Содержание

Введение **3**

1. Математическая модель **4**
1.1. Уравнения Максвелла и Гельмгольца 4
1.2. Вариационная постановка 5
1.3. Вариационная постановка с учетом PML-слоя 7
1.4. Дискретная вариационная постановка 8
1.5. Тетраэдральные конечные элементы 10
1.6. Треугольные конечные элементы 11

2. Вычислительные эксперименты **13**
2.1. Верификация программного комплекса 13
2.1.1 Расчетная область 13
2.1.2 Тестирование на линейных по пространству функциях 13
2.1.3 Тестирование на полиномиальных функциях 14
2.1.4 Определение порядка аппроксимации 14
2.2. Исследование влияния слоя воздуха 15
2.2.1 Описание расчетной области 15
2.2.2 Конечноэлементная сетка 16
2.2.3 Результаты исследования 17
2.3. Исследование эффективности применения PML-слоя 18
2.3.1 Описание расчетной области 19
2.3.2 Варьирование коэффициентов растяжения 20
2.3.3 Варьирование толщины PML-слоя 20
2.3.4 Варьирование размера области, на границе которой вводится PML-слой 21
2.3.5 Проверка выполнения условий на контактных границах 22
2.3.6 Графическое представление результатов 22
2.3.7 Анализ целесообразности применения PML-слоя 24

Заключение **25**

Список литературы **26**

Приложение А. Текст основного модуля программы **28**

Введение

В современном мире сложилась ситуация, что экономика многих стран, в число которых входят Россия, Швеция, Канада, Объединенные Арабские Эмираты, зависит от цены на нефть. Цены на углеводороды могут расти или падать, но конкуренция за обладание ими всегда велика и доходит порой до вооруженных конфликтов. Особенно актуальными в последнее время становятся задачи геологоразведки в недрах земли, скрытых под толщей морской воды, ведь, по оценкам специалистов, на территории только Северного Ледовитого океана может находиться до 25 процентов мировых запасов нефти и газа [1]

К отличительным особенностям задач морской геоэлектрики относится низкая частота источника электромагнитного поля (0.25-100 Гц) [2] и, как следствие, большой размер области моделирования. Кроме того, морское дно имеет сложный рельеф, а электропроводность морской воды может изменяться в зависимости от глубины [1]. Это вызвано различной соленостью и температурой разных слоев морской воды, эти свойства, кроме того, могут меняться от внешних факторов, таких как сезон, погодные условия или интенсивность таяния льдов.

Геометрические размеры локального источника возбуждения электромагнитного поля составляют несколько сотен метров, тогда как размеры области моделирования составляют 6000 м и более. Это приводит к необходимости применения специальных методов для сокращения расчетной области. Для этого нередко в область моделирования не включается воздух, вместо этого на границе раздела сред воздух-вода задаются условия непротекания. Однако такой подход не позволяет правильно учесть физические процессы, протекающие в воздухе [3]. Другим подходом является выделение из области некоторой подобласти меньшего размера и задание на ее границах специальных поглощающих условий. К таким условиям относятся Absorbing Boundary Conditions (ABC) [4], предложенные G. Mur в 1981 году, а также Perfectly Matched Layer (PML) [5, 6], который предложил J.P. Berenger в 1994 году. PML-слой учитывается в вариационной формулировке как подобласть со специальными коэффициентами.

В настоящее время для решения задач морской геоэлектрики наиболее широко используется векторный метод конечных элементов (ВМКЭ). Этот метод подробно освещен в работах [7, 8].

Целью работы является решение трехмерной прямой задачи морской геоэлектрики векторным методом конечных элементов. Для достижения поставленной цели были сформулированы следующие задачи:

1. Исследование влияния слоя воздуха при различной глубине источника электромагнитного возмущения.
2. Исследование целесообразности применения PML-слоя для ограничения области моделирования в задачах морской геоэлектрики на низких частотах.

1. Математическая модель

1.1. Уравнения Максвелла и Гельмгольца

Электромагнитное поле описывается системой уравнений Максвелла [9]:

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \quad \text{– закон Фарадея,} \tag{1}$$

$$\nabla \times \mathbf{H} = \frac{\partial \mathbf{D}}{\partial t} + \sigma \mathbf{E} + \mathbf{J} \quad \text{– закон Ампера,} \tag{2}$$

$$\nabla \cdot \mathbf{B} = 0 \quad \text{– закон Гаусса для магнитной индукции,}$$

$$\nabla \cdot \mathbf{D} = \rho \quad \text{– закон Гаусса для электрической индукции,}$$

где $\mathbf{E}$ – напряженность электрического поля (В/м), $\mathbf{H}$ – напряженность магнитного поля (А/м), $\mathbf{B} = \mu \mathbf{H}$ – магнитная индукция (Тл), $\mathbf{D} = \varepsilon \mathbf{E}$ – электрическая индукция (Кл/м^2), ρ – плотность электрических зарядов (Кл/м^3), σ – электрическая проводимость (См/м), $\varepsilon = \varepsilon_r \varepsilon_0$ – диэлектрическая проницаемость (Ф/м), ε_r – относительная диэлектрическая проницаемость, $\varepsilon_0 = 8.85 \cdot 10^{-12}$ Ф/м – диэлектрическая проницаемость вакуума, $\mu = \mu_r \mu_0$ – магнитная проницаемость (Гн/м), μ_r – относительная магнитная проницаемость, $\mu_0 = 4\pi \cdot 10^{-7}$ Гн/м – магнитная проницаемость вакуума, $\mathbf{J}$ – плотность стороннего электрического тока (А/м^2).

На границе $\Gamma = \Omega^j \cap \Omega^k$ между материалами j и k с различными электрофизическими свойствами должны быть выполнены следующие условия:

$$[\![\mathbf{E} \times \mathbf{n}]\!]_\Gamma = 0 \quad \text{– тангенциальная компонента } \mathbf{E} \text{ непрерывна,} \tag{3}$$

$$[\![\mathbf{B} \cdot \mathbf{n}]\!]_\Gamma = 0 \quad \text{– нормальная компонента } \mathbf{B} \text{ непрерывна,}$$

$$[\![\mathbf{H} \times \mathbf{n}]\!]_\Gamma = \mathbf{J}_\Gamma \quad \text{– тангенциальная компонента } \mathbf{H} \text{ разрывна,}$$

$$[\![\mathbf{D} \cdot \mathbf{n}]\!]_\Gamma = \rho_\Gamma \quad \text{– нормальная компонента } \mathbf{D} \text{ разрывна.} \tag{4}$$

При моделировании электрического поля в частотной области будем полагать, что $\mathbf{E}$ и $\mathbf{J}$ будут зависеть от времени по гармоническому закону:

$$\mathbf{E}(t) = \mathbf{E} e^{i\omega t}, \qquad \mathbf{J}(t) = \mathbf{J} e^{i\omega t}.$$

Используя такое представление, получим из (1) и (2):

$$\nabla \times \mathbf{E} = -i\omega \mathbf{B}, \tag{5}$$

$$\nabla \times \mathbf{H} = i\omega \mathbf{D} + \sigma \mathbf{E} + \mathbf{J}. \tag{6}$$

Выполним следующие преобразования над (5):

$$\nabla \times \mathbf{E} = -i\omega\mu\mathbf{H},$$

$$\mu^{-1}\nabla \times \mathbf{E} = -i\omega\mathbf{H},$$

$$\nabla \times (\mu^{-1}\nabla \times \mathbf{E}) = -i\omega\nabla \times \mathbf{H}. \tag{7}$$

Подставим в (7) (6):

$$\nabla \times (\mu^{-1}\nabla \times \mathbf{E}) = -i\omega(i\omega\varepsilon\mathbf{E} + \sigma\mathbf{E} + \mathbf{J}),$$

$$\nabla \times (\mu^{-1}\nabla \times \mathbf{E}) = \omega^2\varepsilon\mathbf{E} - i\omega\sigma\mathbf{E} - i\omega\mathbf{J},$$

$$\nabla \times (\mu^{-1}\nabla \times \mathbf{E}) + k^2\mathbf{E} = -i\omega\mathbf{J}, \tag{8}$$

где $k^2 = i\omega\sigma - \omega^2\varepsilon$. Уравнение (8) называется уравнением Гельмгольца.

Краевые условия для уравнения (8) можно записать следующим образом:

$$\mathbf{E} \times \mathbf{n}|_{S_1} = \mathbf{E}^g, \tag{9}$$

$$\sigma\mathbf{E} \cdot \mathbf{n}|_{S_2} = 0. \tag{10}$$

В случае удаленных границ (9) принимает вид условия «большого бака»:

$$\mathbf{E} \times \mathbf{n}|_{S_1} = 0. \tag{11}$$

Источником электромагнитного возмущения будет выступать замкнутая токовая петля.

Подействуем оператором $\nabla\cdot$ на уравнение (2):

$$\nabla \cdot (\nabla \times \mathbf{H}) = \nabla \cdot (\frac{\partial \mathbf{D}}{\partial t} + \sigma\mathbf{E} + \mathbf{J}).$$

Так как $\nabla \cdot (\nabla \times \mathbf{H}) = 0$, $\nabla \cdot \frac{\partial \mathbf{D}}{\partial t} = \nabla \cdot \frac{\partial \varepsilon \mathbf{E}}{\partial t} = \nabla \cdot (i\omega\varepsilon\mathbf{E})$ и, так как для замкнутой петли с током выполняется $\nabla \cdot \mathbf{J} = 0$, получим закон сохранения заряда:

$$\nabla \cdot (\sigma + i\omega\varepsilon)\mathbf{E} = 0. \tag{12}$$

1.2. Вариационная постановка

Введем следующие пространства [7, 8]:

$$\mathbb{H}(\mathrm{rot}\,, \Omega) = \{\mathbf{v} \in [\mathbb{L}^2(\Omega)]^3 : \nabla \times \mathbf{v} \in [\mathbb{L}^2(\Omega)]^3\},$$

$$\mathbb{H}_0(\mathrm{rot}\,, \Omega) = \{\mathbf{v} \in \mathbb{H}(\mathrm{rot}\,, \Omega) : \mathbf{v} \times \mathbf{n}|_{\partial\Omega} = 0\}.$$

Скалярное произведение в этих пространствах имеет вид:

$$(\mathbf{u}, \mathbf{v}) = \int\limits_{\Omega} \mathbf{u} \cdot \overline{\mathbf{v}}\, d\Omega.$$

Скалярно умножим (8) на некоторую пробную функцию $\mathbf{v} \in \mathbb{H}_0(\mathrm{rot}\,, \Omega)$:

$$(\nabla \times (\mu^{-1}\nabla \times \mathbf{E}), \mathbf{v}) + (k^2\mathbf{E}, \mathbf{v}) = -(i\omega\mathbf{J}, \mathbf{v}),$$

$$\int\limits_\Omega \nabla \times (\mu^{-1}\nabla \times \mathbf{E}) \cdot \overline{\mathbf{v}}\, d\Omega + \int\limits_\Omega k^2\mathbf{E} \cdot \overline{\mathbf{v}}\, d\Omega = -\int\limits_\Omega i\omega\mathbf{J} \cdot \overline{\mathbf{v}}\, d\Omega.$$

Воспользовавшись первой векторной формулой Грина (13):

$$\int\limits_D \nabla \times \mathbf{u} \cdot \overline{\mathbf{v}}\, dV = \int\limits_D \mathbf{u} \cdot (\nabla \times \overline{\mathbf{v}})\, dV + \int\limits_{\partial D} (\mathbf{n} \times \mathbf{u}) \cdot \overline{\mathbf{v}}\, dS, \tag{13}$$

получим:

$$\int\limits_\Omega \mu^{-1}\nabla \times \mathbf{E} \cdot \nabla \times \overline{\mathbf{v}}\, d\Omega + \int\limits_{\partial\Omega} \mathbf{n} \times (\mu^{-1}\nabla \times \mathbf{E}) \cdot \overline{\mathbf{v}}\, dS + \\ + \int\limits_\Omega k^2\mathbf{E} \cdot \overline{\mathbf{v}}\, d\Omega = -\int\limits_\Omega i\omega\mathbf{J} \cdot \overline{\mathbf{v}}\, d\Omega.$$

Применим тождества $(\mathbf{a} \times \mathbf{b}) \cdot \mathbf{c} = -(\mathbf{a} \times \mathbf{c}) \cdot \mathbf{b}$ и $\mathbf{a} \times \mathbf{b} = -\mathbf{b} \times \mathbf{a}$:

$$\int\limits_\Omega \mu^{-1}\nabla \times \mathbf{E} \cdot \nabla \times \overline{\mathbf{v}}\, d\Omega + \int\limits_\Omega k^2\mathbf{E} \cdot \overline{\mathbf{v}}\, d\Omega = \\ = -\int\limits_\Omega i\omega\mathbf{J} \cdot \overline{\mathbf{v}}\, d\Omega - \int\limits_{\partial\Omega} \overline{\mathbf{v}} \times \mathbf{n} \cdot (\mu^{-1}\nabla \times \mathbf{E})\, dS. \tag{14}$$

Так как $\mathbf{v} \in \mathbb{H}_0(\mathrm{rot}\,, \Omega)$, то из свойств пространства $\mathbb{H}_0(\mathrm{rot}\,, \Omega)$ второй интеграл в правой части равен нулю, тогда уравнение (14) примет вид:

$$\int\limits_\Omega \mu^{-1}\nabla \times \mathbf{E} \cdot \nabla \times \overline{\mathbf{v}}\, d\Omega + \int\limits_\Omega k^2\mathbf{E} \cdot \overline{\mathbf{v}}\, d\Omega = -\int\limits_\Omega i\omega\mathbf{J} \cdot \overline{\mathbf{v}}\, d\Omega. \tag{15}$$

В результате векторная вариационная постановка имеет вид: ***Найти*** $\mathbf{E} \in \mathbb{H}_0(\mathrm{rot}\,, \Omega)$***,*** ***такое что*** $\forall \mathbf{v} \in \mathbb{H}_0(\mathrm{rot}\,, \Omega)$ ***будет выполнено*** (15).

