

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БАШКИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
КАФЕДРА ГЕОФИЗИКИ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ПО ПРОГРАММЕ БАКАЛАВРИАТА

КАДЫРОВ АЛМАЗ ВЕНЕРОВИЧ
«РЕШЕНИЕ ПРЯМОЙ ЗАДАЧИ БОКОВОГО КАРОТАЖНОГО ЗОНДИРОВАНИЯ
МЕТОДАМИ ЧИСЛЕННОГО МОДЕЛИРОВАНИЯ»

Выполнил:

Студент 4 курса очной формы обучения
Направление подготовки – «Физика»
Профиль – «Физика Земли и планет»

Руководитель:

Доцент, к.ф.-м.н.

_____ / И.С. Ремеев

Уфа – 2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1 ПРЯМАЯ ЗАДАЧА БКЗ	5
2 МАТЕМАТИЧЕСКАЯ ПОСТАНОВКА ЗАДАЧИ	8
2.1 Слабая постановка	8
2.2 Аппроксимация	11
3 РАСЧЕТНАЯ СЕТКА	13
3.1 Постановка задачи	13
3.2 Построение одномерной сетки	14
3.3 Построение двумерной сетки	16
3.4 Результаты	32
4 РЕШЕНИЕ	33
4.1 Модель 1	33
4.2 Модель 2	39
ЗАКЛЮЧЕНИЕ	45
БИБЛИОГРАФИЧЕСКИЙ СПИСОК	46
ПРИЛОЖЕНИЕ А	48

ВВЕДЕНИЕ

Проблема и ее актуальность. Геофизические исследования скважин электрическими методами используются для реконструкции пространственного распределения удельного электрического сопротивления (УЭС), которое связано с нефтенасыщением. Задачи электрокаротажа нелинейны, а обратная задача определения параметров по измерениям некорректна (неоднозначна). Решение задач в полных постановках весьма ресурсоемко и плохо подходит для промышленного применения.

Обработка диаграмм бокового каротажного зондирования (БКЗ) заключается в выделении пластов и отсчете существенных значений ρ_k против них, построении кривых зависимости ρ_k от размера зонда — кривых зондирования и кривых БКЗ, сравнении полученных кривых с расчетными для определения удельного сопротивления пластов и выявлении зон проникновения фильтрата промывочной жидкости (ПЖ) в пласт [4].

Основная идея палеточного подхода для решения задач скважинной геоэлектрики заключается в задании функции связи входных и выходных данных решаемой задачи на основе интерполяции по дискретному множеству имеющихся решений. Преимущества рассматриваемого подхода состоят в отсутствии зависимости скорости решения от входных данных и существенном снижении требований к качеству начального приближения, как это требуется в итеративных алгоритмах минимизации невязки теоретических и экспериментальных данных [3]. Недостатком является больший объем данных, необходимых для хранения палетки.

В настоящее время на практике используют палетки, как правило, для двух- и трехслойных моделей, покрывающих наиболее распространенные ситуации.

На практике возникает необходимость в палетках с такими параметрами, которые не опубликованы. Поэтому была поставлена задача разработки метода

решения прямой задачи БКЗ, которая позволила бы задавать необходимые параметры модели для построения палеток.

1 ПРЯМАЯ ЗАДАЧА БКЗ

БКЗ состоит в измерении потенциала электрического поля, создаваемого в среде точечным электрическим зарядом. Далее по измеренному распределению потенциала предпринимается попытка воссоздать геометрию среды, а также ее удельное сопротивление в разных точках пространства.

Поскольку наличие электрического поля в проводящей среде означает возникновение в ней тока, точечный заряд, о котором говорилось выше, фактически представляет собой электрод, находящийся под напряжением относительно бесконечно удаленных точек (условимся называть его излучающим). Технически это достигается за счет использования дополнительного электрода, располагаемого на большом расстоянии от первого; тем самым цепь замыкается, и течение тока становится возможным. В действительности нас не будет интересовать распределение поля на существенном удалении от излучающего электрода, поэтому мы ограничимся моделью, в которой ток растекается от него на условную бесконечность, обладающую нулевым потенциалом. Следует отметить, что на практике как размер излучающего электрода, так и его потенциал являются конечными. При этом, однако, становится необходимым детальный учет геометрии области контакта прибора со средой. Дабы избежать подобных затруднений, мы будем считать электрод точечным, а распределение поля вокруг него – сингулярным, задавая интенсивность излучения не потенциалом излучающего электрода, а силой протекающего через него тока.

Распределение потенциала в среде регистрируется приемными электродами, наличие которых само по себе искажает линии тока в пространстве. Этим эффектом, однако, можно пренебречь, если величина тока, протекающего через приемные электроды, невелика. Далее мы будем считать, что последнее предположение верно.

Расположение приемных электродов варьируется в зависимости от того, какие именно величины предполагается наблюдать непосредственно. Если речь идет об измерении потенциала электрического поля как такового, то каждый

отдельно взятый электрод может рассматриваться как самостоятельный зонд; в этом случае мы имеем дело с потенциал-зондом (строго говоря, техническая реализация потенциал-зонда может быть более сложной, но для теоретических расчетов это значения не имеет). Если же измерять предполагается не сам потенциал, а составляющую его градиента (т.е., с точностью до знака, компоненту напряженности электрического поля), используется пара близко расположенных приемных электродов - так называемый градиент-зонд. При таком подходе рассматриваются не потенциалы электродов в отдельности, а разность их потенциалов, которая и позволяет судить об искомой величине градиента. Независимо от типа зонда, под его длиной мы будем понимать расстояние между излучающим электродом и приемными.

На практике результаты измерений БКЗ принято представлять не значениями потенциалов, которые пропорциональны силе тока в излучающем электроде, а независящими от тока значениями кажущихся удельных сопротивлений. Под кажущимся удельным сопротивлением понимается удельное сопротивление такой однородной среды, Зв которой измерение данным зондом привело бы в точности к тем же результатам, что были получены фактически. Таким образом, при исследовании однородной среды методом БКЗ все зонды, независимо от их типов и длин, приводили бы к одному и тому же значению кажущегося удельного сопротивления, которое соответствовало бы истинному удельному сопротивлению вещества, заполняющего пространство. В случае неоднородной среды никакой прямой аналогии нет, и вычисление ее истинных параметров по кажущимся удельным сопротивлениям становится довольно сложной задачей, которая и представляет собой обратную задачу БКЗ. На практике она может быть сформулирована следующим образом: даны значения кажущегося удельного сопротивления, полученные с нескольких разных зондов (отличающихся, как правило, длиной, но также, возможно, типом и расположением); найти удельное сопротивление среды в каждой точке пространства. Ясно, что в отсутствие дополнительных упрощающих предположений эта задача становится принципиально неразрешимой.

Прямая задача БКЗ заключается в прогнозировании кажущихся удельных сопротивлений, которые должны быть получены в процессе измерения зондами заданных типов и длин. Характеристики сред, в которых осуществляются измерения, также будем считать известными. Прямая задача, в отличие от обратной, всегда разрешима.

2 МАТЕМАТИЧЕСКАЯ ПОСТАНОВКА ЗАДАЧИ

Формализуем задачу, описанную в предыдущем пункте. Во-первых, поместим излучающий электрод в начало координат $\mathbf{x} = 0$ и введем обозначение I для силы протекающего через него тока.

Электрический потенциал $u = u(\mathbf{x})$ электрического поля удовлетворяет задаче Дирихле для уравнения Пуассона [1, с. 67]:

$$\nabla \cdot (\nabla u(\mathbf{x})/\rho(\mathbf{x})) = f(\mathbf{x}), \quad \mathbf{x} \in \Omega, \quad (2.1)$$

$$u(\mathbf{x}) = 0, \quad \mathbf{x} \in \partial\Omega, \quad (2.2)$$

где Ω — бесконечная область пространства, $\partial\Omega$ — граница области Ω , точки которой находятся на бесконечности ($\mathbf{x} \in \partial\Omega : |\mathbf{x}| \rightarrow \infty$), $f = f(\mathbf{x})$ — источниковый член.

