

Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра математического и компьютерного моделирования

**МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ ТЕЧЕНИЙ
МЕЛКОЙ ВОДЫ МЕТОДОМ КОНЕЧНЫХ ЭЛЕМЕНТОВ**

Бакалаврская работа

студента 4 курса 413 группы

направление 01.03.02 - Прикладная математика и информатика

механико-математического факультета

Шарова Александра Вадимовича

Научный руководитель
Старший преподаватель

В.С. Кожанов

Зав. кафедрой
доцент, д.ф. – м. н.

Ю.А. Блинков

Саратов 2018

СОДЕРЖАНИЕ

Стр.

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ	4
ВВЕДЕНИЕ	5
1 Вывод уравнений мелкой воды.....	7
2 Постановка задачи	15
3 Триангуляция двумерной области	17
3.1 Алгоритм Рапперта [Jim Ruppert].....	18
3.2 Алгоритм Чу [L. Paul Chew]	21
4 Метод конечных элементов в двумерной области.....	22
4.1 Метод взвешенных невязок	23
4.2 Метод конечных элементов	26
4.3 МКЭ для рассматриваемой задачи	28
4.3.1 Интегрирование функции двух переменных по произ- вольному треугольнику	28
4.3.2 Уравнения мелкой воды в терминах МКЭ.....	29
5 Решение задачи.....	31
5.1 Построение графиков	31
5.2 Изменение формы водоёма.....	31
5.2.1 Пруд.....	32
5.2.2 Реальный водоём.....	35
5.3 Учёт наличия острова (островов) внутри водоёма	37
5.3.1 Пруд.....	37
5.3.2 Реальный водоём.....	40
5.4 Программная реализация	41
6 Разработка базы данных для хранения информации о про- ведённых экспериментах	42
6.1 Описание базы данных	42
6.2 Формат представления данных в БД	44
6.3 Примеры запросов к БД	47

ЗАКЛЮЧЕНИЕ	48
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	49
ПРИЛОЖЕНИЕ А Исходный код реализации	51
ПРИЛОЖЕНИЕ Б Исходный код Docker контейнера	52
ПРИЛОЖЕНИЕ В Выкладки для МКЭ	53

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

1. МКЭ - метод конечных элементов
2. КЭ - конечный элемент
3. МВ - мелкая вода
4. ГЖС - газожидкостная смесь
5. POLY - тип файла, в котором содержится список вершин и сегментов. Помимо этого, данный файл может содержать в себе информацию о пустотах и вогнутостях, а также некоторую дополнительную информацию [1]
6. СУБД - система управления базами данных
7. БД - база данных
8. NoSQL - термин, обозначающий ряд подходов, направленных на реализацию хранилищ баз данных, имеющих существенные отличия от моделей, используемых в традиционных реляционных СУБД с доступом к данным средствами языка SQL
9. Oracle RDBMS - объектно-реляционная система управления базами данных компании Oracle
10. Нода кластера - один компьютер в группе, объединённой высокоскоростными каналами связи и представляющей с точки зрения пользователя единый аппаратный ресурс
11. Docker - программное обеспечение для автоматизации развёртывания и управления приложениями в среде виртуализации на уровне операционной системы

ВВЕДЕНИЕ

Для корректного построения математических моделей и численных методов расчета течения жидкости важно знать, каким именно образом описываются основные процессы, происходящие в водоеме. Для этого строятся модели, которые учитывают основные характеристики течения в природной среде.

На данный момент во многих случаях не оправдывается применение более сложных математических моделей для исследования течений в прибрежных водах и озерах, чем модели, полученные путем применения осредненных по вертикали характеристик и основанные на численном решении двумерных уравнений - уравнений мелкой воды.

Уравнения мелкой воды - широко известное приближение, на основе которого проводится численное моделирование течений в реках и водоемах, в прибрежных зонах морей и океанов. Трехмерные решения данных уравнений являются нецелесообразными, так как они требуют намного большего количества исходной информации и машинного времени даже с учетом современных вычислительных мощностей.

При выводе данных уравнений предполагается, что среда представляет собой достаточно тонкий слой, глубина которого много меньше его продольного размера, поэтому вертикальной составляющей скорости можно пренебречь и полагать, что продольные скорости постоянны по толщине слоя. Исходя из этого данное приближение успешно применяется для описания течений, где влияние на поток свободной поверхности и рельефа дна значительно.

Целью представленной выпускной квалификационной работы является построение и численная реализация математической модели на основе двумерных уравнений мелкой воды с помощью метода конечных элементов, который уже давно зарекомендовал [2] себя в таких областях, как механика деформируемого тела, электродинамика и, конечно же, гидродинамика. Данный метод является оптимальным для решения поставленной задачи, так как именно он дает возможность применять достаточно гибкую разбивку рассматриваемой области и при его использовании достаточно удовлетворить лишь главным граничным условиям. Для этой задачи требуется составить

и решить систему дифференциальных уравнений с использованием метода конечных элементов. Предполагается, что глубина водоема постоянна, а сам водоем является однородным, то есть не рассматривается случай ГЖС. Необходимо учесть, что озеро подвержено влиянию ветра, а также требуется рассмотреть данную задачу с различными начальными параметрами и проанализировать полученные результаты. Похожая задача для уже решалась ранее, но рассматривался либо только стационарный случай [3], либо использовались другие методы [4].

Актуальность данной работы обусловлена тем, что сейчас есть потребность в более точном и быстром решении уже решенных физических задач. Этой потребности отвечает появление новых быстродействующих систем и методов решения. Так, в данной дипломной работе будет разработана параллельная реализация МКЭ для уравнений МВ, которая позволит более эффективно исследовать различные течения, которые могут быть описаны с помощью уравнений МВ.

В данной работе будут представлены основные определения и понятия для уравнений мелкой воды, которые позволяют составить и решить поставленную задачу, состоящую из двух частей: аналитической и численной. Аналитически будут описаны течения жидкости при пренебрежении температурными эффектами, для которых широко используются классические уравнения Сен-Венана. После этого с помощью МКЭ будет построена и рассчитана математическая модель МВ. В качестве реализации модели будет написана программа на языке Python [5, 6], которая выдает результаты решения системы уравнений мелкой воды, а также строит графики, наглядным образом отображающие полученные результаты.

1 Вывод уравнений мелкой воды

Запишем два основных уравнения для жидкости - уравнение количества движения и уравнение неразрывности [7]:

$$-\frac{\partial p}{\partial x_k} + \frac{\partial \tau_{ik}}{\partial x_i} + \rho b_k = \frac{\partial}{\partial x_i}(\rho v_i v_k) + \frac{\partial}{\partial t}(\rho v_k); \quad (1.1)$$

$$\frac{D\rho}{Dt} + \rho \frac{\partial v_i}{\partial x_i} = 0 \quad (1.2)$$

Если в 1.1 и 1.2 пренебречь температурными эффектами, получим:

$$-\frac{\partial p}{\partial x_k} + \frac{\partial \tau_{ik}}{\partial x_i} + \rho b_k = \frac{D(\rho v_k)}{Dt} \quad (1.3)$$

$$\frac{\partial(\rho v_i)}{\partial t} + \frac{\partial \rho}{\partial t} = 0, \quad (1.4)$$

где p - давление, оказываемое на единицу площади поверхности воды, b_k - массовые силы, приходящиеся на единицу массы, v_k - скорость частицы воды в направлении оси x_k , τ_{ik} - вязкостные составляющие вектора напряжения, ρ - массовая плотность воды.

При математическом моделировании движения мелкой воды сложно применять данные уравнения вследствие наличия свободной поверхности, изменения границ во время приливов и отливов и большого количества переменных.

Данных трудностей можно избежать после ряда упрощений, в результате чего получаются уравнения мелкой воды. Первое упрощение состоит в том, что уравнение количества движения в проекции на ось x_3 записывается в виде:

$$-\frac{\partial p}{\partial x_3} = \rho g, \quad (1.5)$$

где массовые силы отрицательны, поскольку действуют в направлении, противоположном оси x_3 . При выводе формулы 1.5 пренебрегаем всеми членами,

которые характеризуют ускорение, и соответствующими им напряжениями. Проинтегрируем выражение 1.5 и в результате получим:

$$p = \int_{x_3}^{\eta} \rho g dx_3 = \rho g(\eta - x_3) + p_a, \quad (1.6)$$

где p_a - атмосферное давление на поверхности воды, η - возвышение свободной поверхности в соответствии с рисунком 1.1.

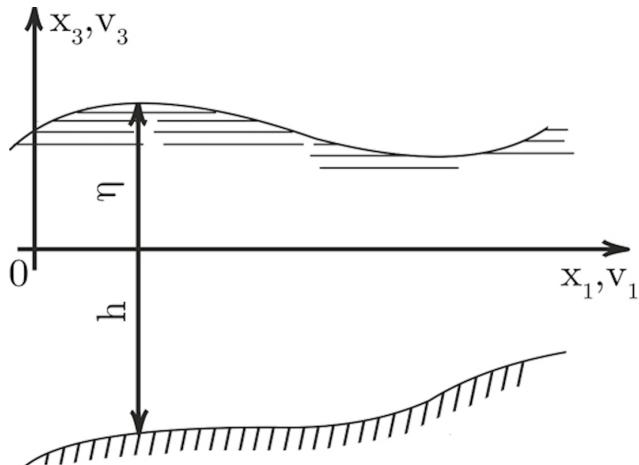


Рисунок 1.1

Оставшиеся два уравнения количества движения по направлениям x_1 и x_2 останутся без изменений:

$$-\frac{\partial p}{\partial x_k} + \frac{\partial \tau_{ik}}{\partial x_i} + \rho b_k = \frac{D(\rho v_k)}{Dt}. \quad (1.7)$$

При $k = 1$ выражение 1.7 дает проекцию на ось x_1 , при $k = 2$ - на ось x_2 . В выражении 1.7 величина v - средняя скорость, ρ - переменная массовая плотность и τ - сумма вязкостных и турбулентных напряжений.

Проинтегрируем выражения 1.4 и 1.7 по x_3 . Для уравнения неразрывности это даст:

$$\int_{-h}^{\eta} \left(\frac{\partial(\rho v_i)}{\partial x_i} + \frac{\partial \rho}{\partial t} \right) dx_3 = 0, \quad (1.8)$$

где h - это глубина, измеряемая от базовой поверхности (в общем случае не горизонтальной).

Определим поток q_k количества жидкости (массу жидкости, приходящуюся на единицу длины и времени):

$$q_i = \int_{-h}^{\eta} \rho v_i dx_3 = \rho \int_{-h}^{\eta} v_i dx_3. \quad (1.9)$$

Предполагается, что $\rho(x_1, x_2)$ не зависит от x_3 .

При интегрировании уравнения 1.8 необходимо использовать кинематическое условие и правило Лейбница [8] для вычисления частной производной интеграла с переменными пределами. Согласно этому правилу:

$$\frac{\partial}{\partial x_1} \int_{h_1(x_1, x_2)}^{h_2(x_1, x_2)} f(x_1, x_2, x_3) dx_3 = \int_{h_1(x_1, x_2)}^{h_2(x_1, x_2)} \frac{\partial f}{\partial x_1} dx_3 + f \Big|_{h_2} \frac{\partial h_2}{\partial x_1} + f \Big|_{h_1} \frac{\partial h_2}{\partial x_1}. \quad (1.10)$$

Аналогично для производной по x_2 .

Кинематическое соотношение для свободной поверхности можно записать следующим образом:

$$v_3 \Big|_{x_2=\eta} = \frac{D\eta}{Dt} = \frac{\partial \eta}{\partial t} + v_1 \Big|_{\eta} \frac{\partial \eta}{\partial x_1} + v_2 \Big|_{\eta} \frac{\partial \eta}{\partial x_2}. \quad (1.11)$$

Применим формулы 1.9-1.11 к уравнению 1.8 и получим:

$$\frac{\partial q_i}{\partial x_i} + \frac{\partial(\rho H)}{\partial t} = 0, \quad (1.12)$$

где $H = \eta + h$.

Чтобы проинтегрировать уравнение количества движения 1.7 по x_3 , определим мгновенные скорости v_1, v_2 :

$$\begin{aligned} v_1 &= \bar{v}_1(x_1, x_2, t) + v'_1(x_1, x_2, x_3, t); \\ v_2 &= \bar{v}_2(x_1, x_2, t) + v'_2(x_1, x_2, x_3, t), \end{aligned} \quad (1.13)$$

где величина \bar{v} означает средние по вертикали скорости, а v' - отклонение от этих средних значений при различных значениях x_3 .

Следовательно:

$$\langle v_k \rangle = \int_{-h}^{\eta} v_k dx_3 = \frac{1}{\rho} q_k, \quad v_k = \frac{1}{H} \langle v_k \rangle, \quad (1.14)$$

так как $\langle v'_k \rangle = 0$, где знак $\langle \rangle$ означает среднее значение стоящее внутри величины.

