Содержание

1 Лекция 1 (10.02)

A Формулы и обозначения

A.1 Векторы

A.1.1 Обозначение

Геометрические вектора обозначаются жирным шрифтом $\mathbf{v}$. Скалярные координаты вектора – через нижний индекс с обозначением оси координат: (v_x, v_y, v_z). Если вектор $\mathbf{u}$ – вектор скорости, то его декартовые координаты имеют специальное обозначение $\mathbf{u} = (u, v, w)$. Единичные вектора, соответствующие осям координат, обозначаются знаком $\hat{\cdot}$: $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$, $\hat{\mathbf{z}}$. Координатные векторы обозначаются по символу первой оси. Например, $\mathbf{x} = (x, y, z)$ или $\boldsymbol{\xi} = (\xi, \eta, \zeta)$.

Операции в векторами имеют следующее обозначение (расписывая в декартовых координатах):

- Умножение на скалярную функцию

$$f\mathbf{u} = (fu_x)\hat{\mathbf{x}} + (fu_y)\hat{\mathbf{y}} + (fu_z)\hat{\mathbf{z}}; \tag{A.1}$$

- Скалярное произведение

$$\mathbf{u} \cdot \mathbf{v} = u_x v_x + u_y v_y + u_z v_z; \tag{A.2}$$

- Векторное произведение

$$\mathbf{u} \times \mathbf{v} = \begin{vmatrix} \hat{\mathbf{x}} & \hat{\mathbf{y}} & \hat{\mathbf{z}} \\ u_x & u_y & u_z \\ v_x & v_y & v_z \end{vmatrix} = (u_y v_z - u_z v_y)\,\hat{\mathbf{x}} - (u_x v_z - u_z v_x)\,\hat{\mathbf{y}} + (u_x v_y - u_y v_x)\,\hat{\mathbf{z}}. \tag{A.3}$$

В двумерном случае можно считать, что $u_z = v_z = 0$. Тогда результатом векторного произведения согласно (A.3) будет вектор, направленный перпендикулярно плоскости xy:

$$\mathbf{u} \times \mathbf{v} = (u_x v_y - u_y v_x)\hat{\mathbf{z}}.$$

При работе с двумерными задачами, где ось $\mathbf{z}$ отсутствует, обычно результатом векторного произведения считают скаляр

$$2D: \ \mathbf{u} \times \mathbf{v} = u_x v_y - u_y v_x. \tag{A.4}$$

Геометрический смысл этого скаляра: площадь параллелограмма, построенного на векторах $\mathbf{u}$ и $\mathbf{v}$.

A.1.2 Набла–нотация

Символ ∇ – есть псевдовектор, который выражает покоординатные производные. Для декартовой системы координат (x, y, z) он запишется в виде

$$\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right).$$

В радиальной (r, ϕ, z):

$$\nabla = \left(\frac{\partial}{\partial r}, \frac{1}{r}\frac{\partial}{\partial \phi}, \frac{\partial}{\partial z} \right).$$

В цилиндрической (r, θ, ϕ):

$$\nabla = \left(\frac{\partial}{\partial r}, \frac{1}{r}\frac{\partial}{\partial \theta}, \frac{1}{r\sin\theta}\frac{\partial}{\partial \phi} \right).$$

Удобство записи дифференциальных выражений с использованием ∇ заключается в независимости записи от вида системы координат. Но если требуется обозначить производную по конкретной координате, то, по аналогии с обычными векторами, это делается через нижний индекс:

$$\nabla_n f = \frac{\partial f}{\partial n}.$$

Для этого символа справедливы все векторные операции, описанные ранее. Так, применение ∇ к скалярной функции аналогично умножению вектора на скаляр (A.1) (здесь и далее приводятся покоординатные выражения для декартовой системы):

$$\nabla f = (\nabla_x f,\ \nabla_y f,\ \nabla_z f) = \frac{\partial f}{\partial x}\hat{\mathbf{x}} + \frac{\partial f}{\partial y}\hat{\mathbf{y}} + \frac{\partial f}{\partial z}\hat{\mathbf{z}}. \tag{A.5}$$

Результатом этой операции является вектор.

Скалярное умножение ∇ на вектор $\mathbf{v}$ по аналогии с (A.2) – есть дивергенция:

$$\nabla \cdot \mathbf{v} = \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z} \tag{A.6}$$

результат которой – скалярная функция.

Двойное применение ∇ к скалярной функции – это оператор Лапласа:

$$\nabla \cdot \nabla f = \nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2} \tag{A.7}$$

Ротор – аналог векторного умножнения (A.3):

$$\nabla \times \mathbf{v} = \begin{vmatrix} \hat{\mathbf{x}} & \hat{\mathbf{y}} & \hat{\mathbf{z}} \\ \nabla_x & \nabla_y & \nabla_z \\ v_x & v_y & v_z \end{vmatrix} = \left(\frac{\partial v_z}{\partial y} - \frac{\partial v_y}{\partial z} \right) \hat{\mathbf{x}} - \left(\frac{\partial v_z}{\partial x} - \frac{\partial v_x}{\partial z} \right) \hat{\mathbf{y}} + \left(\frac{\partial v_y}{\partial x} - \frac{\partial v_x}{\partial y} \right) \hat{\mathbf{z}}. \tag{A.8}$$

A.2 Интегрирование

A.2.1 Формула Гаусса–Остроградского

Формула Гаусса–Остроградского, связывающая интегрирование по объёму E с интегрированием по границе этого объёма Г, для векторного поля $\mathbf{v}$ имеет вид

$$\int\limits_E \nabla \cdot \mathbf{v}\, d\mathbf{x} = \int\limits_\Gamma v_n\, ds, \tag{A.9}$$

где $\mathbf{n}$ – внешняя по отношению к области E нормаль. Смысл этой формулы можно проиллюстрировать на одномерном примере. Пусть одномерное векторное поле $v_x = f(x)$ на отрезке $E = [a, b]$ задано функцией, представленной на рис. 1. Разобъем область на $N = 3$ равномерных подобласти

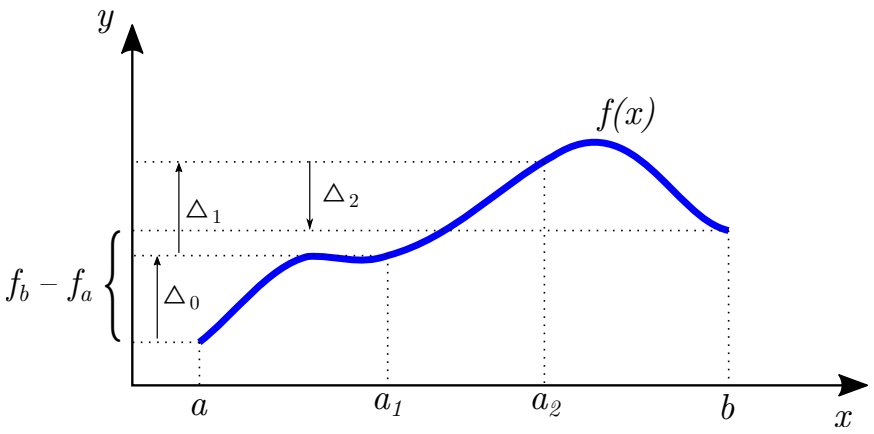


Рис. 1: Формула Гаусса-Остроградского в одномерном случае

длины h. Тогда расписывая интеграл как сумму, а производную через конечную разность, получим

$$\int\limits_E \frac{\partial f}{\partial x}\, dx \approx \sum_{i=0}^{2} h \left(\frac{\partial f}{\partial x}\right)_{i+\frac{1}{2}} \approx \sum_{i=0}^{2} (f_{i+1} - f_i) = \triangle_0 + \triangle_1 + \triangle_2 = f_b - f_a.$$

Очевидно что, при устремлении $N \to \infty$ правая часть предыдущего выражения не изменится. То есть, сумма всех изменений функции в области есть изменение функции по её границам:

$$\int\limits_a^b \frac{\partial f}{\partial x}\, dx = f(b) - f(a).$$

А формула (A.9) – есть многомерное обобщение этого выражения.

A.2.2 Интегрирование по частям

Подставив в (A.9) $\mathbf{v} = f\mathbf{u}$, где f – некоторая скалярная функция, и расписав дивергенцию в виде

$$\nabla \cdot (f\mathbf{u}) = f\nabla\mathbf{u} + \mathbf{u} \cdot \nabla f$$

получим формулу интегрирования по частям

$$\int\limits_E \mathbf{u} \cdot \nabla f \, d\mathbf{x} = \int\limits_\Gamma f u_n \, ds - \int\limits_E f \nabla \cdot \mathbf{u} \, d\mathbf{x} \tag{A.10}$$

Распишем некоторые частные случаи для формулы (A.10). Для $\mathbf{u} = (n_x, 0, 0)$ получим

$$\int\limits_E \frac{\partial f}{\partial x} \, d\mathbf{x} = \int\limits_\Gamma f \cos(\widehat{\mathbf{n}, \mathbf{x}}) \, ds \tag{A.11}$$

При $\mathbf{u} = \nabla g$

$$\int\limits_E f \left(\nabla^2 g\right) \, d\mathbf{x} = \int\limits_\Gamma f \frac{\partial g}{\partial n} \, ds - \int\limits_E \nabla f \cdot \nabla g \, d\mathbf{x} \tag{A.12}$$

При $f = 1$ и $\mathbf{u} = \nabla g$

$$\int\limits_E \nabla^2 g \, d\mathbf{x} = \int\limits_\Gamma \frac{\partial g}{\partial n} \, ds \tag{A.13}$$

A.2.3 Численное интегрирование в заданной области

Квадратурная формула

$$\int\limits_E f(\mathbf{x}) \, d\mathbf{x} = \sum_{i=0}^{N-1} w_i f(\mathbf{x}_i) \tag{A.14}$$

Она определяется заданием узлов интегрирования $\mathbf{x}_i$ и соответствующих весов w_i.

A.3 Интерполяционные полиномы

A.3.1 Многочлен Лагранжа

A.3.1.1 Узловые базисные функции

Рассмотрим функцию $f(\xi)$, заданную в области D. Внутри этой области зададим N узловых точек $\xi_i, i = \overline{0, N-1}$. Приближение функции f будем искать в виде

$$f(\xi) \approx \sum_{i=0}^{N-1} f_i \phi_i(\xi), \tag{A.15}$$

где $f_i = f(\xi_i)$, ϕ_i – узловая базисная функция. Потребуем, чтобы это выражение выполнялось точно для всех заданных узлов интерполяции $\xi = \xi_i$. Тогда, исходя из определения (A.15), запишем условие на узловую базисную функцию

$$\phi_i(\xi_j) = \begin{cases} 1, & i = j, \\ 0, & i \neq j. \end{cases} \tag{A.16}$$

Дополнительно потребуем, чтобы формула (A.15) была точной для постоянных функций

$$f(\xi) = \text{const} \quad \Rightarrow \quad f_i = \text{const}.$$

Тогда для любого ξ должно выполняться условие

$$\sum_{i=0}^{N-1} \phi_i(\xi) = 1, \qquad \xi \in D. \tag{A.17}$$

Задача построения интерполяционной функции состоит в конкретном определении узловых базисов $\phi_i(\xi)$ по заданному набору узловых точек ξ_i и значениям функции в них f_i. Будем искать базисы в виде многочленов вида

$$\phi_i(\xi) = \sum_a A_i^{(a)} \xi^a = A_i^{(0)} + A_i^{(1)} \xi + A_i^{(2)} \xi^2 + \ldots, \qquad i = \overline{0, N-1}. \tag{A.18}$$

Определять коэффициенты $A_i^{(a)}$ будем из условий (A.16), которое даёт N линейных уравнений относительно неизвестных $A_i^{(a)}$ для каждого $i = \overline{0, N-1}$. Таким образом, в выражениях (A.18) должно быть ровно N слагаемых. Будем использовать последовательный набор степеней: $a = \overline{0, N-1}$. Выпишем систему линейных уравнений для 0-ой базисной функции

$$\begin{aligned}
&\phi_0(\xi_0) = A_0^{(0)} + A_0^{(1)} \xi_0 + A_0^{(2)} \xi_0^2 + A_0^{(3)} \xi_0^3 + \ldots = 1, \\
&\phi_0(\xi_1) = A_0^{(0)} + A_0^{(1)} \xi_1 + A_0^{(2)} \xi_1^2 + A_0^{(3)} \xi_1^3 + \ldots = 0, \\
&\phi_0(\xi_2) = A_0^{(0)} + A_0^{(1)} \xi_2 + A_0^{(2)} \xi_2^2 + A_0^{(3)} \xi_2^3 + \ldots = 0, \\
&\ldots
\end{aligned}$$

или в матричном виде

$$\begin{pmatrix} 1 & \xi_0 & \xi_0^2 & \xi_0^3 & \dots \\ 1 & \xi_1 & \xi_1^2 & \xi_1^3 & \dots \\ 1 & \xi_2 & \xi_2^2 & \xi_2^3 & \dots \\ 1 & \xi_3 & \xi_3^2 & \xi_3^3 & \dots \\ \dots & & & & \end{pmatrix} \begin{pmatrix} A_0^{(0)} \\ A_0^{(1)} \\ A_0^{(2)} \\ A_0^{(3)} \\ \vdots \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ \vdots \end{pmatrix}$$

Записывая аналогичные выражения для остальных базисных функций, получим систему матричных уравнений вида $CA = E$:

$$\begin{pmatrix} 1 & \xi_0 & \xi_0^2 & \xi_0^3 & \dots \\ 1 & \xi_1 & \xi_1^2 & \xi_1^3 & \dots \\ 1 & \xi_2 & \xi_2^2 & \xi_2^3 & \dots \\ 1 & \xi_3 & \xi_3^2 & \xi_3^3 & \dots \\ \dots & & & & \end{pmatrix} \begin{pmatrix} A_0^{(0)} & A_1^{(0)} & A_2^{(0)} & A_3^{(0)} & \dots \\ A_0^{(1)} & A_1^{(1)} & A_2^{(1)} & A_3^{(1)} & \\ A_0^{(2)} & A_1^{(2)} & A_2^{(2)} & A_3^{(2)} & \\ A_0^{(3)} & A_1^{(3)} & A_2^{(3)} & A_3^{(3)} & \\ \vdots & & & & \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & \dots \\ 0 & 1 & 0 & 0 & \\ 0 & 0 & 1 & 0 & \\ 0 & 0 & 0 & 1 & \\ \vdots & & & & \end{pmatrix}$$

Отсюда матрица неизвестных коэффициентов A определится как

$$A = C^{-1} = \begin{pmatrix} 1 & \xi_0 & \xi_0^2 & \xi_0^3 & \dots \\ 1 & \xi_1 & \xi_1^2 & \xi_1^3 & \dots \\ 1 & \xi_2 & \xi_2^2 & \xi_2^3 & \dots \\ 1 & \xi_3 & \xi_3^2 & \xi_3^3 & \dots \\ \dots & & & & \end{pmatrix}^{-1}. \tag{A.19}$$

Подставляя полином (A.18) в условие согласованности (A.17), получим требование

$$\sum_{i=0}^{N-1} A_i^{(a)} = \begin{cases} 1, & a = 0, \\ 0, & a = \overline{1, N-1}. \end{cases}$$

То есть сумма всех свободных членов в интерполяционных полиномах должна быть равна единице, а сумма коэффициентов при остальных степенях – нулю. Можно показать, что это свойство выполняется для любой матрицы $A = C^{-1}$, в случае, если первый столбец матрицы C состоит из единиц. То есть условие согласованности требует наличие свободного члена с интерполяционном полиноме.