Источниковый член в случае точечного источника потенциала в точке $\mathbf{x} = 0$:

$$f(\mathbf{x}) = -I\delta(\mathbf{x}), \quad (2.3)$$

где $\delta(\mathbf{x})$ — обобщенная функция дельта-функция Дирака.

2.1 Слабая постановка

Достаточно гладкая функция u , удовлетворяющая обеим равенствам (2.1) и (2.2), известна как классическое решение для краевой задачи. Для задачи Дирихле, функция u является классическим решением, только если она дважды непрерывно дифференцируемая в открытой области Ω ($u \in C^2(\Omega)$) и непрерывная в закрытой области Ω ($u \in C^0(\overline{\Omega})$). В случаях областей с негладкими границами или функций, не являющихся гладкими, f — источниковых членов (к тому же функция УЭС ρ негладкая), функция u , удовлетворяющая (2.1)–(2.2), может не быть гладкой (или обычной) достаточно, чтобы считалась классическим решением. Поскольку f не является гладкой функцией, то вторая производная решения u не является гладкой, и следовательно $u \notin C^2(\Omega)$ и не

существует классического решения. Для таких задач, которые возникают из вполне обоснованных математических моделей, необходима альтернативная запись краевых задач. Поскольку эта альтернативная запись менее ограничивает в плане допустимых входных данных, она названа слабой постановкой [6, с. 14].

FEniCS — популярная вычислительная платформа с открытым исходным кодом (LGPLv3) для решения уравнений в частных производных (ДУЧП). FEniCS позволяет пользователям быстро переводить научные модели в эффективный конечно-элементный код. Имеет высокоуровневые интерфейсы для Python и C++.

FEniCS основан на методе конечных элементов, который является общим и эффективным математическим механизмом для численного решения ДУЧП. Отправной точкой для методов конечных элементов является ДУЧП, выраженное в вариационной форме. Опыт показывает, что возможно работать с FEniCS в качестве инструмента для решения ДУЧП, даже не имея глубоких знаний о методе конечных элементов, при условии, что есть помощь постановки ДУЧП в виде вариационной задачи [7].

Простой рецепт превращения ДУЧП в вариационную задачу — умножить ДУЧП на функцию v , интегрировать полученное уравнение по области Ω и выполнить интегрирование по частям с производными второго порядка. Функция v , которая множит ДУЧП, называется тестовой функцией. Неизвестная функция u , подлежащая аппроксимации, называется пробной функцией. Термины пробная и тестовая функции также используются в программах FEniCS. Пробная и тестовая функции принадлежат к определенным так называемым функциональным пространствам, которые определяют свойства функций.

В данном случае, сначала умножим уравнение Пуассона на тестовую функцию v и интегрируем по заданной области Ω :

$$-\int_{\Omega} \nabla \cdot (\nabla u / \rho) v \, dx = \int_{\Omega} f v \, dx. \quad (2.4)$$

Здесь мы обозначили через dx дифференциальный элемент для интегрирования по области Ω . Позже мы обозначим через ds дифференциальный элемент для интегрирования по границе Ω .

Как правило, когда получают вариационную постановку, избавляются от высших степеней производных. Здесь мы имеем производную второго порядка функции u , которая может быть переведена в производные первого порядка функций u и v посредством применения метода интегрирования по частям. Формула гласит

$$\int_{\Omega} \nabla \cdot (\nabla u / \rho) v \, dx = \int_{\partial\Omega} \frac{1}{\rho} \frac{\partial u}{\partial n} v \, ds - \int_{\Omega} \frac{1}{\rho} \nabla u \cdot \nabla v \, dx, \quad (2.5)$$

где $\frac{\partial u}{\partial n} = \nabla u \cdot n$ есть производная u по направлению внешней нормали n на границе.

Еще одной особенностью вариационных постановок является то, что тестовая функция должна зануляться на участках границы, где известно u . В нашей задаче, это значит, что $v = 0$ на всей границе $\partial\Omega$. Первый член в правой части уравнения следовательно пропадает. Из (2.4) и (2.5) следует

$$-\int_{\Omega} \frac{1}{\rho} \nabla u \cdot \nabla v \, dx = \int_{\Omega} f v \, dx. \quad (2.6)$$

Если мы потребуем, чтобы это уравнение выполнялось для всех тестовых функций v в некотором подходящем пространстве \hat{V} , так называемом тестовом пространстве, то мы получаем четко определенную математическую задачу, которая однозначно определяет решение u , которое лежит в некотором (возможно, другом) функциональном пространстве V , так называемое пробное пространство. Мы обозначим (2.6) как слабую или вариационную форму исходной краевой задачи (2.1) - (2.2).

Правильная постановка нашей вариационной задачи теперь выглядит следующим образом: найти $u \in V$ такой, что

$$-\int_{\Omega} \frac{1}{\rho} \nabla u \cdot \nabla v \, dx = \int_{\Omega} f v \, dx. \quad (2.7)$$

Пробные и тестовые пространства V и \hat{V} в данной задаче определены как

$$V = \{v \in H^1(\Omega) : v = 0 \text{ on } \partial\Omega\}, \quad (2.8)$$

$$\hat{V} = \{v \in H^1(\Omega) : v = 0 \text{ on } \partial\Omega\}. \quad (2.9)$$

Область определения функций пространств есть Ω , и эти функции удовлетворяют граничным условиям: значения функций на $\partial\Omega$ равны нулю.

В кратце, $H^1(\Omega)$ — математически известное пространство Соболева, содержащее такие функции v , что v^2 и $|\nabla v|^2$ имеют конечные интегралы по Ω (по сути это означает, что функции непрерывны). Решение лежащего в основе ДУЧП должно лежать в функциональном пространстве, где производные также являются непрерывными, но пространство Соболева $H^1(\Omega)$ допускает функции с разрывными производными. Это более слабое требование непрерывности u в вариационном утверждении (2.7), как результат интегрирования по частям, имеет большие практические последствия, когда речь идет о построении пространств функций конечных элементов. В частности, он позволяет использовать кусочно-полиномиальные функциональные пространства; то есть функциональные пространства, построенные путем сшивания полиномиальных функций в простых областях, таких как интервалы, треугольники или тетраэдры.

2.2 Аппроксимация

Вариационная задача (2.7) является непрерывной задачей: она определяет решение u в бесконечномерном функциональном пространстве V . Метод конечных элементов для уравнения Пуассона находит приближенное решение вариационной задачи (2.7) путем замены бесконечномерных функциональных пространств V и \hat{V} на дискретные (конечномерные) пробное и тестовые функциональные пространства $V_h \subset V$ и $\hat{V}_h \subset \hat{V}$ [7, с. 14]. Дискретная вариационная задача гласит: найдите $u_h \in V_h \subset V$, для которого

$$\int_{\Omega} \nabla u_h \cdot \nabla v dx = \int_{\Omega} f v dx \quad \forall v \in \hat{V}_h \subset \hat{V}. \quad (2.10)$$

Эта вариационная задача вместе с подходящим определением функциональных пространств V_h и \hat{V}_h однозначно определяют наше приближенное численное решение уравнения Пуассона (2.1). Заметим, что граничные условия записаны в части определения пробного и тестового пространств. Математическая структура может показаться сложной на первый взгляд, но хорошая новость заключается в том, что конечно-элементная вариационная задача (2.10) выглядит так же, как непрерывная вариационная задача (2.7), и FEniCS может автоматически решать вариационную задачу такие проблемы, как (2.10).