Введем еще два пространства [8]

$$\mathbb{H}(\mathrm{grad}\,, \Omega) = \{\varphi \in \mathbb{L}^2(\Omega) : \nabla\varphi \in [\mathbb{L}^2(\Omega)]^3\},$$

$$\mathbb{H}_0(\mathrm{grad}\,, \Omega) = \{\varphi \in \mathbb{H}(\mathrm{grad}\,, \Omega) : \varphi|_{\partial\Omega} = 0\}.$$

В соответствии с комплексом Де Рама (De Rham) [10]

$$\mathbb{H}(\mathrm{grad}\,, \Omega) \xrightarrow{\nabla} \mathbb{H}(\mathrm{rot}\,, \Omega) \xrightarrow{\nabla\times} \mathbb{H}(\mathrm{div}\,, \Omega) \xrightarrow{\nabla\cdot} \mathbb{L}^2(\Omega), \tag{16}$$

будет иметь место вложение $\nabla\varphi \in \mathbb{H}_0(\mathrm{rot}\,, \Omega)$, $\forall\varphi \in \mathbb{H}_0(\mathrm{grad}\,, \Omega)$. Возьмем $\mathbf{v} = \nabla\varphi$, тогда (15) примет вид:

$$\int\limits_\Omega \mu^{-1}\nabla \times \mathbf{E} \cdot \nabla \times (\nabla\overline{\varphi})\, d\Omega + \int\limits_\Omega k^2\mathbf{E} \cdot (\nabla\overline{\varphi})\, d\Omega = -\int\limits_\Omega i\omega\mathbf{J} \cdot (\nabla\overline{\varphi})\, d\Omega.$$

Использовав свойство дивергенции $\nabla \cdot (\varphi \mathbf{F}) = \nabla\varphi \cdot \mathbf{F} + \varphi \nabla \cdot \mathbf{F}$ и применив формулу Остроградского-Гаусса (17)

$$\int\limits_D \nabla \cdot \mathbf{F}\, dV = \int\limits_{\partial D} \mathbf{F} \cdot \mathbf{n}\, dS, \tag{17}$$

получим:

$$\int\limits_\Omega \mu^{-1} \nabla \times \mathbf{E} \cdot \nabla \times (\nabla\overline{\varphi})\, d\Omega + \int\limits_\Omega k^2\, \mathbf{E} \cdot (\nabla\overline{\varphi})\, d\Omega = -\int\limits_\Omega i\omega\overline{\varphi} \nabla \cdot \mathbf{J}\, d\Omega - \int\limits_{\partial\Omega} i\omega\overline{\varphi} \mathbf{J} \cdot \mathbf{n}\, dS.$$

Поскольку $\nabla \times (\nabla\varphi) = 0$, $\nabla \cdot \mathbf{J} = 0$ и $\varphi|_{\partial\Omega} = 0$, в левой части останется только один интеграл:

$$\int\limits_\Omega k^2\, \mathbf{E} \cdot (\nabla\overline{\varphi})\, d\Omega = 0.$$

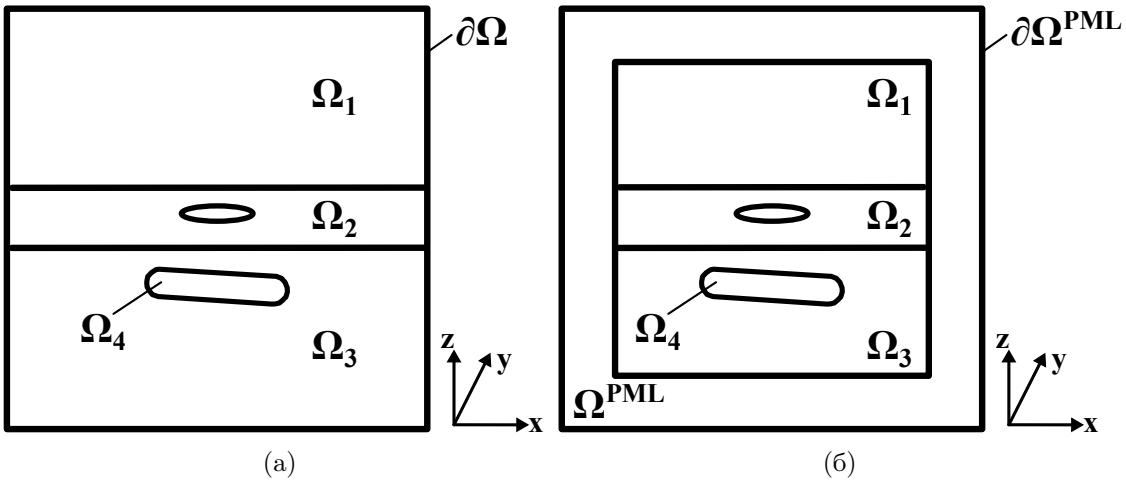
После преобразований получим:

$$\int\limits_\Omega \overline{\varphi} \nabla \cdot (k^2\, \mathbf{E})\, d\Omega = 0,$$

следовательно, решение вариационной задачи (15) удовлетворяет закону сохранения заряда (12) в слабом смысле.

1.3. Вариационная постановка с учетом PML-слоя

Для ограничения расчетной области введем PML-слой Ω^{PML}, который является подобластью основной расчетной области Ω со специальными коэффициентами, построенными таким образом, чтобы обеспечить полное поглощение электрического поля внутри слоя и не допустить его отражения от внутренних границ и прохождения через внешние границы слоя (рисунок 1).



(а) (б)

Рисунок 1 — расчетные области: (а) без PML-слоя и (б) с PML-слоем

PML-слой определяется модифицированными координатами $\tilde{x}$, $\tilde{y}$, $\tilde{z}$, полученными следующей заменой координат [6]:

$$\tilde{x} = \int_0^x s_x(t)\,dt, \qquad \tilde{y} = \int_0^y s_y(t)\,dt, \qquad \tilde{z} = \int_0^z s_z(t)\,dt,$$

где $s_j(\tau) = 1$ вне PML-слоя, а внутри него может быть задано в виде:

$$s_j(\tau) = 1 + \chi \left(\frac{d(\tau)}{\delta}\right)^m, \quad m \geq 1, \tag{18}$$

где $d(\tau)$ – расстояние в j-м направлении от внутренней границы PML-слоя, δ – толщина PML-слоя, χ – некоторое комплексное число, причем $\mathrm{Re}(\chi) \geq 0$, $\mathrm{Im}(\chi) \geq 0$. Оператор ∇ в новых координатах будет иметь вид:

$$\tilde{\nabla} = \left[\frac{1}{s_x}\frac{\partial}{\partial x}, \frac{1}{s_y}\frac{\partial}{\partial y}, \frac{1}{s_z}\frac{\partial}{\partial z}\right].$$

После такой замены, внутри PML-слоя уравнение Гельмгольца (8) будет иметь вид (19)

$$\tilde{\nabla} \times (\mu^{-1}\tilde{\nabla} \times \tilde{\mathbf{E}}) + k^2\tilde{\mathbf{E}} = 0, \tag{19}$$

что приведет к преобразованию уравнения (15) к виду (20):

$$\int_{\Omega^{PML}} \mu^{-1}\tilde{\nabla} \times \tilde{\mathbf{E}} \cdot \tilde{\nabla} \times \overline{\mathbf{v}}\, d\Omega^{PML} + \int_{\Omega^{PML}} k^2\tilde{\mathbf{E}} \cdot \overline{\mathbf{v}}\, d\Omega^{PML} = 0. \tag{20}$$

В результате, если обозначить $\widehat{\Omega} = \Omega \setminus \Omega^{PML}$, то векторная вариационная постановка с учетом PML-слоя примет вид: ***Найти*** $\mathbf{E} \in \mathbb{H}_0(\mathrm{rot}\,, \widehat{\Omega})$ ***и*** $\tilde{\mathbf{E}} \in \mathbb{H}_0(\mathrm{rot}\,, \Omega^{PML})$***, такие что*** $\forall \mathbf{v} \in \mathbb{H}_0(\mathrm{rot}\,, \widehat{\Omega})$ ***и*** $\forall \tilde{\mathbf{v}} \in \mathbb{H}_0(\mathrm{rot}\,, \Omega^{PML})$ ***будет выполнено***:

$$\begin{cases} \displaystyle\int_{\widehat{\Omega}} \mu^{-1}\nabla \times \mathbf{E} \cdot \nabla \times \overline{\mathbf{v}}\, d\widehat{\Omega} + \int_{\widehat{\Omega}} k^2\mathbf{E} \cdot \overline{\mathbf{v}}\, d\widehat{\Omega} = -\int_{\widehat{\Omega}} i\omega\mathbf{J} \cdot \overline{\mathbf{v}}\, d\widehat{\Omega} \\ \displaystyle\int_{\Omega^{PML}} \mu^{-1}\tilde{\nabla} \times \tilde{\mathbf{E}} \cdot \tilde{\nabla} \times \overline{\tilde{\mathbf{v}}}\, d\Omega^{PML} + \int_{\Omega^{PML}} k^2\tilde{\mathbf{E}} \cdot \overline{\tilde{\mathbf{v}}}\, d\Omega^{PML} = 0. \end{cases}$$

1.4. Дискретная вариационная постановка

Разобьем область Ω на m непересекающихся элементов:

$$\Omega = \bigcup_{k=1}^{m} \Omega_k, \quad \forall i \neq j, \quad \Omega_i \cap \Omega_j = \varnothing.$$

Введем конечномерные подпространства:

$$\mathbb{H}_0^h(\mathrm{rot}\,, \Omega) \subset \mathbb{H}_0(\mathrm{rot}\,, \Omega), \quad \mathbb{H}_0^h(\mathrm{grad}\,, \Omega) \subset \mathbb{H}_0(\mathrm{grad}\,, \Omega).$$

Для дискретных подпространств $\mathbb{H}_0^h(\mathrm{rot}\,,\Omega)$ и $\mathbb{H}_0^h(\mathrm{grad}\,,\Omega)$ комплекс Де Рама (16) также будет верен, следовательно закон сохранения заряда (12) будет также выполнен в слабом смысле [9].

Пространство $\mathbb{H}_0^h(\mathrm{rot}\,,\Omega)$ является прямой суммой подпространств [9, 11]

$$\mathbb{H}_0^h(\mathrm{rot}\,,\Omega) = \mathbb{N}_0^h(\mathrm{rot}\,,\Omega) \oplus (\mathbb{N}_0^h(\mathrm{rot}\,,\Omega))^{\perp}, \tag{21}$$

где $\mathbb{N}_0^h(\mathrm{rot}\,,\Omega)$ – ядро rot-оператора, $(\mathbb{N}_0^h(\mathrm{rot}\,,\Omega))^{\perp}$ – его ортогональное дополнение. Для выполнения условий непрерывности (3)-(4) необходимо использовать полный базис (базис II типа) [9, 12–14, 17], состоящий из двух типов базисных функций. Первый тип – роторные базисные функции из пространства $(\mathbb{N}_0^h(\mathrm{rot}\,,\Omega))^{\perp}$, которые обеспечивают непрерывность тангенциальных компонент электрического поля $\mathbf{E}$ (3). Второй – градиентные базисные функции из пространства $\mathbb{N}_0^h(\mathrm{rot}\,,\Omega)$, которые обеспечивают скачок нормальной компоненты электрического поля $\mathbf{E}$ (4) и выполнение закона сохранения заряда (12).

Представим векторнозначную функцию $\mathbf{E}^h$ в виде разложения по базису $\boldsymbol{\psi}_j \in \mathbb{H}_0^h(\mathrm{rot}\,,\Omega)$:

$$\mathbf{E}^h = \sum_{j=1}^{n} q_j \boldsymbol{\psi}_j.$$

В качестве тестовой функции выберем базисную функцию $\boldsymbol{\psi}_i \in \mathbb{H}_0^h(\mathrm{rot}\,,\Omega)$, тогда конечноэлементная аппроксимация вариационного уравнения (15) примет вид:

$$\begin{gathered}\sum_{j=1}^{n}\left(\int\limits_{\Omega} \mu^{-1}\nabla\times\boldsymbol{\psi}_j\cdot\nabla\times\boldsymbol{\psi}_i\,d\Omega + \int\limits_{\Omega} k^2\boldsymbol{\psi}_j\cdot\boldsymbol{\psi}_i\,d\Omega\right) q_j = \\ = -\int\limits_{\Omega} i\omega\mathbf{J}\cdot\boldsymbol{\psi}_i\,d\Omega.\end{gathered} \tag{22}$$

В матрично-векторной форме (22) можно представить в виде следующей системы линейных алгебраических уравнений (СЛАУ):

$$(\mathbf{G}+\mathbf{M})\mathbf{q} = \mathbf{f}, \tag{23}$$

$$\mathbf{G}_{i,j} = \int\limits_{\Omega_k} \mu^{-1}\nabla\times\mathbf{w}_i\cdot\nabla\times\mathbf{w}_j\,d\Omega_k, \quad \mathbf{M}_{i,j} = \int\limits_{\Omega_k} k^2\mathbf{w}_i\cdot\mathbf{w}_j\,d\Omega_k.$$

Матрица СЛАУ будет иметь симметричную разреженную структуру, поэтому ее удобно хранить в формате CSLR (Compressed Sparse (Lower triangle) Row) или CSR (Compressed Sparse Row) [15].