A.3.1.2 Интерполяция в параметрическом отрезке

Будем рассматривать область интерполяции $D = [-1, 1]$. В качестве первых двух узлов интерполяции возьмем границы области: $\xi_0 = -1$, $\xi_1 = 1$.

Линейный базис Будем искать интерполяционный базис в виде

$$\phi_i(\xi) = A_i^{(0)} + A_i^{(1)}\xi.$$

на основе двух условий:

$$\phi_i(-1) = A_i^{(0)} - A_i^{(1)} = \delta_{0i}, \quad \phi_i(1) = A_i^{(0)} + A_i^{(1)}\delta_{1i}.$$

Составим матрицу C, записав эти условия в матричном виде

$$C = \left(\begin{array}{c|cc} & A^{(0)} & A^{(1)} \\ \hline \phi(-1) & 1 & -1 \\ \phi(1) & 1 & 1 \end{array}\right)$$

и, согласно (A.19), найдём матрицу коэффициентов

$$A = \begin{pmatrix} A_0^{(0)} & A_1^{(0)} \\ A_0^{(1)} & A_1^{(1)} \end{pmatrix} = C^{-1} = \left(\begin{array}{c|cc} & \phi_0 & \phi_1 \\ \hline 1 & \frac{1}{2} & \frac{1}{2} \\ \xi & -\frac{1}{2} & \frac{1}{2} \end{array}\right).$$

Отсюда узловые базисные функции примут вид (рис. 2)

$$\begin{aligned} \phi_0(\xi) &= \frac{1-\xi}{2}, \\ \phi_1(\xi) &= \frac{1+\xi}{2}. \end{aligned} \tag{A.20}$$

Окончательно интерполяционная функция из определения (A.15) примет вид

$$f(\xi) \approx \frac{1-\xi}{2} f(-1) + \frac{1+\xi}{2} f(1).$$

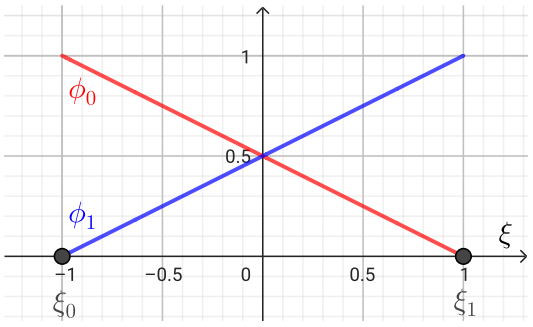


Рис. 2: Линейный базис в параметрическом отрезке

Квадратичный базис Будем искать интерполяционный базис в виде

$$\phi_i(\xi) = A_i^{(0)} + A_i^{(1)}\xi + A_i^{(2)}\xi^2.$$

По сравнению с линейным случаем, в форму базиса добавился ещё один неизвестый коэффициент $A_i^{(2)}$, поэтому в набор условий (A.16) требуется ещё одно уравнение (ещё одна узловая точка). Поме-

стим её в центр параметрического сегмента $\xi_2 = 0$. Далее будем действовать по аналогии с линейным случаем:

$$C = \left(\begin{array}{c|ccc} & A^{(0)} & A^{(1)} & A^{(2)} \\ \hline \phi(-1) & 1 & -1 & 1 \\ \phi(1) & 1 & 1 & 1 \\ \phi(0) & 1 & 0 & 0 \end{array}\right) \quad \Rightarrow \quad A = C^{-1} = \left(\begin{array}{c|ccc} & \phi_0 & \phi_1 & \phi_2 \\ \hline 1 & 0 & 0 & 1 \\ \xi & -\frac{1}{2} & \frac{1}{2} & 0 \\ \xi^2 & \frac{1}{2} & \frac{1}{2} & -1 \end{array}\right).$$

Узловые базисные функции для квадратичной интерполяции примут вид (рис. 3)

$$\begin{aligned} \phi_0(\xi) &= \frac{\xi^2 - \xi}{2}, \\ \phi_1(\xi) &= \frac{\xi^2 + \xi}{2}, \\ \phi_2(\xi) &= 1 - \xi^2. \end{aligned} \tag{A.21}$$

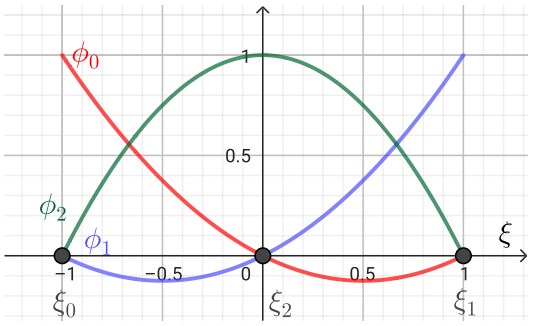


Рис. 3: Квадратичный базис в параметрическом отрезке

Кубический базис Интерполяционный базис будет иметь вид

$$\phi_i(\xi) = A_i^{(0)} + A_i^{(1)}\xi + A_i^{(2)}\xi^2 + A_i^{(3)}\xi^3.$$

Для нахождения четырёх коэффициентов нам понадобится четыре узла интерполяции. Две из них – это границы параметрического отрезка. Остальные две разместим так, чтобы разбить отрезок на равные интервалы: $\xi_2 = -\frac{1}{3}$, $\xi_3 = \frac{1}{3}$. Далее вычислим матрицу коэффициентов:

$$C = \left(\begin{array}{c|cccc} & A^{(0)} & A^{(1)} & A^{(2)} & A^{(3)} \\ \hline \phi(-1) & 1 & -1 & 1 & -1 \\ \phi(1) & 1 & 1 & 1 & 1 \\ \phi(-\frac{1}{3}) & 1 & -\frac{1}{3} & \frac{1}{9} & -\frac{1}{27} \\ \phi(\frac{1}{3}) & 1 & \frac{1}{3} & \frac{1}{9} & \frac{1}{27} \end{array}\right) \quad \Rightarrow \quad A = C^{-1} = \frac{1}{16}\left(\begin{array}{c|cccc} & \phi_0 & \phi_1 & \phi_2 & \phi_3 \\ \hline 1 & -1 & -1 & 9 & 9 \\ \xi & 1 & -1 & -27 & 27 \\ \xi^2 & 9 & 9 & -9 & -9 \\ \xi^3 & -9 & 9 & 27 & -27 \end{array}\right)$$

Узловые базисные функции для квадратичной интерполяции примут вид (рис. 4)

$$\begin{aligned}
\phi_0(\xi) &= \frac{1}{16}\left(-1+\xi+9\xi^2-9\xi^3\right), \\
\phi_1(\xi) &= \frac{1}{16}\left(-1-\xi+9\xi^2+9\xi^3\right), \\
\phi_2(\xi) &= \frac{1}{16}\left(9-27\xi-9\xi^2+27\xi^3\right), \\
\phi_3(\xi) &= \frac{1}{16}\left(9+27\xi-9\xi^2-27\xi^3\right),
\end{aligned} \tag{A.22}$$

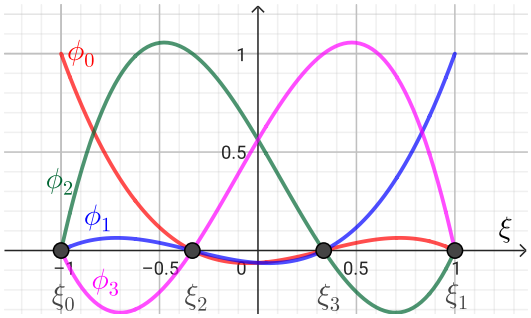


Рис. 4: Кубический базис в параметрическом отрезке

На рис. 5 представлено сравнение результатов аппроксимации функции $f(x) = -x + \sin(2x+1)$ линейным, квадратичным и кубическим базисом. Видно, что все интерполяционные приближения точно попадают в функцию в своих узлах интерполяции, а между узлами происходит аппроксимация полиномом соответствующей степени.

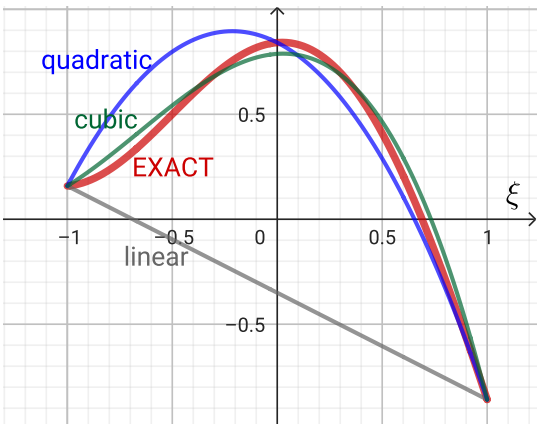


Рис. 5: Результат интерполяции

A.3.1.3 Интерполяция в параметрическом треугольнике

Теперь рассмотрим двумерное обобщение формулы

Линейный базис

$$\phi_i(\xi,\eta) = A_i^{(00)} + A_i^{(10)}\xi + A_i^{(01)}\eta.$$

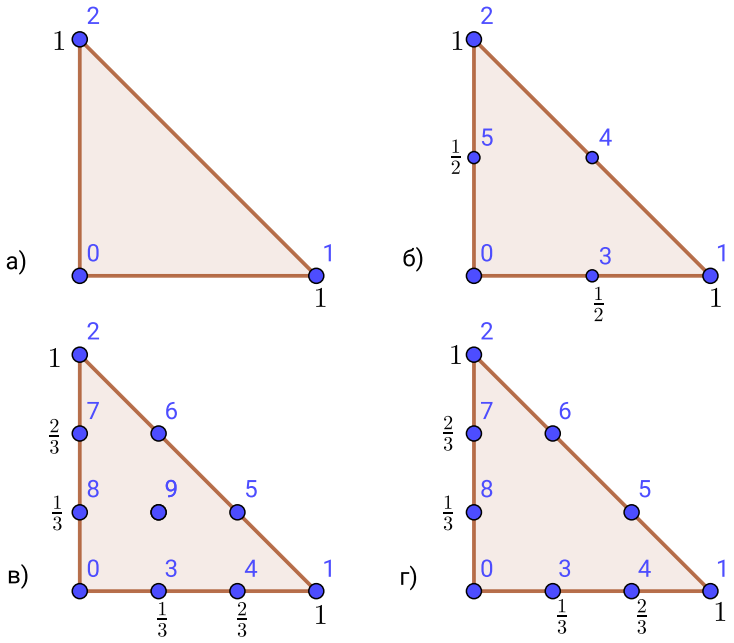


Рис. 6: Расположение узловых точек в параметрическом треугольнике. а) линейный базис, б) квадратичный базис, в) кубический базис, г) неполный кубический базис

$$C = \left(\begin{array}{c|ccc} & A^{(00)} & A^{(10)} & A^{(01)} \\ \hline \phi(0,0) & 1 & 0 & 0 \\ \phi(1,0) & 1 & 1 & 0 \\ \phi(0,1) & 1 & 0 & 1 \end{array}\right) \quad \Rightarrow \quad A = C^{-1} = \left(\begin{array}{c|ccc} & \phi_0 & \phi_1 & \phi_2 \\ \hline 1 & 1 & 0 & 0 \\ \xi & -1 & 1 & 0 \\ \eta & -1 & 0 & 1 \end{array}\right)$$

$$\begin{aligned} \phi_0(\xi,\eta) &= 1-\xi-\eta, \\ \phi_1(\xi,\eta) &= \xi, \\ \phi_2(\xi,\eta) &= \eta, \end{aligned} \tag{A.23}$$

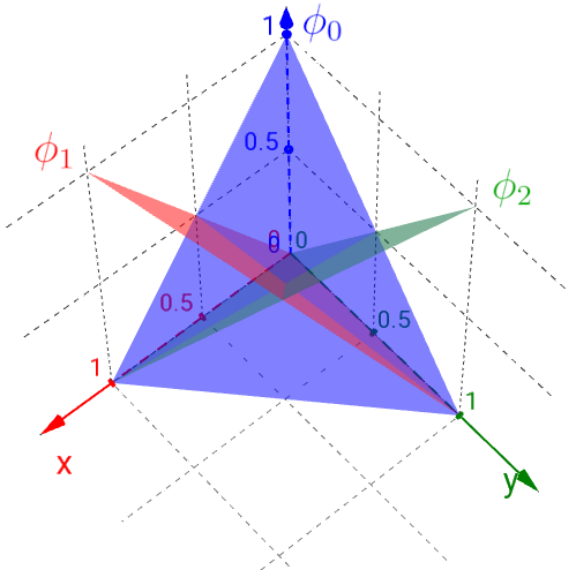


Рис. 7: Линейный базис в параметрическом треугольнике

Квадратичный базис

$$\phi_i(\xi,\eta) = A_i^{(00)} + A_i^{(10)}\xi + A_i^{(01)}\eta + A_i^{(11)}\xi\eta + A_i^{(20)}\xi^2 + A_i^{(02)}\eta^2.$$

$$C = \left(\begin{array}{c|cccccc} & A^{(00)} & A^{(10)} & A^{(01)} & A^{(11)} & A^{(20)} & A^{(02)} \\ \hline \phi(0,0) & 1 & 0 & 0 & 0 & 0 & 0 \\ \phi(1,0) & 1 & 1 & 0 & 0 & 1 & 0 \\ \phi(0,1) & 1 & 0 & 1 & 0 & 0 & 1 \\ \phi(\frac{1}{2},0) & 1 & \frac{1}{2} & 0 & 0 & \frac{1}{4} & 0 \\ \phi(\frac{1}{2},\frac{1}{2}) & 1 & \frac{1}{2} & \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \phi(0,\frac{1}{2}) & 1 & 0 & \frac{1}{2} & 0 & 0 & \frac{1}{4} \end{array}\right) \quad \Rightarrow \quad A = \left(\begin{array}{c|cccccc} & \phi_0 & \phi_1 & \phi_2 & \phi_3 & \phi_4 & \phi_5 \\ \hline 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ \xi & -3 & -1 & 0 & 4 & 0 & 0 \\ \eta & -3 & 0 & -1 & 0 & 0 & 4 \\ \xi\eta & 4 & 0 & 0 & -4 & 4 & -4 \\ \xi^2 & 2 & 2 & 0 & -4 & 0 & 0 \\ \eta^2 & 2 & 0 & 2 & 0 & 0 & -4 \end{array}\right)$$