Приближенное решение разыскивается в виде:

$$u_h = \sum_{i=1}^N c_i \varphi_i(\mathbf{x}) \quad (2.11)$$

где $\varphi_1(\mathbf{x}), \varphi_2(\mathbf{x}), \dots, \varphi_N(\mathbf{x})$ — некоторые заданные (базисные) функции, удовлетворяющие граничным условиям; c_1, c_2, \dots, c_N — неизвестные, подлежащие вычислению. Их находят, решая систему алгебраических уравнений [2, с. 14]:

$$\int_{\Omega} \frac{1}{\rho} \nabla u_h \cdot \nabla \varphi_i \, dx = \int_{\Omega} f v \, dx, \quad i = 1, 2, \dots, N. \quad (2.12)$$

$\varphi_i(\mathbf{x})$ — базисная функция, непрерывная на Ω , равная нулю на $\partial\Omega$ и заданная таким образом:

$$\varphi_i(\mathbf{x}_j) = \begin{cases} 0, & \text{если } \mathbf{x}_i \neq \mathbf{x}_j, \\ 1, & \text{если } \mathbf{x}_i = \mathbf{x}_j; \end{cases} \quad (2.13)$$

где \mathbf{x}_i — координаты i -того узла сетки.

3 РАСЧЕТНАЯ СЕТКА

Сложность численного расчета данной задачи следующая. При применении метода конечных элементов к данной задаче возникают затруднения, связанные с тем, что искомый потенциал поля имеет особенность (уходит на бесконечность) в окрестности источника тока. Это приводит к необходимости сгущения расчётной сетки в этой окрестности, но сильное сгущение приводит к чрезмерному росту числа узлов сетки и, как следствие, сильному росту времени, требуемому на решение.

Поэтому представляет актуальность задача построения оптимальной сетки, в которой сочетаются высокая точность получаемого решения и малое число узлов.

3.1 Постановка задачи

В качестве модельной рассматривалась задача для трёхслойной осесимметричной модели системы скважина–пласт: задача рассматривается в цилиндрической системе координат (r, z) , где $r > 0$ — радиальная координата, z — осевая координата, по которой область однородна (единственный однородный пласт бесконечной мощности), первый слой — сама скважина ($0 < r < 1$) с УЭС $\rho_c = 1$, второй слой — зона проникновения ($1 < r < r_{3\pi} = 9$) с УЭС $\rho_{3\pi} = 15$, третий слой — пласт ($r > r_\pi$) с УЭС $\rho_\pi = 10$. В начале координат размещается источник тока силой 1.

В качестве основы для расчётной сетки использовалась одномерная сетка, узлы которой рассчитаны в параграфе 3.2, отложенная на лучах, исходящих из начала координат под различными углами. Затем эта основа дополнялась узлами, лежащими на границах разрыва УЭС $r = 1$ и $r = r_\pi$. Также производилось дополнительное адаптивное сгущение сетки в областях, в которых наблюдалось сильное искажение решения по сравнению с эталонным (полученным на сетке с большим числом узлов во всей области).

3.2 Построение одномерной сетки

Задача БКЗ моделируется, как правило, для осесимметричного случая, что приводит к двумерным задачам. Известно, что точное решение (потенциал электрического поля) для случая одинакового во всём пространстве УЭС есть константа, делённая на расстояние до источника:

$$\varphi(r, z) = \frac{C}{\sqrt{r^2 + z^2}}$$

(здесь начало координат совпадает с источником тока)

Поскольку для любого распределения УЭС решение в малой окрестности источника тока практически определяется УЭС в этой же окрестности (а она считается постоянной, равной УЭС жидкости в скважине), решение будет всегда иметь одну и ту же особенность в начале координат и изменяться плавно вдали от источника тока. В качестве первого приближения к оптимальной сетке можно использовать оптимальную одномерную сетку для известного распределения $y = 1/x$, и затем эту одномерную сетку распространять на несколько лучей, исходящих из начала координат.

Поэтому первой поставленной задачей было построение оптимальной одномерной сетки для интерполяции известной функции $y = 1/x$:

Задача. Данна функция $y = 1/x$ на интервале $[\delta, +\infty)$ ($\delta > 0$). Построить на ней сетку $\{x_i, i = 0, 1, \dots\}$ из наименьшего числа узлов (начиная с $x_0 = \delta$) так, чтобы разность между функцией y и приближающим её сплайном первого порядка на этой сетке не превышала заданного значения ε .

Математическая постановка:

$$f(x) = \frac{1}{x} \text{ — исходная функция}$$

$$X = (x_i), i \in [0, N] \text{ — сетка}$$

$$x_0 = \delta \text{ — известное}$$

$$F(x) = \begin{cases} F_0(x), & x \in [x_0, x_1] \\ F_1(x), & x \in [x_1, x_2] \\ \dots \\ F_{N-1}(x), & x \in [x_{N-1}, x_N] \end{cases} \quad (3.1)$$

$$F_i(x) = A_i x + B_i, \quad x \in [x_i, x_{i+1}] \quad (3.2)$$

$$\begin{cases} F_i(x_i) = f(x_i) \\ F_i(x_{i+1}) = f(x_{i+1}) \end{cases} \Rightarrow A_i, B_i \quad (3.3)$$

$$R(x) = \begin{cases} R_0(x), & x \in [x_0, x_1] \\ R_1(x), & x \in [x_1, x_2] \\ \dots \\ R_{N-1}(x), & x \in [x_{N-1}, x_N] \end{cases} \quad (3.4)$$

$$R_i(x) = |F_i(x) - f(x)|, \quad x \in [x_i, x_{i+1}] \quad (3.5)$$

$$R'_i(x) = 0 \Rightarrow x_{\max} \quad (3.6)$$

$$R_i(x_{\max}) = \varepsilon \Rightarrow x_{i+1} \quad (3.7)$$

Решение:

$$\begin{cases} F_i(x_i) = f(x_i) \\ F_i(x_{i+1}) = f(x_{i+1}) \end{cases} \Rightarrow \begin{cases} A_i x_i + B_i = f(x_i) \\ A_i x_{i+1} + B_i = f(x_{i+1}) \end{cases} \Rightarrow$$

$$\begin{cases} A_i = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} \\ B_i = f(x_i) - A_i x_i \end{cases} \quad (3.8)$$

$$R_i(x) = A_i x + B_i - \frac{1}{x} \quad (3.9)$$

$$R'_i(x) = A_i + \frac{1}{x^2} = 0 \Rightarrow x_{\max} = \frac{1}{\sqrt{-A_i}} \quad (3.10)$$

$$R_i(x_{\max}) = A_i \frac{1}{\sqrt{-A_i}} + B_i - \sqrt{-A_i} = \varepsilon \Rightarrow \quad (3.11)$$

$$x_{i+1} = \begin{cases} \frac{\varepsilon x_i^2 + x_i - 2\sqrt{\varepsilon x_i^3}}{\varepsilon^2 x_i^2 - 2\varepsilon x_i + 1}, & \text{— выражение для нового левого узла} \\ \frac{\varepsilon x_i^2 + x_i + 2\sqrt{\varepsilon x_i^3}}{\varepsilon^2 x_i^2 - 2\varepsilon x_i + 1}; & \text{— выражение для нового правого узла} \end{cases} \quad (3.12)$$

Решение уравнения (3.11) было получено посредством библиотеки SymPy в Python.

3.3 Построение двумерной сетки

Для построения вершин шестиугольников использованы одномерные сетки: узлы решения для гиперболы и узлы, равноотстоящие в логарифмическом масштабе. Использована реализация триангуляции библиотеки triangle на Python с различными настройками: режимы добавления новых узлов, учет графа, значения ограничения снизу значений углов треугольников.

1. Построены вершины шестиугольников из узлов решения для гиперболы. На рис. 3.1 Построены отрезки графа внутреннего и внешнего границ (граф изображен с красным цветом) и обозначена метка дыры (изображена красным крестиком), в которой исключена триангуляция. Вызвана триангуляция библиотеки triangle с настройкой 'pq30', где 'p' сообщает библиотеке о вводе графа, 'q30' — ограничение снизу 30 градусов значений углов треугольников триангуляции.

421 узлов.