1.5. Тетраэдральные конечные элементы

В качестве конечных элементов для представления расчетной области, будем пользоваться тетраэдрами. На тетраэдральном конечном элементе определим $\mathcal{L}$-координаты, называемые также барицентрическими координатами [16]. Введем нумерацию вершин и ребер, показанную на рисунке 2:

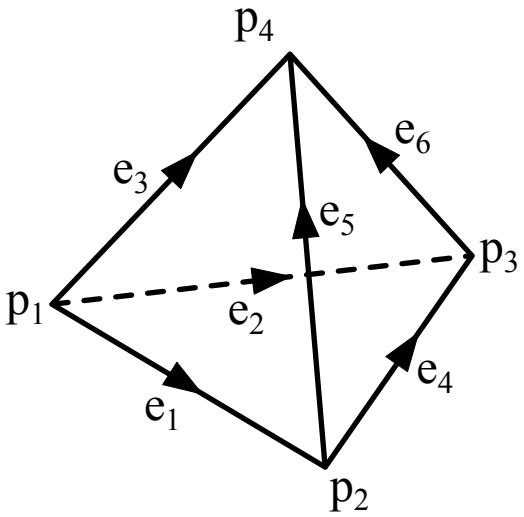


Рисунок 2 — тетраэдральный конечный элемент

Под $\mathcal{L}$-координатами понимают функции следующего вида:

$$\mathcal{L}_i(x, y, z) = \alpha_{i,1}x + \alpha_{i,2}y + \alpha_{i,3}z + \alpha_{i,4}, \quad i = \overline{1..4}.$$

Коэффициенты $\alpha_{i,j}$ могут быть определены по формуле (24):

$$\begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} \\ \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} \\ \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} \\ \alpha_{4,1} & \alpha_{4,2} & \alpha_{4,3} & \alpha_{4,4} \end{bmatrix} = \begin{bmatrix} p_{1_x} & p_{2_x} & p_{3_x} & p_{4_x} \\ p_{1_y} & p_{2_y} & p_{3_y} & p_{4_y} \\ p_{1_z} & p_{2_z} & p_{3_z} & p_{4_z} \\ 1 & 1 & 1 & 1 \end{bmatrix}^{-1}. \tag{24}$$

Задав $\mathcal{L}$-координаты, можно определить на тетраэдре базисные функции. В отличие от узлового метода конечных элементов, в векторном методе конечных элементов базисные функции ассоциированы не с узлами, а с ребрами (edge), гранями (face) и объемами (volume) [17, 18]. Так как будут использованы полные (II типа) базисы первого и второго порядков, то ограничимся рассмотрением только базисных функций, ассоциированных с ребрами и гранями.

Иерархический векторный базис Вебба второго порядка второго типа на тетраэдрах имеет вид [19]:

$$\begin{gathered}
\mathbf{w}_i^{1,\mathrm{I}} = \mathcal{L}_k\nabla\mathcal{L}_l - \mathcal{L}_l\nabla\mathcal{L}_k;\ i{=}1,...,6;\ k,l{=}1,...,4;\ k{<}l,\\
\mathbf{w}_i^{1,\mathrm{II}} = \mathcal{L}_k\nabla\mathcal{L}_l + \mathcal{L}_l\nabla\mathcal{L}_k;\ i{=}7,...,12;\ k,l{=}1,...,4;\ k{<}l,\\
\mathbf{w}_i^{2,\mathrm{I}} = \mathcal{L}_k\mathcal{L}_l\nabla\mathcal{L}_j + \mathcal{L}_j\mathcal{L}_l\nabla\mathcal{L}_k - 2\mathcal{L}_j\mathcal{L}_k\nabla\mathcal{L}_l;\ i{=}13,...,16;\ j,k,l{=}1,...,4;\ j{<}k{<}l,\\
\mathbf{w}_i^{2,\mathrm{I}} = \mathcal{L}_k\mathcal{L}_l\nabla\mathcal{L}_j - 2\mathcal{L}_j\mathcal{L}_l\nabla\mathcal{L}_k + \mathcal{L}_j\mathcal{L}_k\nabla\mathcal{L}_l;\ i{=}17,...,20;\ j,k,l{=}1,...,4;\ j{<}k{<}l,\\
\mathbf{w}_i^{2,\mathrm{II}} = \nabla(\mathcal{L}_j\mathcal{L}_k\mathcal{L}_l);\ i{=}21,...,24;\ j,k,l{=}1,...,4;\ j{<}k{<}l,\\
\mathbf{w}_i^{2,\mathrm{II}} = \nabla(\mathcal{L}_j\mathcal{L}_k(\mathcal{L}_j - \mathcal{L}_k));\ i{=}25,...,30;\ j,k{=}1,...,4;\ j{<}k,
\end{gathered} \tag{25}$$

где $\mathbf{w}_1^{1,\mathrm{I}},...,\mathbf{w}_6^{1,\mathrm{I}}$ – базисные функции первого порядка первого типа, ассоциированные с ребрами, $\mathbf{w}_7^{1,\mathrm{II}},...,\mathbf{w}_{12}^{1,\mathrm{II}}$ – базисные функции первого порядка второго типа, ассоциированные с ребрами, $\mathbf{w}_{13}^{2,\mathrm{I}},...,\mathbf{w}_{20}^{2,\mathrm{I}}$ – базисные функции второго порядка первого типа, ассоциированные с гранями, $\mathbf{w}_{21}^{2,\mathrm{II}},...,\mathbf{w}_{24}^{2,\mathrm{II}}$ – базисные функции второго порядка второго типа, ассоциированные с гранями, $\mathbf{w}_{25}^{2,\mathrm{II}},...,\mathbf{w}_{30}^{2,\mathrm{II}}$ – базисные функции второго порядка второго типа, ассоциированные с ребрами. Так как этот базис иерархический, то для получения базиса меньшего порядка следует ограничиться меньшим количеством функций. Так, для базиса первого порядка второго типа следует использовать функции $\mathbf{w}_1^{1,\mathrm{I}},...,\mathbf{w}_{12}^{1,\mathrm{II}}$.

Для вычисления интегралов в (22) воспользуемся кубатурной формулой численного интегрирования (формулой Гаусса) [20]:

$$\int\limits_{\Omega_k} f(x,y,z)\,d\Omega_k = \sum_{i=1}^{m} f(x_i,y_i,z_i)w_i,$$

где (x_i,y_i,z_i) – точки Гаусса, m – число точек Гаусса, w_i – соответствующие веса. При работе с базисными функциями второго порядка нужно использовать формулы, которые бы обеспечивали восьмой порядок интегрирования [21]. Для базисных функций первого порядка будет достаточно и меньших порядков интегрирования [20].

1.6. Треугольные конечные элементы

Границы области Ω являются двумерными и представляют собой треугольники. Для учета краевых условий (9) требуется построить разложение $\mathbf{E}^g$ по базису соответствующей границы в смысле МНК, для этого нужно решать СЛАУ вида

$$\mathbf{M}^{S_1}\tilde{\mathbf{q}} = \mathbf{b}^{S_1}, \tag{26}$$

где $\mathbf{M}_{i,j}^{S_1} = \int\limits_{S_1} \tilde{\mathbf{w}}_i\cdot\tilde{\mathbf{w}}_j\,dS_1$, $\mathbf{b}_i^{S_1} = \int\limits_{S_1} \mathbf{E}^g\cdot\tilde{\mathbf{w}}_i\,dS_1$, $\tilde{\mathbf{w}}_i$ и $\tilde{\mathbf{w}}_j$ – базисные функции на треугольниках.

Определим $\mathcal{L}$-координаты на треугольниках таким же образом, как и на тетраэдрах. Введем нумерацию вершин и ребер согласно рисунку 3:

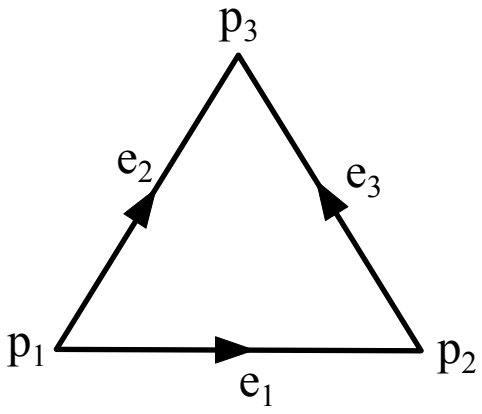


Рисунок 3 — треугольный конечный элемент

Тогда $\mathcal{L}$-координаты примут вид:

$$\mathcal{L}_i(x,y) = \alpha_{i,1}x + \alpha_{i,2}y + \alpha_{i,3}, \quad i = \overline{1..3}.$$

Коэффициенты $\alpha_{i,j}$ могут быть определены по формуле (27):

$$\begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} \\ \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} \\ \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} \end{bmatrix} = \begin{bmatrix} p_{1x} & p_{2x} & p_{3x} \\ p_{1y} & p_{2y} & p_{3y} \\ 1 & 1 & 1 \end{bmatrix}^{-1}. \tag{27}$$

Иерархический векторный базис Вебба второго порядка второго типа на треугольниках имеет вид:

$$\tilde{\mathbf{w}}_i^{1,\mathrm{I}} = \mathcal{L}_k\nabla\mathcal{L}_l - \mathcal{L}_l\nabla\mathcal{L}_k;\ i{=}1,...,3;\ k,l{=}1,...,3;\ k{<}l,$$
$$\tilde{\mathbf{w}}_i^{1,\mathrm{II}} = \mathcal{L}_k\nabla\mathcal{L}_l + \mathcal{L}_l\nabla\mathcal{L}_k;\ i{=}4,...,6;\ k,l{=}1,...,3;\ k{<}l,$$
$$\tilde{\mathbf{w}}_7^{2,\mathrm{I}} = \mathcal{L}_2\mathcal{L}_3\nabla\mathcal{L}_1 + \mathcal{L}_1\mathcal{L}_3\nabla\mathcal{L}_2 - 2\mathcal{L}_1\mathcal{L}_2\nabla\mathcal{L}_3,$$
$$\tilde{\mathbf{w}}_8^{2,\mathrm{I}} = \mathcal{L}_2\mathcal{L}_3\nabla\mathcal{L}_1 - 2\mathcal{L}_1\mathcal{L}_3\nabla\mathcal{L}_2 + \mathcal{L}_1\mathcal{L}_2\nabla\mathcal{L}_3,$$
$$\tilde{\mathbf{w}}_9^{2,\mathrm{II}} = \nabla(\mathcal{L}_1\mathcal{L}_2\mathcal{L}_3),$$
$$\tilde{\mathbf{w}}_i^{2,\mathrm{II}} = \nabla(\mathcal{L}_j\mathcal{L}_k(\mathcal{L}_j - \mathcal{L}_k));\ i{=}10,...,12;\ j,k{=}1,...,3;\ j{<}k,$$

где $\tilde{\mathbf{w}}_1^{1,\mathrm{I}}, ..., \tilde{\mathbf{w}}_3^{1,\mathrm{I}}$ – базисные функции первого порядка первого типа, $\tilde{\mathbf{w}}_4^{1,\mathrm{II}}, ..., \tilde{\mathbf{w}}_6^{1,\mathrm{II}}$ – базисные функции первого порядка второго типа, $\tilde{\mathbf{w}}_7^{2,\mathrm{I}}, \tilde{\mathbf{w}}_8^{2,\mathrm{I}}$ – базисные функции второго порядка первого типа, $\tilde{\mathbf{w}}_9^{2,\mathrm{II}}, ..., \tilde{\mathbf{w}}_{12}^{2,\mathrm{II}}$ – базисные функции второго порядка второго типа. Для базиса первого порядка второго типа следует использовать функции $\tilde{\mathbf{w}}_1^{1,\mathrm{I}}, ..., \tilde{\mathbf{w}}_6^{1,\mathrm{II}}$.

Для вычисления интегралов в (26) воспользуемся формулой Гаусса [20]:

$$\int\limits_{\Omega_k} f(x,y)\, d\Omega_k = \sum_{i=1}^{m} f(x_i, y_i) w_i,$$

где (x_i, y_i) – точки Гаусса, m – число точек Гаусса, w_i – соответствующие веса. Так же, как и для тетраэдров, для работы с базисом второго порядка нужно использовать формулы, обеспечивающие восьмой порядок интегрирования [21]. Для базиса первого порядка достаточно и меньших порядков интегрирования [20].

2. Вычислительные эксперименты

2.1. Верификация программного комплекса

Верификация полученной конечноэлементной аппроксимации будет проводиться на тестовой задаче, имеющей аналитическое решение.

2.1.1. Расчетная область

Расчетная область представляет собой куб со следующими параметрами: $x \in [0, 1]$, $y \in [0, 1]$, $z \in [0, 1]$. Куб разбивается на регулярную тетраэдральную сетку согласно рисунку 4:

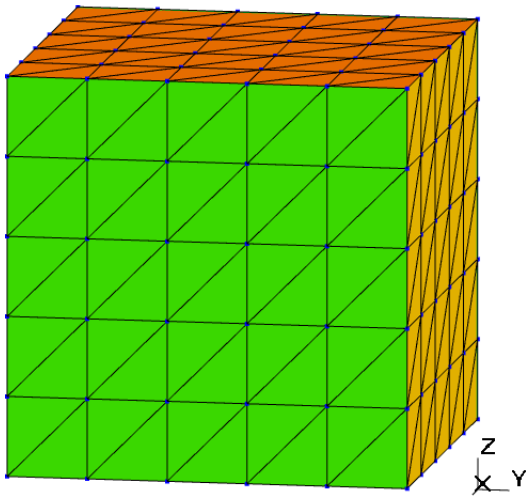


Рисунок 4 — конечноэлементная сетка для верификации

Всего в сетке 750 тетраэдров, 300 треугольников по границе, 216 узлов, 1115 ребер и 1650 граней.

Физические параметры среды заданы следующим образом: $\varepsilon = \varepsilon_0$, $\mu = \mu_0$, $\sigma = 10$ См/м. Частота источника поля $\nu = \frac{100}{2\pi}$ Гц. На всех внешних гранях расчетной области заданы краевые условия первого рода (9).

2.1.2. Тестирование на линейных по пространству функциях

В качестве аналитического решения уравнения (8) выберем функцию

$$\mathbf{E} = (y + z, x + z, x + y)^T.$$

Тестирование будем проводить на базисных функциях первого и второго порядка второго типа. Погрешности в норме пространства $\mathbb{L}^2$ полученных решений приведены в таблице 1.