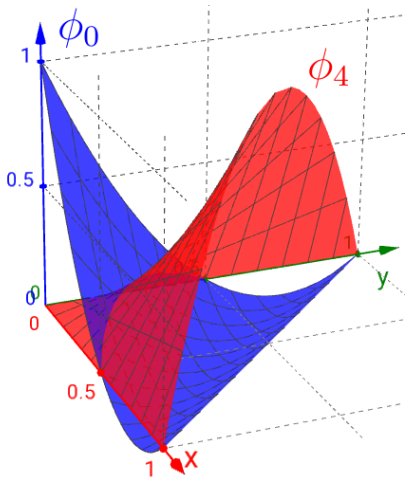


Рис. 8: Квадратичные функции ϕ_0, ϕ_4 в параметрическом треугольнике

Кубический базис TODO

Неполный кубический базис TODO

A.3.1.4 Интерполяция в параметрическом квадрате

Билинейный базис

$$\phi_i = A_i^{00} + A_i^{10}\xi + A_i^{01}\eta + A_i^{11}\xi\eta.$$

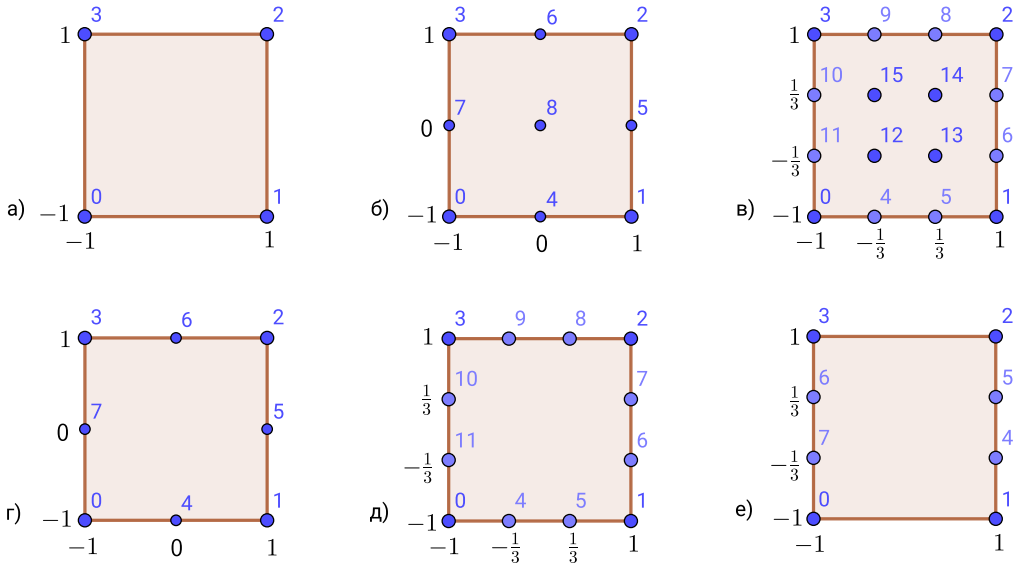


Рис. 9: Расположение узловых точек в параметрическом квадратре

$$C = \left(\begin{array}{c|cccc} & A^{(00)} & A^{(10)} & A^{(01)} & A^{(11)} \\ \hline \phi(-1,-1) & 1 & -1 & -1 & 1 \\ \phi(1,-1) & 1 & 1 & -1 & -1 \\ \phi(1,1) & 1 & 1 & 1 & 1 \\ \phi(-1,1) & 1 & -1 & 1 & -1 \end{array}\right) \quad \Rightarrow \quad A = C^{-1} = \frac{1}{4}\left(\begin{array}{c|cccc} & \phi_0 & \phi_1 & \phi_2 & \phi_3 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \xi & -1 & 1 & 1 & -1 \\ \eta & -1 & -1 & 1 & 1 \\ \xi\eta & 1 & -1 & 1 & -1 \end{array}\right)$$

$$\begin{aligned} \phi_0(\xi,\eta) &= \frac{1-\xi-\eta+\xi\eta}{4} \\ \phi_1(\xi,\eta) &= \frac{1+\xi-\eta-\xi\eta}{4} \\ \phi_2(\xi,\eta) &= \frac{1+\xi+\eta+\xi\eta}{4} \\ \phi_3(\xi,\eta) &= \frac{1-\xi+\eta-\xi\eta}{4} \end{aligned} \tag{A.24}$$

Определение двумерных базисов через комбинацию одномерных Обратим внимание, что в искомые билинейные базисные функции линейны в каждом из направлений ξ, η, если брать их по отдельности. Значит можно представить эти функции как комбинацию одномерных линейных базисов (A.20) в каждом из направлений. Узлы двумерного параметрического квадрата можно выразить через узлы линейного базиса в параметрическом одномерном сегменте, рассмотренном в п. A.3.1.2:

$$\boldsymbol{\xi}_0 = \left(\xi_0^{1D}, \xi_0^{1D}\right), \quad \boldsymbol{\xi}_1 = \left(\xi_1^{1D}, \xi_0^{1D}\right), \quad \boldsymbol{\xi}_2 = \left(\xi_1^{1D}, \xi_1^{1D}\right), \quad \boldsymbol{\xi}_3 = \left(\xi_0^{1D}, \xi_1^{1D}\right).$$

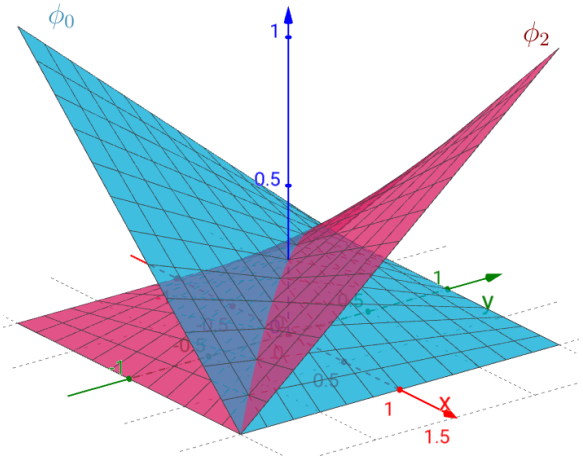


Рис. 10: Билинейные функции ϕ_0, ϕ_2 в параметрическом квадрате

Значит и соответствующие базисные функции можно выразить через линейный одномерный базис ϕ^{1D} из соотношений (A.20):

$$\phi_0(\xi,\eta) = \phi_0^{1D}(\xi)\phi_0^{1D}(\eta) = \frac{1-\xi}{2}\frac{1-\eta}{2},$$
$$\phi_1(\xi,\eta) = \phi_1^{1D}(\xi)\phi_0^{1D}(\eta) = \frac{1+\xi}{2}\frac{1-\eta}{2},$$
$$\phi_2(\xi,\eta) = \phi_1^{1D}(\xi)\phi_1^{1D}(\eta) = \frac{1+\xi}{2}\frac{1+\eta}{2},$$
$$\phi_3(\xi,\eta) = \phi_0^{1D}(\xi)\phi_1^{1D}(\eta) = \frac{1-\xi}{2}\frac{1+\eta}{2}.$$

Раскрыв скобки можно убедится, что мы получили тот же билинейный базис, что и ранее (A.24).

Биквадратичный базис Применим этот метод для вычисления биквадратичного базиса, определённого в точках на рис. 9б. В качестве основе возьмём квадратичный одномерный базис ϕ_i^{1D} из (A.21).

$$\begin{aligned}
&\phi_0(\xi,\eta) = \phi_0^{1D}(\xi)\phi_0^{1D}(\eta) = \frac{\xi^2-\xi}{2}\frac{\eta^2-\eta}{2}, && \phi_1(\xi,\eta) = \phi_1^{1D}(\xi)\phi_0^{1D}(\eta) = \frac{\xi^2+\xi}{2}\frac{\eta^2-\eta}{2},\\
&\phi_2(\xi,\eta) = \phi_1^{1D}(\xi)\phi_1^{1D}(\eta) = \frac{\xi^2+\xi}{2}\frac{\eta^2+\eta}{2}, && \phi_3(\xi,\eta) = \phi_0^{1D}(\xi)\phi_1^{1D}(\eta) = \frac{\xi^2-\xi}{2}\frac{\eta^2+\eta}{2},\\
&\phi_4(\xi,\eta) = \phi_2^{1D}(\xi)\phi_0^{1D}(\eta) = (1-\xi^2)\frac{\eta^2-\eta}{2}, && \phi_5(\xi,\eta) = \phi_1^{1D}(\xi)\phi_2^{1D}(\eta) = \frac{\xi^2+\xi}{2}(1-\eta^2),\\
&\phi_6(\xi,\eta) = \phi_2^{1D}(\xi)\phi_1^{1D}(\eta) = (1-\xi^2)\frac{\eta^2+\eta}{2}, && \phi_7(\xi,\eta) = \phi_0^{1D}(\xi)\phi_2^{1D}(\eta) = \frac{\xi^2-\xi}{2}(1-\eta^2),\\
&\phi_8(\xi,\eta) = \phi_2^{1D}(\xi)\phi_2^{1D}(\eta) = (1-\xi^2)(1-\eta^2).
\end{aligned} \tag{A.25}$$

Бикубический базис

Неполный биквадратичный базис

Неполный бикубический базис

A.4 Геометрические алгоритмы

A.4.1 Линейная интерполяция

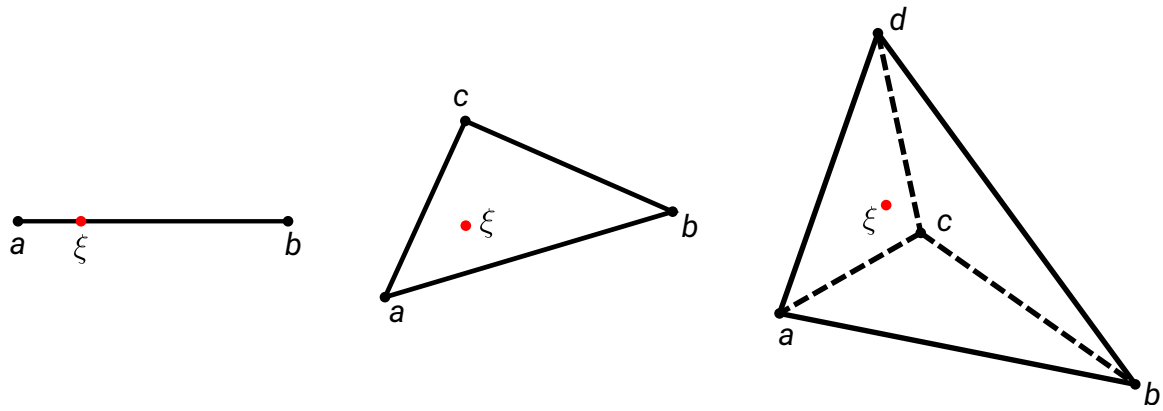


Рис. 11: Порядок нумерации точек одномерного, двумерного и трёхмерного симплекса при линейной интерполяции

Пусть функция u задана в узлах симплекса, имеющего нумерацию согласно рис. 11. Необходимо найти значение этой функции в точке $\boldsymbol{\xi}$ (эта точка вообще говоря не обязана лежать внутри симлекса).

Интерполяция в одномерном, двумерном и трёхмерном виде запишется как

$$u(\xi) = \frac{|\triangle_{\xi a}|u(b) + |\triangle_{b\xi}|u(a)}{|\triangle_{ba}|} \tag{A.26}$$

$$u(\xi) = \frac{|\triangle_{ab\xi}|u(c) + |\triangle_{bc\xi}|u(a) + |\triangle_{ca\xi}|u(b)}{|\triangle_{abc}|} \tag{A.27}$$

$$u(\xi) = \frac{|\triangle_{abc\xi}|u(d) + |\triangle_{cbd\xi}|u(a) + |\triangle_{cda\xi}|u(b) + |\triangle_{adb\xi}|u(c)}{|\triangle_{abcd}|}, \tag{A.28}$$

где $|\triangle|$ – знаковый объём симплекса, вычисляемый как

$$|\triangle_{ab}| = b - a,$$

$$|\triangle_{abc}| = \left(\frac{(\mathbf{b}-\mathbf{a}) \times (\mathbf{c}-\mathbf{a})}{2}\right)_z,$$

$$|\triangle_{abcd}| = \frac{(\mathbf{b}-\mathbf{a}) \cdot ((\mathbf{c}-\mathbf{a}) \times (\mathbf{d}-\mathbf{a}))}{6}.$$

A.4.2 Преобразование координат

Рассмотрим преобразование из двумерной параметрической системы координат $\boldsymbol{\xi}$ в физическую систему $\mathbf{x}$. Такое преобразование полностью определяется покоординатными функциями $\mathbf{x}(\boldsymbol{\xi})$. Далее получим соотношения, связывающие операции дифференцирования и интегрирования в физической и параметрической областях.

