt");
    fout << "t\tintegral\tnorm\tuniform_integral\tuniform_norm\ttime" << endl;
    for (int i = 0; i < grid.size(); i++) {
        auto v = performer[i];
        fout
            << grid(i) << "\t"
            << (isnan(v.integral_residual) ? integral_residual_max : v.integral_residual) << "\t"
            << (isnan(v.norm_residual) ? norm_residual_max : v.norm_residual) << "\t"
            << uniform_value.integral_residual << "\t"
            << uniform_value.norm_residual << "\t"
            << v.time << endl;
    }
    fout.close();
}

//---------------------------------------------------------------------------
void investigate_t2_changing(
    int n,
    const string& filename,
    const function<fem_result_t(double, double)>& ft
) {
    async_performer_t<fem_result_t, pair<int, int>> performer;

    grid_generator_t grid(-1, 1, n);
    for (int i = 0; i < grid.size(); i++) {
        for (int j = 0; j < grid.size(); j++) {
            performer.add([i, j, ft, grid] () -> fem_result_t {
                return ft(grid(i), grid(j));
            }, {i, j});
        }
    }

    performer.finish();

    auto integral_residual_max = max_element_ignore_nan(performer.begin(), performer.end(), [] (auto& a) -> double { return a.second.integral_residual; });
    auto norm_residual_max = max_element_ignore_nan(performer.begin(), performer.end(), [] (auto& a) -> double { return a.second.norm_residual; });

    ofstream fout(filename + ".integral.txt");
    ofstream fout2(filename + ".norm.txt");
    ofstream fout3(filename + ".time.txt");
    int last_line = 0;
    for (int i = 0; i < grid.size(); i++) {
        for (int j = 0; j < grid.size(); j++) {
            auto v = performer[{i, j}];
            fout << (isnan(v.integral_residual) ? integral_residual_max : v.integral_residual) << "\t";
            fout2 << (isnan(v.norm_residual) ? norm_residual_max : v.norm_residual) << "\t";
            fout3 << v.time << "\t";
        }
        fout << endl;
        fout2 << endl;
        fout3 << endl;
    }
    fout.close();
    fout2.close();

    fout.open(filename + ".x.txt");
    for (int i = 0; i < grid.size(); i++)
        fout << grid(i) << endl;
    fout.close();

    fout.open(filename + ".y.txt");
    for (int i = 0; i < grid.size(); i++)
        fout << grid(i) << endl;
    fout.close();
}

//---------------------------------------------------------------------------
void investigate_functions(
    const string& filename,
    const function<fem_result_t(const function_3d_t&)>& f
) {
    vector<pair<function_3d_t, string>> spaces, times;

    spaces.push_back({[] (double x, double y, double t) -> double { return 1; }, "$1$"});
    spaces.push_back({[] (double x, double y, double t) -> double { return x+y; }, "$x+y$"});
    spaces.push_back({[] (double x, double y, double t) -> double { return x*x+y*y; }, "$x^2+y$"});
    spaces.push_back({[] (double x, double y, double t) -> double { return x*x*y+y*y*y; }, "$x^2y+y^3$"});
    spaces.push_back({[] (double x, double y, double t) -> double { return x*y*y; }, "$xy^2$"});
```

```
    spaces.push_back({[] (double x, double y, double t) -> double { return x*x*x*x+y*y*y*y; }, "$x^4+y^4$"});
    spaces.push_back({[] (double x, double y, double t) -> double { return exp(x*y); }, "$e^{xy}$"});

    times.push_back({[] (double x, double y, double t) -> double { return 0; }, "$0$"});
    times.push_back({[] (double x, double y, double t) -> double { return t; }, "$t$"});
    times.push_back({[] (double x, double y, double t) -> double { return t*t; }, "$t^2$"});
    times.push_back({[] (double x, double y, double t) -> double { return t*t*t; }, "$t^3$"});
    times.push_back({[] (double x, double y, double t) -> double { return t*t*t; }, "$t^4$"});
    times.push_back({[] (double x, double y, double t) -> double { return exp(t); }, "$e^t$"});

    async_performer_t<fem_result_t, pair<string, string>> performer;

    for (auto& i : spaces) {
        for (auto& j : times) {
            performer.add([i, j, &f]() -> fem_result_t {
                return f(function_3d_t([&] (double x, double y, double t) -> double { return i.first(x, y, t) + j.first(x, y, t); }));
            }, {i.second, j.second});
        }
    }

    performer.finish();

    ofstream fout(filename);
    fout << "a\t";
    for (auto& i : times)
        fout << i.second << "\t";
    for (auto& i : spaces) {
        fout << endl << i.second << "\t";
        for (auto& j : times) {
            auto v = performer[{i.second, j.second}];
            fout << "\\scalebox{.75}{\\tcell{$" << write_for_latex_double(v.integral_residual, 2) << "$\\\\$" << write_for_latex_double(v.norm_residual, 2) << "$\\\\$" << int(v.time/1000) << "$}}\t";
        }
    }
    fout.close();
}

int main() {
    cout << calc_time_microseconds([](){
        investigate_functions(
            "functions_table_10_10_10.txt",
            [] (const function_3d_t& u) -> fem_result_t {
                return calc_fem_residual(u, grid_generator_t(0, 1, 10), grid_generator_t(0, 1, 10), grid_generator_t(0, 1, 10));
            }
        );

        investigate_functions(
            "functions_table_50_50_50.txt",
            [](const function_3d_t& u) ->  fem_result_t {
                return calc_fem_residual(u, grid_generator_t(0, 1, 50), grid_generator_t(0, 1, 50), grid_generator_t(0, 1, 50));
            }
        );

        vector<pair<function_3d_t, int>> u_space_mas;
        u_space_mas.push_back({[] (double x, double y, double t) -> double { return x*x + y*y + t*t; }, 0});
        u_space_mas.push_back({[] (double x, double y, double t) -> double { return x*x*x*x + y*y*y*x + t*t*t*t; }, 1});
        u_space_mas.push_back({[] (double x, double y, double t) -> double { return exp(x*y) + exp(t*t); }, 2});
        u_space_mas.push_back({[] (double x, double y, double t) -> double { return exp((1-x)*(1-y)) + exp((1-t)*(1-t)); }, 3});
        u_space_mas.push_back({[] (double x, double y, double t) -> double { return x*x*x + y*y*y*y*x*x*t + t*t*exp(t); }, 4});

        for (auto& i : u_space_mas) {
            auto& u = i.first;
            investigate_t_changing(
                750,
                "time_tgrid_" + to_string(i.second),
                [u] (double tt) -> fem_result_t {
                    return calc_fem_residual(u, grid_generator_t(0, 1, 10), grid_generator_t(0, 1, 10), grid_generator_t(0, 1, 10, tt));
                }
            );

            investigate_t2_changing(
                75,
                "space_tgrid_" + to_string(i.second),
                [u] (double tx, double ty) -> fem_result_t {
                    return calc_fem_residual(u, grid_generator_t(0, 1, 10, tx), grid_generator_t(0, 1, 10, ty), grid_generator_t(0, 1, 10));
```

```
                }
            );
        }
    })/1000/1000 << "s" << endl;
    system("pause");
}
```