Будем предполагать, что массовые силы обусловлены только эффектом Кориолиса. Таким образом получаем:

$$b_1 = \rho f v_2, \quad b_2 = -\rho f v_1. \quad (1.15)$$

Предположим, что наклоны поверхности и дна малы по сравнению с единицей, тогда составляющие внутреннего напряжения можно аппроксимировать следующим образом (в соответствии с рисунком 1.2):

$$\begin{aligned} \tau_1|_s &\approx \left\{ -\tau_{11} \frac{\partial \theta}{\partial x_1} - \tau_{12} \frac{\partial \theta}{\partial x_2} + \tau_{13} \right\}, \\ \tau_1|_b &\approx \left\{ \tau_{11} \frac{\partial h}{\partial x_1} + \tau_{12} \frac{\partial h}{\partial x_2} - \tau_{13} \right\}. \end{aligned} \quad (1.16)$$

Аналогичные значения можно выписать для величин $\tau_2|_s$ и $\tau_2|_b$.

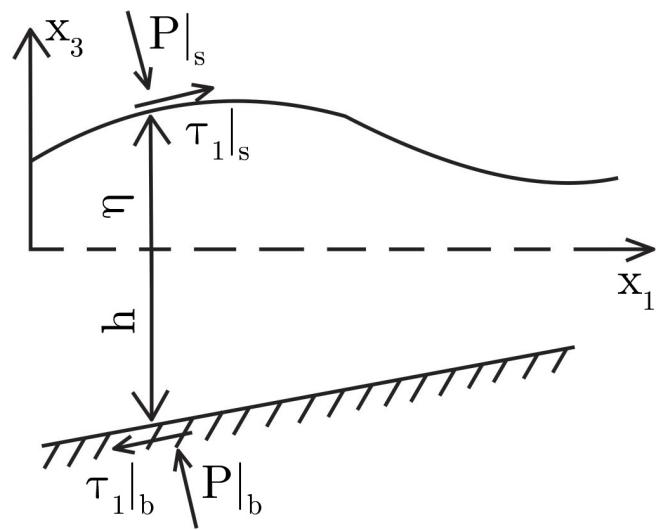


Рисунок 1.2

Величины $\tau_2|_s$ и $\tau_2|_b$ можно интерпретировать как компоненты внешней силы, приложенные к поверхности и дну.

Подставим теперь соотношения 1.13 - 1.15 в уравнения количества движения, проинтегрированные по x_3 , и дополнительно используем правило Лейбница и кинематическое условие. Тогда:

$$\begin{aligned} \frac{\partial q_1}{\partial t} + \frac{\partial}{\partial x_1} \left(\frac{q_1^2}{H} \right) + \frac{\partial}{\partial x_2} \left(\frac{q_1 q_2}{H} \right) &= -\frac{\partial N_p}{\partial x_2} + \frac{\partial N_{11}}{\partial x_1} + \frac{\partial N_{12}}{\partial x_2} \\ &\quad + f q_2 + p \left| \frac{\partial \eta}{\partial x_1} + \tau_1 \right|_s + p \left| \frac{\partial h}{\partial x_1} - \tau_1 \right|_b; \\ \frac{\partial q_2}{\partial t} + \frac{\partial}{\partial x_1} \left(\frac{q_1 q_2}{H} \right) + \frac{\partial}{\partial x_2} \left(\frac{q_2^2}{H} \right) &= -\frac{\partial N_p}{\partial x_1} + \frac{\partial N_{22}}{\partial x_2} + \frac{\partial N_{21}}{\partial x_1} \\ &\quad + f q_1 + p \left| \frac{\partial \eta}{\partial x_2} + \tau_2 \right|_s + p \left| \frac{\partial h}{\partial x_2} - \tau_2 \right|_b, \end{aligned} \quad (1.17)$$

где

$$\begin{aligned} N_p &= \langle p \rangle = \int_{-h}^{\eta} pdx_3 = \rho g \frac{H^2}{2} + Hp_a; \\ N_{11} &= \langle \tau_{11} \rangle - \langle pv'_1 v'_1 \rangle; \\ N_{22} &= \langle \tau_{22} \rangle - \langle pv'_2 v'_2 \rangle; \\ N_{12} &= \langle \tau_{12} \rangle - \langle pv'_1 v'_2 \rangle; \end{aligned} \quad (1.18)$$

Более того, элементы N_{ik} могут быть аппроксимированы следующими выражениями:

$$\begin{aligned} N_{11} &\approx 2\varepsilon_{11} \frac{\partial q_1}{\partial x_1}; \\ N_{22} &\approx 2\varepsilon_{22} \frac{\partial q_2}{\partial x_2}; \\ N_{12} &\approx \varepsilon_{12} \left(\frac{\partial q_2}{\partial x_1} + \frac{\partial q_1}{\partial x_2} \right), \end{aligned} \quad (1.19)$$

где ε_{ik} обобщенные коэффициенты вихревой вязкости. Для изотропного характера течения $\varepsilon_{11} = \varepsilon_{22} = \varepsilon_{12} = \varepsilon$.

Касательные напряжения на дне обычно определяются соотношениями:

$$\begin{aligned}\tau_1 \Big|_b &= \frac{g}{c^2} \frac{1}{\rho} \frac{q_1 \sqrt{(q_1^2 + q_2^2)}}{H^2}; \\ \tau_2 \Big|_b &= \frac{g}{c^2} \frac{1}{\rho} \frac{q_2 \sqrt{(q_1^2 + q_2^2)}}{H^2},\end{aligned}\quad (1.20)$$

где g - ускорение силы тяжести, c - коэффициент трения, ρ - плотность воды.

Составляющие напряжения трения на поверхности воды обычно обусловлены действием ветра и могут быть найдены по формулам:

$$\begin{aligned}\tau_1 \Big|_s &= \gamma^2 \rho_a W^2 \cos(\theta); \\ \tau_2 \Big|_s &= \gamma^2 \rho_a W^2 \sin(\theta),\end{aligned}\quad (1.21)$$

где γ^2 - коэффициент ветрового напряжения, ρ_a - плотность воздуха, W - скорость ветра, θ - угол между осью x_1 и направлением ветра.

Уравнения 1.17 перепишем в виде:

$$\begin{aligned}\frac{\partial q_1}{\partial t} + \frac{\partial}{\partial x_1} \left(\frac{q_1^2}{H} \right) + \frac{\partial}{\partial x_2} \left(\frac{q_1 q_2}{H} \right) &= \frac{\partial}{\partial x_1} (N_{11} - N_p) + \frac{\partial N_{12}}{\partial x_2} + B_1; \\ \frac{\partial q_2}{\partial t} + \frac{\partial}{\partial x_1} \left(\frac{q_1 q_2}{H} \right) + \frac{\partial}{\partial x_2} \left(\frac{q_2^2}{H} \right) &= \frac{\partial}{\partial x_2} (N_{22} - N_p) + \frac{\partial N_{12}}{\partial x_1} + B_2,\end{aligned}\quad (1.22)$$

где

$$\begin{aligned}B_1 &= f q_2 + \gamma^2 \rho_a W^2 \cos(\theta) - \left(\frac{g}{c^2} \right) \frac{1}{\rho} \frac{q_1 \sqrt{(q_1^2 + q_2^2)}}{H^2} + p_a \frac{\partial H}{\partial x_1} + \rho g H \frac{\partial h}{\partial x_1}; \\ B_2 &= -f q_1 + \gamma^2 \rho_a W^2 \sin(\theta) - \left(\frac{g}{c^2} \right) \frac{1}{\rho} \frac{q_2 \sqrt{(q_1^2 + q_2^2)}}{H^2} + p_a \frac{\partial H}{\partial x_2} + \rho g H \frac{\partial h}{\partial x_2}.\end{aligned}$$

Для решения окончательной системы уравнений 1.22, дополненных условием 1.12, необходимо установить требуемые граничные условия. Будем считать, что граница S состоит из двух частей: твердой границы S_1 и жидкой

S_2 , представляющей границу рассматриваемого водоема с открытым морем в соответствии с рисунком 1.3.

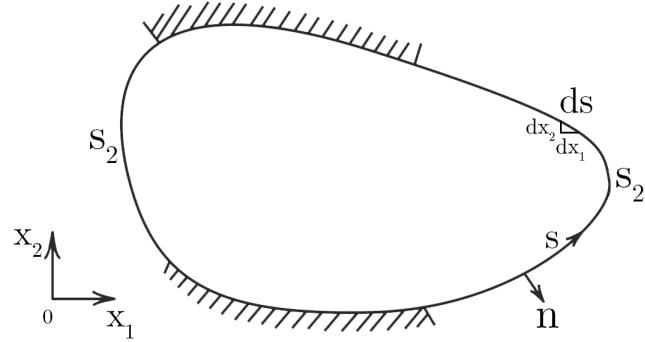


Рисунок 1.3

В системе координат $s - n$ связанной с границей течения, расход массы жидкости можно записать через q_s и q_n :

$$q_n = \int_{-h}^{\eta} \rho v_n dx_3 = \alpha_{n1} q_1 + \alpha_{n2} q_2; \\ q_s = \int_{-h}^{\eta} \rho v_s dx_3 = -\alpha_{n2} q_1 + \alpha_{n1} q_2, \quad (1.23)$$

где $\alpha_{n1} = \cos(\bar{n}, x_1); \alpha_{n2} = \cos(\bar{n}, x_2)$.

Для установления результирующих сил можно воспользоваться формулами:

$$N_{n1} = \alpha_{n1}(N_{11} - N_p) + \alpha_{n2}N_{12}; \\ N_{n2} = \alpha_{n1}N_{12} + \alpha_{n2}(N_{22} - N_p). \quad (1.24)$$

По значениям N_{n1} и N_{n2} определяется нормальная и касательная составляющие результирующей силы для наклонной площадки:

$$\begin{aligned} N_{nn} &= \alpha_{n1}N_{n1} + \alpha_{n2}N_{n2}; \\ N_{ns} &= \alpha_{n2}N_{n1} + \alpha_{n1}N_{n2}. \end{aligned} \quad (1.25)$$

Если же в исследуемую акваторию впадает река, то на S_1 :

$$\begin{aligned} q_n &= \overline{q_n} = |q|; \\ q_s &= 0, \end{aligned} \quad (1.26)$$

где $|q|$ - поток втекающей реки.

На жидкой границе S_2 необходимо задать нормальные и касательные силы:

$$\begin{aligned} N_{nn} &= \overline{N_{nn}}; \\ N_{ns} &= \overline{N_{ns}}. \end{aligned} \quad (1.27)$$

Но так как слагаемыми, учитывающими вихревую вязкость в уравнении 1.22 можно пренебречь, то касательные силы или скорости не могут быть заданы. Таким образом, граничные условия сводятся к следующим:

$$\begin{aligned} q_n &= 0 \text{ или } q_n = \overline{q_n} \text{ на } S_1 \\ N_{nn} &= \overline{N_{nn}} = -N_p \text{ на } S_2. \end{aligned} \quad (1.28)$$

2 Постановка задачи

Рассматривается течение мелкой воды в закрытом водоеме при учете влияния на этот водоем ветра. Это явление описывается системой дифференциальных уравнений в частных производных [9], которая состоит из двух уравнений мелкой воды и уравнения неразрывности:

$$\begin{cases} \frac{\partial q_i}{\partial x_i} + \frac{\partial(\rho H)}{\partial t} = 0 \\ \frac{\partial q_1}{\partial t} + \frac{\partial}{\partial x_1} \left(\frac{q_1^2}{H} \right) + \frac{\partial}{\partial x_2} \left(\frac{q_1 q_2}{H} \right) = \frac{\partial}{\partial x_1} (N_{11} - N_p) + \frac{\partial N_{12}}{\partial x_2} + B_1 \\ \frac{\partial q_2}{\partial t} + \frac{\partial}{\partial x_1} \left(\frac{q_1 q_2}{H} \right) + \frac{\partial}{\partial x_2} \left(\frac{q_2^2}{H} \right) = \frac{\partial}{\partial x_2} (N_{22} - N_p) + \frac{\partial N_{12}}{\partial x_1} + B_2 \end{cases}$$

где

$$B_1 = f q_2 + \gamma^2 \rho_a W^2 \cos(\theta) - \left(\frac{g}{c^2} \right) \frac{1}{\rho} \frac{q_1 \sqrt{(q_1^2 + q_2^2)}}{H^2} + p_a \frac{\partial H}{\partial x_1} + \rho g H \frac{\partial h}{\partial x_1}; \quad (2.1)$$

$$B_2 = -f q_1 + \gamma^2 \rho_a W^2 \sin(\theta) - \left(\frac{g}{c^2} \right) \frac{1}{\rho} \frac{q_2 \sqrt{(q_1^2 + q_2^2)}}{H^2} + p_a \frac{\partial H}{\partial x_2} + \rho g H \frac{\partial h}{\partial x_2}. \quad (2.2)$$

В системе задействованы следующие переменные физические величины:

ρ - плотность воды	1000 [кг/м ³]
c - коэффициент трения Шэзи	10 [м ^{1/2} с ⁻¹]
g - ускорение свободного падения	9,832 [м/с ²]
γ^2 - коэффициент ветрового напряжения	0.002
p_a - атмосферное давление на поверхности воды	10 ⁵ [Па]
ρ_a - плотность воздуха	1,2754 [кг/м ³]
f - сила Кориолиса	0.973 · 10 ⁻⁴ [1/c]

Также в системе используются следующие переменные:

W - скорость ветра [м/с]

θ - угол между x_1 и направлением ветра

h - возвышение свободной поверхности [м]

Для данного двумерного случая системы дифференциальных уравнений требуется написать программную реализацию решения с помощью метода конечных элементов. Реализация должна работать с любой произвольно заданной областью. В качестве примеров таких областей должны быть как искусственно вырытые пруды, так и реальные озера.