A.4.2.1 Матрица Якоби

Будем рассматривать двумерное преобразование $(\xi, \eta) \to (x, y)$. Линеаризуем это преобразование (разложим в ряд Фурье до линейного слагаемого)

$$x(\xi_0 + d\xi, \eta_0 + d\eta) \approx x_0 + \left.\frac{\partial x}{\partial \xi}\right|_{\xi_0,\eta_0} d\xi + \left.\frac{\partial x}{\partial \eta}\right|_{\xi_0,\eta_0} d\eta,$$
$$y(\xi_0 + d\xi, \eta_0 + d\eta) \approx y_0 + \left.\frac{\partial y}{\partial \xi}\right|_{\xi_0,\eta_0} d\xi + \left.\frac{\partial y}{\partial \eta}\right|_{\xi_0,\eta_0} d\eta,$$

где $x_0 = x(\xi_0, \eta_0)$, $y_0 = y(\xi_0, \eta_0)$. Переписывая это выражение в векторном виде, получим

$$\mathbf{x}(\boldsymbol{\xi}_0 + \mathbf{d\xi}) - \mathbf{x}_0 = J(\boldsymbol{\xi}_0)\, \mathbf{d\xi}. \tag{A.29}$$

Матрица J (зависящая от точки приложения в параметрической плоскости) называется матрицей Якоби:

$$J = \begin{pmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{pmatrix} = \begin{pmatrix} \dfrac{\partial x}{\partial \xi} & \dfrac{\partial x}{\partial \eta} \\ \dfrac{\partial y}{\partial \xi} & \dfrac{\partial y}{\partial \eta} \end{pmatrix} \tag{A.30}$$

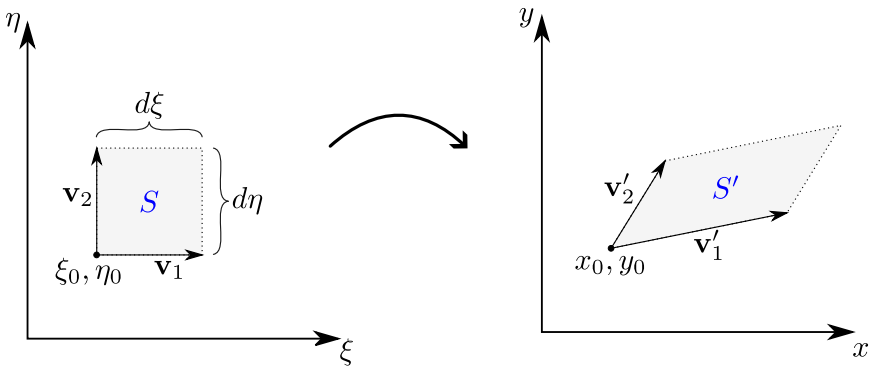


Рис. 12: Преобразование элементарного объёма

Якобиан Определитель матрицы Якоби (якобиан), взятый в конкретной точке параметрической плоскости $\boldsymbol{\xi}_0$, показывает, во сколько раз увеличился элементарный объём около этой точки в результате преобразования. Действительно, рассмотрим два перпендикулярных элементарных вектора в параметрической системе координат: $\mathbf{v}_1 = (d\xi, 0)$ и $\mathbf{v}_2 = (0, d\eta)$ отложенных от точки $\boldsymbol{\xi}_0$ (см. рис. 12). В результате преобразования по формуле (A.29) получим следующие преобразования концевых точек и векторов:

$$\begin{aligned}
&(\xi_0, \eta_0) \to (x_0, y_0), \\
&(\xi_0 + d\xi, \eta_0) \to (x_0 + J_{11} d\xi, y_0 + J_{21} d\xi) \quad \Rightarrow \quad \mathbf{v}_1 \to \mathbf{v}'_1 = (J_{11} d\xi, J_{21} d\xi), \\
&(\xi_0, \eta_0 + d\eta) \to (x_0 + J_{12} d\eta, y_0 + J_{22} d\eta) \quad \Rightarrow \quad \mathbf{v}_2 \to \mathbf{v}'_2 = (J_{12} d\eta, J_{22} d\eta).
\end{aligned}$$

Элементарный объём равен площади параллелограмма, построенного на элементарных векторах. В параметрической плоскости согласно (A.4) получим

$$|S| = \mathbf{v}_1 \times \mathbf{v}_2 = d\xi d\eta,$$

и аналогично для физической плоскости:

$$|S'| = \mathbf{v}'_1 \times \mathbf{v}'_2 = (J_{11}J_{22} - J_{12}J_{21})d\xi d\eta = |J|d\xi d\eta$$

Сравнивая два последних соотношения приходим к выводу, что элементарный объём в результате преобразования увеличился в $|J|$ раз. Тогда можно записать

$$dx\,dy = |J|\,d\xi\,d\eta \tag{A.31}$$

Многомерным обобщением этой формулы будет

$$d\mathbf{x} = |J|\,d\boldsymbol{\xi} \tag{A.32}$$

A.4.2.2 Дифференцирование в параметрической плоскости

Пусть задана некоторая функция $f(x, y)$. Распишем её производную по параметрическим координатам:

$$\frac{\partial f}{\partial \xi} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial \xi},$$
$$\frac{\partial f}{\partial \eta} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial \eta} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial \eta}.$$

Вспоминая определение (A.30), запишем

$$\begin{pmatrix} \dfrac{\partial f}{\partial \xi} \\ \dfrac{\partial f}{\partial \eta} \end{pmatrix} = J^T \begin{pmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{pmatrix} = \begin{pmatrix} J_{11} & J_{21} \\ J_{12} & J_{22} \end{pmatrix} \begin{pmatrix} \dfrac{\partial f}{\partial \xi} \\ \dfrac{\partial f}{\partial \eta} \end{pmatrix}$$

Обратная зависимость примет вид

$$\begin{pmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{pmatrix} = \left(J^T\right)^{-1} \begin{pmatrix} \dfrac{\partial f}{\partial \xi} \\ \dfrac{\partial f}{\partial \eta} \end{pmatrix} = \frac{1}{|J|} \begin{pmatrix} J_{22} & -J_{21} \\ -J_{12} & J_{11} \end{pmatrix} \begin{pmatrix} \dfrac{\partial f}{\partial \xi} \\ \dfrac{\partial f}{\partial \eta} \end{pmatrix}$$

В многомерном виде запишем

$$\nabla_{\mathbf{x}} f = \left(J^T\right)^{-1} \nabla_{\boldsymbol{\xi}} f. \tag{A.33}$$

A.4.2.3 Интегрирование в параметрической плоскости

Пусть в физической области $\mathbf{x}$ задана область D_x. Интеграл функции $f(\mathbf{x})$ по этой области можно расписать, используя замену (A.32)

$$\int\limits_{D_x} f(\mathbf{x})\, d\mathbf{x} = \int\limits_{D_\xi} f(\boldsymbol{\xi})\, |J(\boldsymbol{\xi})| d\boldsymbol{\xi}, \tag{A.34}$$

где $f(\boldsymbol{\xi}) = f(\mathbf{x}(\boldsymbol{\xi}))$, а D_ξ – образ области D_x в параметрической плоскости.

A.4.2.4 Двумерное линейное преобразование. Параметрический треугольник

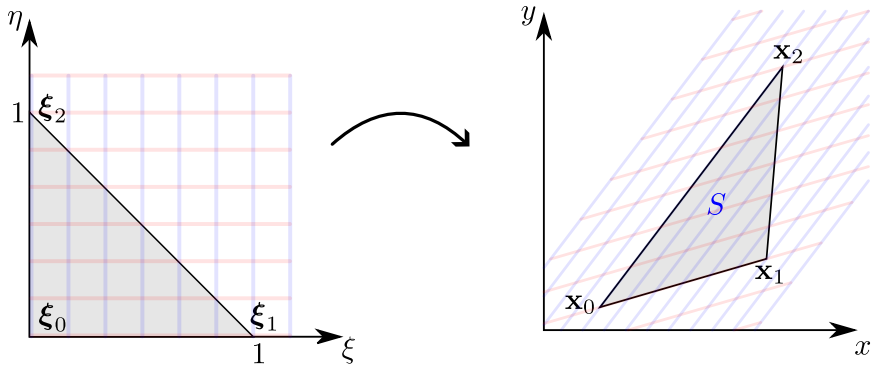


Рис. 13: Преобразование из параметрического треугольника

Рассмотрим двумерное преобразование, при котором определяющие функции являются линейными. То есть представимыми в виде

$$x(\xi, \eta) = A_x \xi + B_x \eta + C_x,$$
$$y(\xi, \eta) = A_y \xi + B_y \eta + C_y.$$

Для определения шести констант, определяющих это преобразование, достаточно выбрать три любые (не лежащие на одной прямой) точки: $(\xi_i, \eta_i) \to (x_i, y_i)$ для $i = 0, 1, 2$. В результате получим систему из шести линейных уравнений (три точки по две координаты), из которой находятся константы $A_{x,y}, B_{x,y}, C_{x,y}$. Пусть три точки в параметрической плоскости образуют единичный прямоугольный треугольник (рис. 13):

$$\xi_0, \eta_0 = (0, 0), \quad \xi_1, \eta_1 = (1, 0), \quad \xi_2, \eta_2 = (0, 1).$$

Тогда система линейных уравнений примет вид

$$x_0 = C_x, \quad y_0 = C_y,$$
$$x_1 = A_x + C_x, \quad y_1 = A_y + C_y,$$
$$y_2 = B_x + C_x, \quad y_2 = B_y + C_y.$$

Определив коэффициенты преобразования их этой системы, окончательно запишем преобразование

$$x(\xi, \eta) = (x_1 - x_0)\xi + (x_2 - x_0)\eta + x_0,$$
$$y(\xi, \eta) = (y_1 - y_0)\xi + (y_2 - y_0)\eta + y_0. \tag{A.35}$$

Матрица Якоби этого преобразования (A.30) не будет зависеть от параметрических координат ξ, η:

$$J = \begin{pmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{pmatrix}. \tag{A.36}$$

Якобиан преобразования будет равен удвоенной площади треугольника S, составленного из определяющих точек в физической плоскости:

$$|J| = (x_1 - x_0)(y_2 - y_0) - (y_1 - y_0)(x_2 - x_0) = (\mathbf{x}_1 - \mathbf{x}_0) \times (\mathbf{x}_2 - \mathbf{x}_0) = 2|S|. \tag{A.37}$$

Распишем интеграл по треугольнику S по формуле (A.34). Вследствии линейности преобразования якобиан постоянен и, поэтому, его можно вынести его из-под интеграла:

$$\int\limits_S f(x,y)\,dxdy = |J| \int\limits_0^1 \int\limits_0^{1-\xi} f(\xi,\eta) d\eta d\xi. \tag{A.38}$$

A.4.2.5 Двумерное билинейное преобразование. Параметрический квадрат

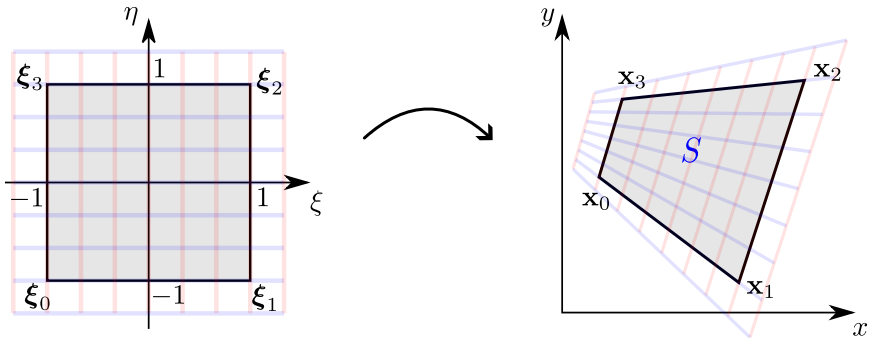


Рис. 14: Преобразование из параметрического квадрата

A.4.2.6 Трёхмерное линейное преобразование. Параметрический тетраэдр

TODO

A.4.3 Свойства многоугольника

A.4.3.1 Площадь многоугольника

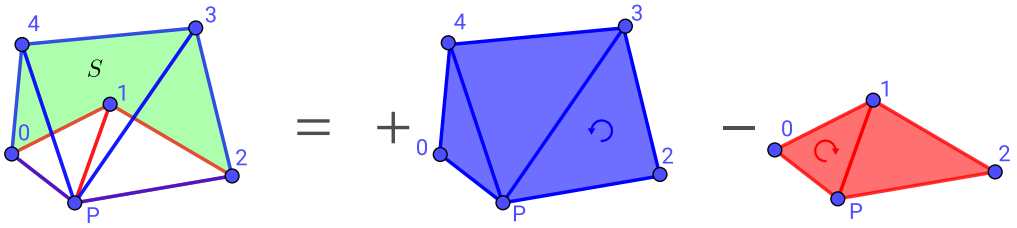


Рис. 15: Площадь произвольного многоугольника

Рассмотрим произвольный несамопересекающийся N-угольник S, заданный координатами своих узлов $\mathbf{x}_i$, $i = \overline{0, N-1}$, пронумерованных последовательно против часовой стрелки (рис. 15). Далее введём произвольную точку $\mathbf{p}$ и от этой точки будем строить ориентированные треугольники до граней многоугольника:

$$\triangle_i^p = (\mathbf{p}, \mathbf{x}_i, \mathbf{x}_{i+1}), \quad i = \overline{0, N-1},$$

(для корректности записи будем считать, что $\mathbf{x}_N = \mathbf{x}_0$). Тогда площадь исходного многоугольника S будет равна сумме знаковых площадей треугольников $\triangle_i^p$:

$$|S| = \sum_{i=0}^{N-1} |\triangle_i^p|, \qquad |\triangle_i^p| = \frac{(\mathbf{x}_i - \mathbf{p}) \times (\mathbf{x}_{i+1} - \mathbf{p})}{2}.$$

Знак площади ориентированного треугольника зависит от направления закрутки его узлов: она положительна для закрутки против часовой стрелки и отрицательна, если узлы пронумерованы по часовой стрелке. В частности, на рисунке 15 видно, что треугольники, отмеченные красным: $P01, P12$, будут иметь отрицательную площадь, а синие треугольники $P23$, $P34$, $P40$ – положительную. Сумма этих площадей с учётом знака даст искомую площадь многоугольника.

Для сокращения вычислений воспользуемся произвольностью положения $\mathbf{p}$ и совместим её с точкой $\mathbf{x}_0$. Тогда треугольники $\triangle_0^p$, $\triangle_{N-1}^p$ выродятся (будут иметь нулевую площадь). Обозначим такую последовательную триангуляцию как

$$\triangle_i = (\mathbf{x}_0, \mathbf{x}_i, \mathbf{x}_{i+1}), \qquad i = \overline{1, N-2}. \tag{A.39}$$

Знаковая площадь ориентированного треугольника будет равна

$$|\triangle_i| = \frac{(\mathbf{x}_i - \mathbf{x}_0) \times (\mathbf{x}_{i+1} - \mathbf{x}_0)}{2}. \tag{A.40}$$

Тогда окончательно формула определения площади примет вид

$$|S| = \sum_{i=1}^{N-2} |\triangle_i|. \tag{A.41}$$

Плоский полигон в пространтве Если плоский полигон S расположен в трёхмерном пространстве, то правая часть формулы (A.40) согласно определению векторного произведения в трёхмерном пространстве (A.3) – есть вектор. Чтобы получить скалярную площадь, нужно спроецировать этот вектор на единичную нормаль к плоскости многоугольника:

$$\mathbf{n} = \frac{\mathbf{k}}{|\mathbf{k}|}, \qquad \mathbf{k} = (\mathbf{x}_1 - \mathbf{x}_0) \times (\mathbf{x}_2 - \mathbf{x}_0).$$

Эта формула записана из предположения, что узел $\mathbf{x}_2$ не лежит на одной прямой с узлами $\mathbf{x}_0$, $\mathbf{x}_1$. Иначе вместо $\mathbf{x}_2$ нужно выбрать любой другой узел, удовлетворяющий этому условию. Тогда площадь ориентированного треугольника, построенного в трёхмерном пространстве запишется через смешанное произведение:

$$|\triangle_i| = \frac{((\mathbf{x}_i - \mathbf{x}_0) \times (\mathbf{x}_{i+1} - \mathbf{x}_0)) \cdot \mathbf{n}}{2}. \tag{A.42}$$

Формула для определения площади полигона (A.41) будет по прежнему верна. При этом итоговый знак величины S будет положительным, если закрутка полигона положительная (против часовой стрелки) при взгляде со стороны вычисленной нормали $\mathbf{n}$.