Таблица 1 — относительные погрешности в норме $\mathbb{L}^2$

Порядок базисных ф-й	$\frac{\|\mathbf{E}-\mathbf{E}^h\|_{\mathbb{L}^2}}{\|\mathbf{E}\|_{\mathbb{L}^2}}$	$\frac{\|\mathbf{E}_x-\mathbf{E}_x^h\|_{\mathbb{L}^2}}{\|\mathbf{E}_x\|_{\mathbb{L}^2}}$	$\frac{\|\mathbf{E}_y-\mathbf{E}_y^h\|_{\mathbb{L}^2}}{\|\mathbf{E}_y\|_{\mathbb{L}^2}}$	$\frac{\|\mathbf{E}_z-\mathbf{E}_z^h\|_{\mathbb{L}^2}}{\|\mathbf{E}_z\|_{\mathbb{L}^2}}$
1	5.277e-11	5.313e-11	5.345e-11	5.169e-11
2	8.064e-11	8.111e-11	8.056e-11	8.025e-11

Как и следовало ожидать, метод хорошо аппроксимировал линейную по пространству функцию.

2.1.3. Тестирование на полиномиальных функциях

В качестве аналитического решения уравнения (8) выберем функцию

$$\mathbf{E} = \begin{pmatrix} e^{-(0.5-y)^2-(0.5-z)^2} \\ e^{-(0.5-x)^2-(0.5-z)^2} \\ e^{-(0.5-x)^2-(0.5-y)^2} \end{pmatrix}. \tag{28}$$

Тестирование будем проводить на базисных функциях первого и второго порядка второго типа. Погрешности в норме пространства $\mathbb{L}^2$ полученных решений приведены в таблице 2.

Таблица 2 — относительные погрешности в норме $\mathbb{L}^2$

Порядок базисных ф-й	$\frac{\|\|\mathbf{E}\|-\|\mathbf{E}^h\|\|_{\mathbb{L}^2}}{\|\|\mathbf{E}\|\|_{\mathbb{L}^2}}$	$\frac{\|\mathbf{E}_x-\mathbf{E}_x^h\|_{\mathbb{L}^2}}{\|\mathbf{E}_x\|_{\mathbb{L}^2}}$	$\frac{\|\mathbf{E}_y-\mathbf{E}_y^h\|_{\mathbb{L}^2}}{\|\mathbf{E}_y\|_{\mathbb{L}^2}}$	$\frac{\|\mathbf{E}_z-\mathbf{E}_z^h\|_{\mathbb{L}^2}}{\|\mathbf{E}_z\|_{\mathbb{L}^2}}$
1	6.608e-3	7.869e-3	5.877e-3	5.877e-3
2	1.775e-4	1.895e-4	1.712e-4	1.712e-4

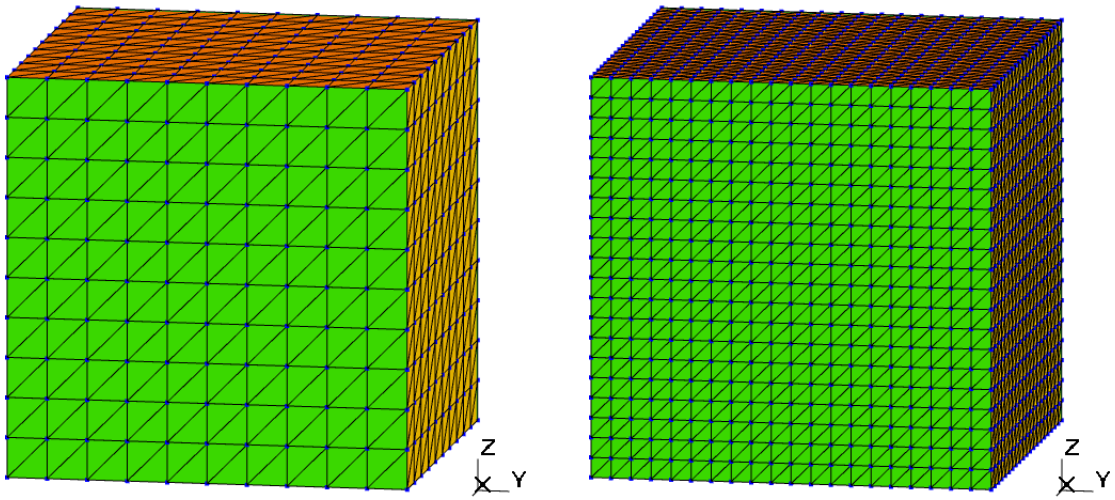
Метод достаточно хорошо аппроксимировал и не полиномиальных функцию.

2.1.4. Определение порядка аппроксимации

В качестве аналитического решения уравнения (8) выберем функцию (28). Проведем исследование на порядок аппроксимации. Измельчим сетку расчетной области, изображенную на рисунке 4, в 2 и 4 раза, после чего сравним погрешности полученных решений. Измельченные сетки приведены на рисунках 5а и 5б, погрешности в норме пространства $\mathbb{L}^2$ полученных решений и порядок аппроксимации – в таблице 3.

Таблица 3 — относительные погрешности в норме $\mathbb{L}^2$

Порядок базиса	$\frac{\|\mathbf{E}-\mathbf{E}^h\|\mathbb{L}^2}{\|\mathbf{E}\|_{\mathbb{L}^2}}$	$\frac{\|\mathbf{E}-\mathbf{E}^{h/2}\|_{\mathbb{L}^2}}{\|\mathbf{E}\|_{\mathbb{L}^2}}$	$\frac{\|\mathbf{E}-\mathbf{E}^{h/4}\|_{\mathbb{L}^2}}{\|\mathbf{E}\|_{\mathbb{L}^2}}$	$\log_2 \frac{\|\mathbf{E}-\mathbf{E}^h\|_{\mathbb{L}^2}}{\|\mathbf{E}-\mathbf{E}^{h/2}\|_{\mathbb{L}^2}}$	$\log_2 \frac{\|\mathbf{E}-\mathbf{E}^{h/2}\|_{\mathbb{L}^2}}{\|\mathbf{E}-\mathbf{E}^{h/4}\|_{\mathbb{L}^2}}$
1	6.608e-3	1.637e-3	5.051e-4	2.013	1.696
2	1.775e-4	2.164e-5	3.582e-6	3.036	2.595



(а) (б)

Рисунок 5 — конечноэлементные сетки для определения порядка аппроксимации

Из результатов видно, что порядок аппроксимации получился второй для базисных функций первого порядка и третий для базисных функций второго порядка, что совпадает с теоретическими значениями.

2.2. Исследование влияния слоя воздуха

Проведем исследование влияния слоя воздуха в модельной задаче морской геоэлектрики при различной глубине погружения в воду источника электромагнитного возмущения.

В этом исследовании будем пользоваться базисными функциями второго полного порядка.

2.2.1. Описание расчетной области

Схематичное изображение расчетной области показано на рисунке 6, где Ω_1 – воздух ($\sigma = 10^{-6}$ См/м, $\mu = \mu_0$, $\varepsilon = \varepsilon_0$); Ω_2 – морская вода ($\sigma = 3.3$ См/м, $\mu = \mu_0$, $\varepsilon = \varepsilon_0$); Ω_3 – грунт ($\sigma = 0.2$ См/м, $\mu = \mu_0$, $\varepsilon = \varepsilon_0$); Ω_4 – углеводороды ($\sigma = 10^{-2}$ См/м, $\mu = \mu_0$, $\varepsilon = \varepsilon_0$); L_1, L_2 и L_3 – размеры области моделирования по осям x, y и z соответственно; $L_1 = L_2 = L_3 = 6000$ м; $h_1 = 600$ м – толщина Ω_2; $l_1 = 400$ м, $h_3 = 75$ м, $h_2 = 135$ м – длина, толщина и глубина объекта Ω_4 соответственно.

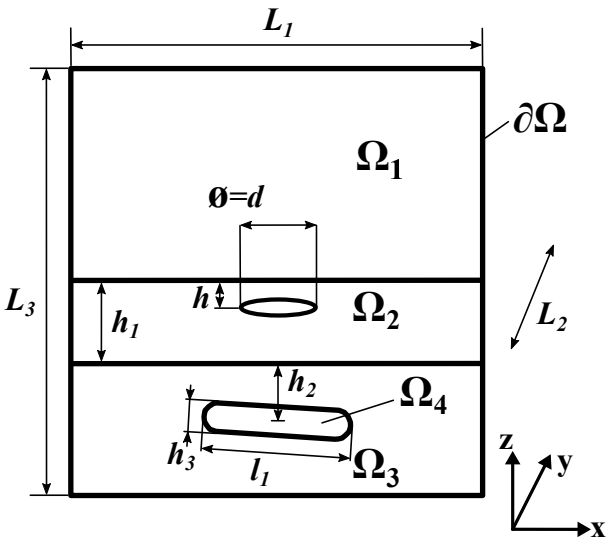


Рисунок 6 — схематичное изображение расчетной области

Объект Ω_4 представляет собой скругленный прямоугольный параллелепипед с двумя равными сторонами, наклоненный под углом 5°. Источником электрического поля является токовая петля диаметром $d = 100$ м с током частотой 1 Гц, глубина h которой варьируется в ходе исследования.

2.2.2. Конечноэлементная сетка

Фрагмент $x \in [-600, 0]$, $y \in [-600, 600]$ $z \in [-1000, 600]$ одной из конечноэлементных сеток, использованных для проведения исследования, представлен на рисунке 7.

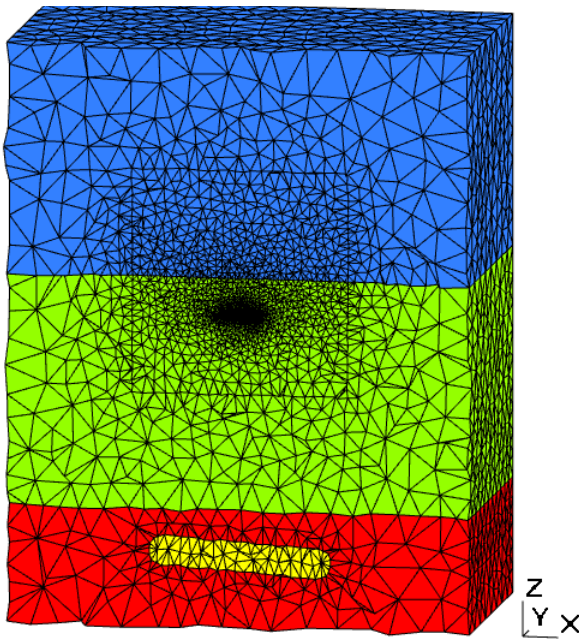


Рисунок 7 — фрагмент конечноэлементной сетки

2.2.3. Результаты исследования

Разности решений в норме $\mathbb{L}^2$ в объеме $[-600, 600] \times [-600, 600] \times [-1000, 0]$ между областью, в которой присутствует слой воздуха, и областью, в которой заданы условия непротекания (10), для некоторых значений глубины петли h показаны в таблице 4. В форме графика эти данные приведены на рисунке 8.

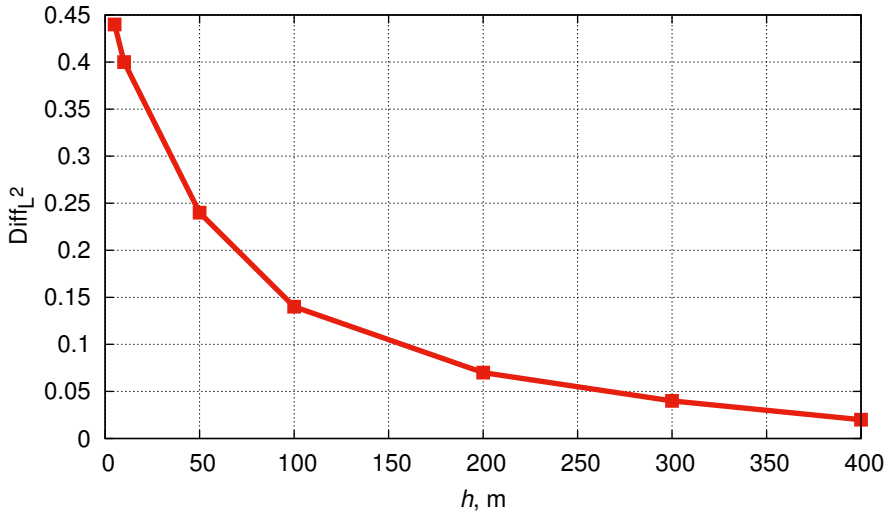
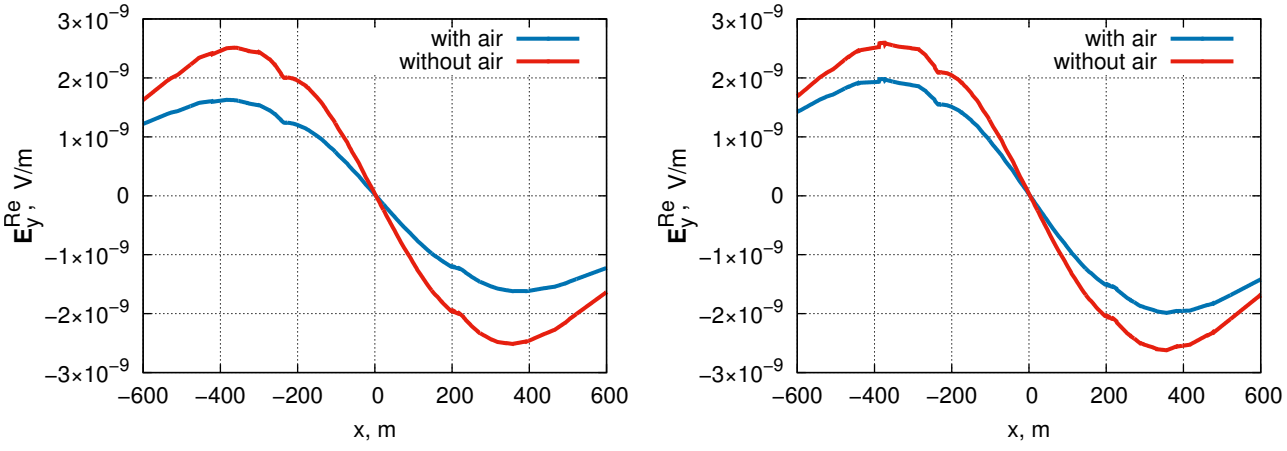


Рисунок 8 — график изменения относительной разности решений при изменении глубины