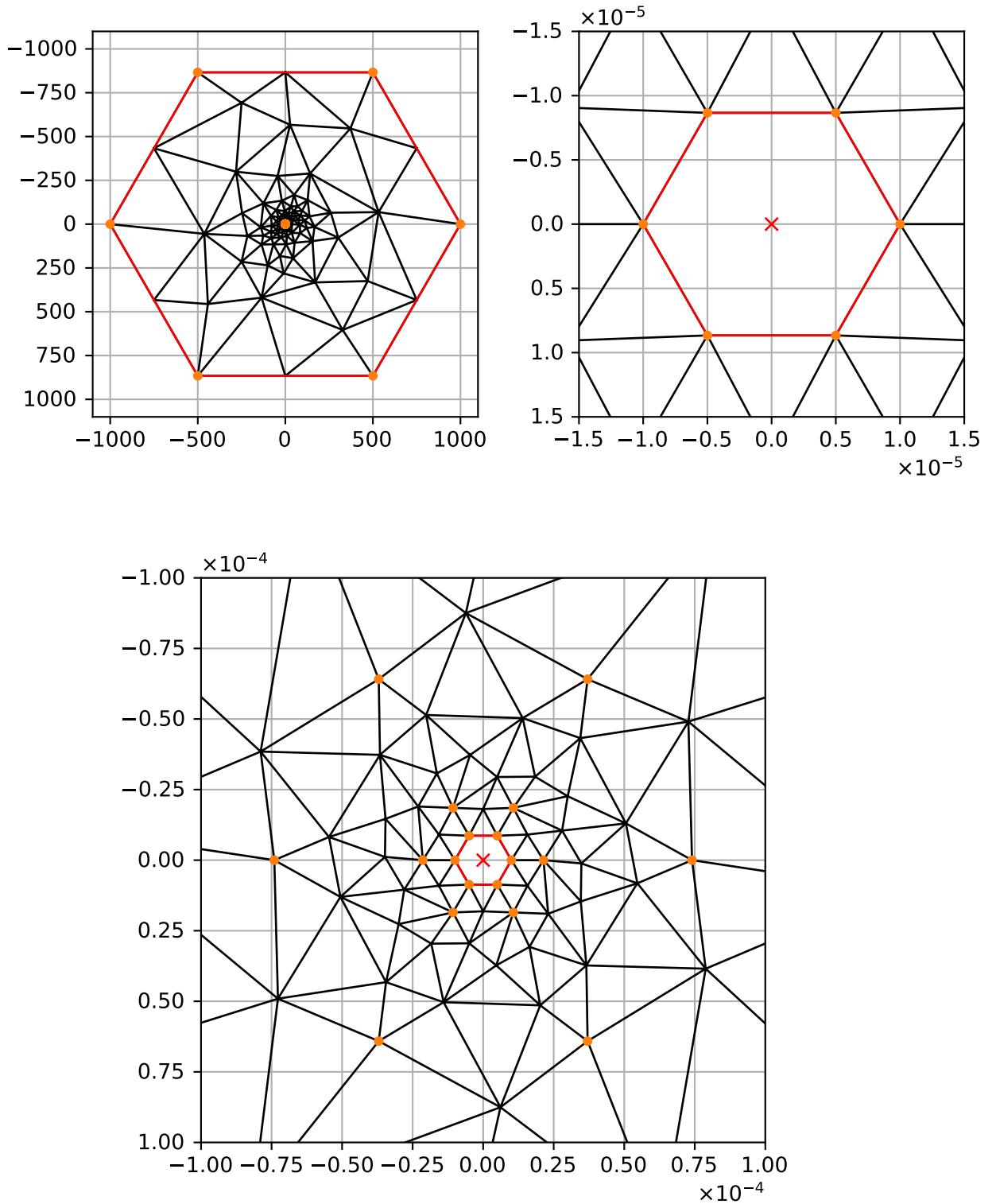
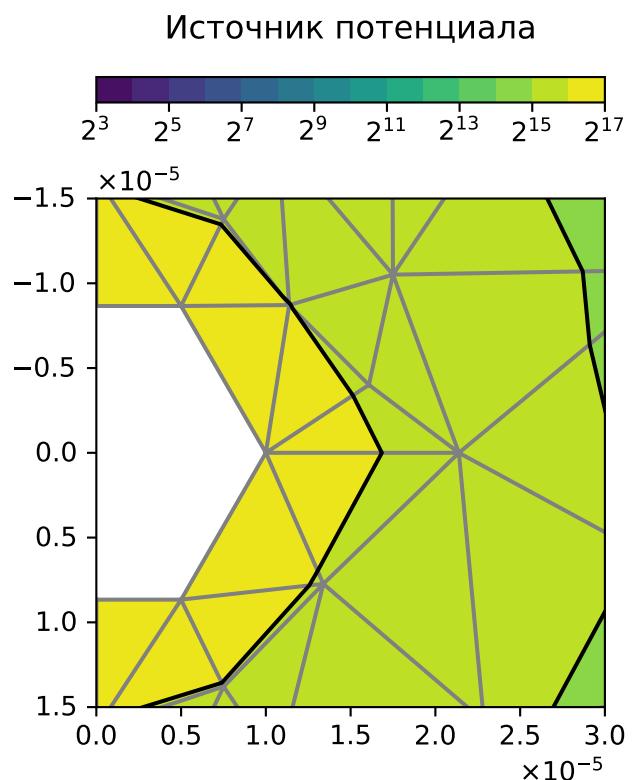
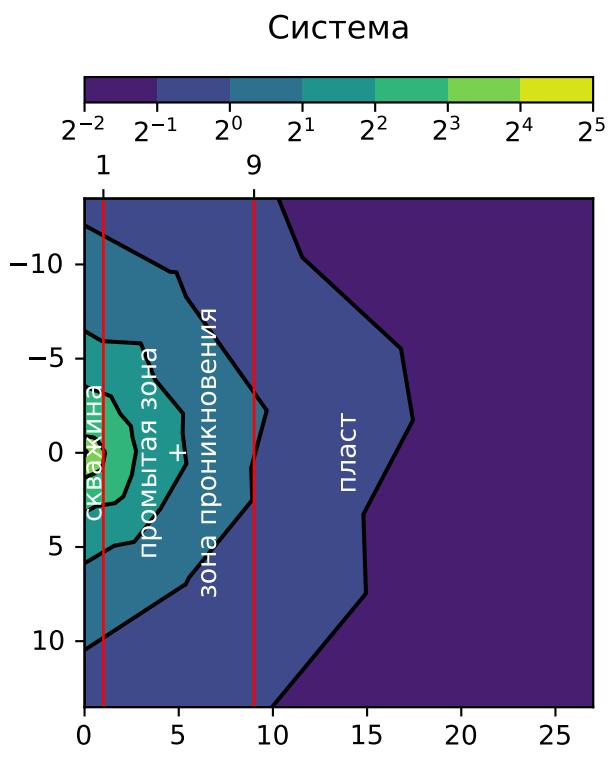
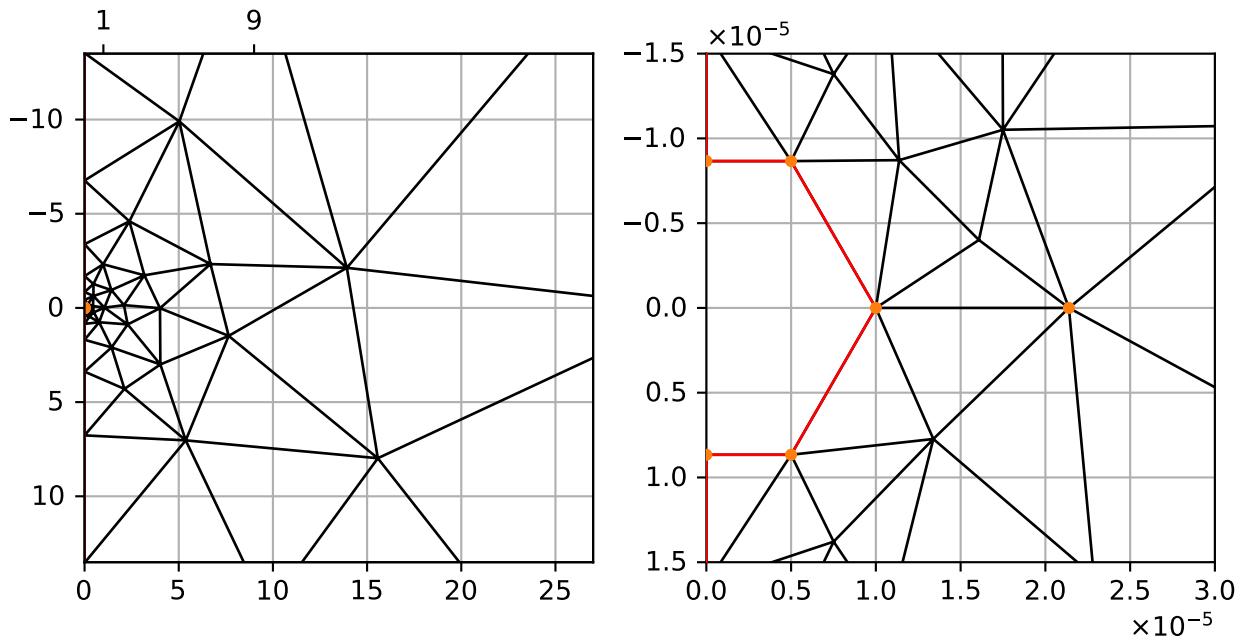
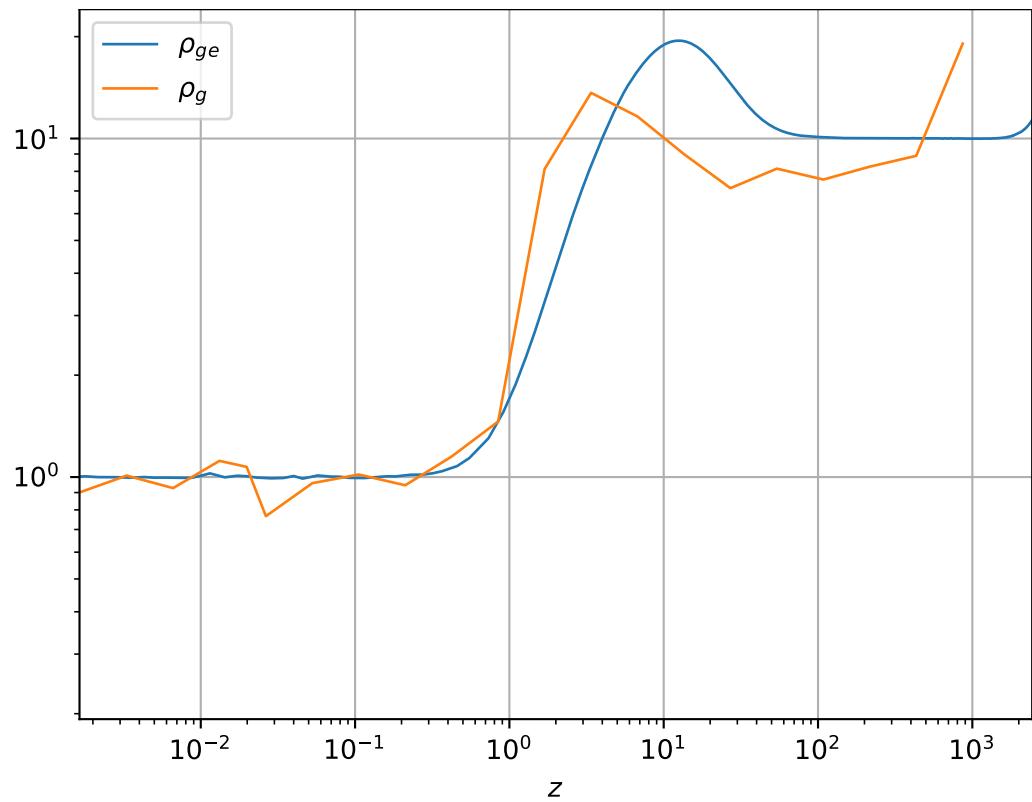