Так как задача рассматривается двумерная, то КЭ будет представлять из себя треугольник. Простые двумерные области разбиваются на квадраты, которые впоследствии разбиваются на треугольники. В случае с более сложной геометрией триангуляция рассматриваемой области осуществляется с помощью более сложных алгоритмов, которые необходимо рассмотреть перед решением поставленной задачи, для того чтобы корректно генерировать сетку, на которой будет найдено решение поставленной задачи.

3 Триангуляция двумерной области

В геометрии триангуляция дискретного множества точек $P \subset \mathbb{R}^{n+1}$ в наиболее общем значении — это разбиение выпуклой оболочки некоторого набора точек, которое представляет собой планарный граф [10]. Одна из фигур разбиения является выпуклой оболочкой разбиваемого множества, а остальные симплексами — геометрическими фигурами, которые являются n -мерным обобщением треугольника. В любой триангуляции (T) формально должны выполняться следующие свойства:

1. любые два симплекса в T пересекаются в общей грани ребра или вершины или вообще не пересекаются;
2. множество точек, являющихся вершинами симплексов разбиения, совпадает с множеством P ;
3. нельзя добавить ни одного нового ребра в граф без нарушения планарности.

Одно и то же множество можно триангулировать разными способами. Триангуляция дает тем лучшую аппроксимацию, чем больше её минимальный угол, при этом формируемые симплексы стремятся к равноугольности. Очень важна максимизация минимального угла в вычислительных задачах, когда точность производимых вычислений очень сильно зависит от размера минимального угла триангуляции. Наилучшей в этом смысле триангуляцией является триангуляция Делоне. Простейшим способом её построения является инкрементальный алгоритм, работающий за $o(n^2)$ операций, но он не поддерживает вырожденные случаи, когда 4 точки из множества лежат на одной окружности: в этом случае триангуляция Делоне не уникальна, и способов разбиения существует несколько, но минимальные углы этих триангуляций равны. Иногда даже минимальный угол триангуляции Делоне оказывается слишком малым для устойчивой работы использующего её алгоритма, и тогда можно произвести улучшение, используя алгоритм Рапперта [J. Ruppert]. При этом будут добавлены новые вершины триангуляции, а также образованы дополнительные треугольники. Стабильность численного алгоритма (метода конечных элементов, к примеру) может возрасти многократно за счет появления нижней границы для углов.

3.1 Алгоритм Рапперта [Jim Ruppert]

Алгоритм Рапперта [J. Ruppert], или же усовершенствованный алгоритм Делоне, довольно новый, он был опубликован в 1994 году. Данный алгоритм адаптирован для МКЭ, а это значит, что он удовлетворяет всем требованиям, которые предъявляются к конечно-элементным сеткам, а именно:

1. треугольники не должны быть сильно вытянутыми, так как наличие таких треугольников отрицательно сказывается на точности результатов расчета;
2. должна быть учтена геометрия рассматриваемой области, и в местах со сложной геометрией сетка должна сгущаться.

Если говорить точнее, алгоритм Рапперта [J. Ruppert] не является алгоритмом генерации сеток. Он является лишь алгоритмом улучшения качества сетки, от чего и произошло название. В общем случае входными данными для него будут являться геометрия конструкции в виде замкнутых полигонов и первичное разбиение конструкции на треугольники.

При описании алгоритма Рапперта [J. Ruppert] следует ввести следующую терминологию:

- Элемент по смыслу совпадает с конечным элементом. Это элементарная часть пространства, на множество которых мы хотим разбить более сложную область этого пространства;
- Узел — точка в которой сходятся грани и соприкасаются несколько элементов;
- Сегмент — это отрезок, соединяющий две соседние точки лежащие на границе области разбиения; Другими словами, этот отрезок принадлежит контурам области;
- Грань — это отрезок, по которому граничат два соприкасающихся треугольника;
- Включенная точка — любая вершина текущей сетки, находящаяся внутри окружности, радиусом которой является любой сегмент;
- «Неправильный треугольник» — треугольник, не удовлетворяющий глобальным условиям, которые накладываются на генерируемую сетку;

- Вытянутый треугольник — треугольник, имеющий одну сторону, намного отличающуюся от других двух. То есть треугольник слишком вытянут вдоль некоторой прямой.

Алгоритм опирается на две базовые процедуры:

1. Разбиение неправильного треугольника с введением нового узла.
 - (a) Вычисляются координаты центра окружности, описанной вокруг треугольника, подлежащего разбиению
 - (b) В найденную точку добавляется новый узел
 - (c) Удаляется треугольник, подлежащий разбиению, и прилегающие к нему треугольники, а затем добавляются новые в соответствии с рисунком 3.1
2. Разбиение сегмента, принадлежащего границе области разбиения с введением нового узла.
 - (a) Если в окружность, диаметром которой является сегмент, принадлежащий границе области разбиения, попадает точка, не принадлежащая этому сегменту, то сегмент делится на две части;
 - (b) Удаляется треугольник, которому принадлежал первоначальный сегмент, и добавляются два новых треугольника в соответствии с рисунком 3.2.

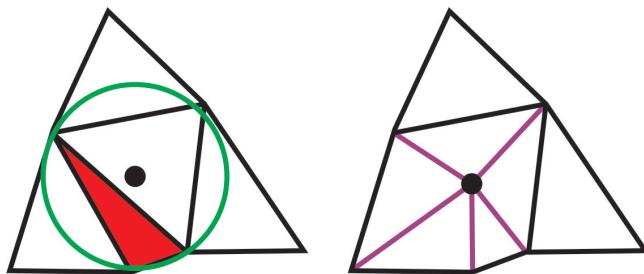


Рисунок 3.1 — Разбиение «неправильного» треугольника

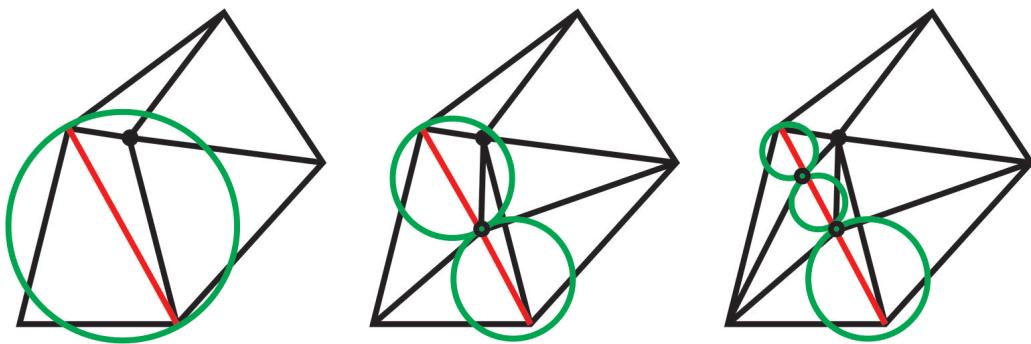


Рисунок 3.2 — Разбиение «неправильного» сегмента посередине и добавление двух новых треугольников

Принцип работы алгоритма состоит в цикле определения «неправильных» треугольников. В каждом треугольнике сетки определяется минимальный угол и, затем сравниваются минимальные углы для всех треугольников с целью нахождения минимального угла сетки. Другими словами, минимальный угол в сетке равен минимально возможному углу, который образован двумя сегментами, либо двумя гранями, либо сегментом и гранью, имеющими общий узел.

Критерий минимального угла автоматически обеспечивает сгущение сетки вблизи мелких подробностей: отверстий, углов и вырезов.

Все описанные процедуры выполняются в определенной последовательности, они имеют разный приоритет. Общая схема работы алгоритма следующая:

1. Производится поиск включенной точки перебором всех точек и сегментов (только сегментов, но не граней).
2. Если такая точка найдена, то над сегментом, который включает ее, выполняется процедура деления сегмента пополам и производится возврат на шаг 1. Если включенных точек не было найдено, то алгоритм переходит на следующий шаг.
3. Производится поиск треугольника с минимальным углом.
4. Если минимальный угол меньше заданного параметра, то над треугольником, и его содержащим производится процедура деления и производится возврат на шаг 1. В противном случае происходит переход на следующий шаг.
5. Производится поиск треугольника максимальной площадью.

6. Если максимальная площадь больше заданного параметра, то над таким треугольником производится процедура деления и производится возврат на шаг 1. В противном случае происходит переход на следующий шаг.

7. Все условия удовлетворены. Конец работы алгоритма

Автор в своей работе [11] приводит строгие математические доказательства, гарантирующие правильную работу алгоритма при минимальном угле $\alpha < 20^\circ$. Данный алгоритм имеет ряд теоретических и практических преимуществ для генерации сетки. Для не острого ввода и минимального угла порога около 20.70° алгоритм гарантированно завершает работу и создает сетку оптимального размера с постоянным коэффициентом, а также дает теоретически подтвержденные гарантии качества получаемой с его помощью сетки.

3.2 Алгоритм Чу [L. Paul Chew]

В некоторых случаях в алгоритме Рапперда [J. Ruppert] получаются слишком большие сегменты, а для МКЭ очень хорошо иметь как можно более мелкие элементы в сетке. Для того чтобы этого избежать, был разработан алгоритм Чу [L. Chew], в котором сетка получается мельче за счет того, что он более консервативен в отношении разбиения на сегменты, когда граница угла меньше 30° . Главное преимущество данного алгоритма по сравнению с алгоритмом Рапперда [J. Ruppert] состоит в том, что КЭ в сетке получаются с углом до 28.6° , что является несомненным плюсом, так как элементы в сетке получаются приблизительно одинаковые. Алгоритм построен на более сложной теоретической базе [12], и главное его отличие от алгоритма Рапперта в том, что Рапперт только гарантирует хорошее разбиение и угол меньший, чем 20.7° , а Чу обязательно даст хорошее разбиение, правда, с углом до 26.5° . На практике данные алгоритмы дают примерно одинаковые разбиения, но алгоритм Чу является более популярным среди разработчиков математических пакетов.

4 Метод конечных элементов в двумерной области

Метод конечных элементов [13, 14] - численный метод решения дифференциальных уравнений в частных производных, возникающими при решении задач прикладной физики. В его основе лежат две главные идеи: дискретизация исследуемого объекта и кусочно-элементная аппроксимация исследуемых функций (в физической интерпретации - температуры, давления, перемещения и т.д.).

Идея состоит в том, что область, в которой ищется решение дифференциальных уравнений, разбивается на конечное количество элементов. В каждом из элементов произвольно выбирается вид аппроксимирующей функции. Вне своего элемента аппроксимирующая функция равна нулю. Значения функций в узлах являются решением задачи и заранее неизвестны. Коэффициенты аппроксимирующих функций обычно ищутся из условия равенства значения соседних функций на границах между элементами (в узлах). Затем эти коэффициенты выражаются через значения функций в узлах элементов. Составляется система уравнений. Количество уравнений равно количеству неизвестных значений в узлах, на которых ищется решение исходной системы, и прямо пропорционально количеству элементов. Так как каждый из элементов связан с ограниченным количеством соседних, система уравнений имеет разрежённый вид, что упрощает её решение.

Если говорить в матричных терминах, то собираются так называемые матрицы жёсткости и масс. Далее на эти матрицы накладываются граничные условия (например, при условиях Неймана в матрицах не меняется ничего, а при условиях Дирихле из матриц вычёркиваются строки и столбцы, соответствующие граничным узлам, так как в силу краевых условий значение соответствующих компонент решения известно). После собирается система уравнений, тип которых может различаться в зависимости от поставленной задачи. Они могут быть как алгебраическими, так и дифференциальными. После того как система получена, она решается любым из доступных методов.

Основное отличие МКЭ от классических алгоритмов вариационных принципов и от методов невязок заключается в выборе базисных функций. Они берутся в виде кусочно-непрерывных функций, которые обращаются в ноль

всюду, кроме ограниченных подобластей, являющихся конечными элементами. Это в свою очередь ведет к разреженной структуре матрицы коэффициентов разрешающей системы уравнений.