A.4.3.2 Интеграл по многоугольнику

Рассмотрим интеграл функции $f(x, y)$ по N-угольнику S, заданному последовательными координатами своих узлов $\mathbf{x}_i$. Введём последовательную триангуляцию согласно (A.39). Тогда интеграл по многоугольнику можно расписать как сумму интегралов по ориентированным треугольникам:

$$\int\limits_S f(x,y)\,dxdy = \sum_{i=1}^{N-2} \int\limits_{\triangle_i} f(x,y)\,dxdy. \tag{A.43}$$

Далее для вычисления интегралов в правой части воспользуемся преобразованием к параметрическому треугольнику (п. A.4.2.4). Следуя формуле интегрирования (A.38), распишем интеграл по i-ому треугольнику:

$$\int\limits_{\triangle_i} f(x,y)\,dxdy = |J_i| \int\limits_0^1 \int\limits_0^{1-\xi} f_i(\xi,\eta)\,d\eta d\xi,$$

где якобиан $|J_i|$ согласно (A.37) есть удвоенная площадь ориентированного треугольника $\triangle_i$ (положительная при закрутке против часовой стрелке и отрицаетельная иначе):

$$|J_i| = 2|\triangle_i| = (\mathbf{x}_i - \mathbf{x}_0) \times (\mathbf{x}_{i+1} - \mathbf{x}_0),$$

а функция $f_i(\xi, \eta)$ есть функция от преобразованных согласно (A.35) переменных:

$$f_i(\xi,\eta) = f\left((\mathbf{x}_i - \mathbf{x}_0)\,\xi + (\mathbf{x}_{i+1} - \mathbf{x}_0)\,\eta + \mathbf{x}_0\right).$$

Окончательно запишем

$$\int\limits_S f(x,y)\,dxdy = 2\sum_{i=1}^{N-2} |\triangle_i| \int\limits_0^1 \int\limits_0^{1-\xi} f_i(\xi,\eta)\,d\eta d\xi. \tag{A.44}$$

Отметим, что эта формула работает и в том случае, когда полигон расположен в трёхмерном пространстве (знаковую площадь при этом следует вычислять по (A.42)).

A.4.3.3 Центр масс многоугольника

По определению, координаты центра масс $\mathbf{c}$ области S равны среднеинтегральным значениям координатных функций. То есть

$$c_x = \frac{1}{|S|}\int\limits_S x\,dxdy, \quad c_y = \frac{1}{|S|}\int\limits_S y\,dxdy.$$

Далее распишем интеграл в правой части через последовательную триангуляцию согласно (A.43) с учётом линейного преобразования (A.35):

$$\begin{aligned}\int\limits_S x\,dxdy &= \sum_{i=1}^{N-2}\int\limits_{\triangle_i} x\,dxdy \\ &= \sum_{i=1}^{N-2}|J_i|\int\limits_0^1\int\limits_0^{1-\xi}((x_i-x_0)\xi+(x_{i+1}-x_0)\eta+x_0)\,d\eta d\xi \\ &= \sum_{i=1}^{N-2}\frac{|J_i|}{2}\frac{x_0+x_i+x_{i+1}}{3} \\ &= \sum_{i=1}^{N-2}|\triangle_i|\frac{x_0+x_i+x_{i+1}}{3}.\end{aligned}$$

Итого, с учётом (A.41), координаты центра масс примут вид

$$\mathbf{c} = \frac{\sum\limits_{i=1}^{N-2}\frac{\mathbf{x}_0+\mathbf{x}_i+\mathbf{x}_{i+1}}{3}|\triangle_i|}{\sum\limits_{i=1}^{N-2}|\triangle_i|}.$$

Если полигон расположен в двумерном пространстве xy, то знаковая площадь треугольников вычисляется по формуле (A.40). В случае трёхмерного пространтва должна использоваться формула (A.42).

A.4.4 Свойства многогранника

A.4.4.1 Объём многогранника

TODO

A.4.4.2 Интеграл по многограннику

TODO

A.4.4.3 Центр масс многогранника

TODO

A.4.5 Поиск многоугольника, содержащего заданную точку

TODO

B Работа с инфраструктурой проекта CFDCourse

В настоящем параграфе будут даны инструкции для разворачивания инфраструктуры сборки проекта для операционных систем Linux(Ubuntu) и Windows. Для других версий Linux и MacOS алгоритм останется тем же за исключением работы с менеджером пакетов (`apt` – в Ubuntu, `brew` – в MacOS, `pacman` – в ArchLinux и т.д.).

В процессе настройки будет необходимо установить систему контроля версий **Git**, систему контейнеризации **Docker** и (опционально) интегрированную среду разработки. Проект позволяет работать в любой средой разработки и даже вообще без неё. Ниже будут приведены инструкции для настройки **vscode**.

Процесс сборки и запуска программ будет осуществляться в системе, развёрнутой в докере на основе Ubuntu 24.04. В дальнейшем систему, установленную непосредственно на компьютере, будем называть хост системой. А систему, развёрнутую в докере – контейнером.

Для успешной установке на хосте должно быть около 5Гб свободного места.

B.1 Клонирование

Для клонирования проекта на локальный компьютер необходимо установить систему контроля версий Git и открыть терминал на хост-системе в папке, в которой планируется хранить папку с репозиторием. Если в качестве это папки будет использоваться домашняя пользовательская папка. В нижеследующих инструкциях в качестве такой папки будет использоваться домашняя папка пользователя.

Ubuntu

Откройте терминал и в нём

```
> sudo apt install git  # установка гита
> cd ~                  # переходим в папку,
                        # где будет хранится папка с репозиторием
```

Windows

В Windows необходимо скачать и установить дистрибутив https://github.com/git-for-windows/git/releases/download/v2.51.0.windows.1/Git-2.51.0-64-bit.exe. Далее откройте командную строку `cmd`. И в ней перейдите в целевую папку

```
> cd %USERPROFILE%
```

Далее необходимо клонировать репозиторий

```
> git clone https://github.com/kalininei/CFDCourse26
```

В результате в домашней папке должна появиться папка с `CFDCourse26`.

B.2 Разворачивание контейнера

Установим докер

Ubuntu+apt

Запустить скрипт, написанный на основе инструкций с официального сайта https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository:

```
cd ~/CFDCourse26    # перейдём в репозиторий
./scripts/ubuntu_docker_install.sh
```

Windows/MacOs+DockerDesktop

Скачайте и установите дистрибутив с официального сайта

- https://docs.docker.com/desktop/setup/install/windows-install/ для Windows
- https://docs.docker.com/desktop/setup/install/mac-install/ для MacOs

Далее следуйте процедуре установки десктопного приложения. После установки запустите `DockerDesktop`. Этап регистрации при запуске опционален и может быть пропущен.

Далее в терминале находясь в директории `CFDCourse26` развернём контейнер:

```
docker compose up --build -d
```

На этом этапе будет скачаны необходимые образы и запущен контейнер. По окончании можно убедится, что контейнер работает

```
docker ps
```

В случае работы с `DockerDesctop` запущенный контейнер будет виден в графическом интерфейсе во вкладке `Containers`.

B.3 Базовая разработка

Чтобы скомпилировать проект необходимо войти в терминал контейнера. Далее

```
docker ps                   # Убедимся, что контейнер cfd26 запущен
docker exec -it cfd26 bash  # Войдём в него
```

Папка хоста с репозиторием `CFDCourse` примонтирована к папке контейнера `\app`. Войдём туда

```
cd /app   # заходим в директорию
ls -alh   # смотрим список файлов
```

Для удобства в контейнере установлен консольный файловый менеджер `mc`, который можно использовать для операций с файлами. Так же есть его псевдоним `mcc`, который отличается от базового `mc` тем, что запоминает текущую директорию при выходе. Благодаря чему его можно использовать для навигации вместо `cd`.

B.3.1 Особенности проекта

- Проект состоит из статически линкуемой библиотеки `libcfd26.a` и исполняемого файла с тестовыми программами для этой библиотеки. Исходники библиотеки лежат в директории `src/cfd`, исходники тестов – `src/test`,
- Используется 20-ый стандарт C++,
- Сборка осуществляется в системе cmake,
- Для написания тестов используется фреймворк `Catch2`,
- В проекте установлены жёсткие правила работы с проедупреждениями компилляции, из-за которых они как обрабатываются ошибки,
- В проекте установлены форматтеры для исходных кодов на C++, python, cmake. При сохрании любого исходного файла этих форматов в vscode форматтер будет вызван автоматически. Если исходники модифицировались иначе, то запустить форматтер для всех файлов проекта можно скриптом `scripts/formatall.sh` из папки `\app`.

B.3.2 Сборка в отладочном режиме

Из папки `\app` контейнера:

```
mkdir build                         # создаём папку, в которую будут строится программа
cd build                            # Заходим
cmake .. -DCMAKE_BUILD_TYPE=Debug   # Строим сборочные скрипты
make -j4                            # Компилируем на 4-х потоках
```

Сама папка `build` является временной папкой для построения. Она не подключена к системе контроля версий. И может быть удалена и создана заново (например для полной гарантированной очистки кэша построения). Бинарные файлы будут построены в папку `\app\build\bin`. Для запуска тестов зайдём в эту папку:

```
cd bin
./cfd26_test               # запуск всех тестов
./cfd26_test [grid1]       # запуск единственного тест кейса
```

B.3.3 Сборка в релизном режиме

Отличается от сборки в отладочном режиме только флагом `cmake`. Будем строить в папку `build_release`. Также от папки `\app`

```
mkdir build_release                           # создаём папку
cd build_release                              # Заходим
cmake .. -DCMAKE_BUILD_TYPE=RelWithDebInfo    # Или просто Release,
                                              # если отладочная информация не нужна
```

```
make -j4                                    # Компилируем на 4-х потоках
cd bin                                      # заходим в папку с исполняемым файлом
./cfd26_test                                # запуск всех тестов
./cfd26_test [grid1]                        # запуск единственного тест кейса
```

B.3.4 Работа с кодом

Работать с исходными файлами можно находясь на хосте и используя любой удобный текстовый редактор. (например
`notepad++` на Windows). Порядок работы будет выглядеть следующим образом

- Редакатировать исходные файлы на хосте в папке `CFDCourse26/src`
- Для сборки переключиться на терминал и выполнить вышеописанную процедуру сборки
- Если сборка не прошла из-за ошибок в коде, эти ошибки будут распечатаны в терминал с указанием файла и номера строки
- Для отладки можно использовать консольный отладчик `gdb` из терминала

B.4 Разработка в vscode

B.4.1 Подключение к контейнеру

Необходимо установить vscode на хосте следуя инструкциям с официального сайта https://code.visualstudio.com/Download. В самом vscode нужно установить расширение
`Remote Explorer, Dev Containers` для удалённой разработки в контейнере. Далее нужно переключиться на вкладку RemoteExplorer (см. рис. 16).

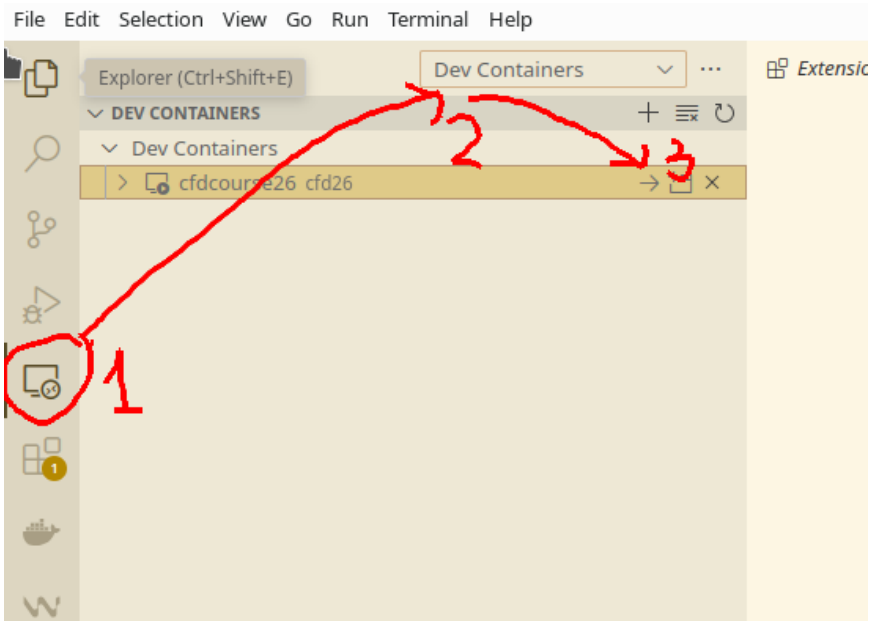


Рис. 16: Подключения vscode к контейнеру

B.4.2 Настройки vscode

Далее необходимо открыть папку `\app`:
`File->Open Folder...`. Контейнер содержит в себе базовые настройки vscode, которые при сборке контейнера копируются в папки `.vscode, .vscode-server`.

Следует отметить, что удалённо подключённый vscode не может пользоваться расширениями, установленными на хосте. Все необходимые расширения нужно устанавливать в контейнер заново. Список базовых расширений, необходимых для работы, уже содержится в настроечных файлах. Когда вы откроете папку `app` в vscode, вам будет предложено эти расширения установить. Нужно согласиться.

Настроечные папки
`.vscode, .vscode-server` игнорируются системой контроля версий и расположены на хосте. То есть вы можете дополнительно установить туда любые свои расширения и донастроить vscode как вам удобно. Эти настройки не будут зависеть от ветки гита и не будут затираться при пересборке контейнера.

Если всё же понадобится обнулить настройки vscode, нужно

1. удалить папки `.vscode, .vscode_server`
2. удалить все настройки из конфигурационного файла контейнера (`ctrl+shift+p, open container configuration file`)
3. выйти из контейнера на vscode: `file->close remote connection`
4. остановить и пересобрать контейнер на хосте

```
docker stop cfd26
docker compose up --build -d
```

B.4.3 Сборка и отладка

В папке `.vscode` лежат базовые таски и лаунчи для компилляции и запуска программы. В случае необходимости можете дополнить базовый набор своими командами. Согласно настройкам по умолчанию по нажатию `F5` происходит сборка программы в отладочном режиме и запуск отладки тестовой программы с прогоном всех тестов. Посмотреть результаты можно на вкладке `TERMINAL`. В случае ошибок компилляции, список этих ошибок будет виден на вкладке `PROBLEMS`.