Таблица 4 — относительные разности решений

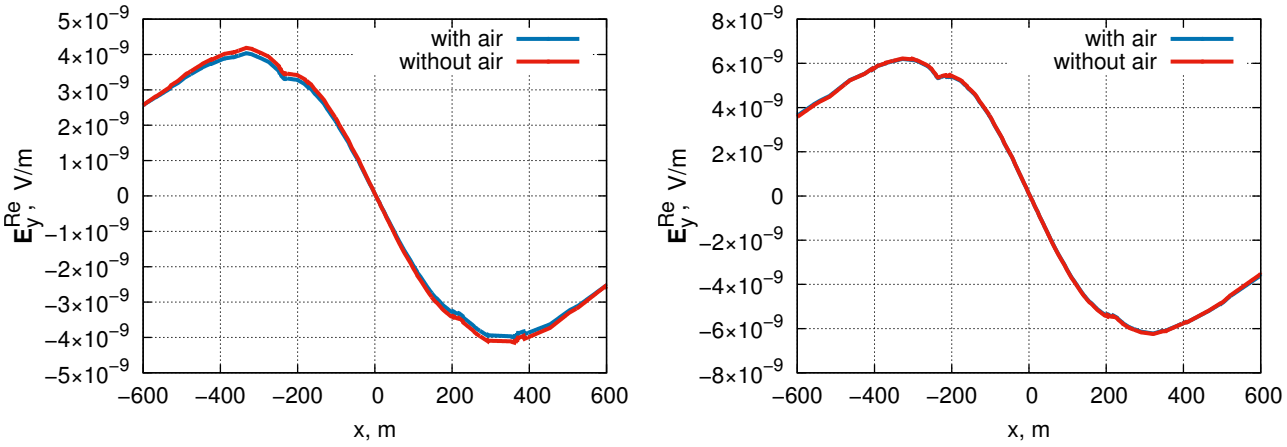
Глубина петли	5	10	50	100	200	300	400
$\frac{\|\|\|\mathbf{E}^{air} - \mathbf{E}^{noair}\|\|\|_{\mathbb{L}^2}}{\|\|\|\mathbf{E}^{air}\|\|\|_{\mathbb{L}^2}}$	0.44	0.40	0.24	0.14	0.07	0.04	0.02

Графики вещественной компоненты $\mathbf{E}_y$ по линии $y = 0$, $z = -610$ для различных глубин петли представлены на рисунках 9 и 10.



(а) (б)

Рисунок 9 — $\mathrm{Re}(\mathbf{E}_y)$ по линии $y = 0$, $z = -610$, глубина (а) 5 м и (б) 50 м



(а) (б)

Рисунок 10 — $\mathrm{Re}(\mathbf{E}_y)$ по линии $y = 0$, $z = -610$, глубина (а) 200 м и (б) 300 м

Из результатов следует, что слой воздуха оказывает значительное влияние на получаемое решение при расположении источника электромагнитного возмущения на малой глубине (меньше трехсот метров для рассмотренной конфигурации).

2.3. Исследование эффективности применения PML-слоя

Цель вычислительных экспериментов: определение эффективности применения PML-слоя. Геометрические характеристики PML-слоя показаны на рисунке 11.

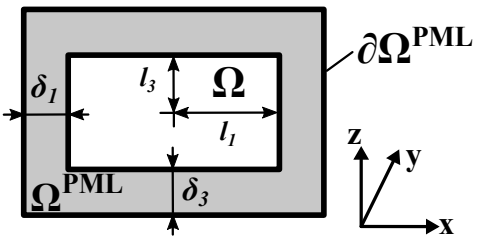
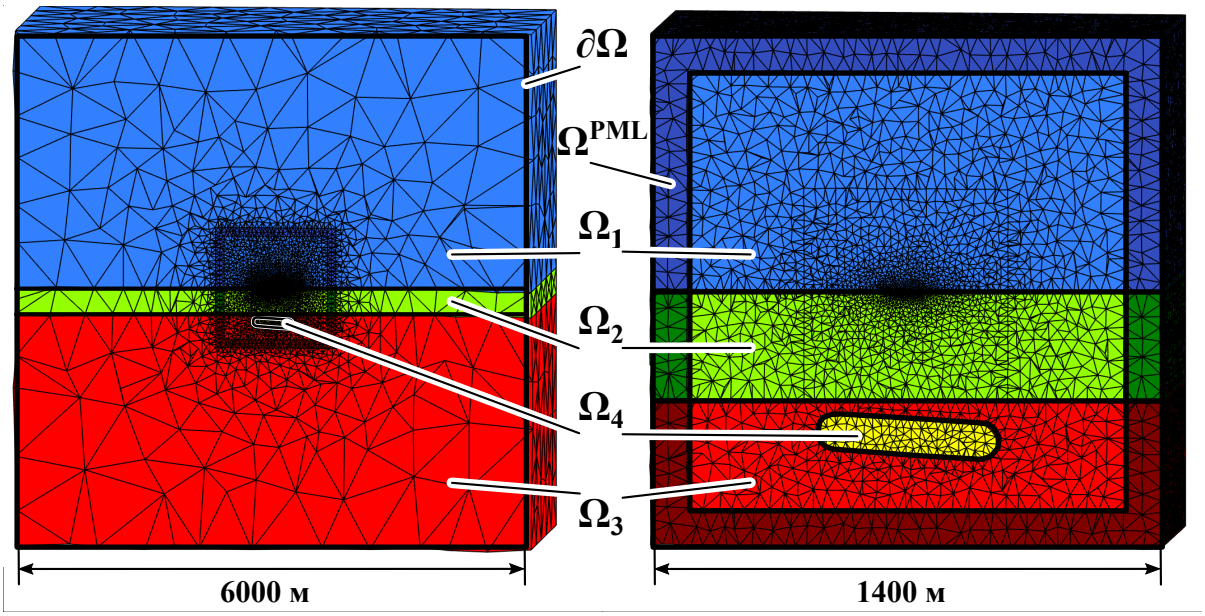


Рисунок 11 — геометрические характеристики PML-слоя

Вычислительные эксперименты проводились следующим образом: последовательно варьировался каждый из параметров PML-слоя: толщина слоя в k-м направлении δ_k, где $k = \overset{x}{1}, \overset{y}{2}, \overset{z}{3}$, расстояние от центра области до внутренних границ слоя l_k, коэффициент комплексного растяжения координат χ (18), оставшиеся параметры фиксировались, что позволило определить параметры, оказывающие наибольшее влияние на характеристики PML-слоя.

Фрагменты тетраэдральных конечноэлементных сеток с «большим баком» и с PML-слоем приведены на рисунках 12а и 12б.



(а) (б)

Рисунок 12 — конечноэлементные сетки: (а) «большой бак» и (б) PML-слой

В этом исследовании будем пользоваться базисными функциями первого полного порядка.

2.3.1. Описание расчетной области

Схематичное изображение расчетной области показано на рисунке 13, где Ω_1 – воздух ($\sigma = 10^{-6}$ См/м, $\mu = \mu_0$, $\varepsilon = \varepsilon_0$); Ω_2 – морская вода ($\sigma = 3.3$ См/м, $\mu = \mu_0$, $\varepsilon = \varepsilon_0$); Ω_3 – грунт ($\sigma = 0.2$ См/м, $\mu = \mu_0$, $\varepsilon = \varepsilon_0$); Ω_4 – углеводороды ($\sigma = 10^{-2}$ См/м, $\mu = \mu_0$, $\varepsilon = \varepsilon_0$); L_1, L_2 и L_3 – размеры области моделирования по осям x, y и z соответственно; $L_1 = L_2 = L_3 = 6000$ м; $h_1 = 300$ м – толщина Ω_2; $l_1 = 400$ м, $h_3 = 100$ м, $h_2 = 100$ м – длина, толщина и глубина объекта Ω_4 соответственно.

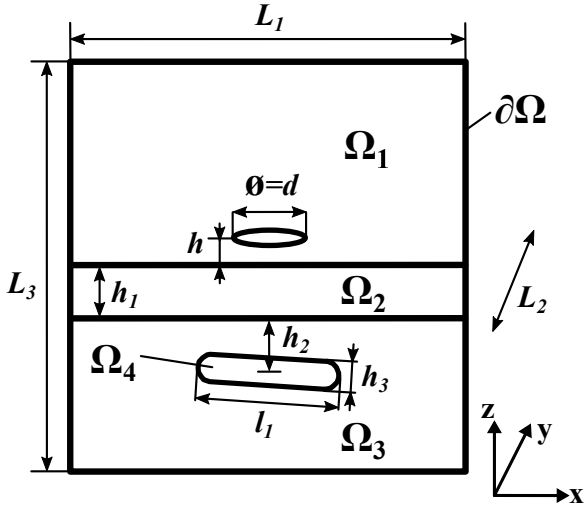


Рисунок 13 — расчетная область

Объект Ω_4 представляет собой скругленный прямоугольный параллелепипед с двумя равными сторонами, наклоненный под углом 5°. Источником электрического поля является петля диаметром $d = 100$ м с током частотой 1 Гц, расположенная в воздухе на расстоянии $h = 5$ м от границы раздела сред воздух-вода. Также рассматривается случай, когда петля расположена в воде ($h = -5$ м).

Выделим внутри области с условием «большого бака» и области, на границе которой задан PML-слой, подобласть Ω' размером $1000 \times 1000 \times 1000$ м^3. Для этой подобласти в данном исследовании будем оценивать разность в норме $\mathbb{L}^2$ между действительными компонентами $\mathbf{E}_y$ векторов решений $\mathbf{E}_y^{\text{бак}}$ и $\mathbf{E}_y^{\text{PML}}$, полученных с применением «большого бака» и PML-слоя соответственно.

2.3.2. Варьирование коэффициентов растяжения

Зафиксируем $\delta_k = 100$ м, $l_k = 600$ м, $m = 3$, $h = 5$ м и будем варьировать коэффициент комплексного растяжения координат χ. Размер получаемой СЛАУ для «большого бака» – 653814, с PML-слоем – 616180. Результаты приведены в таблице 5. Для случая $h = -5$ м размер получаемой СЛАУ для «большого бака» – 652396, с PML-слоем – 614504. Результаты приведены в таблице 6.

Таблица 5 — варьирование коэффициентов растяжения при $h = 5$ м

$\text{Re}(\chi)$ в Ω_1	$\text{Im}(\chi)$ в Ω_1	$\text{Re}(\chi)$ в Ω_2	$\text{Im}(\chi)$ в Ω_2	$\text{Re}(\chi)$ в Ω_3	$\text{Im}(\chi)$ в Ω_3	$\frac{\|\text{Re}(\mathbf{E}_y^{\text{бак}}) - \text{Re}(\mathbf{E}_y^{\text{PML}})\|}{\|\text{Re}(\mathbf{E}_y^{\text{бак}})\|}$	Время, бак	Время, PML
3	0	1	5	3	1	0.106636	650	592
3	1	0	6	2	1	0.0925		599
4	0	1	5	3	1	0.0947		731
4	1	0	6	2	1	0.0910		591

Таблица 6 — варьирование коэффициентов растяжения при $h = -5$ м

$\text{Re}(\chi)$ в Ω_1	$\text{Im}(\chi)$ в Ω_1	$\text{Re}(\chi)$ в Ω_2	$\text{Im}(\chi)$ в Ω_2	$\text{Re}(\chi)$ в Ω_3	$\text{Im}(\chi)$ в Ω_3	$\frac{\|\text{Re}(\mathbf{E}_y^{\text{бак}}) - \text{Re}(\mathbf{E}_y^{\text{PML}})\|}{\|\text{Re}(\mathbf{E}_y^{\text{бак}})\|}$	Время, бак	Время, PML
4	0	1	5	3	1	0.0929047	309	344
4	0	1	6	3	1	0.0870		294
4	0	1	6	3	2	0.0809		253
4	1	1	6	3	2	0.0658		306

2.3.3. Варьирование толщины PML-слоя

Зафиксируем $\chi_{\Omega_1} = (4, 1)$, $\chi_{\Omega_2} = (0, 6)$, $\chi_{\Omega_3} = (2, 1)$, $l_k = 600$ м, $m = 3$, $h = 5$ м и будем варьировать толщину PML-слоя δ_k. Результаты приведены в таблице 7. Для случая

$h = -5$ м выберем $\chi_{\Omega_1} = (4, 0)$, $\chi_{\Omega_2} = (1, 6)$, $\chi_{\Omega_3} = (3, 2)$. Результаты приведены в таблице 8.

Таблица 7 — варьирование толщины PML-слоя при $h = 5$ м

δ_k	$\frac{\|\mathrm{Re}(\mathbf{E}_y^{\text{бак}}) - \mathrm{Re}(\mathbf{E}_y^{\text{PML}})\|}{\|\mathrm{Re}(\mathbf{E}_y^{\text{бак}})\|}$	Время, бак	Время, PML	Размер СЛАУ, бак	Размер СЛАУ, PML
80	0.1199	673	1289	659858	618128
100	0.0910	650	591	653814	616180
120	0.0784	609	1142	654354	617324

Таблица 8 — варьирование толщины PML-слоя при $h = -5$ м

δ_k	$\frac{\|\mathrm{Re}(\mathbf{E}_y^{\text{бак}}) - \mathrm{Re}(\mathbf{E}_y^{\text{PML}})\|}{\|\mathrm{Re}(\mathbf{E}_y^{\text{бак}})\|}$	Время, бак	Время, PML	Размер СЛАУ, бак	Размер СЛАУ, PML
80	0.1201	359	297	652312	614822
100	0.0809	309	253	652396	614504
120	0.0623	250	859	652422	615394

2.3.4. Варьирование размера области, на границе которой вводится PML-слой

Зафиксируем $\chi_{\Omega_1} = (4, 1)$, $\chi_{\Omega_2} = (0, 6)$, $\chi_{\Omega_3} = (2, 1)$, $\delta_k = 100$ м, $m = 3$, $h = 5$ м и будем варьировать l_k – размер области, на границе которой вводится PML-слой. Результаты приведены в таблице 9. Для случая $h = -5$ м выберем $\chi_{\Omega_1} = (4, 0)$, $\chi_{\Omega_2} = (1, 6)$, $\chi_{\Omega_3} = (3, 2)$. Результаты приведены в таблице 10.