Главные достоинства МКЭ состоят в следующем:

1. исследуемые объекты могут иметь любую форму и различную физическую природу, это твёрдые тела, жидкости, газы, электромагнитные среды
2. конечные элементы могут иметь различную криволинейную форму и различные размеры
3. можно исследовать однородные и неоднородные, изотропные и анизотропные тела с линейными и нелинейными свойствами
4. можно решать как стационарные, так и нестационарные задачи
5. можно моделировать любые граничные условия
6. вычислительный алгоритм, представленный в матричной форме, формально единообразен для различных физических задач и для задач различной размерности. Это удобно для вычисления на компьютере, так как методология не меняется и фактически используется единая программа численного решения
7. на одной и той же сетке конечных элементов можно решать различные физические задачи, что облегчает анализ связанных между собой задач
8. разрешающая система уравнений имеет разреженную симметричную ленточную матрицу жёсткости, что ускоряет вычислительный процесс

Перед тем как рассматривать метод конечных элементов, рассмотрим метод взвешенных невязок, который является частным случаем метода конечных элементов с одним элементом.

4.1 Метод взвешенных невязок

Рассмотрим дифференциальное уравнение вида

$$\mathcal{A} = \mathcal{L}\phi + p = 0 \quad (4.1)$$

в некоторой области Ω .

Здесь \mathcal{L} - некоторый линейный дифференциальный оператор, а r не зависит от неизвестной функции ϕ .

Решение уравнения 4.1 должно удовлетворять условию

$$\mathcal{B} = \mathcal{M}\phi + r = 0 \quad (4.2)$$

на замкнутой кривой Γ , ограничивающей область Ω .

Здесь \mathcal{M} - соответствующий линейный дифференциальный оператор, а r не зависит от неизвестной функции ϕ .

Построим аппроксимацию для решения ϕ , которая на граничной кривой Γ принимает те же значения, что и ϕ . Если найти некоторую функцию ψ , принимающую одинаковые с ϕ значения на Γ , т.е. $\psi|_{\Gamma} = \phi|_{\Gamma}$, и ввести систему линейно независимых базисных функций $\{N_m; m = 1, 2, \dots, N\}$, то на Ω можно предложить следующую аппроксимацию $\hat{\phi}$ для ϕ :

$$\phi \approx \hat{\phi} = \psi + \sum_{m=1}^M a_m N_m, \quad (4.3)$$

где $a_m, m = \overline{1, M}$ - некоторые параметры, вычисляемые так, чтобы получить хорошее приближение, а функция ψ и базисные функции N_m выбраны таким образом, что

$$\mathcal{M}\psi = -r, \quad \mathcal{M}N_m = 0, \quad m = \overline{1, M} \text{ на } \Gamma, \quad (4.4)$$

и поэтому ϕ автоматически удовлетворяет краевым условиям [?] при произвольных коэффициентах a_m . Отметим, что система базисных функций (которые также называют функциями формы) должна быть выбрана так, чтобы гарантировать улучшение аппроксимации при возрастании числа базисных функций для M .

Для того чтобы найти аппроксимации производных от ϕ , продифференцируем 4.3 и получим:

$$\begin{aligned}\phi &\approx \hat{\phi} = \psi + \sum_{m=1}^M a_m N_m, \\ \frac{\partial \phi}{\partial x} &\approx \frac{\partial \hat{\phi}}{\partial x} = \frac{\partial \psi}{\partial x} + \sum_{m=1}^M a_m \frac{\partial N_m}{\partial x}, \\ \frac{\partial^2 \phi}{\partial x^2} &\approx \frac{\partial^2 \hat{\phi}}{\partial x^2} = \frac{\partial^2 \psi}{\partial x^2} + \sum_{m=1}^M a_m \frac{\partial^2 N_m}{\partial x^2}\end{aligned}$$

и т.д.

Так как построенное разложение 4.3 удовлетворяет краевым условиям 4.2, то для получения аппроксимации искомой функции ϕ осталось гарантировать, что $\hat{\phi}$ - приближённое решение уравнения 4.1. Для этого вначале введём невязку R_Ω в аппроксимации, определяемую по формуле:

$$R_\Omega = A(\hat{\phi}) = \mathcal{L}\hat{\phi} + p = \mathcal{L}\psi + \left(\sum_{m=1}^M a_m \mathcal{L}N_m \right) + p. \quad (4.5)$$

Чтобы уменьшить эту невязку, потребуем равенства нулю соответствующего числа интегралов от погрешности [8], взятых с различными весами, т.е.

$$\int_{\Omega} W_l R_\Omega d\Omega = \int_{\Omega} W_l \left\{ \mathcal{L}\psi + \left(\sum_{m=1}^M a_m \mathcal{L}N_m \right) + p \right\} d\Omega = 0; \quad l = \overline{1, M}, \quad (4.6)$$

где $\{W_l : l = 1, 2, \dots, M\}$ - это множество линейно независимых весовых (или пробных) функций.

Таким образом, система уравнений метода взвешенных невязок 4.6 сводится к системе линейных алгебраических уравнений для неизвестных коэффициентов a_m , которую можно записать следующим образом:

$$Ka = f, \quad (4.7)$$

где

$$a = (a_1, a_2, \dots, a_M)^T, \quad (4.8)$$

$$K_{lm} = \int_{\Omega} W_l \mathcal{L} N_m d\Omega, \quad l, m = \overline{1, M}; \quad (4.9)$$

$$f_l = - \int_{\Omega} W_l p d\Omega - \int_{\Omega} W_l \mathcal{L} \psi d\Omega \quad l, m = \overline{1, M}. \quad (4.10)$$

Вычислив элементы матрицы K и столбца свободных членов f и решив затем полученную систему, мы найдем a_m , где $m = \overline{1, M}$, и тем самым закончим процесс построения приближенного решения 4.1.

4.2 Метод конечных элементов

В методе взвешенных невязок предполагалось, что базисные функции N_m определены одним выражением на всей области Ω . При этом интегралы в аппроксимирующем уравнении 4.5 вычисляются по всей области.

С другой стороны, область Ω можно разбить на несколько непересекающихся подобластей (конечных элементов Ω^e), которые могут иметь различную форму и размеры. В результате разбивки создается сетка из границ элементов:

$$\Omega = \bigcup_{e=1}^E \Omega^e.$$

Пересечение границ областей Ω^e образуют узлы. Выбор типа и размера КЭ напрямую зависит от рассматриваемой задачи.

После того как мы разбили рассматриваемую область Ω сеткой конечных элементов, можно аппроксимировать решение уравнения отдельно для каждой подобласти Ω^e , где базисные функции могут быть определены различным образом для каждой из подобластей. В этом случае определённые интегралы, входящие в аппроксимирующие уравнения, можно посчитать просуммировав, вклады по каждому элементу:

$$\int_{\Omega} W_l R_{\Omega} d\Omega = \sum_{e=1}^E \int_{\Omega^e} W_l R_{\Omega} d\Omega^e,$$

$$\int_{\Gamma} \overline{W}_l R_{\Gamma} d\Gamma = \sum_{e=1}^E \int_{\Gamma^e} W_l R_{\Gamma} d\Gamma^e. \quad (4.11)$$

В данных интегралах:

$$\sum_{e=1}^E \Omega^e = \Omega, \quad (4.12)$$

$$\sum_{e=1}^E \Gamma^e = \Gamma, \quad (4.13)$$

где E - число подобластей, на которые разбивается область Ω ; Γ^e - часть границы Ω^e , лежащая на Γ .

Система уравнений метода конечных элементов сводится к аналогичной системе метода взвешенных невязок, но за исключением того, что:

$$K_{lm} = \sum_{e=1}^E K_{lm}^e = \sum_{e=1}^E \int_{\Omega^e} W_l^e \mathcal{L} N_m^e d\Omega^e, \quad l, m = \overline{1, M}; \quad (4.14)$$

$$f_l = \sum_{e=1}^E f_l^e = \sum_{e=1}^E \left(- \int_{\Omega^e} W_l^e p d\Omega^e - \int_{\Omega^e} W_l^e \mathcal{L} \psi d\Omega^e \right) \quad l, m = \overline{1, M}. \quad (4.15)$$

Отметим также, что для матрицы K^e не нужно вычислять каждую компоненту, так как её компонента K_{lm}^e равна нулю, если узлы l и m не принадлежат элементу e .

Идея метода конечных элементов заключается в следующем: если подобласти имеют простую форму и базисные функции на этих подобластях вычисляются однотипно, то решать задачу указанным выше способом становиться очень просто.

4.3 МКЭ для рассматриваемой задачи

4.3.1 Интегрирование функции двух переменных по произвольному треугольнику

Интегрирование по конечному элементу является очень важной составляющей МКЭ. Так как в общем случае симплексы в разбиении являются различными, то необходимо уметь интегрировать функцию двух переменных по произвольному треугольнику. В декартовой системе координат это является достаточно сложной задачей, и именно поэтому необходимо осуществить переход к барицентрической системе координат, в которой эта задача является достаточно простой.

Барицентрическая система координат представляет собой систему координат, в которой расположение точки симплекса определяется как центр массы или барицентр, как правило, неравных масс, размещенных в его вершинах.

Мы можем записать декартовы координаты точки \mathbf{r} через декартовы компоненты треугольных вершин $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3$ где $\mathbf{r}_i = (x_i, y_i)$, и в терминах барицентрической системы координат переход будет выглядеть следующим образом:

$$x = \lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3 \quad (4.16)$$

$$y = \lambda_1 y_1 + \lambda_2 y_2 + \lambda_3 y_3 \quad (4.17)$$

$$\lambda_3 = 1 - \lambda_1 - \lambda_2 \quad (4.18)$$

Вычисление двойного интеграла в барицентрических координатах осуществляется посредством следующей формулы [15]:

$$\int_{\Delta} f(\mathbf{r}) d\mathbf{r} = 2S_{\Delta} \int_0^1 \int_0^{1-\lambda_2} f(\lambda_1 \mathbf{r}_1 + \lambda_2 \mathbf{r}_2 + (1 - \lambda_1 - \lambda_2) \mathbf{r}_3) d\lambda_1 d\lambda_2 \quad (4.19)$$

4.3.2 Уравнения мелкой воды в терминах МКЭ

В рассматриваемой системе 2.1-2.1 можно пренебречь членами $-fq_1$ и fq_2 , так как коэффициент Кориолиса очень мал, а также конвективными членами. После отбрасывания неучитываемых членов в 2.1-2.1 можно перейти к конечно-элементной формулировке. Для этого необходимо представить функции $q_1(x_1, x_2, t)$, $q_2(x_1, x_2, t)$, $H(x_1, x_2, t)$ как разложения:

$$q_1(x_1, x_2, t) = \sum_{m=1}^M a_m(t) N_m(x, y) \quad (4.20)$$

$$q_2(x_1, x_2, t) = \sum_{m=1}^M a_{m+k}(t) N_m(x, y) \quad (4.21)$$

$$H(x_1, x_2, t) = \sum_{m=1}^M a_{2m+k}(t) N_m(x, y) \quad (4.22)$$

Важно заметить, что в данных разложениях применяются одни и те же весовые функции N_k .

Следующим шагом необходимо подставить полученные разложения в исходную систему уравнений. После этого каждое из уравнений необходимо умножить на $\sum_{l=1}^M W_l(x_1, x_2)$ и проинтегрировать во треугольнику. Также каждое из уравнений необходимо разрешить относительно производной, чтобы получить систему обыкновенных дифференциальных уравнений [16] вида:

$$A \cdot \frac{da}{dt} + B \cdot a + \dots = f \quad (4.23)$$

Важно заметить, что каждое из трех исходных уравнений в частных производных дало по M обыкновенных дифференциальных уравнений, и в конечном итоге задача сводится к решению системы из $3 \cdot M$ дифференциальных уравнений.

В данной дипломной работе конечная система уравнений не была получена аналитически, так как для упрощения решения и минимизации ошибок

был использован программный пакет symPy [17]. С помощью его и системы Jupiter Notebook три формулы для q_1 , q_2 и H соответственно были выведены, а далее запрограммированы с помощью функции solve ivp из пакета scipy и приведены в приложении В.

5 Решение задачи

5.1 Построение графиков

Для того чтобы визуализировать результаты численных экспериментов, была написана функция, которая строит графики для q_1 , q_2 и H в различные моменты времени. Для лучшей визуализации был написан алгоритм, который комбинирует данные графики в GIF анимацию. Так как количество графиков для некоторых из примеров было достаточно велико, то для создания анимации в случае некоторых экспериментов пришлось писать данные в буфер обмена напрямую, так как в противном случае происходило переполнение и программа переставала работать.

Все графики строятся по изначальным точкам разбиения области. Для того чтобы получить более детальную картинку, была произведена кубическая интерполяция.