Рис. 3.1

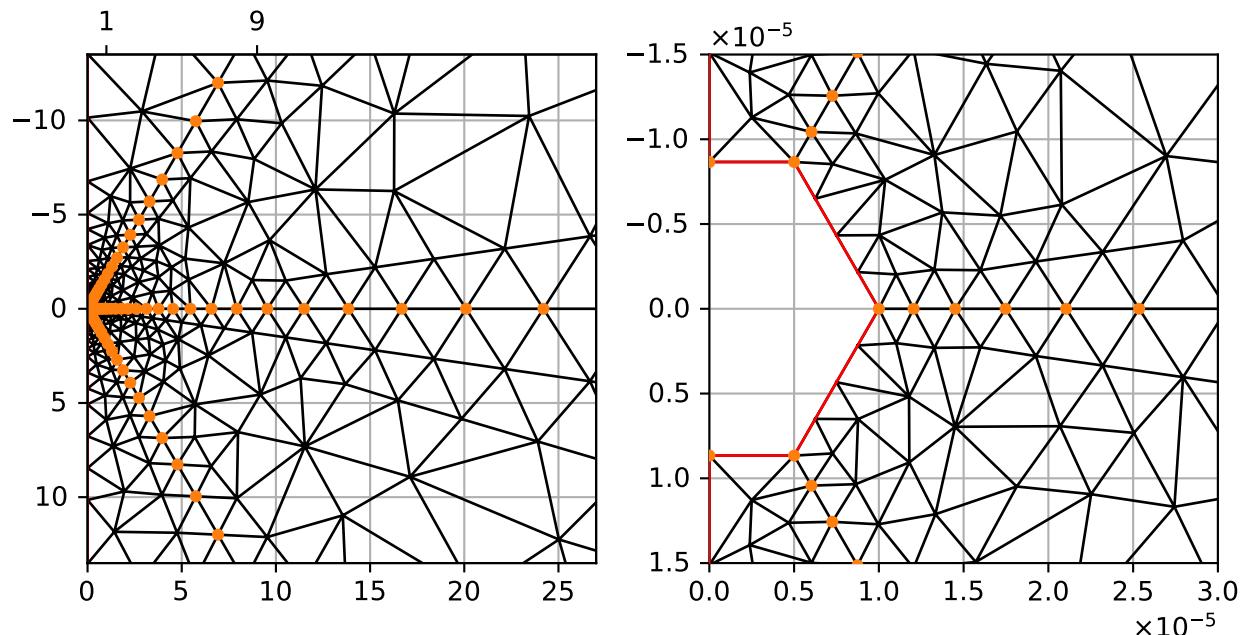
2. Для расчетной сетки выбраны узлы, расположенные справа от оси $r = 0$, и построены сегменты половины внутреннего и внешнего границ. Использованы лагранжевые элементы \mathcal{P}_1 .