Таблица 9 — варьирование размера области, на границе которой вводится PML-слой, при $h = 5$ м

l_k	$\frac{\|\mathrm{Re}(\mathbf{E}_y^{\text{бак}}) - \mathrm{Re}(\mathbf{E}_y^{\text{PML}})\|}{\|\mathrm{Re}(\mathbf{E}_y^{\text{бак}})\|}$	Время, бак	Время, PML	Размер СЛАУ, бак	Размер СЛАУ, PML
500	0.187456	628	587	659130	621390
600	0.0909998	650	591	652396	614504
800	0.0440642	718	658	794310	744856

Таблица 10 — варьирование размера области, на границе которой вводится PML-слой, при $h = -5$ м

l_k	$\frac{\|\mathrm{Re}(\mathbf{E}_y^{\text{бак}}) - \mathrm{Re}(\mathbf{E}_y^{\text{PML}})\|}{\|\mathrm{Re}(\mathbf{E}_y^{\text{бак}})\|}$	Время, бак	Время, PML	Размер СЛАУ, бак	Размер СЛАУ, PML
500	0.1751746	317	238	659814	621390
600	0.0809429	309	253	652396	614504
800	0.0348019	357	329	793272	743780

2.3.5. Проверка выполнения условий на контактных границах

Проверим, что в случае независимого варьирования коэффициентов комплексного растяжения координат χ, на границе двух соседних PML-слоев с различными характеристиками не нарушаются условия на контактных границах (3)-(4). Для этого рассмотрим напряженность электрического поля $\mathbf{E}$ вдоль линии $x = 650$, $y = 0$, $z = [-0.005, 0.005]$ (середина PML-слоя по x-направлению):

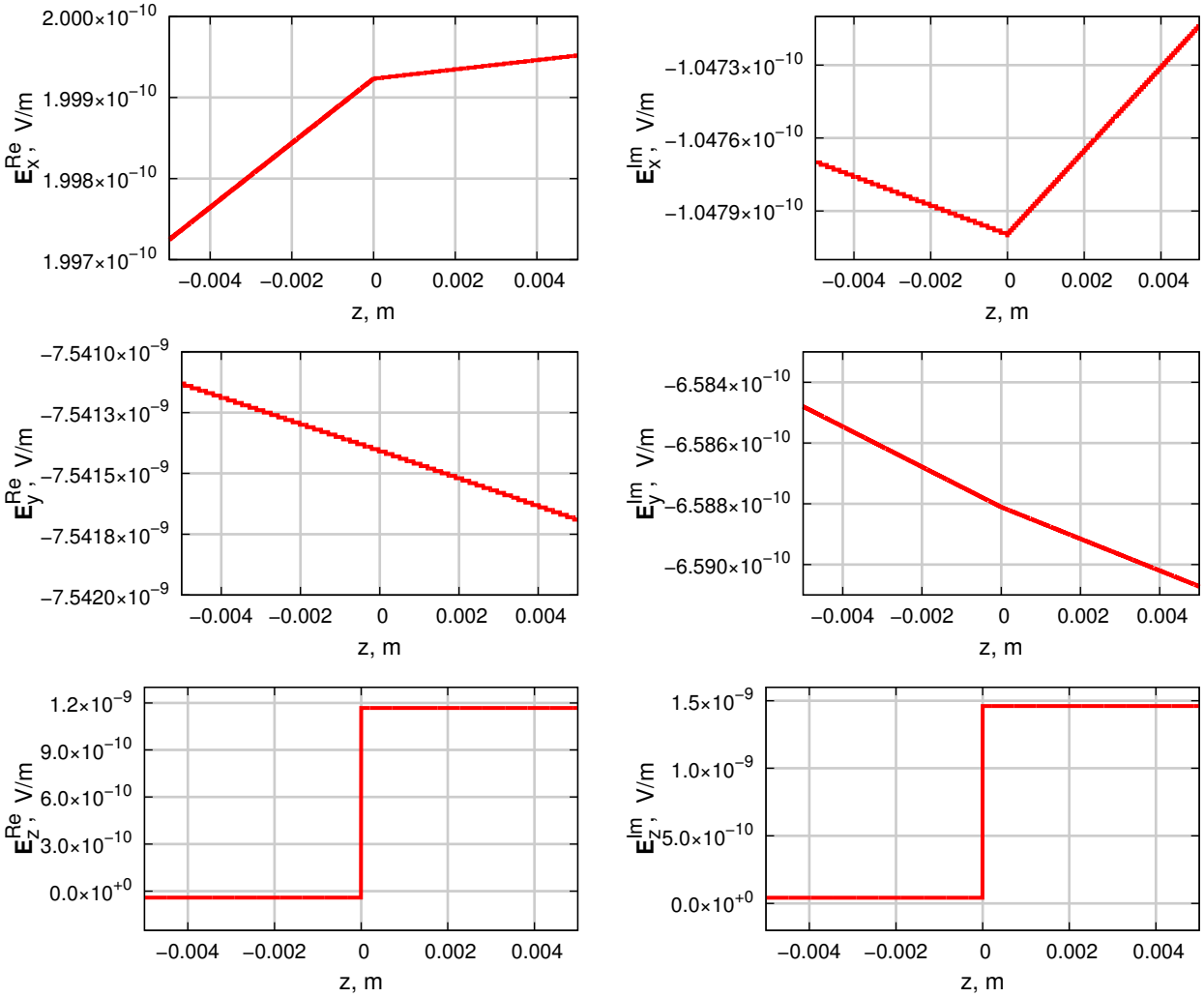
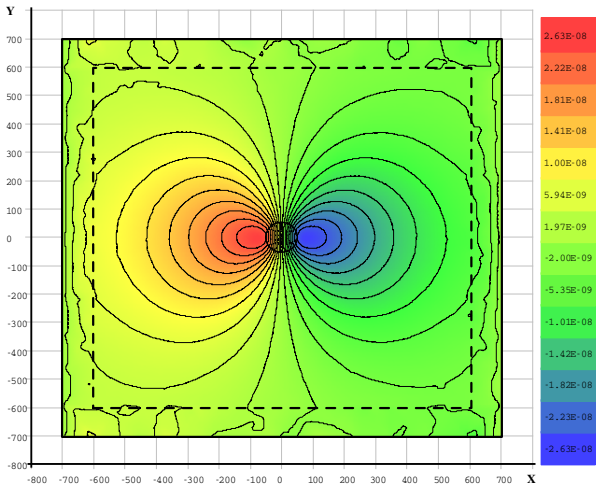


Рисунок 14 — графики компонент электрического поля на контактных границах

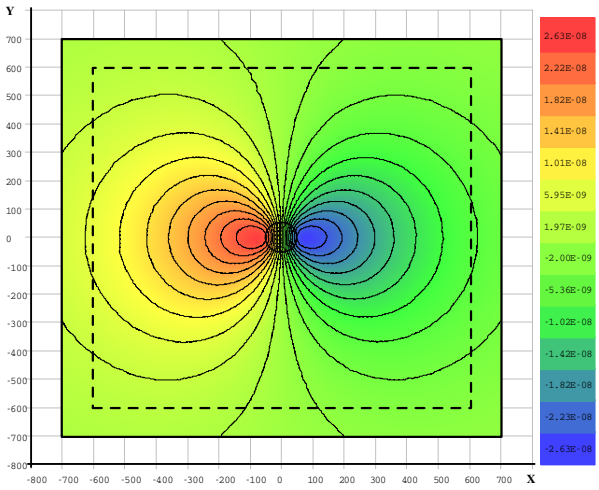
Как видно из графиков на рисунке 14, разрывна только нормальная компонента $\mathbf{E}_z$, следовательно, условия (3)-(4) выполнены.

2.3.6. Графическое представление результатов

Рассмотрим результаты, полученные при параметрах $h = 5$ м, $\chi_{\Omega_1} = (4, 0)$, $\chi_{\Omega_2} = (1, 6)$, $\chi_{\Omega_2} = (3, 2)$, $m = 3$, $l_k = 600$ м и $\delta_k = 100$ м. На рисунках 15 и 16 показаны картины электрического поля в сечении плоскостью $z = -10$ м. На рисунках 15а и 16а представлено решение с PML-слоем; 15б и 16б – решение с «большим баком».

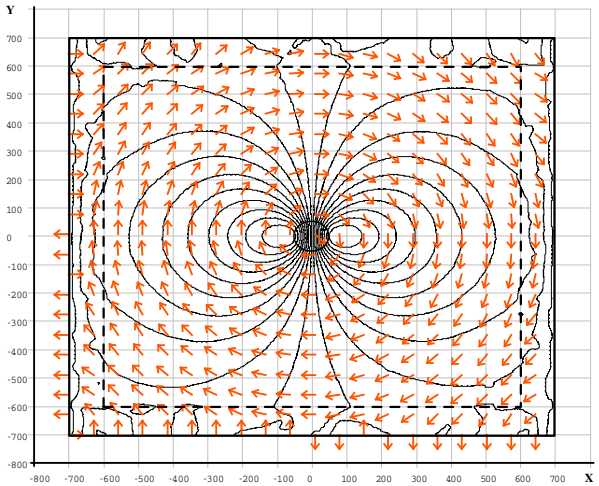


(а)

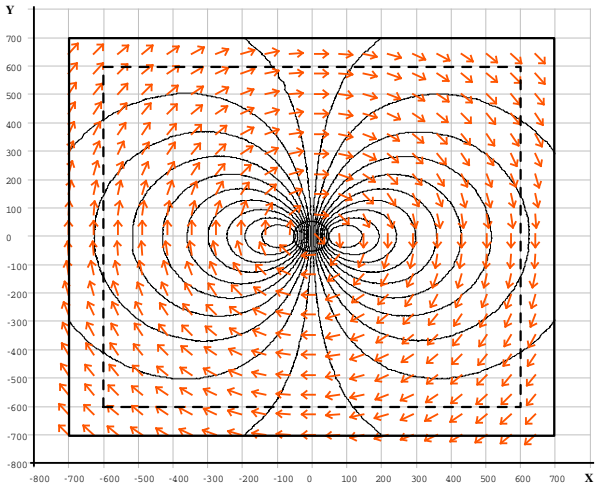


(б)

Рисунок 15 — $\mathrm{Re}(\mathbf{E}_y)$

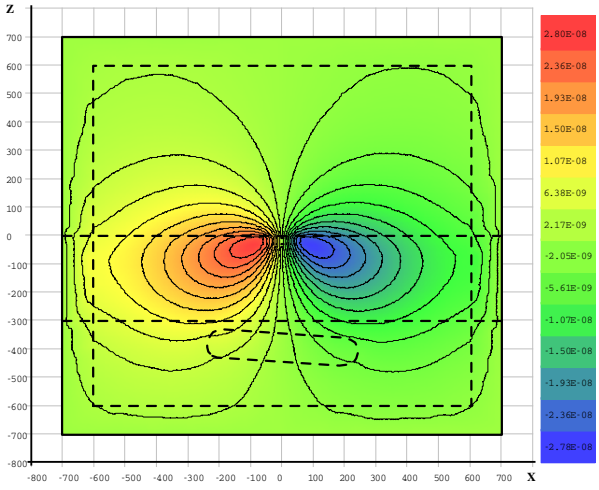


(а)

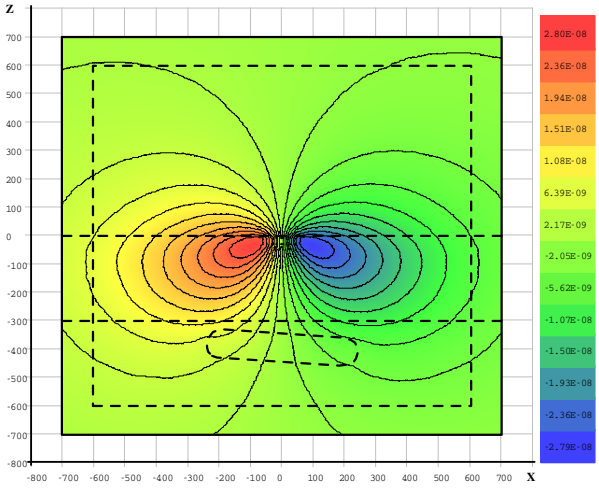


(б)

Рисунок 16 — изолинии $\mathrm{Re}(\mathbf{E}_y)$ и векторы $(\mathrm{Re}(\mathbf{E}_x), \mathrm{Re}(\mathbf{E}_y))^T$



(а)



(б)

Рисунок 17 — $\mathrm{Re}(\mathbf{E}_y)$

На рисунке 17 показаны картины электрического поля в сечении плоскостью $y = 0$. На рисунках 17а и 17б представлено решение с PML-слоем и решение с «большим баком» соответственно.

2.3.7. Анализ целесообразности применения PML-слоя

Наибольшее влияние на точность получаемого решения при введении PML-слоя оказывают коэффициент комплексного растяжения координат χ и l – размер области, на границе которой вводится PML-слой. При увеличении размера внутренней области ожидаемо увеличивается размер СЛАУ, поэтому этот параметр при решении реальных задач следует не варьировать, а выбрать как некоторое ограничение. Таким образом, в первую очередь стоит проводить поиск оптимального значения именно для коэффициента растяжения χ. Выбор толщины PML-слоя влияет на точность решения незначительно, поэтому достаточно лишь убедиться, что она выбрана в подходящих для задачи пределах – не слишком большой и не слишком маленькой.

В рассмотренном виде применение PML-слоя для сокращения области моделирования задач низкочастотной морской геоэлектрики незначительно уменьшает размерность СЛАУ и, в большинстве случаев, не приводит к существенному сокращению времени решения задачи. Одной из причин этого может быть то, что основное растяжение приходится на вещественные компоненты координат. Это приводит к значительной «вытянутости» тетраэдров внутри PML-слоя и, как следствие, сильному ухудшению свойств матрицы СЛАУ и увеличению времени решения. Параллелепипедальные конечные элементы лишены подобного недостатка, поэтому для них можно проводить комплексное растяжение в гораздо больших диапазонах. Однако, такие элементы не подходят для аппроксимации сколь-либо сложных областей. Для использования в одной сетке и тетраэдральных, и параллелепипедальных конечных элементов можно применять специальные переходные элементы [22, 23], либо воспользоваться неконформными методами [24–28].