Для получения дополнительной информации об эксперименте была написана функция для визуализации графика функции тока:

$$\Psi = \int q_1(x_1, x_2) dx_2$$

Визуализация данной функции дает лучшее понимание о том, каким именно образом распространяются волны в водоеме. Для данных графиков тоже была произведена кубическая интерполяция.

5.2 Изменение формы водоёма

Для программной реализации триангулирования произвольной области был использован Python модуль triangle [1], который реализует триангуляцию как с помощью алгоритма Рапперта, так и с помощью алгоритма Чу.

Данный модуль получает на вход геометрию в формате POLY и возвращает список вершин, сегментов, пустых областей, треугольников, маркеров вершин и маркеров сегментов, которые представляют собой нерегулярную сетку для исходной области. Для структуризации этой информации была написана своя реализация для объекта «точка» и «треугольник». Экземпляр

класса «точка» хранит в себе координаты (x, y) и номер точки. Экземпляр класса «треугольник» хранит в себе список из трех точек, а также имеет методы для вычисления площади, базисных функций и интегрирования по данному треугольнику. Таким образом, в реализации результат триангуляции представляет из себя массив экземпляров класса «треугольник».

5.2.1 Пруд

В качестве самой простой геометрии был выбран квадратный пруд показанный на рисунке 5.1

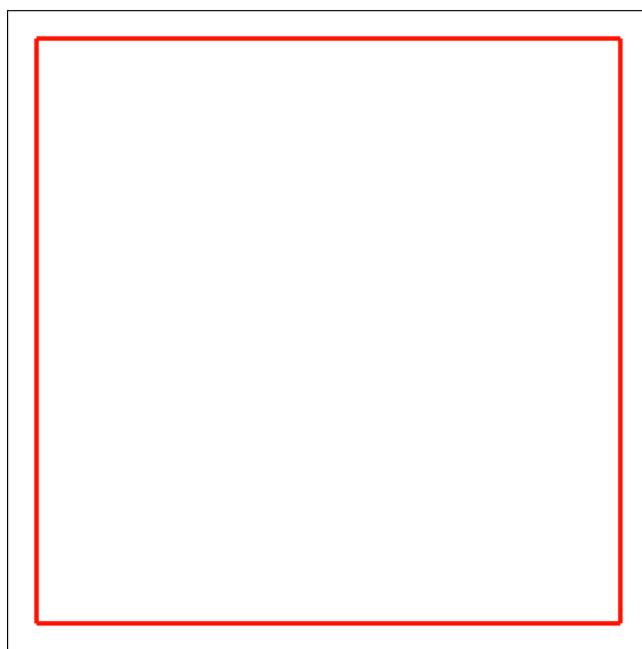


Рисунок 5.1

Для него было построено разбиение в соответствии с рисунком 5.2.

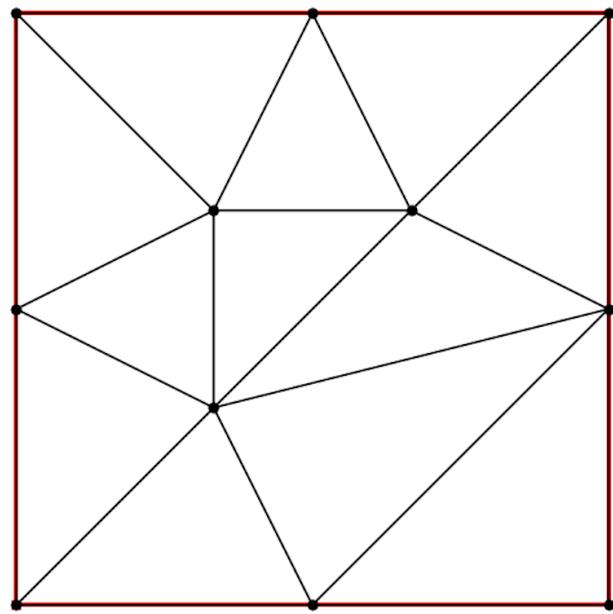


Рисунок 5.2 — Триангуляция прямоугольного пруда

Данный пример рассматривался на моментах времени 0с, 1с и 2с.

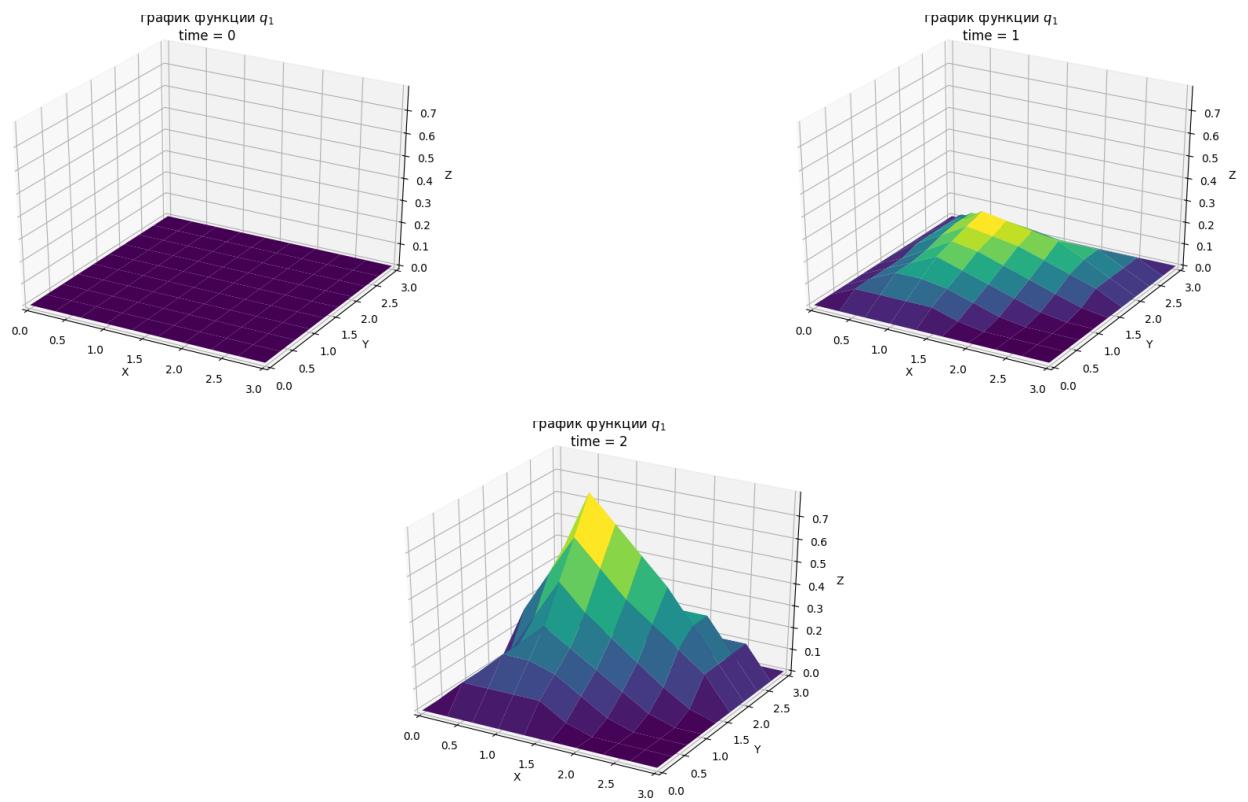


Рисунок 5.3 — Изменение первой функции потока жидкости в различные промежутки времени

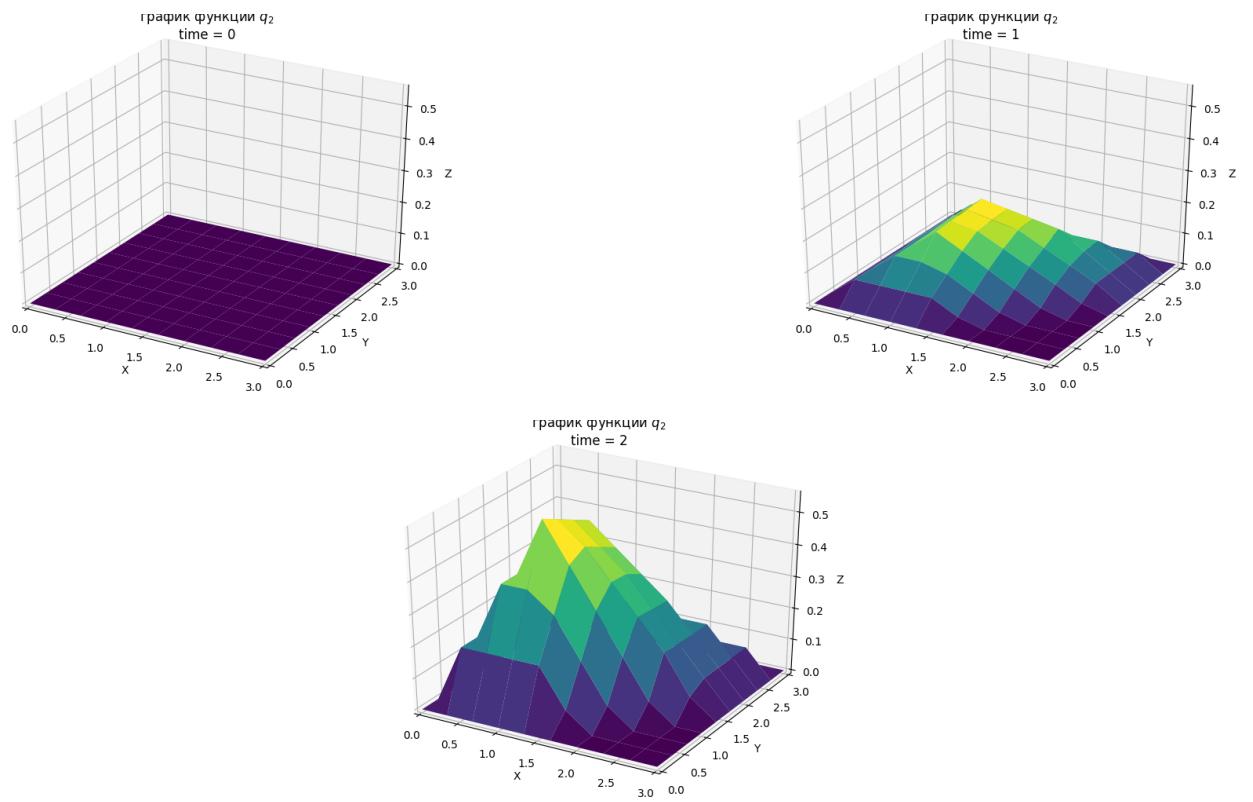


Рисунок 5.4 — Изменение второй функций потока жидкости в различные промежутки времени