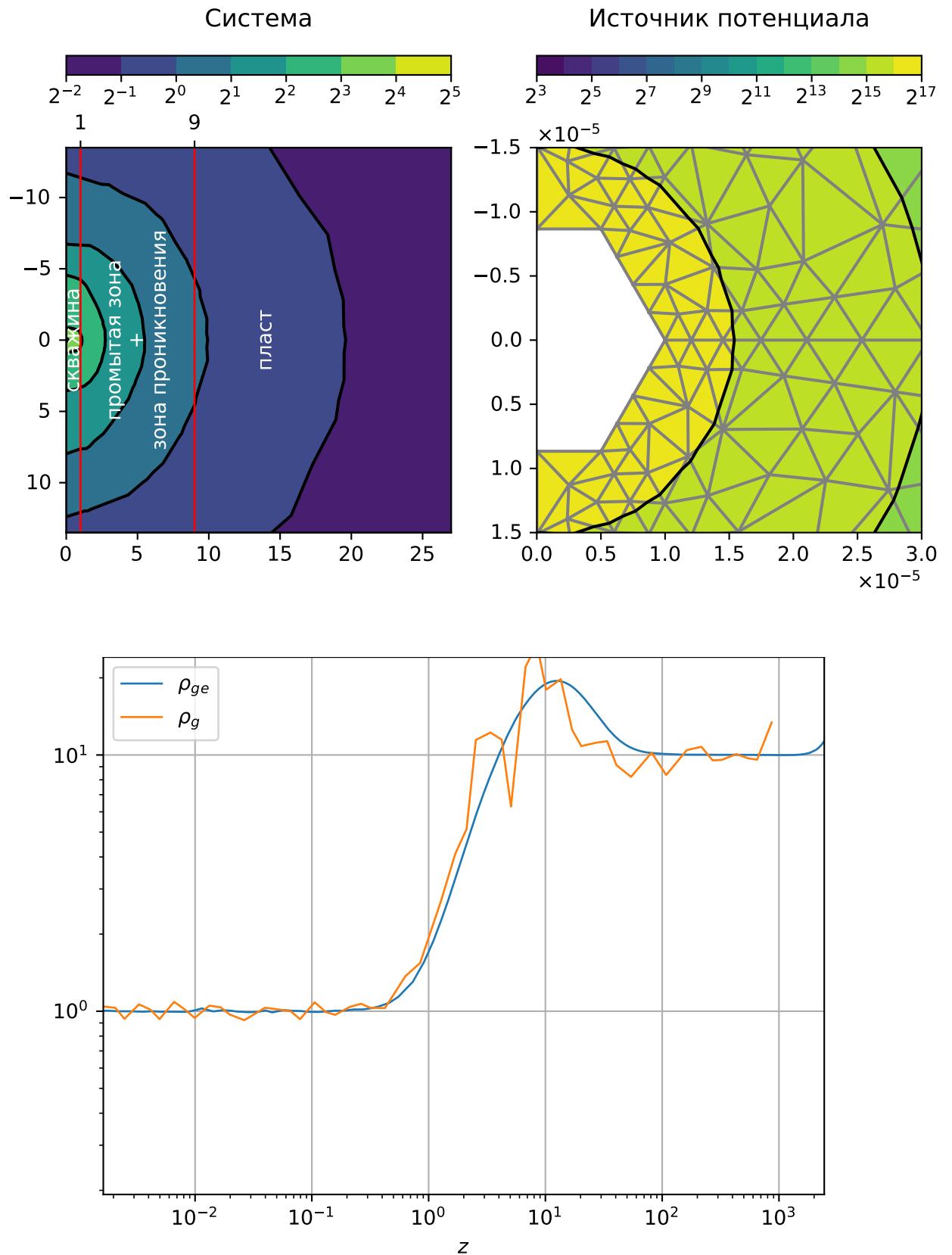
221 узлов.



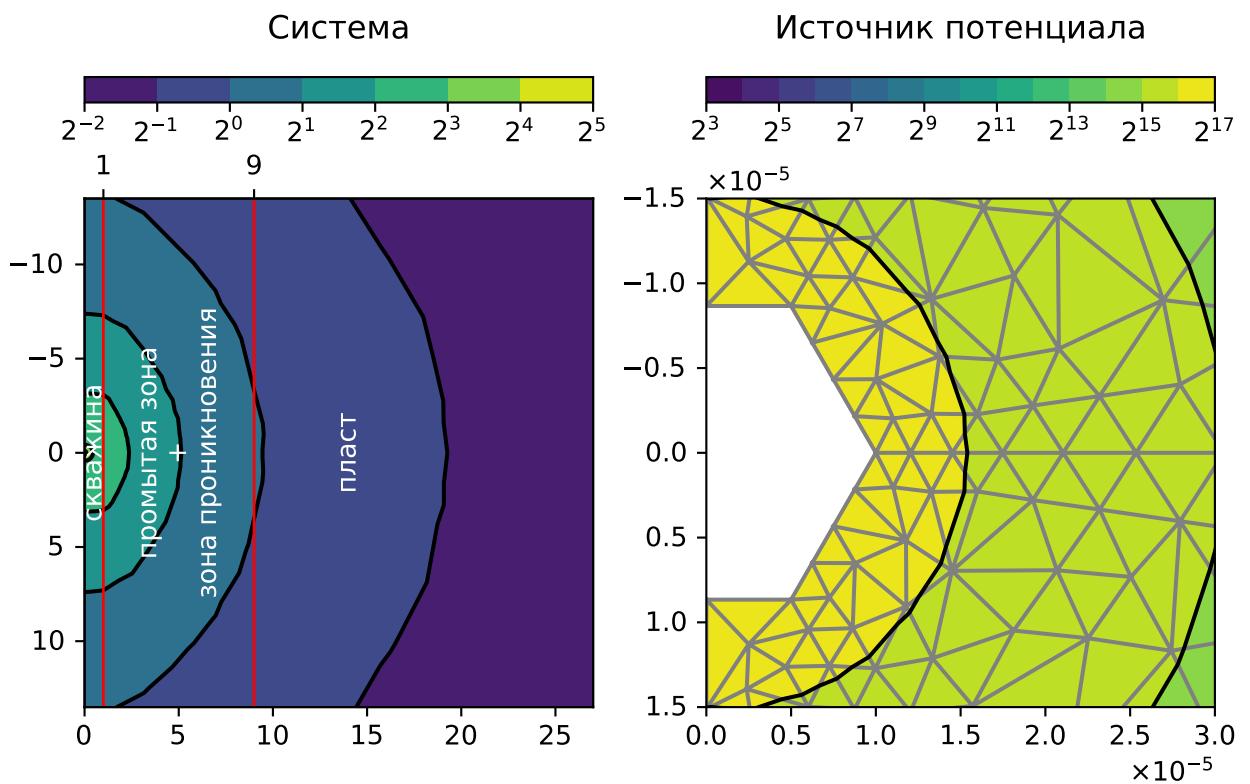
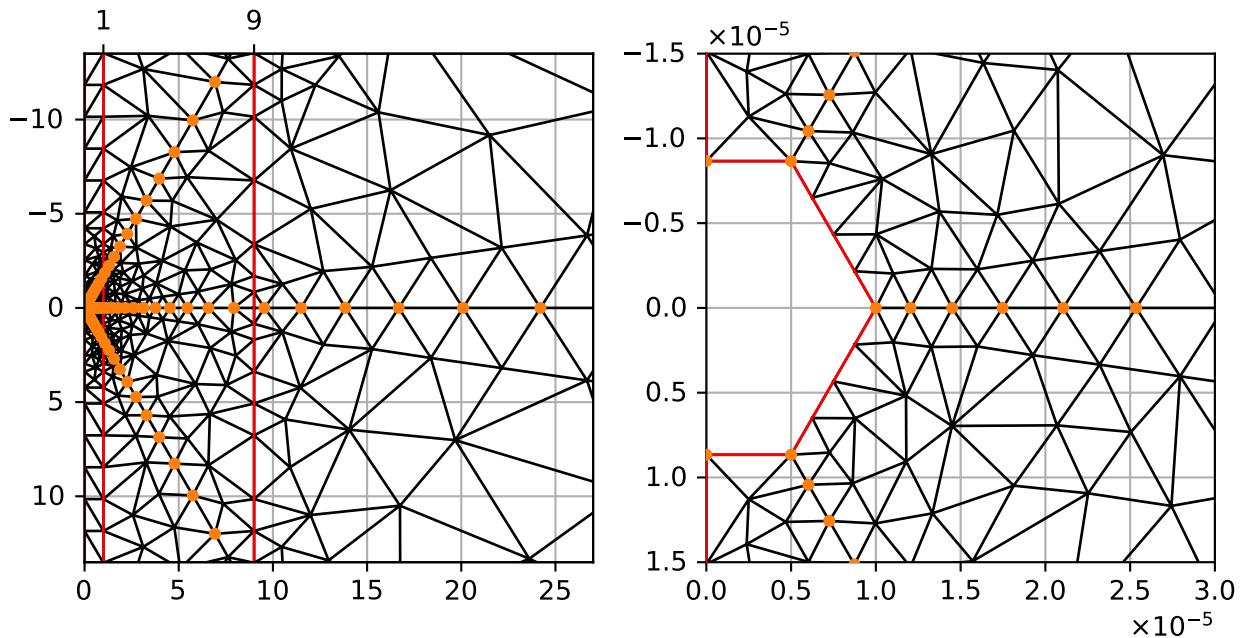