Заключение

В работе были реализованы алгоритмы на базе векторного метода конечных элементов. Эти алгоритмы были положены в основу программного комплекса, который позволяет моделировать электромагнитные поля в областях разнообразной структуры. С помощью этого программного комплекса были решены модельные задачи морской геоэлектрики на низких частотах, проведены исследования возможности сокращения области моделирования без внесения дополнительных погрешностей.

На основании исследований были сделаны выводы, что расчеты, в которых в область моделирования не включается воздух, допустимы только при расположении источника электромагнитного поля на большой глубине, иначе, из-за неправильного учета физических процессов, происходящих в воздухе, полученное решение будет неверным.

Применение PML-слоя позволило получить достаточно точные решения, однако его применение не привело к резкому уменьшению размерности систем уравнений и, как следствие, к уменьшению времени решения. Было выдвинуто предположение, что увеличить эффективность применения PML-слоя можно с помощью применения неконформных методов, в которых конечноэлементная сетка может содержать геомитрические носители разного типа: тетраэдры и параллелепипеды.

Список литературы

1. Шурина, Э.П. Морская геоэлектрика – задачи и перспективы / Э.П. Шурина, М.И. Эпов, А.В. Мариенко // Тез. докл. всеросс. науч.-техн. конф. «Научное и техническое обеспечение исследования и освоения шельфа Северного Ледовитого океана». – 2010. – 9-13 августа. – С. 7-12.
2. Gabrielsen, P.T. 3D CSEM for Hydrocarbon Exploration in the Barents Sea / P.T. Gabrielsen, D.V. Shantsev, S. Fanavoll // 5th Saint Petersburg International Conference & Exhibition – Geosciences: Making the most of the Earth's resources. – 2012. – 2-5 April. – Pp. 1-5.
3. Anderson, C. An integrated approach to marine electromagnetic surveying using a towed streamer and source / C. Anderson, J. Mattsson // First Break. – May 2010. – Vol. 28, Iss. 5. – Pp. 71-75.
4. Mur G. Absorbing boundary conditions for the finite-difference approximation of the time-domain electromagnetic-field equations // Electromagnetic Compatibility, IEEE Transactions on. – 1981. – No. 4. – Pp. 377-382.
5. Berenger, J.P. A perfectly matched layer for the absorption of electromagnetic waves / J.P. Berenger // Journal of computation physics. – 1994 – No. 114. – Pp. 185-200.
6. Wiik, T. A Discontinuous Galerkin Method for Modelling Marine Controlled Source Electromagnetic Data / T. Wiik, M.V. De Hoop, B. Ursin // Proceedings of the Project Review, Geo-Mathematical Imaging Group, Purdue University, West Lafayette, IN. – 2013 – Vol. 1 – P. 75-102.
7. Баландин, М.Ю. Векторный метод конечных элементов : Учеб. пособие / М.Ю. Баландин, Э.П. Шурина. – Новосибирск : Изд-во НГТУ, 2001. – 69 с.
8. Monk P. Finite element methods for Maxwell's equations. / P. Monk – Oxford University Press, 2003.
9. Эпов, М.И. Параллельные конечноэлементные вычислительные схемы в задачах геоэлектрики / М.И. Эпов, Э.П. Шурина, Д.А. Архипов // Вычислительные технологии. – 2013. – Том 18, №2. – С. 94-112.
10. Schwarzbach C. Stability of finite element solutions to Maxwell's equations in frequency domain. / C. Schwarzbach – 2009.
11. Hiptmair R. Multigrid method for Maxwell's equations / R. Hiptmair // SIAM Journal on Numerical Analysis. – 1998. – Vol. 36. – No. 1. – Pp. 204-225.
12. Nédélec J.C. Mixed finite elements in $\mathbb{R}3$ / J.C. Nédélec // Numerische Mathematik. – 1980. – Vol. 35. – No. 3. – Pp. 315-341.
13. Nédélec J.C. A new family of mixed finite elements in $\mathbb{R}3$ / J.C. Nédélec // Numerische Mathematik. – 1986. – Vol. 50. – No. 1. – Pp. 57-81.
14. Webb J.P. Edge elements and what they can do for you / J.P. Webb // Magnetics, IEEE Transactions on. – 1993. – Vol. 29. – No. 2. – Pp. 1460-1465.

15. Баландин М.Ю. Методы решения СЛАУ большой размерности: Учеб. пособие / М.Ю. Баландин, Э.П. Шурина. – Новосибирск : Изд-во НГТУ, 2000. – 70 с.
16. Соловейчик, Ю.Г. Метод конечных элементов для решения скалярных и векторных задач : учеб. пособие / Ю.Г. Соловейчик, М.Э. Рояк, М.Г. Персова. – Новосибирск : Изд-во НГТУ, 2007. – 896 с.
17. Webb, J.P. Hierarchal Vector Basis Functions of Arbitrary Order for Triangular and Tetrahedral Finite Elements / J.P. Webb // IEEE transactions on antennas and propagation. – 1999. – Vol. 47. – Pp. 1244-1253.
18. Nechaev, O.V. Multilevel iterative solver for the edge fem solution of the 3D Maxwell equation / O.V. Nechaev, E.P. Shurina, M.A. Botchev // Computers and Mathematics with Applications. – 2008. – No. 55. – Pp. 2346-2362.
19. Михайлова Е.И., Шурина Э.П. Математическое моделирование высокочастотного электромагнитного поля в волноводных устройствах / Е.И. Михайлова, Э.П. Шурина // Вестник НГУ. Серия: Математика, механика, информатика. – 2013. – Т. 13. – №. 4. – С. 102-118.
20. Мысовских И.П. Интерполяционные кубатурные формулы. – Наука. Гл. ред. физ.-мат. лит., 1981.
21. Zhang L. et al. A set of symmetric quadrature rules on triangles and tetrahedra / L. Zhang, T. Cui, H. Liu // J. Comput. Math. – 2009. – Vol. 27. – No. 1. – Pp. 89-96.
22. Solin P., Segeth K., Dolezel I. Higher-order finite element methods. / P. Solin, K. Segeth, I. Dolezel – CRC Press, 2003. – 388 p.
23. Bergot M., Duruflé M. High-order optimal edge elements for pyramids, prisms and hexahedra / M. Bergot, M. Duruflé // Journal of Computational Physics. – 2013. – Vol. 232. – No. 1. – Pp. 189-213.
24. Dosopoulos S., Zhao B., Lee J.F. Non-conformal and parallel discontinuous Galerkin time domain method for Maxwell's equations: EM analysis of IC packages / S. Dosopoulos, B. Zhao, J.F. Lee // Journal of Computational Physics. – 2013. – Vol. 238. – Pp. 48-70.
25. Perugia I., Schötzau D. The *hp*-local discontinuous Galerkin method for low-frequency time-harmonic Maxwell equations / I. Perugia, D. Schötzau // Mathematics of Computation. – 2003. – Vol. 72. – No. 243. – Pp. 1179-1214.
26. Dosopoulos S., Lee J.F. Interconnect and lumped elements modeling in interior penalty discontinuous Galerkin time-domain methods / S. Dosopoulos, J.F. Lee // Journal of Computational Physics. – 2010. – Vol. 229. – No. 22. – Pp. 8521-8536.
27. Christophe A. et al. An Overlapping Nonmatching Grid Mortar Element Method for Maxwell's Equations / A. Christophe et al. // Magnetics, IEEE Transactions on. – 2014. – Vol. 50. – Iss. 2. – Pp. 409-412.
28. Gopalakrishnan J., Pasciak J.E. Multigrid for the mortar finite element method / J. Gopalakrishnan, J.E. Pasciak // SIAM Journal on Numerical Analysis. – 2000. – Vol. 37. – No. 3. – Pp. 1029-1052.