Для отладки конкретного теста необходимо запустить тестовую программу с аргументом (например `cfd26_test [grid1]`. Чтобы передать программе аргумент нужно этот аргумент прописать в файле `.vscode/launch.json` в поле `args`. Либо создать ещё одну конфигурацию запуска с вашими аргументами и указать эту конфигурацию в настройке `Run And Dubug`.

Чтобы собрать программу без запуска нужно выполнить таск (`ctrl+shift+b`):
`cmake: build debug, cmake: build release` для отладочного и релизного режима соответсвенно.

В целом сборка на vscode представляет из себя автоматизированный алгоритм, представленный в п. B.3. То есть исполняемая программа в дебаговой версии кладётся в папку `build`, в релизной – в `build_release` и её можно запустить из терминала. Удаление этих папок ведёт к полной очистке кеша построения.

B.5 Работа с системой контроля версий

Работать с гитом можно как с хоста (из папки `CFDCourse26`), так и из контейнера (из папки `\app`). Ниже будут даны инструкции для работы с гитом в консоли. Альтернативно, можно установить графический интерфейс (например `GitExtensions` для Windows) или командами vscode на вкладке `Source Control`.

Из системы контроля версий исключены следующие каталоги:

- build*/ – папки со сборками,
- .vscode, .vscode-server – настройки и рисширения vscode,
- local_data – папка для хранения любых пользовательских данных.

Изменения из этих папках не будут отслежены и скоммичены.

B.5.1 Порядок работы с репозиторием CFDCourse

Основная ветка проекта – `master`. После каждой лекции в эту ветку будет отправлен коммит с сообщением `lect{index}`. В этом коммите будет дополнен pdf документ с содержанием лекции, задание по итогам лекции и необходимые для этого задания изменения в коде.

B.5.1.1 Получение последнего коммита

Таким образом, **после лекции**, после того, как изменение
`lect{index}` придёт на сервер, необходимо выполнить следующие команды

```
git checkout master  # перейти на основную ветку
git pull             # получить изменения
```

Если изменения не содержали никаких изменений в настроечных файлах контейнера `Dockerfile, docker-compose.yaml`, то для сборки проекта рестарт контейнера не требуется. Иначе требуется пересобрать контейнер:

```
docker stop cfd26            # остановить текущий контейнер
docker compose up --build -d # пересобрать новый
```

B.5.1.2 Создание коммита с текущим дз

Перед началом лекции, если была сделана какая то работа по заданиям,

```
> git checkout -b hw-lect{index}      # создать локальную ветку, содержащую задание
> git add .
> git commit -m "{свой комментарий}"  # скоммитить свои изменения в эту ветку
```

Даже если задание выполнено не до конца, вы в любой момент можете переключиться на ветку с заданием и его доделать

```
git checkout work-lect{index}
```

B.5.1.3 Создание коммита с прошлым дз

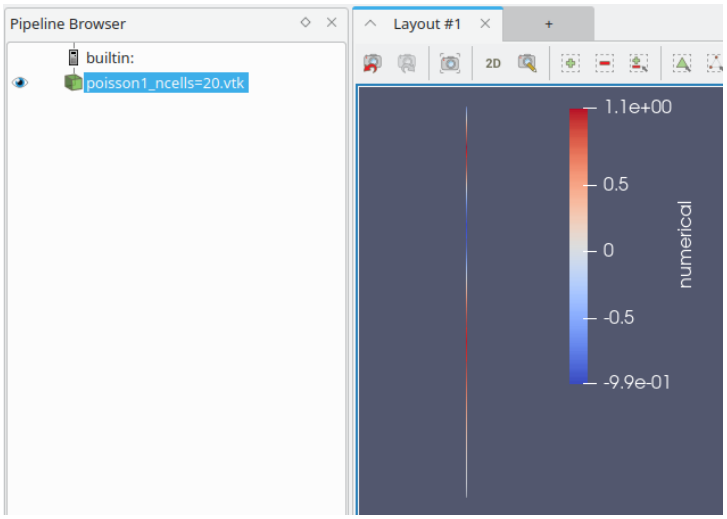
Если вы не сделали задание вовремя и решили вернутся к нему позже, то нужно

```
git checkout master            # перейти на основную ветку
git log --oneline              # в списке всех коммитов найти хэш коммита
                               # lect{index} той лекции которую нужно сделать
git checkout <...>             # переключиться на этот коммит по его хэшу
git checkout -b hw-lect{index} # создать ветку от этого коммита и работать в этой ветке
...
git commit -m "comment"        # по окончании работы скоммитить изменения
git checkout master            # и вернуться в основную ветку
```

B.6 Paraview

B.6.1 Данные на одномерных сетках

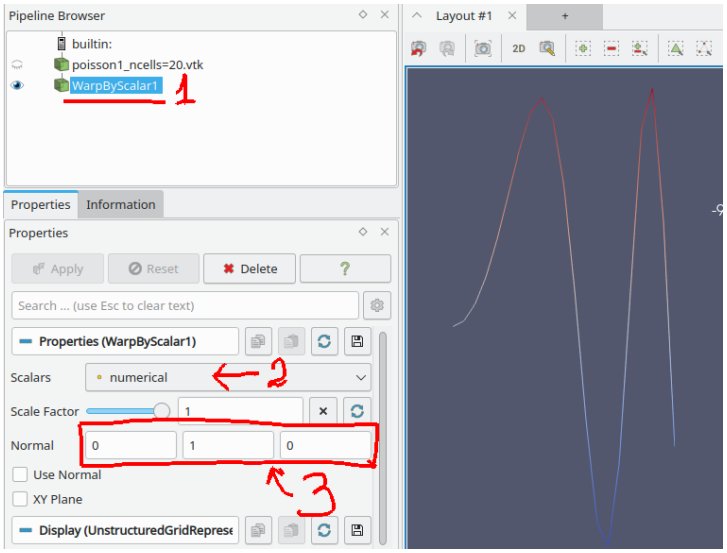
Заданные на сетке данные паравью показывает цветом. Поэтому при загрузке одномерных сеток можно видеть картинку типа



Развернуть изображение в плоскость xy

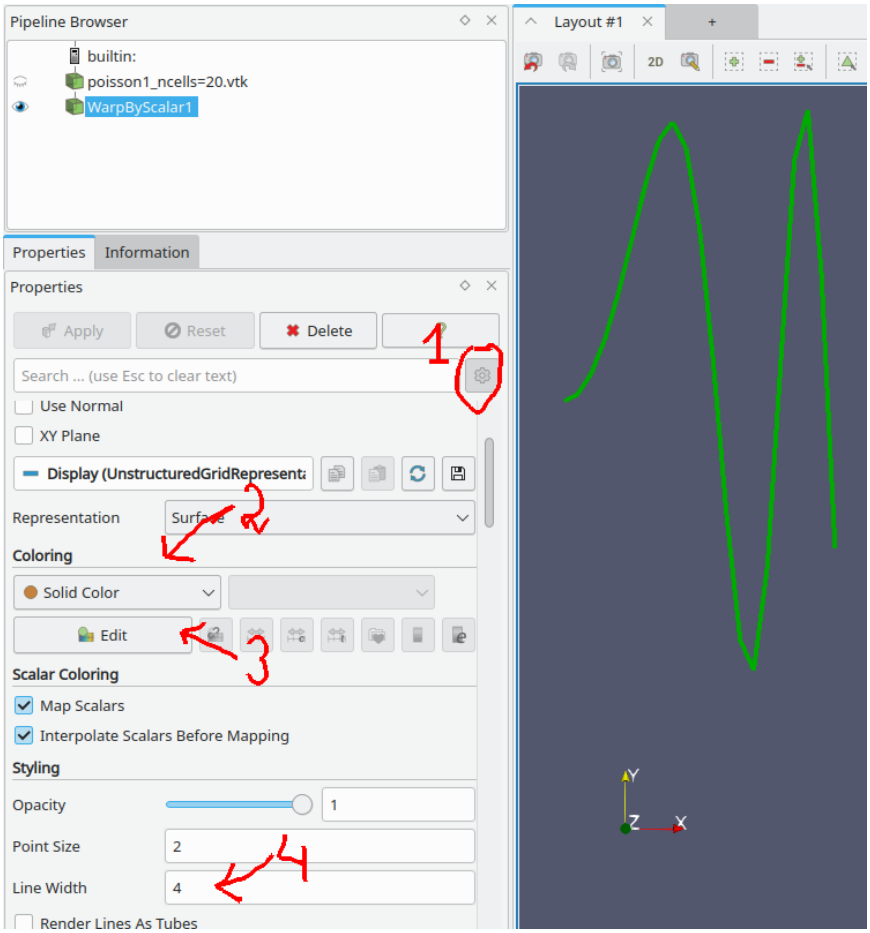
Отобразить данные в виде y-координаты Для того, что бы данные отображались в качестве значения по оси ординат, к загруженному файлу необходимо

1. применить фильтр `WarpByScalar` (В меню `Filters->Alphabetical->Warp By Scalar`)
2. в меню настройки фильтра указать поле данных, для отображения (numerical в примере ниже)
3. И настроить нормаль, вдоль которой будут проецироваться данные (в нашем случае ось y)



Цвет и толщина линии

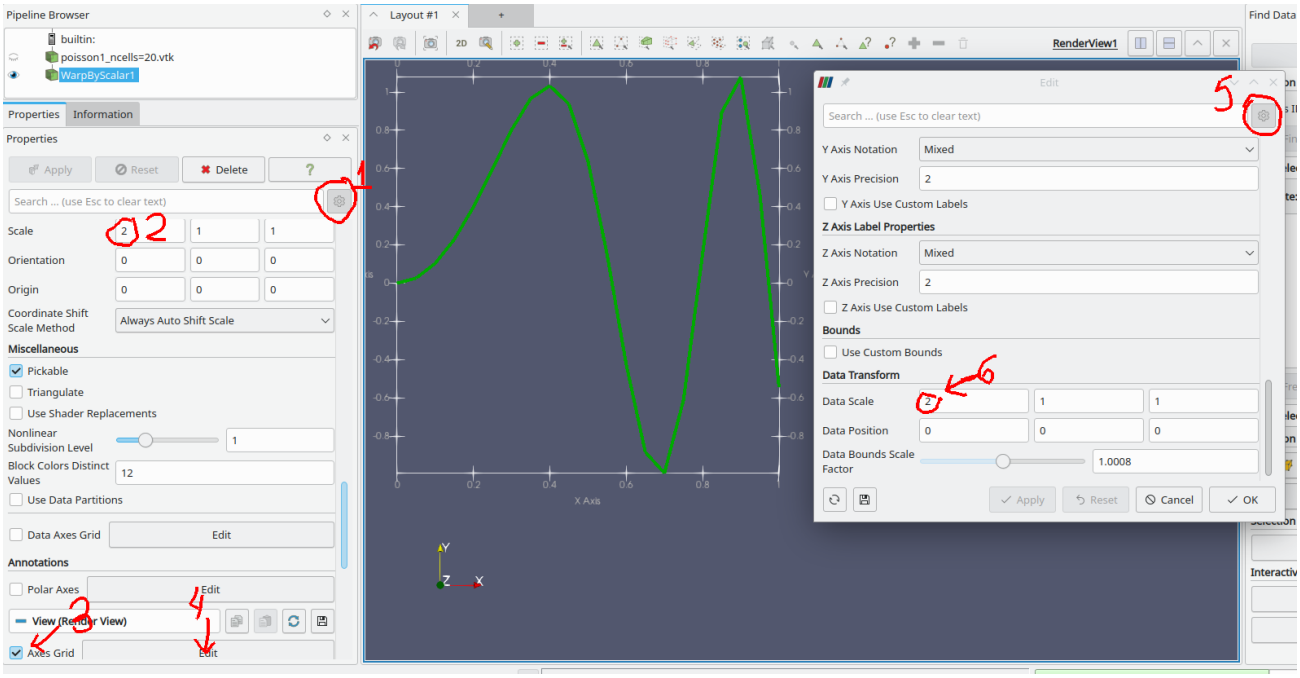
1. Включить подробные опции фильтра
2. Сменить стиль на `Solid Color`
3. В меню `Edit` выбрать желаемый цвет
4. В строке `Line Width` указать толщину линии



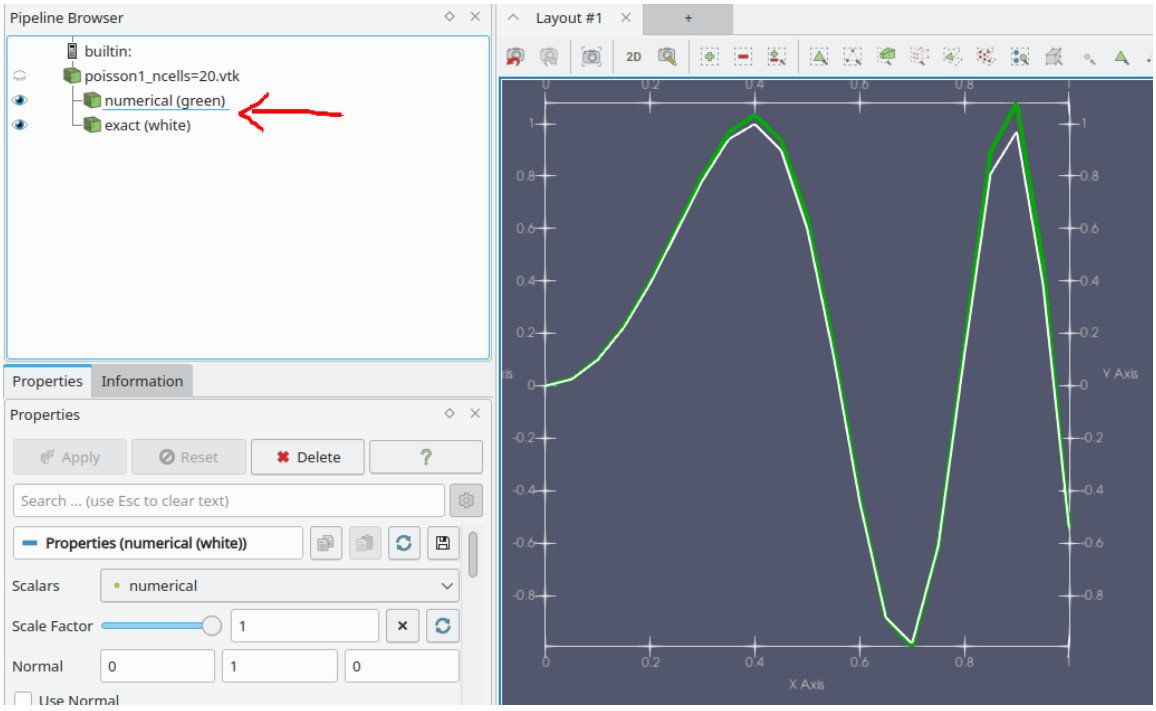
Настрока масштабов и отображение осей координат

1. Отметье подробные настройки фильтра
2. В поле `Transforming/Scale` Установите желаемые масштабы (в нашем случае растянуть в два раза по оси x)
3. Установите галку на отображение осей
4. откройте меню натройки осей
5. В нём включите подробные настроки
6. И также поставьте растяжение осей

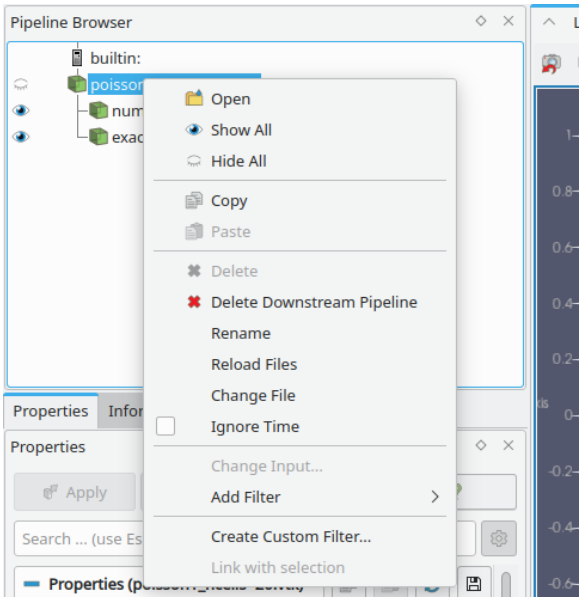
В случае, если масштабировать график не нужно, достаточно выполнить шаг 3.