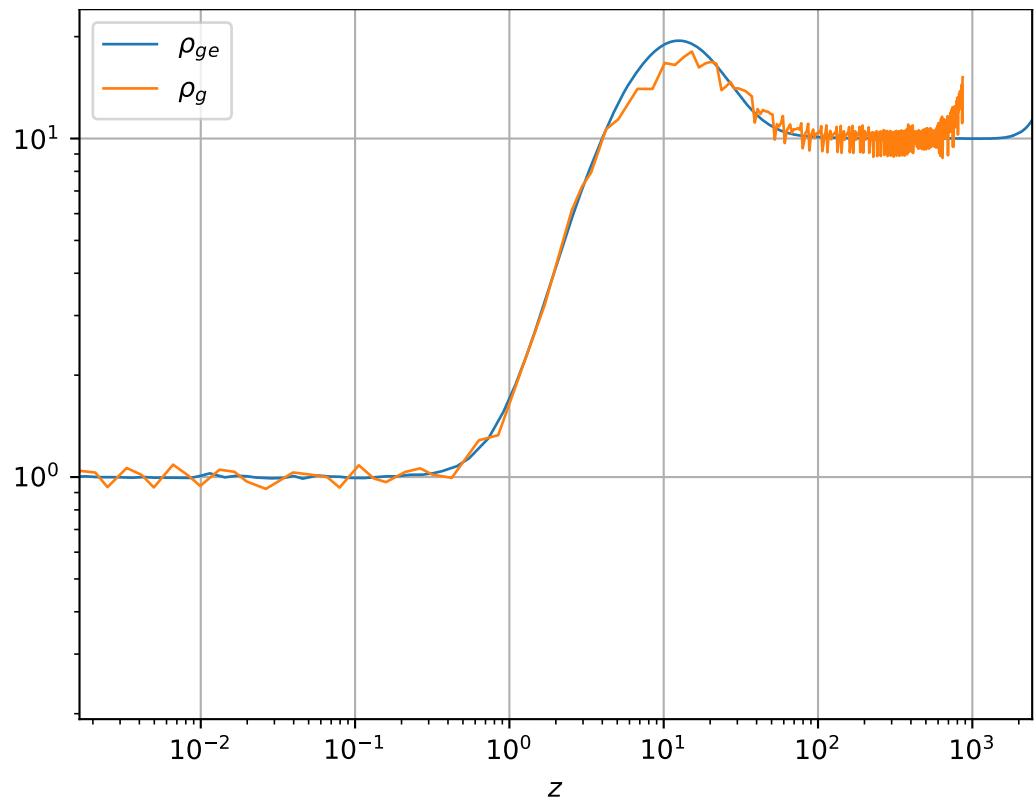
3. Использованы узлы, равноотстоящие в логарифмическом масштабе.
1510 узлов.





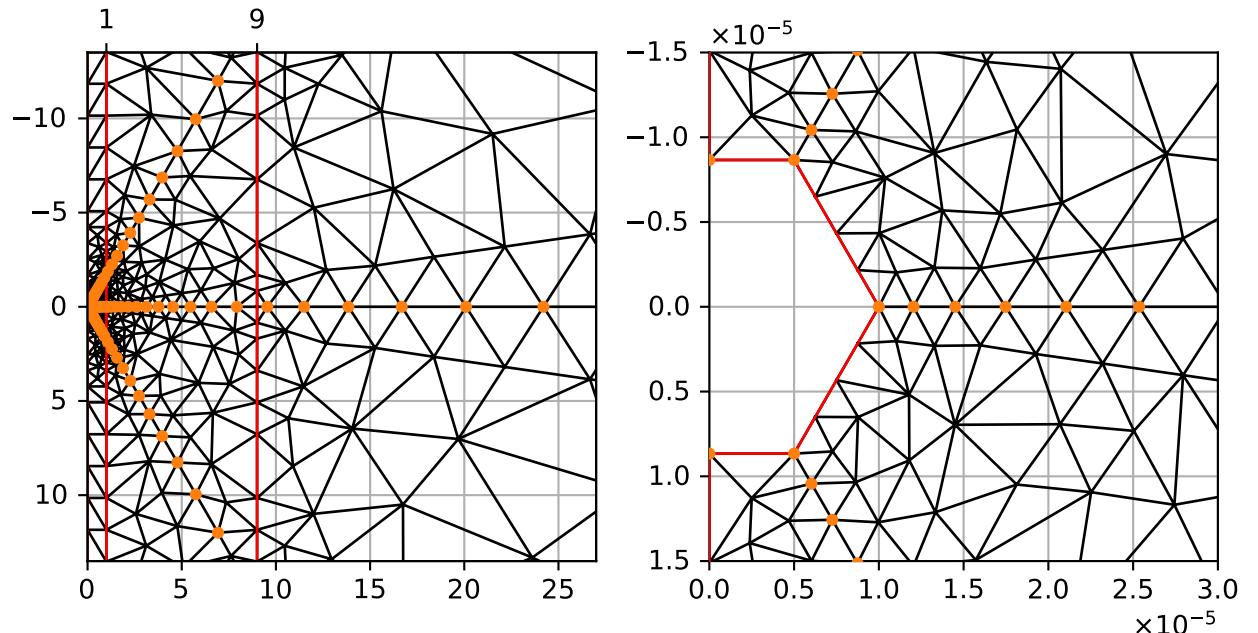
4. Добавлены сегменты на местах разрыва коэффициента УЭС.
7201 узлов.