Приложение А. Текст основного модуля программы

```
// Класс векторный МКЭ
class VFEM
{
public:
    // Конфигурация
    config_type config;

    // Ввод данных
    bool input_phys(const string & phys_filename);
    bool input_mesh(const string & gmsh_filename);
    // Процедура для сборки СЛАУ
    void make_struct();
    void make_data();
    // Запуск решения СЛАУ
    void solve();
    // Вывод данных в 3D сетке
    bool output(const string & tecplot_filename);
    // Вывод данных в 2D сетке
    bool output_slice(const string & tecplot_filename, char slice_var, double slice_val,
                      char var1, double min_var1, double max_var1, size_t num_var_1,
                      char var2, double min_var2, double max_var2, size_t num_var_2);
    // Вывод данных по линии
    bool output_line(const string & tecplot_filename, char line_var1, double line_val1,
                     char line_var2, double line_val2, char var3, double min_var3,
                     double max_var3, size_t num_var);

    // Поиск конечного элемента по точке
    finite_element * get_fe(const point & p) const;

    // Решение в точке
    cvector3 solution(const point & p) const;
    cvector3 solution(const point & p, const finite_element * fe) const;
    // Ротор решения в точке
    cvector3 rotor(const point & p) const;
    cvector3 rotor(const point & p, const finite_element * fe) const;

    // Конечные элементы (тетраэдры)
    vector<finite_element> fes;

    void calculate_diff();

    // Основная СЛАУ
    SLAE slae;
    // СЛАУ на ядре
    SLAE ker_slae;
    // СЛАУ по границе
    SLAE surf_slae;

    // Узлы
    vector<point> nodes;
    // Ребра
    set<edge> edges;
    // Грани
    set<face> faces;
    // Ребра с источниками
    vector<edge_src> edges_src;
    // Треугольники
    vector<triangle *> trs;
    // Треугольники простые
    // (не используется при config.boundary_enabled == true)
    vector<triangle_base> trs_base;
    // Треугольники полные, для первых краевых
    // (не используется при config.boundary_enabled == false)
    vector<triangle_full> trs_full;
    // Точечные источники
    vector<pair<point, cvector3> > pss;
    // Физические области
    map<phys_id, phys_area> phys;

    // Число степеней свободы
    size_t dof_num;
    // Число степеней свободы ядра
    size_t ker_dof_num;
    // Соответствие глобальных степеней свободы и по границе
    // (не используется при config.boundary_enabled == false)
    map<size_t, size_t> global_to_local;
    // Степени свободы с первыми краевыми
    // (не используется при config.boundary_enabled == true)
    set<size_t> dof_first;
    // Степени свободы с первыми краевыми у ядра
    set<size_t> ker_dof_first;
    // Восьмиричное дерево поиска
    octree<finite_element> tree;

    // Получение степеней свободы тетраэдра в глобальной матрице
    size_t get_tet_dof(const tetrahedron_base * fe, size_t i) const;
    // Получение степеней свободы тетраэдра в матрице ядра
    size_t get_tet_ker_dof(const tetrahedron_base * fe, size_t i) const;
    // Получение степеней свободы треугольника в глобальной матрице
    size_t get_tr_dof(const triangle * tr, size_t i) const;
    // Получение степеней свободы треугольника в матрице ядра
    size_t get_tr_ker_dof(const triangle * tr, size_t i) const;
    // Получение степеней свободы треугольника в матрице по границе
    size_t get_tr_surf_dof(const triangle * tr, size_t i) const;
```

```
    // Генерация портрета глобальной матрицы
    void generate_portrait();
    // Генерация портрета глобальной матрицы ядра
    void generate_ker_portrait();
    // Сборка глобальной матрицы
    void assemble_matrix();
    // Применение краевых условий
    void applying_bound();
    // Генерация портрета по границе
    void generate_surf_portrait();
    // Применение источников на ребрах
    void apply_edges_sources();
    // Применение точечных источников
    void apply_point_sources();

protected:
    // Добавление локальных матриц от одного КЭ в глобальную
    template<typename U, typename V>
    void process_fe(const U * fe, const V *);

    // Добавление ребра в множество ребер
    size_t add_edge(edge ed, set<edge> & edges_set);
    // Добавление грани в множество граней
    size_t add_face(face fc, set<face> & faces_set);

    cpoint convert_point_to_pml(const point * p, const finite_element * fefe) const;
    void input_pml();
    // Параметры PML
    phys_pml_area phys_pml;

    // Проектирование на пространство ядра
    void to_kernel_space(const complex<double> * in, complex<double> * out) const;
    // Интерполяция на полное пространство
    void to_full_space(const complex<double> * in, complex<double> * out) const;
    // Скалярное произведение
    double dot_prod_self(const complex<double> * a) const;
    // Умножение матрицы с полного пространства на вектор
    void mul_matrix(const complex<double> * f, complex<double> * x) const;
    // Подсчет невязки
    void calc_residual(const complex<double> * x0, complex<double> * p) const;
};

void VFEM::generate_portrait()
{
    cout << "Generating portrait ..." << endl;

    set<size_t> * portrait = new set<size_t> [dof_num];
    for(size_t k = 0; k < fes.size(); k++)
    {
        show_progress("step 1", k, fes.size());

        array_t<size_t> dof(config.basis.tet_bf_num);
        for(size_t i = 0; i < config.basis.tet_bf_num; i++)
            dof[i] = get_tet_dof(&fes[k], i);

        for(size_t i = 0; i < config.basis.tet_bf_num; i++)
        {
            size_t a = dof[i];
            for(size_t j = 0; j < i; j++)
            {
                size_t b = dof[j];
                if(b > a)
                    portrait[b].insert(a);
                else
                    portrait[a].insert(b);
            }
        }
    }

    size_t gg_size = 0;
    for(size_t i = 0; i < dof_num; i++)
        gg_size += portrait[i].size();

    slae.alloc_all(dof_num, gg_size);

    slae.ig[0] = 0;
    slae.ig[1] = 0;
    size_t tmp = 0;
    for(size_t i = 0; i < dof_num; i++)
    {
        show_progress("step 2", i, dof_num);

        for(set<size_t>::iterator j = portrait[i].begin(); j != portrait[i].end(); ++j)
        {
            slae.jg[tmp] = *j;
            tmp++;
        }
        slae.ig[i + 1] = slae.ig[i] + portrait[i].size();

        portrait[i].clear();
    }

    delete [] portrait;
}

void VFEM::generate_ker_portrait()
{
    cout << "Generating kernel portrait ..." << endl;

    set<size_t> * portrait = new set<size_t> [ker_dof_num];
    for(size_t k = 0; k < fes.size(); k++)
    {
        show_progress("step 1", k, fes.size());
```

```
        array_t<size_t> ker_dof(config.basis.tet_ker_bf_num);
        for(size_t i = 0; i < config.basis.tet_ker_bf_num; i++)
            ker_dof[i] = get_tet_ker_dof(&fes[k], i);

        for(size_t i = 0; i < config.basis.tet_ker_bf_num; i++)
        {
            size_t a = ker_dof[i];
            for(size_t j = 0; j < i; j++)
            {
                size_t b = ker_dof[j];
                if(b > a)
                    portrait[b].insert(a);
                else
                    portrait[a].insert(b);
            }
        }
    }

    size_t gg_size = 0;
    for(size_t i = 0; i < ker_dof_num; i++)
        gg_size += portrait[i].size();

    ker_slae.alloc_all(ker_dof_num, gg_size);

    ker_slae.ig[0] = 0;
    ker_slae.ig[1] = 0;
    size_t tmp = 0;
    for(size_t i = 0; i < ker_dof_num; i++)
    {
        show_progress("step 2", i, ker_dof_num);

        for(set<size_t>::iterator j = portrait[i].begin(); j != portrait[i].end(); ++j)
        {
            ker_slae.jg[tmp] = *j;
            tmp++;
        }
        ker_slae.ig[i + 1] = ker_slae.ig[i] + portrait[i].size();

        portrait[i].clear();
    }

    delete [] portrait;
}

void VFEM::generate_surf_portrait()
{
    cout << "Generaing surface portrait ..." << endl;

    size_t dof_surf_num = global_to_local.size();
    set<size_t> * portrait = new set<size_t> [dof_surf_num];
    for(size_t k = 0; k < trs.size(); k++)
    {
        show_progress("step 1", k, trs.size());

        array_t<size_t> dof_surf(config.basis.tr_bf_num);
        for(size_t i = 0; i < config.basis.tr_bf_num; i++)
            dof_surf[i] = get_tr_surf_dof(trs[k], i);

        if(trs[k]->phys->type_of_bounds == 1)
        {
            for(size_t i = 0; i < config.basis.tr_bf_num; i++)
            {
                size_t a = dof_surf[i];
                for(size_t j = 0; j < i; j++)
                {
                    size_t b = dof_surf[j];
                    if(b > a)
                        portrait[b].insert(a);
                    else
                        portrait[a].insert(b);
                }
            }
        }
    }

    size_t gg_size = 0;
    for(size_t i = 0; i < dof_surf_num; i++)
        gg_size += portrait[i].size();

    surf_slae.alloc_all(dof_surf_num, gg_size);

    surf_slae.ig[0] = 0;
    surf_slae.ig[1] = 0;
    size_t tmp = 0;
    for(size_t i = 0; i < dof_surf_num; i++)
    {
        show_progress("step 2", i, dof_surf_num);

        for(set<size_t>::iterator j = portrait[i].begin(); j != portrait[i].end(); ++j)
        {
            surf_slae.jg[tmp] = *j;
            tmp++;
        }
        surf_slae.ig[i + 1] = surf_slae.ig[i] + portrait[i].size();

        portrait[i].clear();
    }

    delete [] portrait;
}

// Добавление локальных матриц от одного КЭ в глобальную
```

```
template<typename U, typename V>
void VFEM::process_fe(const U * curr_fe, const V *)
{
    // Получение физических параметров для заданного КЭ
    phys_area ph = curr_fe->get_phys_area();
    complex<double> k2(- ph.epsilon * ph.omega * ph.omega, ph.omega * ph.sigma);

    // Инициализация параметров вычислителей для правой части
    pair<const config_type *, array_t<evaluator<complex<double> > *, 3> >
            params_object(& config, array_t<evaluator<complex<double> > *, 3>());
    if(config.right_enabled)
    {
        map<size_t, array_t<evaluator<complex<double> >, 3> >::iterator
                it = config.right.values.find(ph.gmsh_num);
        if(it != config.right.values.end())
            for(size_t i = 0; i < 3; i++)
                params_object.second[i] = &(it->second[i]);
        else
            for(size_t i = 0; i < 3; i++)
            {
                evaluator<complex<double> > * ev_curr = &(config.right.default_value[i]);
                params_object.second[i] = ev_curr;
                ev_curr->set_var("epsilon", ph.epsilon);
                ev_curr->set_var("sigma", ph.sigma);
                ev_curr->set_var("mu", ph.mu);
                ev_curr->set_var("J0", ph.J0);
                ev_curr->set_var("k2", k2);
            }
    }

    // Получение степеней свободы
    array_t<size_t> dof(config.basis.tet_bf_num);
    for(size_t i = 0; i < config.basis.tet_bf_num; i++)
        dof[i] = get_tet_dof(curr_fe, i);

    // Получение локальных матриц и правой части
    V matrix_G = curr_fe->G();
    V matrix_M = curr_fe->M();
    array_t<complex<double> > array_rp = curr_fe->rp(func_rp, &params_object);

    // Основная матрица
    for(size_t i = 0; i < config.basis.tet_bf_num; i++)
    {
        complex<double> add;
        size_t i_num = dof[i];
        for(size_t j = 0; j < i; j++)
        {
            size_t j_num = dof[j];
            add = matrix_G[i][j] / ph.mu + matrix_M[i][j] * k2;
            slae.add(i_num, j_num, add);
        }
        add = matrix_G[i][i] / ph.mu + matrix_M[i][i] * k2;
        slae.di[i_num] += add;
        add = array_rp[i];
        slae.rp[i_num] += add;
    }

    // Матрица ядра
    if(config.v_cycle_enabled)
    {
        array_t<size_t> ker_dof(config.basis.tet_ker_bf_num);
        for(size_t i = 0; i < config.basis.tet_ker_bf_num; i++)
            ker_dof[i] = get_tet_ker_dof(curr_fe, i);

        V matrix_K = curr_fe->K();
        for(size_t i = 0; i < config.basis.tet_ker_bf_num; i++)
        {
            size_t i_num = ker_dof[i];
            for(size_t j = 0; j < i; j++)
                ker_slae.add(i_num, ker_dof[j], matrix_K[i][j] * k2);
            ker_slae.di[i_num] += matrix_K[i][i] * k2;
        }
    }
}

void VFEM::assemble_matrix()
{
    cout << "Assembling matrix ..." << endl;

    cout << " > Assembling matrix ..." << endl;
    // Сборка основной матрицы
    for(size_t k = 0; k < fes.size(); k++)
    {
        show_progress("", k, fes.size());
        if(!is_pml(fes[k].barycenter, &fes[k], &phys_pml))
            process_fe(fes[k].to_std(), (matrix_t<double> *)NULL);
        else
            process_fe(&fes[k], (l_matrix *)NULL);
    }
}

void VFEM::applying_bound()
{
    cout << " > Applying first bound ..." << endl;

    if(config.boundary_enabled)
    {

        // Решение СЛАУ по границе
        if(global_to_local.size() > 0)
        {
            for(size_t k = 0; k < trs.size(); k++)
            {
```

```
            show_progress("building matrix", k, trs.size());

            phys_area ph = trs[k]->get_phys_area();
            if(ph.type_of_bounds == 1)
            {
                // Получение физических параметров для текущего треугольника границы
                complex<double> k2(- ph.epsilon * ph.omega * ph.omega, ph.omega * ph.sigma);

                // Инициализация параметров вычислителей для первого краевого
                pair<const config_type *, array_t<evaluator<complex<double> > *, 3> >
                        params_object(& config, array_t<evaluator<complex<double> > *, 3>());
                map<size_t, array_t<evaluator<complex<double> >, 3> >::iterator
                        it = config.boundary.values.find(ph.gmsh_num);
                if(it != config.boundary.values.end())
                    for(size_t i = 0; i < 3; i++)
                        params_object.second[i] = &(it->second[i]);
                else
                    for(size_t i = 0; i < 3; i++)
                    {
                        evaluator<complex<double> > * ev_curr = &(config.boundary.default_value[i]);
                        params_object.second[i] = ev_curr;
                        ev_curr->set_var("epsilon", ph.epsilon);
                        ev_curr->set_var("sigma", ph.sigma);
                        ev_curr->set_var("mu", ph.mu);
                        ev_curr->set_var("J0", ph.J0);
                        ev_curr->set_var("k2", k2);
                    }

                // Получение степеней свободы
                array_t<size_t> dof_surf(config.basis.tr_bf_num);
                for(size_t i = 0; i < config.basis.tr_bf_num; i++)
                    dof_surf[i] = get_tr_surf_dof(trs[k], i);

                // Получение локальных матриц и правой части
                matrix_t<double> M_surf = trs[k]->M();
                array_t<complex<double> > b_surf = trs[k]->rp(func_b1, &params_object);

                // Добавляем полученное в матрицу с краевыми
                for(size_t i = 0; i < config.basis.tr_bf_num; i++)
                {
                    for(size_t j = 0; j < i; j++)
                        surf_slae.add(dof_surf[i], dof_surf[j], M_surf[i][j]);
                    surf_slae.di[dof_surf[i]] += M_surf[i][i];
                    surf_slae.rp[dof_surf[i]] += b_surf[i];
                }
            }
        }
        surf_slae.solve(config.eps_slae_bound, config.max_iter);
    }

    // Учет первых краевых
    for(size_t k = 0; k < slae.n; k++)    // Пробегаем по всей матрице
    {
        show_progress("applying", k, slae.n);

        map<size_t, size_t>::iterator g2l_k = global_to_local.find(k);
        if(g2l_k != global_to_local.end())
        {
            complex<double> val = surf_slae.x[g2l_k->second];
            slae.rp[k] = val;
            slae.di[k] = 1.0;
            slae.x[k] = val; // Начальное приближение сразу знаем

            for(size_t i = slae.ig[k]; i < slae.ig[k + 1]; i++)
            {
                if(global_to_local.find(slae.jg[i]) == global_to_local.end())
                    slae.rp[slae.jg[i]] -= slae.gg[i] * val;
                slae.gg[i] = 0.0;
            }
        }
        else
        {
            for(size_t i = slae.ig[k]; i < slae.ig[k + 1]; i++)
            {
                if(global_to_local.find(slae.jg[i]) != global_to_local.end())
                {
                    complex<double> val = surf_slae.x[global_to_local[slae.jg[i]]];
                    slae.rp[k] -= slae.gg[i] * val;
                    slae.gg[i] = 0.0;
                }
            }
        }
    }

}
else
{

    for(size_t k = 0; k < slae.n; k++)    // Пробегаем по всей матрице
    {
        show_progress("", k, slae.n);

        if(dof_first.find(k) != dof_first.end())
        {
            slae.rp[k] = 0.0;
            slae.di[k] = 1.0;

            for(size_t i = slae.ig[k]; i < slae.ig[k + 1]; i++)
                slae.gg[i] = 0.0;
        }
        else
        {
            for(size_t i = slae.ig[k]; i < slae.ig[k + 1]; i++)
```

```
                {
                    if(dof_first.find(slae.jg[i]) != dof_first.end())
                        slae.gg[i] = 0.0;
                }
            }
        }

    }

    if(config.v_cycle_enabled)
    {
        // Первые краевые для матрицы ядра
        for(size_t k = 0; k < ker_slae.n; k++)
        {
            // Пробегаем по всей матрице
            if(ker_dof_first.find(k) != ker_dof_first.end())
            {
                ker_slae.di[k] = 1.0;
                for(size_t i = ker_slae.ig[k]; i < ker_slae.ig[k + 1]; i++)
                    ker_slae.gg[i] = 0.0;
            }
            else
            {
                for(size_t i = ker_slae.ig[k]; i < ker_slae.ig[k + 1]; i++)
                {
                    if(ker_dof_first.find(ker_slae.jg[i]) != ker_dof_first.end())
                        ker_slae.gg[i] = 0.0;
                }
            }
        }
    }
}

void VFEM::apply_edges_sources()
{
    cout << " > Applying edges sources ..." << endl;
    for(size_t k = 0; k < edges_src.size(); k++)
    {
        show_progress("", k, edges_src.size());
        size_t pos = edges_src[k].num; // Заносим только в роторные функции!
        slae.rp[pos] += complex<double>(0.0, -1.0) * edges_src[k].phys->J0 * edges_src[k].phys->omega * edges_src[k].direction;
    }
}

finite_element * VFEM::get_fe(const point & p) const
{
    finite_element * fe = NULL;
    fe = tree.find(p.x, p.y, p.z);
    if(!fe)
        cout << "Warning: Point " << p << " is outside area!" << endl;
    return fe;
}

cvector3 VFEM::solution(const point & p) const
{
    return solution(p, get_fe(p));
}

cvector3 VFEM::solution(const point & p, const finite_element * fe) const
{
    cvector3 result;
    if(fe)
    {
        for(size_t i = 0; i < config.basis.tet_bf_num; i++)
            result = result + slae.x[get_tet_dof(fe, i)] * fe->w(i, p);
    }
    return result;
}

cvector3 VFEM::rotor(const point & p) const
{
    return rotor(p, get_fe(p));
}

cvector3 VFEM::rotor(const point & p, const finite_element * fe) const
{
    cvector3 result;
    if(fe)
    {
        for(size_t i = 0; i < config.basis.tet_bf_num; i++)
            result = result + slae.x[get_tet_dof(fe, i)] * fe->rotw(i, p);
    }
    return result;
}

void VFEM::make_struct()
{
    if(config.boundary_enabled && global_to_local.size() > 0)
        generate_surf_portrait();
    generate_portrait();
    if(config.v_cycle_enabled)
        generate_ker_portrait();
}

void VFEM::make_data()
{
    assemble_matrix();
    apply_edges_sources();
    applying_bound();
}
```