Построение графиков для нескольких данных Если требуется нарисовать рядом несколько графиков для разных данных из одного файла, примените фильтр `Warp By Scalar` для этого файла ещё раз, изменив поле `Scalars` в настройке фильтра. Для наглядности измените имя узла в Pipeline Browser на осмысленные



Обновление данных при изменении исходного файла В случае, если исходный файл был изменён, нужно в контекстном меню узла соответствующего файла выбрать `Reload Files` (или нажать F5). Если те же самые фильтры нужно применить для просмотра другого файла нужно в этом меню нажать `Change File`.



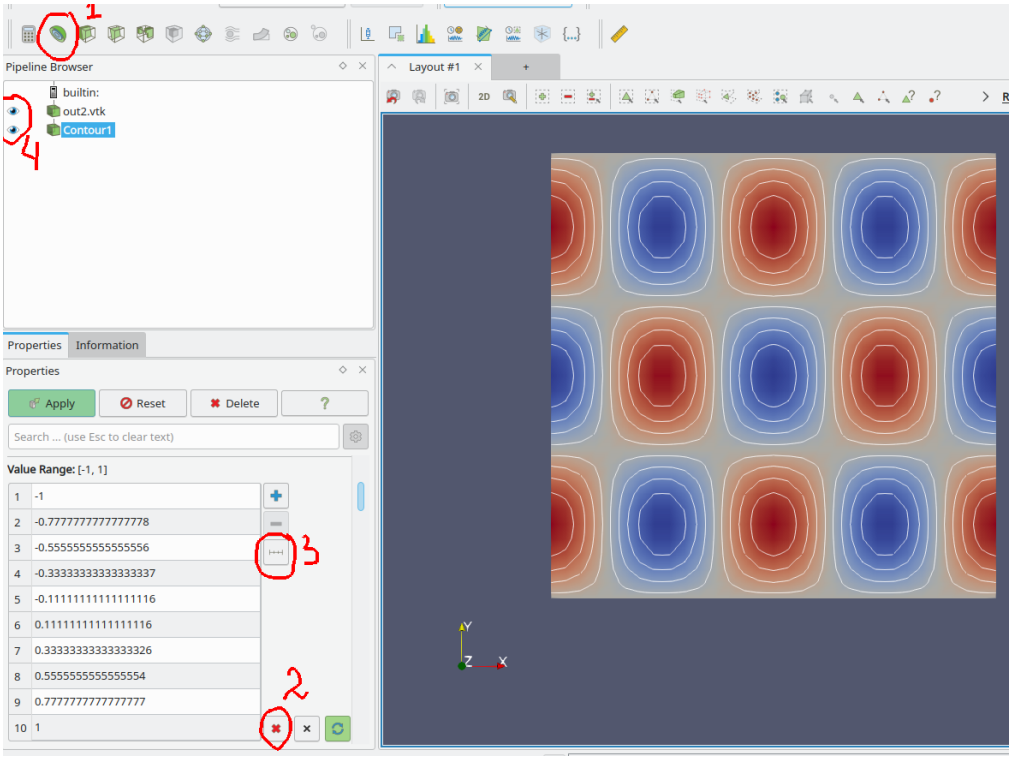
B.6.2 Изолинии для двумерного поля

1. Нажмите иконку `Contour` (или `Filters/Contour`) В настройках фильтра Contour by выберитее данные, по которым нужно строить изолинии.

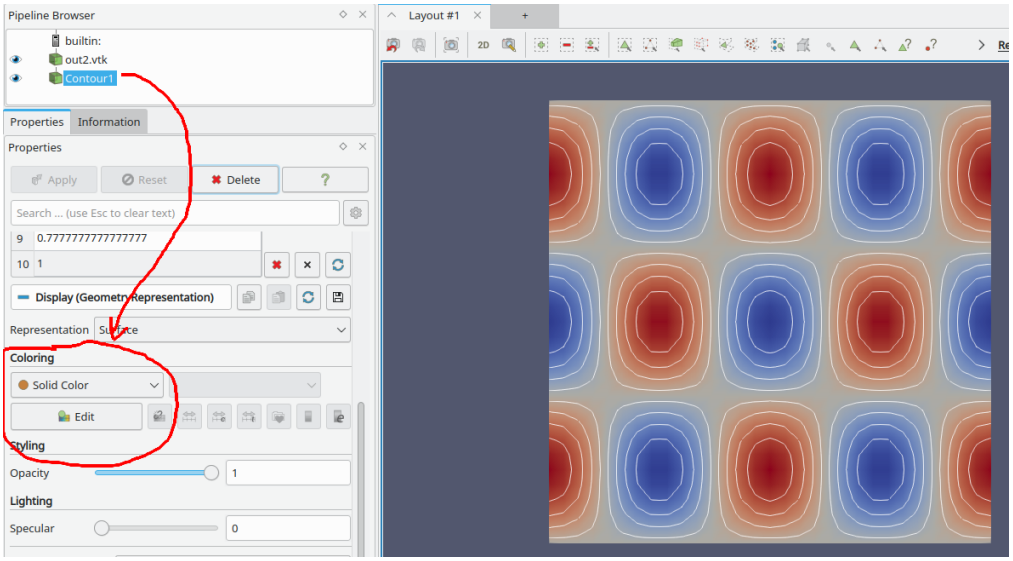
2. В настройках фильтра удалите все существующие записи о значениях для изолиний

3. Добавьте равномерные значения. В появившемся меню установите необходимое количество изолиний и их диапазон.

4. Если необходимо, включите одновременное отображения цветного поля и изолиний.



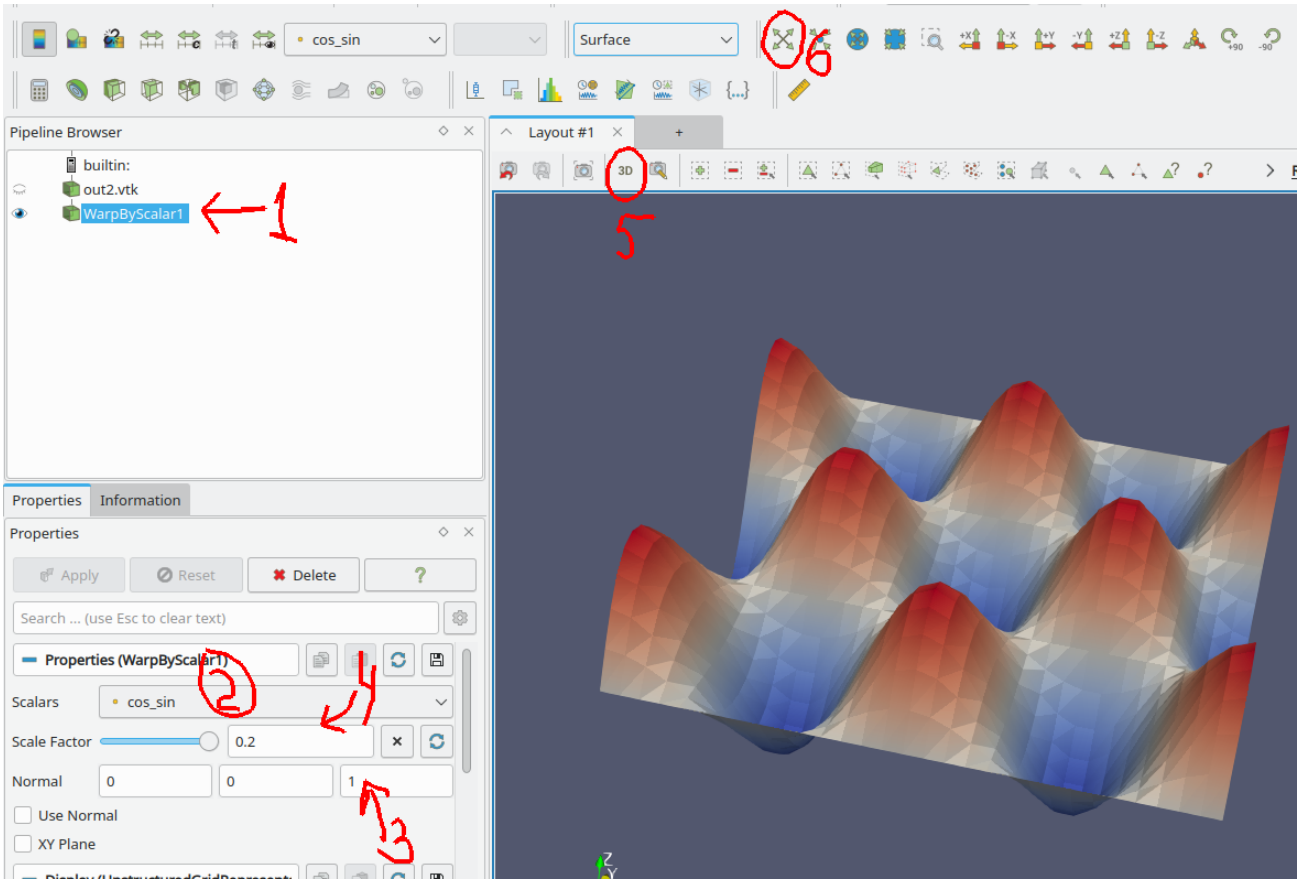
Задание цвета и толщины изолинии В случае, если нужно сделать изолинии одного цвета, установите поле `Coloring/Solid color` в настройках фильтра. Там же в меню `Edit` можно выбрать цвет. Для установления толщины линии включите подробные настройки и найдите там опцию `Styling/Line Width`.



B.6.3 Данные на двумерных сетках в виде поверхности

По аналогии с одномерным графиком (п. B.6.1), двумерные поля так же можно отобразить, проектируя данные на геометрическую координату для получения объёмного графика. Для этого

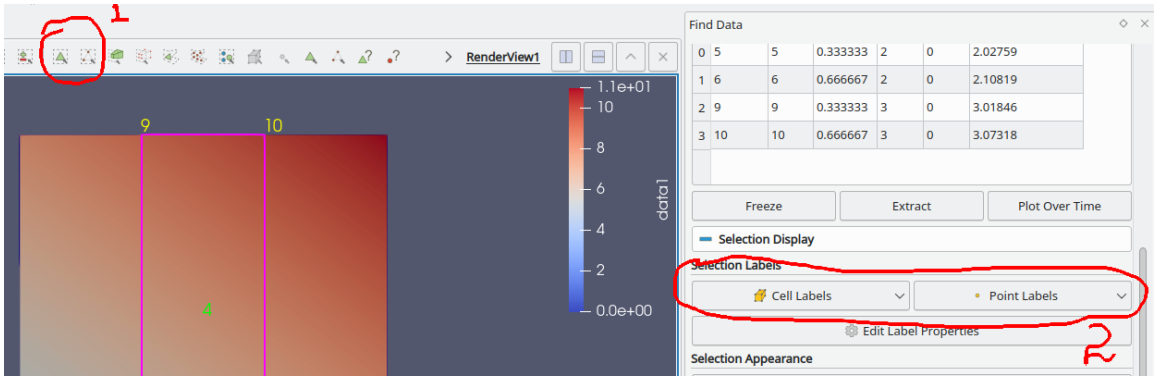
1. Включите фильтр `Filters/Warp By Scalar`
2. В настройках фильтра установите данные, которые будут проектироваться на координату z
3. Установите нормаль для проецирования (ось z)
4. Если нужно, выберите масштабирования для этой координаты
5. После нажатия `Apply` включите трёхмерное отображение
6. Если данные не видно, обновите экран.



B.6.4 Числовых значения в точках и ячейках

Иногда в процессе отладки или анализа результатов расчёта требуется знать точное значение поля в заданном узле или ячейке сетки. Для этого

1. Включить режим выделения точек или ячеек (иконка (1 на рисунке) или горячие клавиши `s`, `d`). Выделить мышкой интересующую область
2. В окне `Find data` (или `Selection Inspector` для старых версий Paraview) отметить поле, которое должно отображаться в центрах ячеек и в точках (2 на рисунке). Если такого окна нет, включить его из основного меню `View`.