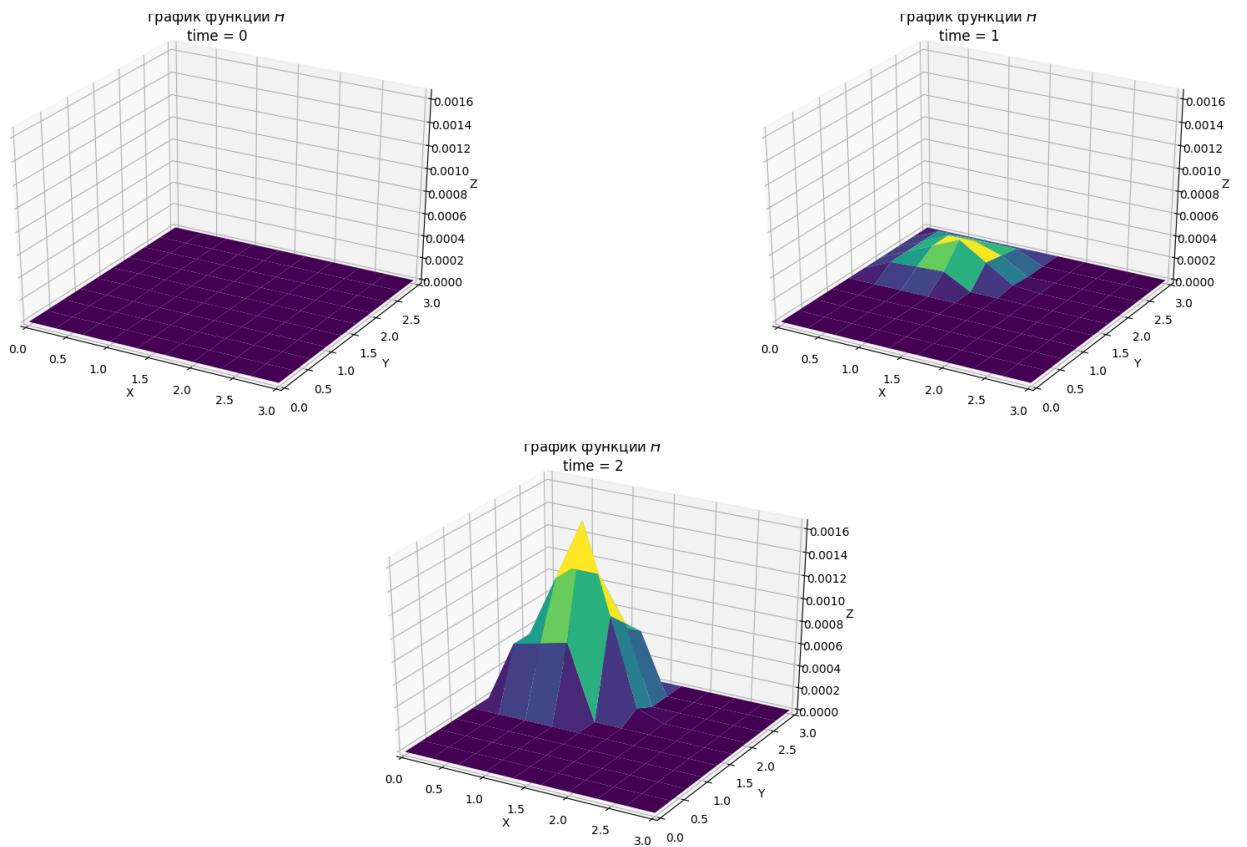


Рисунок 5.5 — Изменение функции H в различные промежутки времени

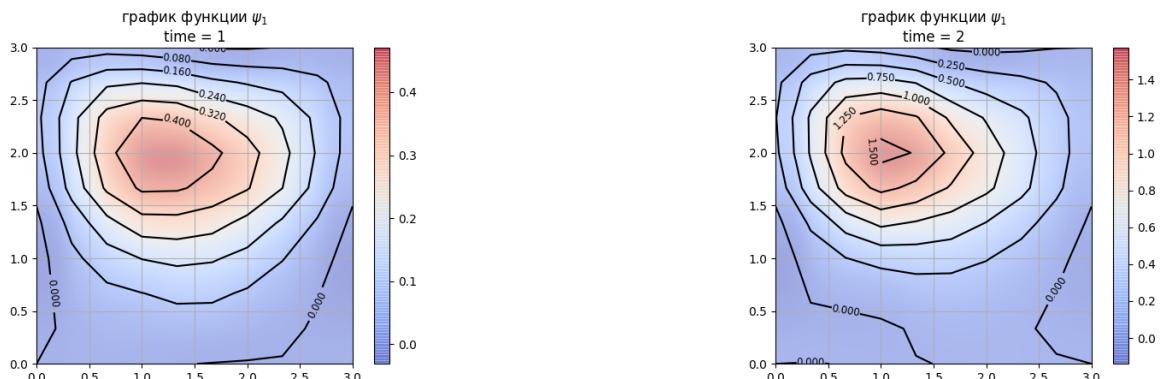


Рисунок 5.6 — Изменение функции тока в различные промежутки времени

5.2.2 Реальный водоём

В качестве реального водоема, у которого нет островов, было выбрано озеро Эльтон - солёное бессточное самосадочное озеро на севере Прикаспийской низменности. Данное озеро имеет площадь в 152 квадратных километра и

наибольшую глубину до 1.5 метров. В среднем глубина данного озера составляет 0.05 – 0.07 метра, а это значит, что для данного озера будут справедливыми уравнения мелкой воды.

Данные для расчета задачи по озеру Эльтон были получены на основании данных из системы OSM [18] - некоммерческого веб-картографического проекта. Для этого был использован сервис overpass-turbo [19], который является интерфейсом к API OSM и позволяет очень просто выполнять необходимые запросы. Запрос для озера Эльтон выглядит следующим образом:

```
[out:json][timeout:25];
(
  node["name"="Elton"]({{bbox}});
  way["name"="Elton"]({{bbox}});
  relation["name"="Elton"]({{bbox}});
);
out body;
>;
out skel qt;
```

Одним из результатов запроса будет *geojson* файл, в котором содержатся точки запрашиваемого географического объекта и некоторая метаинформация. Так как для триангуляции требуется файл с геометрией в формате POLY, то была написана соответствующая функция на языке *Python*, которая конвертирует формат *geojson* в формат POLY.

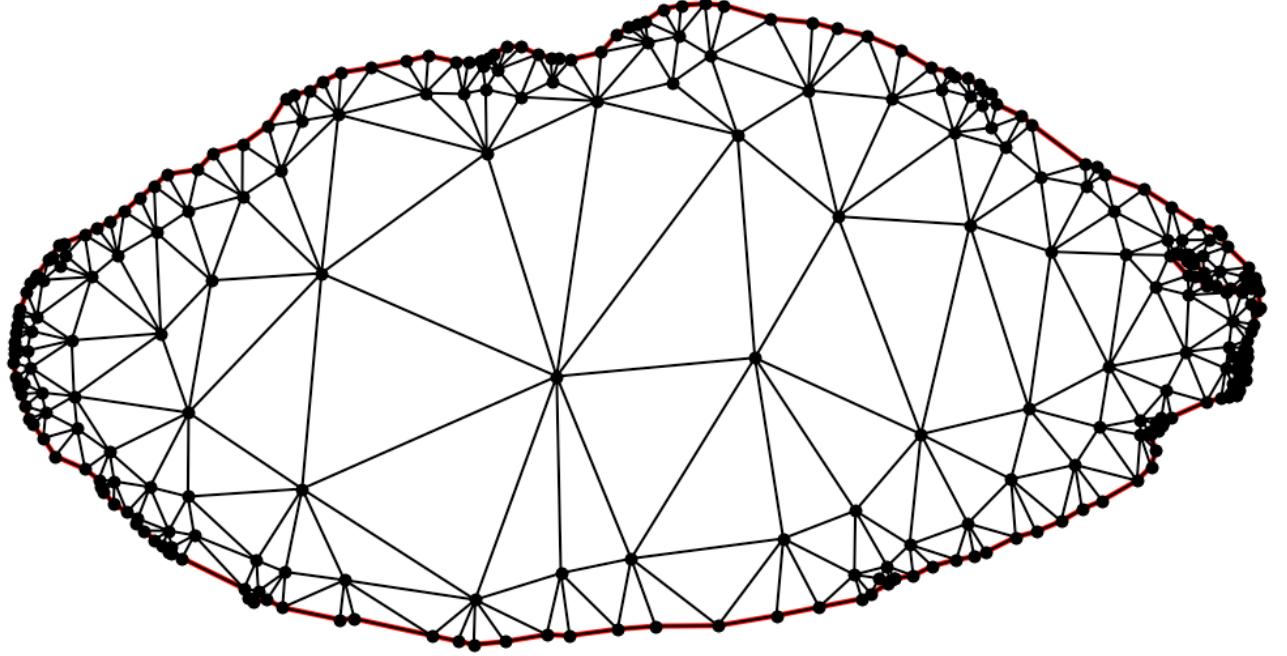


Рисунок 5.7 — Триангуляция озера Эльтон

TODO: тут будут добавлены аналогичные графики после того, как озеро досчитается

5.3 Учёт наличия острова (островов) внутри водоёма

5.3.1 Пруд

Для того чтобы протестировать корректность алгоритма на сетках с «дырами», в пруд, рассмотренный ранее, добавились два острова, как показано на рисунке 5.8

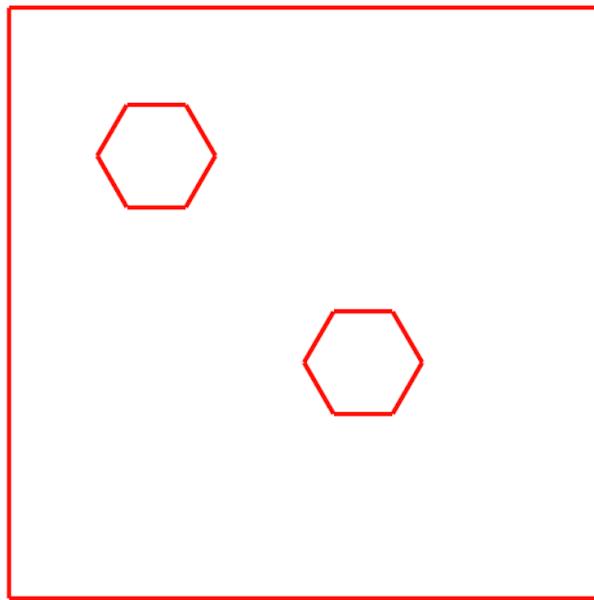


Рисунок 5.8

Для этой геометрии было построено разбиение в соответствии с рисунком 5.9.

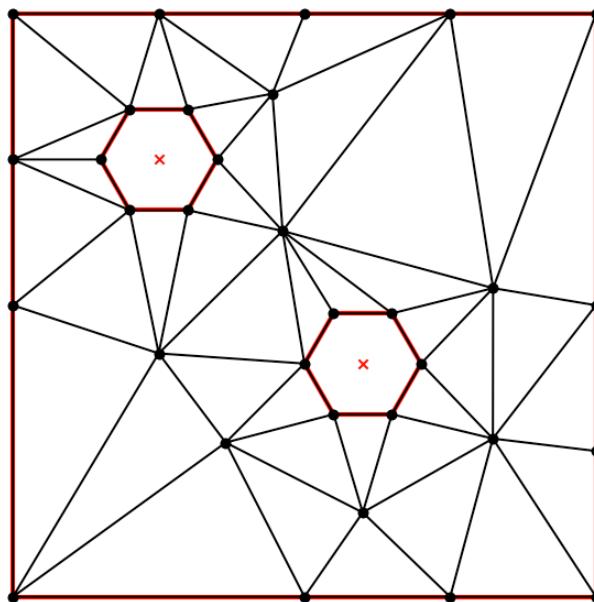


Рисунок 5.9 — Триангуляция прямоугольного пруда с двумя островами

Данный пример рассматривался на двух моментах времени: 1.91с и 1.99с.

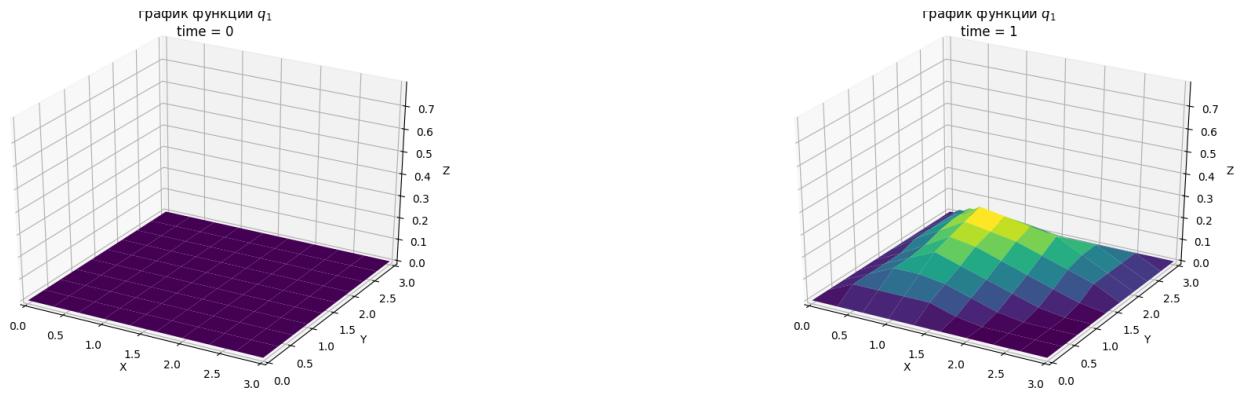


Рисунок 5.10 — Изменение первой функций потока жидкости в различные промежутки времени

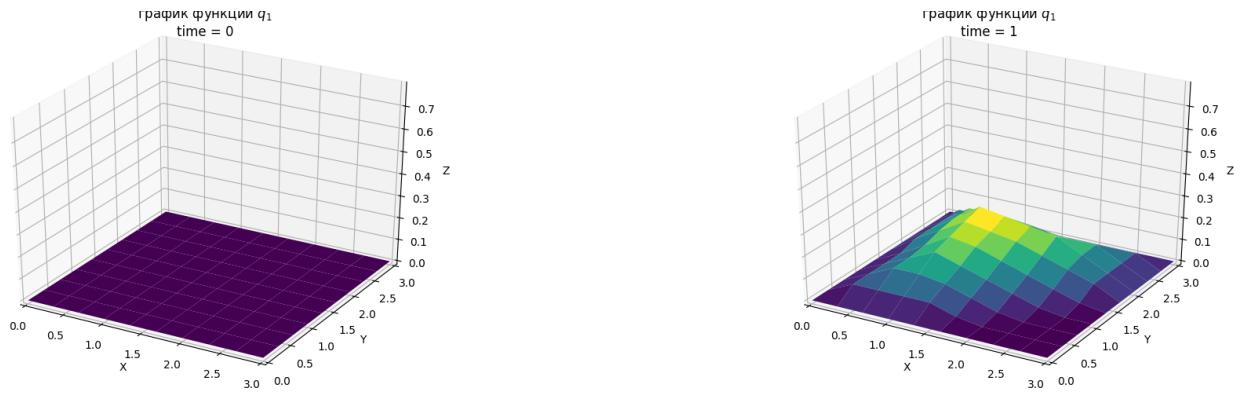


Рисунок 5.11 — Изменение второй функций потока жидкости в различные промежутки времени

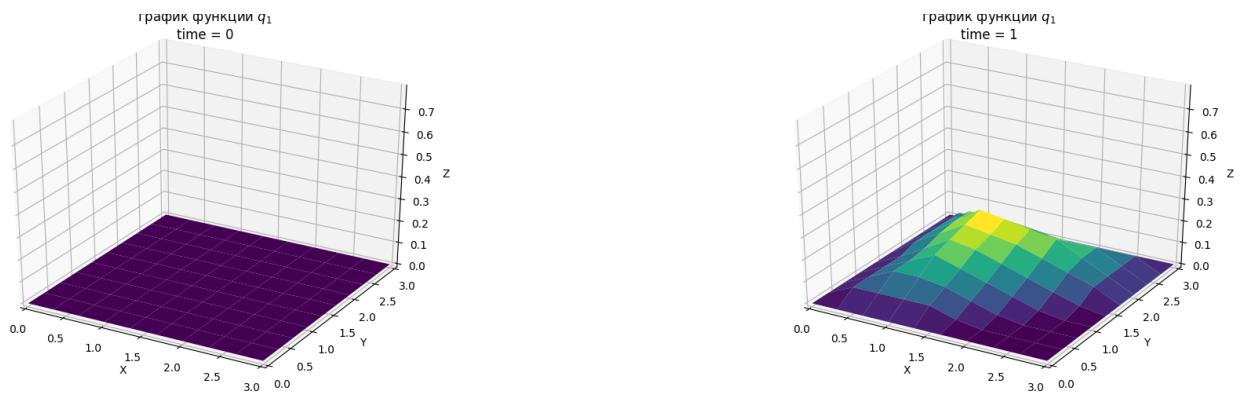


Рисунок 5.12 — Изменение функции H в различные промежутки времени

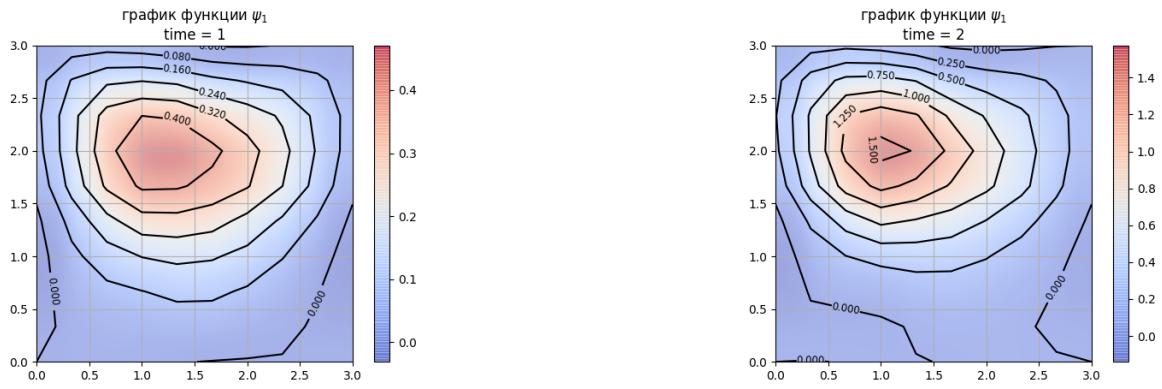


Рисунок 5.13 — Изменение функции тока в различные промежутки времени

5.3.2 Реальный водоём

В качестве реального водоема, у которого есть острова, было выбрано озеро Верхнее. Оно находится в Северной Америке и является самым крупным и глубоким в системе Великих озёр. С физической точки зрения глубина этого озера составляет максимум 406 метров, а его размеры - 563 x 257 км. Получается, что для него можно использовать уравнения мелкой воды.

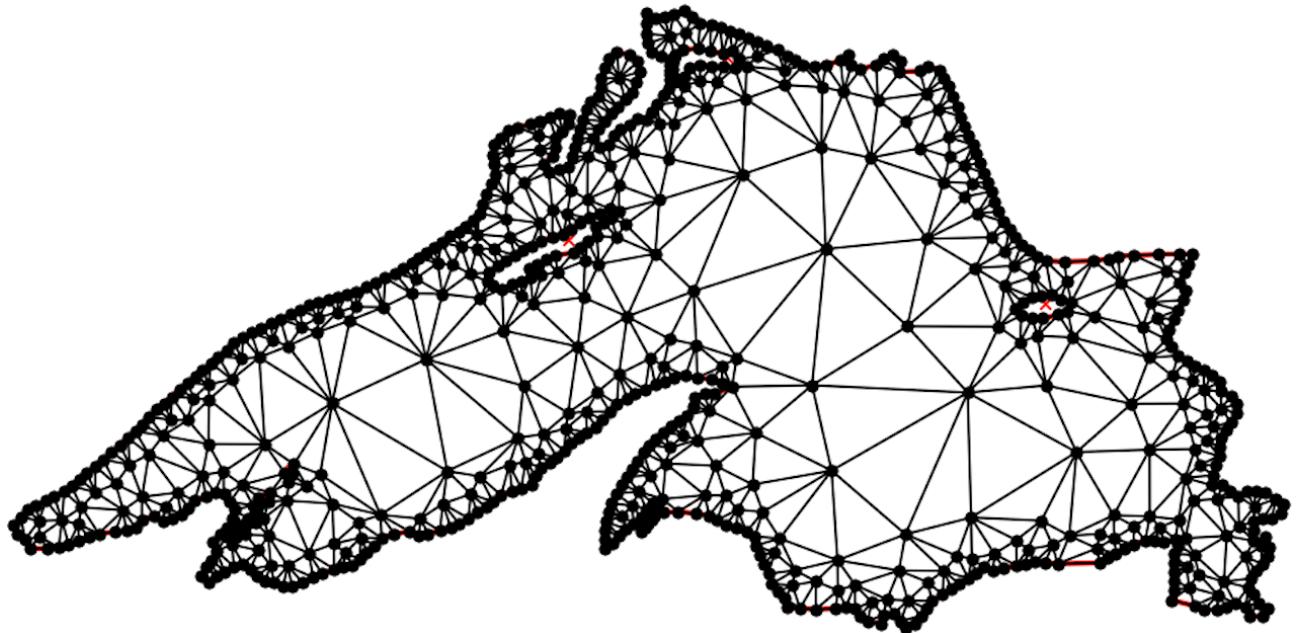


Рисунок 5.14

TODO: тут будут добавлены аналогичные графики после того, как озеро досчитается

5.4 Программная реализация

МКЭ является очень ресурсоемким методом, и именно из-за этого он долго не применялся, а существовал только в теории. Все современные задачи считаются на больших сетках, и занимает это достаточно большое количество времени.

Именно по этой причине реализация МКЭ в данной дипломной работе является многопоточной. В первую очередь это удалось сделать благодаря самому МКЭ, который позволяет считать элементы в сетке независимо друг от друга. Для решения этой задачи был написан пулл потоков, который позволяет в один и тот же момент времени считать от 128 до 256 элементов. Цифры могут меняться, так как параметр является конфигурируемым и максимальное значение одновременно вычисляемых элементов зависит только от компьютера, на котором производятся вычисления. Так как при использовании потоков Python не может задействовать все ядра процессора для выполнения нескольких потоков одновременно, то вместо них в реализации задачи были использованы процессы из библиотеки `multiprocessing`.

Помимо этого все модули приложения вместе с внешними зависимостями были упакованы в Docker контейнер, который позволяет очень просто переносить приложение на любую систему, которая поддерживает данную технологию. Данная упаковка позволила производить вычисления не на личном стационарном компьютере, а на удаленном сервере, который имеет намного большие вычислительные мощности и позволяет быстрее получить результат.

6 Разработка базы данных для хранения информации о проведённых экспериментах

Данные, полученные в результате численных экспериментов, являются разрозненными, так как они представляют из себя не только GIF анимацию, но и некоторую метаинформацию, которая представлена в виде JSON файлов. Из-за этого, помимо общепринятых требований к БД, следует основываясь на приведенной выше специфике данных, которые были получены в результате решения поставленной задачи, ввести следующие требования, которым должны удовлетворять БД и СУБД, управляющая ей:

1. Запись об эксперименте должна позволять хранить любую метаинформацию, которая может различаться в зависимости от проведенного эксперимента
2. БД должна позволять добавлять поля к уже существующим данным. Это, к примеру, нужно для добавления в запись данных, которые получены в результате постобработки эксперимента человеком
3. БД должна легко масштабироваться, так как данных может быть много
4. Должно быть отсутствие необходимости сопоставления объекта в базе и программе для упрощения разработки
5. Хранение графической информации должно быть реализовано максимально просто и эффективно

6.1 Описание базы данных

Основываясь на приведенных выше требованиях, в качестве СУБД решено выбрать MongoDB, которая классифицируется как NoSQL. MongoDB — документо-ориентированная система управления базами данных с открытым исходным кодом, не требующая описания схемы таблиц. Каждая запись — это документ без жёстко заданной схемы, который может содержать вложенные документы.

Так как документо-ориентированные базы данных еще не так распространены, как реляционные, но их популярность с каждым годом все увеличивается, целесообразно будет разъяснить некоторые нюансы терминологии документо-ориентированных баз данных.

В учебнике The Little MongoDB Book [20] приведены шесть основных концепций MongoDB:

1. MongoDB концептуально является тем же самым, что и привычная нам база данных (в терминологии Oracle RDBMS - схема). Внутри MongoDB может быть ноль или более баз данных, каждая из которых является контейнером для прочих сущностей.
2. База данных может иметь ноль или более «коллекций». «Коллекция» настолько похожа на традиционную «таблицу», что можно смело считать их одним и тем же.
3. «Коллекции» состоят из нуля или более «документов», каждый из которых можно рассматривать как «строку».
4. «Документ» состоит из одного или более «полей», которые подобны «колонкам».
5. «Индексы» в MongoDB почти идентичны таковым в реляционных базах данных.
6. «Курсыры» отличаются от предыдущих пяти концепций, но они очень важны. Когда мы запрашиваем у MongoDB какие-либо данные, то БД возвращает «курсор», с которым мы можем делать все что угодно: подсчитывать, пропускать определённое число предшествующих записей, - при этом не загружая сами данные.

Если резюмировать сказанное выше, то MongoDB состоит из «баз данных», которые, в свою очередь, состоят из «коллекций». «Коллекции» состоят из «документов», а каждый «документ» - из «полей». «Коллекции» могут быть проиндексированы, что улучшает производительность выборки и сортировки. И наконец, получение данных из MongoDB сводится к получению «курсора», который отдаёт эти данные по мере необходимости.

Также стоит привести основные достоинства данной базы данных:

1. Документо-ориентированное хранилище (простая и мощная JSON-подобная схема данных)
2. Достаточно гибкий язык для формирования запросов
3. Динамические запросы
4. Полная поддержка индексов
5. Профилирование запросов

6. Эффективное хранение двоичных данных больших объёмов, например фото и видео
7. Журнализование операций, модифицирующих данные в БД
8. Поддержка отказоустойчивости и масштабируемости: асинхронная репликация, набор реплик и шардинг

Основными плюсами MongoDB, помимо её документо-ориентированности, для данной дипломной работы стали:

1. Схожий с реляционными СУБД подход к хранению данных, который построен на следующем: база данных-коллекция-документ-поле
2. Формат хранения данных. Данные в MongoDB хранятся в формате BSON. BSON - это бинарный формат, используемый для хранения документов и осуществления удаленного вызова процедур в MongoDB. Тот факт, что данные хранятся именно в бинарном виде, значительно ускоряет время их автоматической обработки
3. Простота разработки. MongoDB, в отличии от конкурентов, достаточно просто разворачивается и администрируется, а также имеет хороший API
4. GridFS - файловая система, которая поставляется вместе с MongoDB и служит для хранения больших файлов

6.2 Формат представления данных в БД

Так как в качестве СУБД была выбрана документо-ориентированная MongoDB, то и структура базы данных будет приводится в её терминологии.