B.6.5 Векторные поля

Открыть файл vtk или vtk.series, который содержит векторное поле. Далее

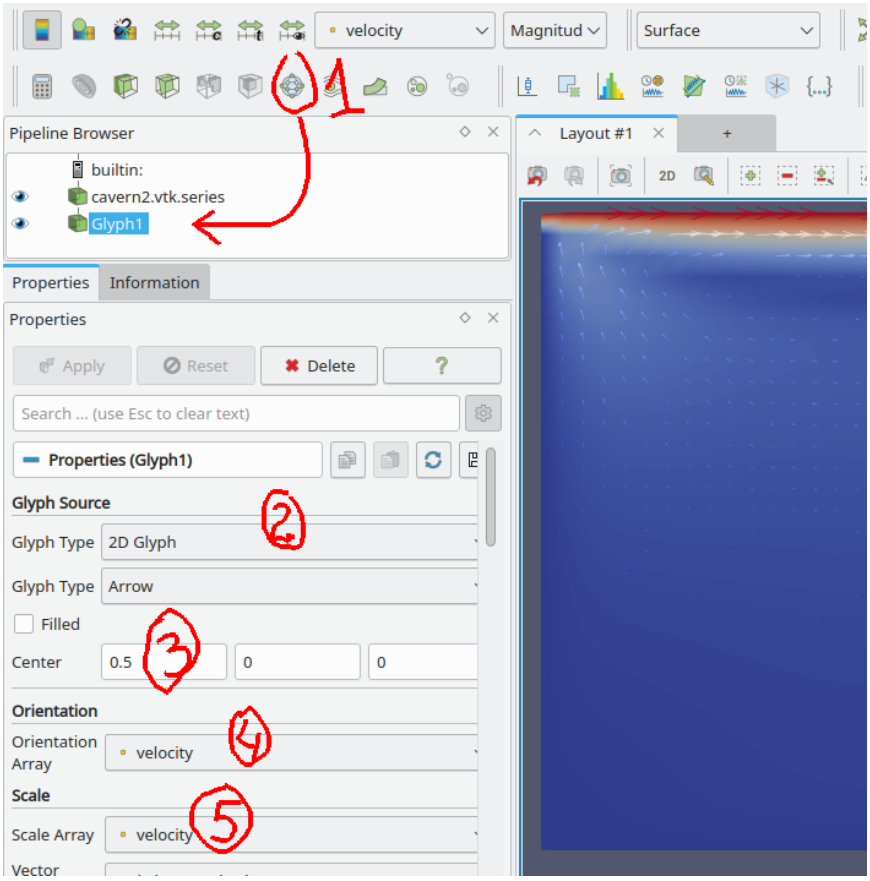
1. Создать фильтр `Glyph`

2. Задать двумерный тип стрелки

3. Сместить центр стрелки, чтобы она исходила из точки, к которой приписана

4. Отметить необходимое векторное поле в качестве ориентации

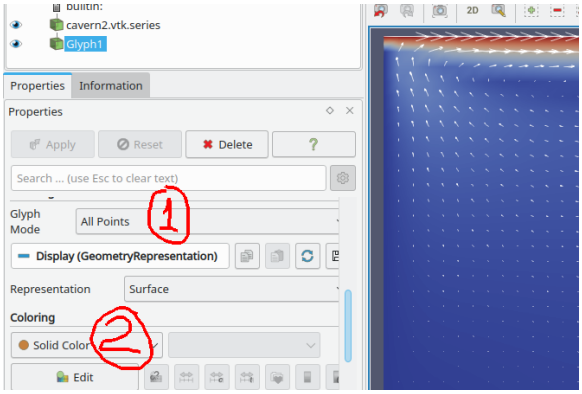
5. Отметить необходимое векторное поле для масштабирования Нажать `Apply`.



Настройка отображения стрелок

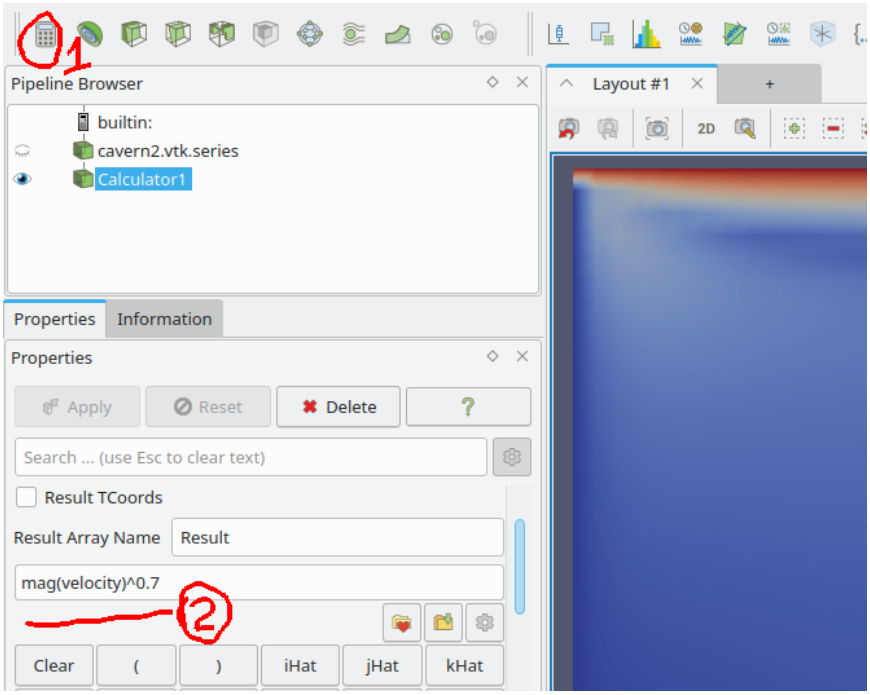
1. Выбрать необходимый `Glyph-mode`. Если сетка небольшая, то можно `All Points`.

2. Установить белый цвет для стрелок. Нажать Apply.



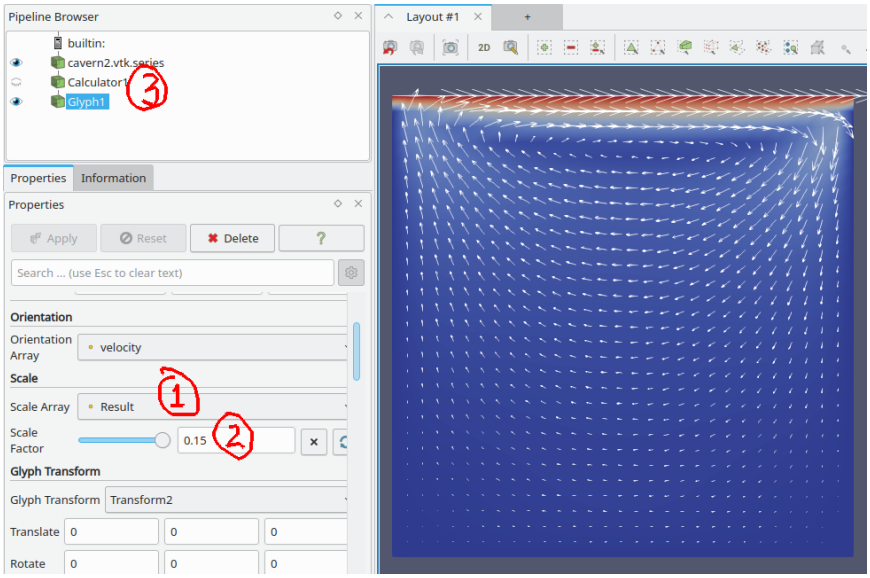
Уменьшения разброса по длине стрелок Если разброс по длинам стрелок слишком велик, его можно подравнять, введя новую функцию $|\mathbf{v}|^\alpha$ – длина вектора в степени меньше единицы (например, $\alpha = 0.7$). Такую функцию можно создать через калькулятор

1. Начиная от загруженного файла создать фильтр `Calculator`
2. Там вбить необходимую формулу



Созданную функцию нужно прокинуть в `Glyph` в качестве коэффициента масштабирования

1. В `Scale Array` фильтра `Glyph` указать уже результат работы `Calculator`-а (`Result` по умолчанию),
2. Подтянуть значение `Scale Factor` до приемлимого
3. Не забыть отключить вспомогательное поле `Calculator` из отображения



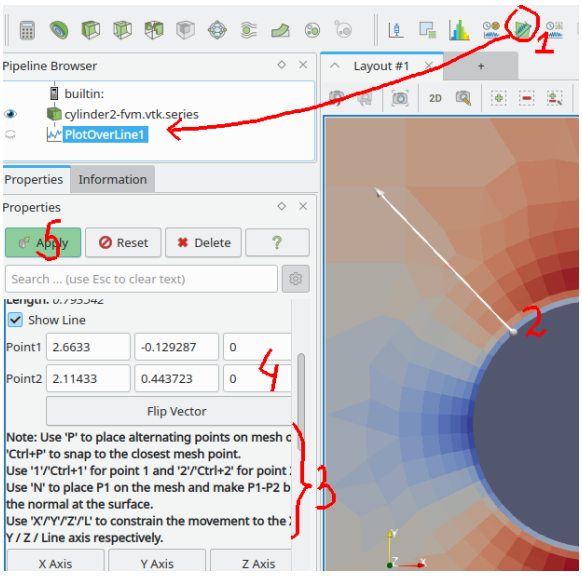
B.6.6 Значение функции вдоль линии

1. Выбрать фильтр `Plot Over Line` иконкой или в меню `Filters`
2. Установить начальную и конечную точку сечения

3. Можно использовать привязку к узлам сетки с помощью горячих клавиш (в подсказках написано)

4. Можно установить координаты руками в соответствующем поле. Для двумерных задач проследить, что координата Z равна нулю

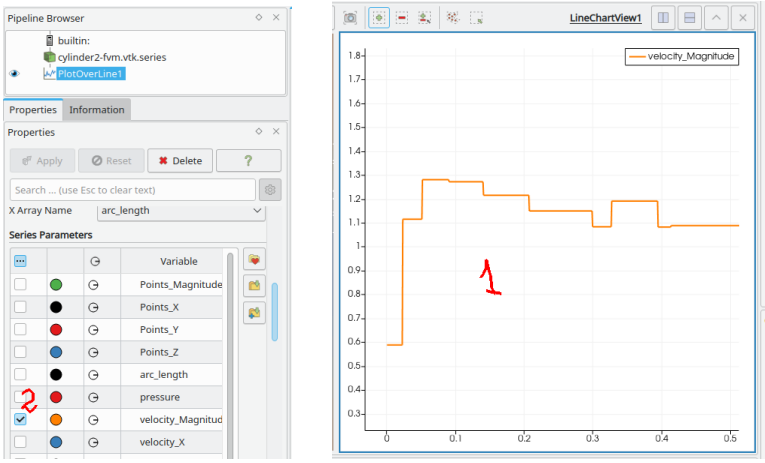
5. Нажать `Apply`



Настройка графика

1. После установок появится дополнительное окно типа `Line Chart View` с нарисованным графиком.

2. Сделав это окно активным в настройках фильтра `PlotOverLine` можно выбрать, какие поля рисовать (`Series Parameters`)

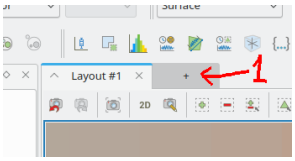


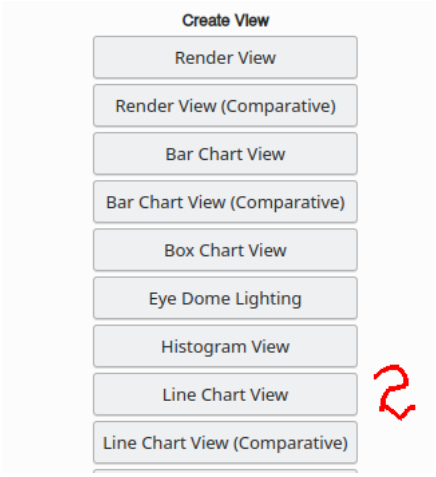
Отрисовка в отдельном окне

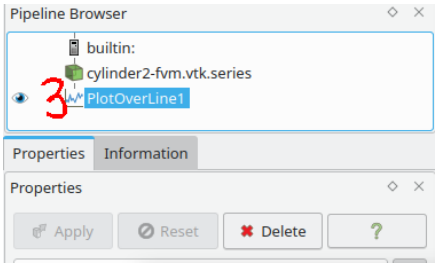
1. Открыть новую вкладку

2. Выбрать `Line Chart View`

3. Выбрать предварительно созданный фильтр с одномерным графиком


Layout #1
1


Create View
Render View
Render View (Comparative)
Bar Chart View
Bar Chart View (Comparative)
Box Chart View
Eye Dome Lighting
Histogram View
Line Chart View
Line Chart View (Comparative)
2


Pipeline Browser
builtin:
cylinder2-fvm.vtk.series
PlotOverLine1
3
Properties
Information
Properties
Apply
Reset
Delete

B.7 Hybmesh

Генератор сеток на основе композитного подхода. Работает на основе python-скрипотов. Полная документация http://kalininei.github.io/HybMesh/index.html

B.7.1 Работа в Windows

Инсталлятор программы следует скачать по ссылке https://github.com/kalininei/HybMesh/releases и установить стандартным образом.

Для запуска скрипта построения `script.py` нужно открыть консоль, перейти в папку с нужным скриптом, оттуда выполнить (при условии, что программа была установлена в папку `C:\Program Files`):

```
> "C:\Program Files\HybMesh\bin\hybmesh.exe" -sx script.py
```

B.7.2 Работа в Linux

Версию для линукса нужно собирать из исходников. Либо, если собрать не получилось, можно строить сетки в Windows и переносить полученные vtk-файлы на рабочую систему.

Перед сборкой в систему необходимо установить dev-версии пакетов `suitesparse` и `libxml2`. Также должны быть доступны компилляторы
`gcc-c++` и `gcc-fortan` и `cmake`. Программа работает со скиптами python2. Лучше установить среду anaconda (https://docs.anaconda.com/free/anaconda/install/index.html) И в ней создать окружение с python-2.7:

```
> conda create -n py27 python=2.7   # создать среду с именем py27
> conda activate py27               # активировать среду py27
> pip install decorator             # установить пакет decorator
```

Сначала следует склонировать репозиторий в папку с репозиториями гита:

```
> cd D:/git_repos
> git clone https://github.com/kalininei/HybMesh
```

Поскольку программа не предназначена для запуска из под анаконды, в сборочные скрипты нужно внести некоторые изменения. В корневом сборочном файле `HybMesh/CMakeLists.txt` нужно закомментировать все строки в диапазоне

```
# ========================== Python check
....
# ========================== Windows installer options
```

а в файле `HybMesh/src/CMakeLists.txt` последнюю строку

```
#add_subdirectory(bindings)
```

Далее, находясь в корневой директории репозитория HybMesh, запустить сборку

```
> mkdir build
> cd build
> cmake .. -DCMAKE_BUILD_TYPE=Release
> make -j8
> sudo make install
```

Для запуска скриптов нужно создать скрипт-прокладку

```
import sys
sys.path.append("/path/to/HybMesh/src/py/")  # вставить полный путь к Hybmesh/src/py
execfile(sys.argv[1])
```

и сохранить его в любое место. Например в `path/to/HybMesh/hybmesh.py`.

Для запуска скрипта построения сетки следует перейти в папку, где находится нужный скрипт `script.py`, убедится, что анаконда работает в нужной среде (то есть `conda activate py27` был вызван), и запустить

```
> python /path/to/HybMesh/hybmesh.py script.py
```