Основной и единственной базой данных является база «эксперименты» ("experiments"). В ней находится коллекция «данные» («data»). Данная коллекция содержит в себе документы, каждый из которых является проведенным экспериментом. Каждый из документов хранит в себе следующие поля:

1. дата проведения эксперимента
2. время выполнения эксперимента
3. уникальный идентификатор, указывающий на изображения, полученные в результате экспериментов, которые хранятся в GridFS

4. матрица решения системы ОДУ
5. вектор, хранящий в себе моменты времени, в которых была решена ОДУ
6. тип и файл сетки
7. количество базисных функций

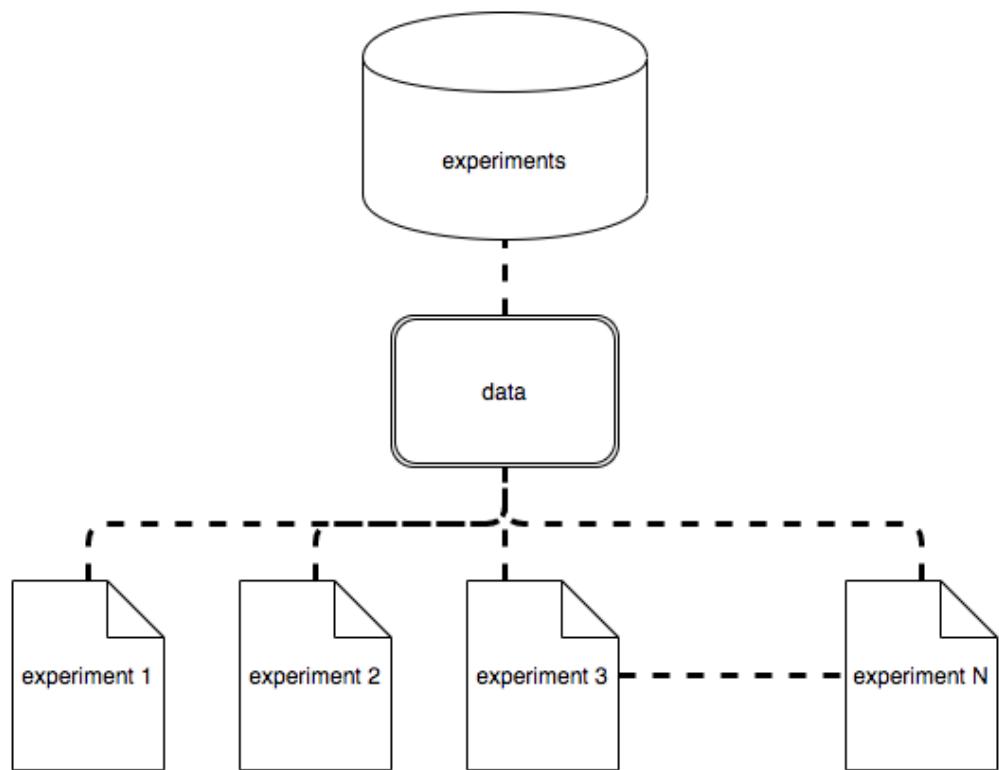


Рисунок 6.1 — Структура БД

Документ, который описывает эксперимент, имеет следующую структуру (в подобном виде он хранится в MongoDB):

```

{
  "date": <date>,
  "images": {
    "frame": {
      "fluid_flow": {
        "1": frame_q1_id,
        "2": frame_q2_id
      },
      "surface_elevation": frame_H_id
    },
    "surface": {
      "fluid_flow": {
        "1": surf_q1_id,
        "2": surf_q2_id
      },
      "surface_elevation": surf_H_id
    },
    "stream_function": {
      "1": wave_psi1_id,
      "2": wave_psi2_id
    }
  },
  "solution": {
    "matrix": <matrix_data>,
    "times": <times_data>
  },
  "meta": {
    "mesh": {
      "type": mesh_type,
      "name": mesh_name
    },
    "execution_time": <time>,
    "quantity_of_basis_functions": <quantity_of_fe>
  }
}

```

Поля с постфиксами `id` являются уникальными идентификаторами для GIF анимаций, которые сохранены на файловой системе GridFS. Благодаря этому изображения результатов экспериментов не пропадут в случае отказа одной из нод кластера MongoDB. Таким образом обеспечивается надежная сохранность данных.

6.3 Примеры запросов к БД

Для того чтобы использовать данные, сохраненные в БД в результате экспериментов, можно использовать запросы вида:

```
db.data.find({ "date" : "2018-04-24-20-07" })
```

Это не очень удобно, так как результатами экспериментов является не только текстовая информация, но еще и графическая. В случае прямого запроса в ответе будут бинарные объекты, с которыми тяжело работать в чистом виде. Именно из-за этого для более простой работы с базой данных была написана функция `save(db, data, dir)`, которая находится в модуле `fem.db.mongo`. На вход данной функции подается JSON, который является результатом прямого запроса к базе, а также папка, в которую будут сохранены графики эксперимента. Помимо функции `save(db, data, dir)`, в том же пакете находится функция `read(db, collection, date)`, которая позволяет пользователю не делать прямые запросы в базу и облегчает работу с результатами. Комбинированный пример получения данных из базы выглядит следующим образом:

```
save(db, read(db, 'data', '2018-04-24-20-07'), '/data')
```

Данный вызов, во-первых, найдет в базе данных документ, в котором хранится информация о результатах эксперимента, который был проведен 2018-04-24-20-07, а во-вторых, сохранит все необходимые результаты на диск в папку `/data`. Это очень удобно, так как не требует от конечного пользователя знаний о структуре запросов в MongoDB.

ЗАКЛЮЧЕНИЕ

В представленной бакалаврской работе были получены уравнения мелкой воды при пренебрежении температурными эффектами, а также написана многопоточная программная реализация МКЭ для решений данных уравнений.

Реализована возможность переносимости разработанного алгоритма, а также возможность его удаленного использования посредством контейнеризации.

В качестве примера были произведены расчеты на различных сетках, в том числе и на реальных озерах(Эльтон и Верхнее). Для данных расчетов были построены графики функций потока жидкости, тока и $H(?)$. Благодаря этому удалось качественно оценить полученные результаты проведенных экспериментов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. A Two-Dimensional Quality Mesh Generator and Delaunay Triangulator. — URL: <https://www.cs.cmu.edu/~quake/triangle.html> (online; accessed: 22.04.2017).
2. Норри., Д., Фриз., Ж. Введение в метод конечных элементов. — М.: Мир, 1981. — С. 304.
3. Панкратов, И.А., Рымчук, Д.С. Расчет течения мелкой воды // Математика. Механика. — 2014. — Т. 16. — С. 121–124.
4. Панкратов, И.А. Об аппроксимации нестационарных уравнений мелкой воды // Juvenis scientia. — 2016. — Т. 5. — С. 3–4.
5. Lutz, M. Learning Python, 5th Edition. — O'Reilly Media, 2013. — Р. 1648. — ISBN: 978-1449355739.
6. Kiusalaas, J. Numerical Methods in Engineering with Python 3, 3rd Edition. — New York: Cambridge University Press, 2013. — Р. 432. — ISBN: 978-1107033856.
7. Коннор, Дж., Бреббиа, К. Метод конечных элементов в механике жидкости. — Л.: Судостроение, 1979. — С. 264.
8. Зорич, В. А. Математический анализ. — М.: ФАЗИС, 1997. — Т. 1. — С. 554. — ISBN: 5-7036-0031-6.
9. Юрко, В. А. Уравнения математической физики : учеб. пособие для студентов механико-математического и физического факультетов. — Саратов: Изд-во Сарат. ун-та, 2004. — С. 118.
10. Скворцов, А. В. Триангуляция Делоне и её применение. — Издательство Томского университета, 2002.
11. Ruppert, J. A Delaunay Refinement Algorithm for Quality 2-Dimensional Mesh Generation. — NASA Ames Research Center, 1995.

12. Rand, A. Where and How Chew's Second Delaunay Refinement Algorithm Works. — URL: <http://2011.cccg.ca/PDFschedule/papers/paper91.pdf> (дата обращения: 20.04.2017).
13. Панкратов, И.А. Введение в методы взвешенных невязок. — URL: http://www.sgu.ru/sites/default/files/textdocsfiles/2013/12/10/fem_introduction.pdf (дата обращения: 20.04.2017).
14. Зенкевич, О., Морган, К. Конечные элементы и аппроксимация. — М.: Мир, 1986. — С. 318.
15. Голованов, Н. Н. Геометрическое моделирование. — М.: Издательство Физико-математической литературы, 2002. — С. 472.
16. Гуревич, А. П. Основы теории обыкновенных дифференциальных уравнений. — Саратов: Изд-во СГУ, 2013. — С. 176.
17. Mehta, H. K. Mastering Python Scientific Computing. — Packt Publishing, 2015. — P. 345. — ISBN: 978-1783288823.
18. OpenStreetMap. — URL: <https://www.openstreetmap.org> (online; accessed: 22.04.2017).
19. Over pass turbo. — URL: <https://overpass-turbo.eu/> (online; accessed: 22.04.2017).
20. Seguin., K. The Little MongoDB Book. — Self-Published, 2013. — P. 34. — ISBN: 978-1493786602.

ПРИЛОЖЕНИЕ А

Исходный код реализации

TODO: код будет добавлен в самом конце, после того, как я перестану вносить в него изменения

ПРИЛОЖЕНИЕ Б

Исходный код Docker контейнера

Предполагается, что рядом с файлом для сборки контейнера лежит файл requirements.txt, а также исходный код реализации в модуле fem.

Листинг Б.1: Dockerfile

```
1 FROM python:3
2
3 RUN mkdir -p /opt/diploma
4 WORKDIR /opt/diploma/fem
5
6 ENV DISPLAY=:0.0 \
7     MPLBACKEND="agg"
8
9 COPY ./python/requirements.txt /opt/diploma
10 COPY ./python/fem /opt/diploma
11 RUN pip3 install --upgrade pip
12 RUN pip3 install --no-cache-dir -r /opt/diploma/
    requirements.txt
```

Листинг Б.2: requirements.txt

```
1 pymongo
2 numpy
3 sympy
4 scipy
5 matplotlib
6 shapely
7 descartes
8 triangle
9 jupyter
10 ipython
11 bson
12 pillow
13 imageio
```

ПРИЛОЖЕНИЕ В

Выкладки для МКЭ

Запишем уравнение неразрывности:

$$\frac{d}{dx_1}q_1 + \frac{d}{dx_2}q_2 + \frac{\partial}{\partial t}(H\rho) = 0 \quad (\text{B.1})$$

Подставим в него разложения q_1, q_2, H :

$$\begin{aligned} \frac{\partial}{\partial t} \left(\rho \left(H_0(x_1, x_2) + \sum_{k=1}^M N_k(x_1, x_2, k) a_{2m+k}(t) \right) \right) + \\ \frac{\partial}{\partial x_1} \sum_{k=1}^M N_k(x_1, x_2, k) a_k(t) + \frac{\partial}{\partial x_2} \sum_{k=1}^M N_k(x_1, x_2, k) a_{m+k}(t) = 0 \end{aligned}$$

Умножим уравнение на $\sum_{l=1}^M W_l$, где W_l - весовая функция, и проинтегрируем по треугольнику Δ :

$$\begin{aligned} \sum_{l=1}^M \int_{\Delta} \left(\rho \sum_{k=1}^M N_k(x_1, x_2) \frac{d}{dt} a_{2m+k}(t) + \sum_{k=1}^M a_k(t) \frac{\partial}{\partial x_1} N_k(x_1, x_2) + \right. \\ \left. \sum_{k=1}^M a_{m+k}(t) \frac{\partial}{\partial x_2} N_k(x_1, x_2) \right) W_l(x_1, x_2) d\Delta = 0 \end{aligned}$$

Упростим выражение:

$$\begin{aligned} \sum_{l=1}^M \int_{\Delta} \rho W_l(x_1, x_2) \frac{d}{dt} a_{2m+k}(t) \sum_{k=1}^M N_k(x_1, x_2) d\Delta = \\ \left[- \sum_{l=1}^M \left(\int_{\Delta} W_l(x_1, x_2) a_k(t) \sum_{k=1}^M \frac{\partial}{\partial x_1} N_k(x_1, x_2) d\Delta + \right. \right. \\ \left. \left. \int_{\Delta} W_l(x_1, x_2) a_{m+k}(t) \sum_{k=1}^M \frac{\partial}{\partial x_2} N_k(x_1, x_2) d\Delta \right) \right] \end{aligned}$$

Разрешим его относительно $\frac{d}{dt}a_{2m+k}(t)$:

$$\begin{aligned}\frac{d}{dt} a_{2m+k}(t) &= \frac{1}{\sum_{l=1}^M \int_{\Delta} \rho W_l(x_1, x_2) \sum_{k=1}^M N_k(x_1, x_2) d\Delta} \cdot \\ &\left[- \sum_{l=1}^M \left(\int_{\Delta} W_l(x_1, x_2) a_k(t) \sum_{k=1}^M \frac{\partial}{\partial x_1} N_k(x_1, x_2) d\Delta + \right. \right. \\ &\quad \left. \left. \int_{\Delta} W_l(x_1, x_2) a_{m+k}(t) \sum_{k=1}^M \frac{\partial}{\partial x_2} N_k(x_1, x_2) d\Delta \right) \right]\end{aligned}$$

TODO: тут будут добавлены расчеты для eq1, eq2