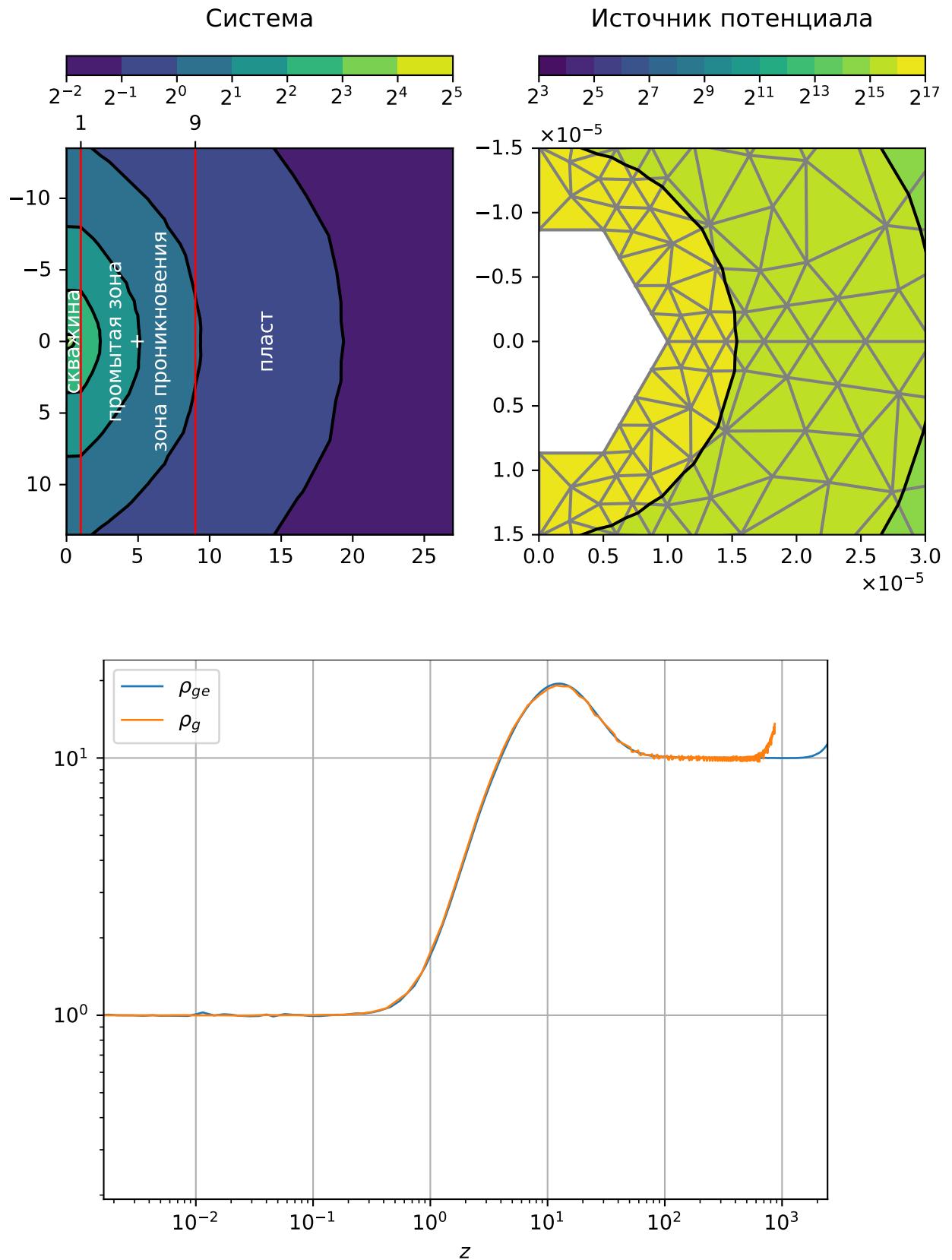




5. Использованы лагранжевые элементы \mathcal{P}_3 .

7201 узлов.

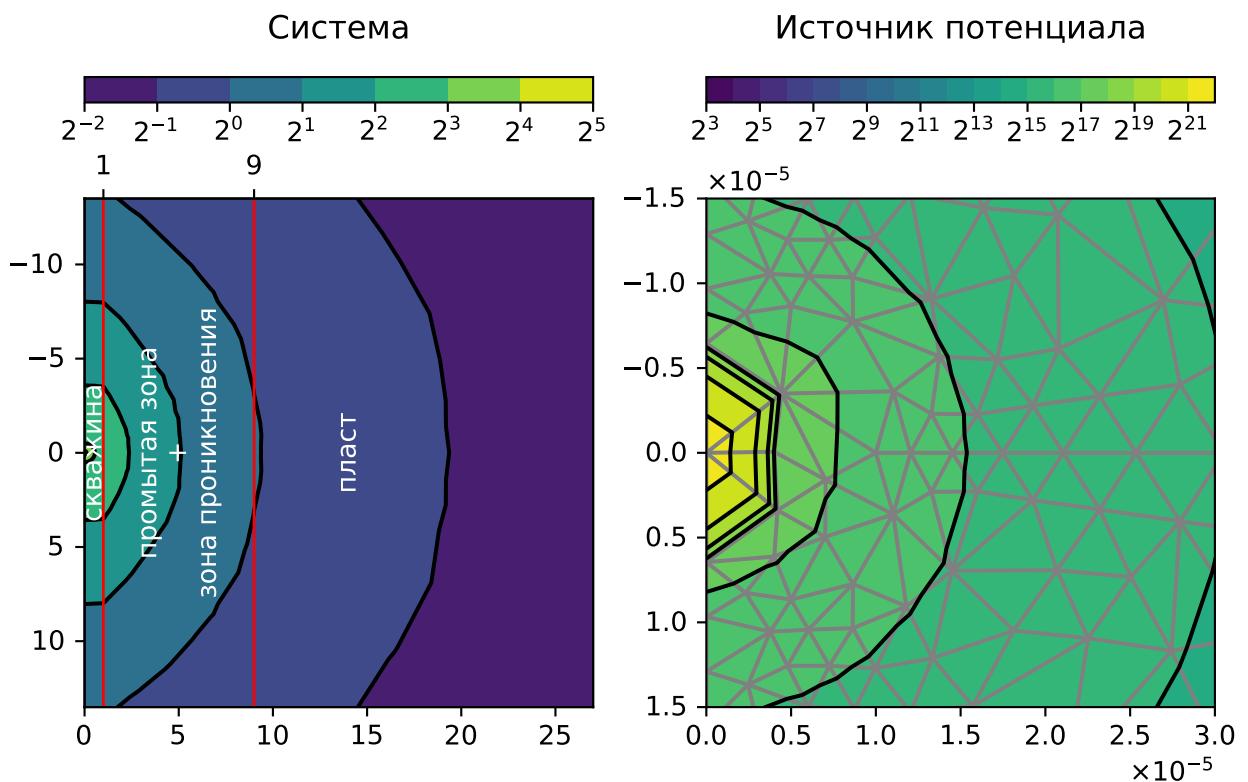
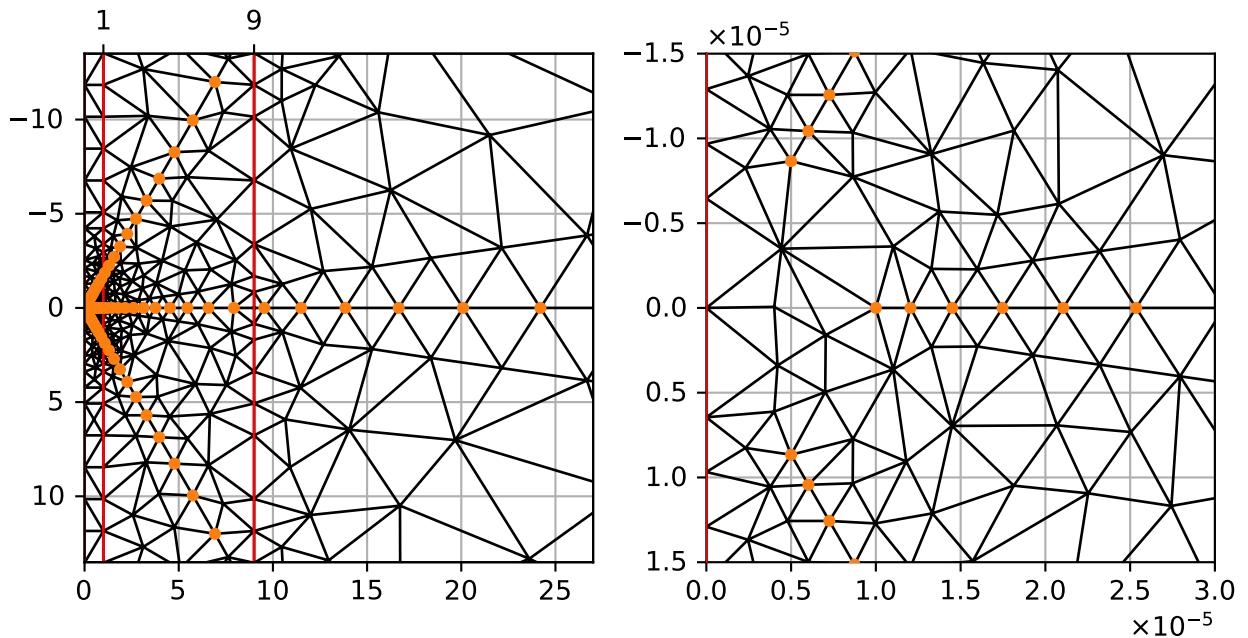


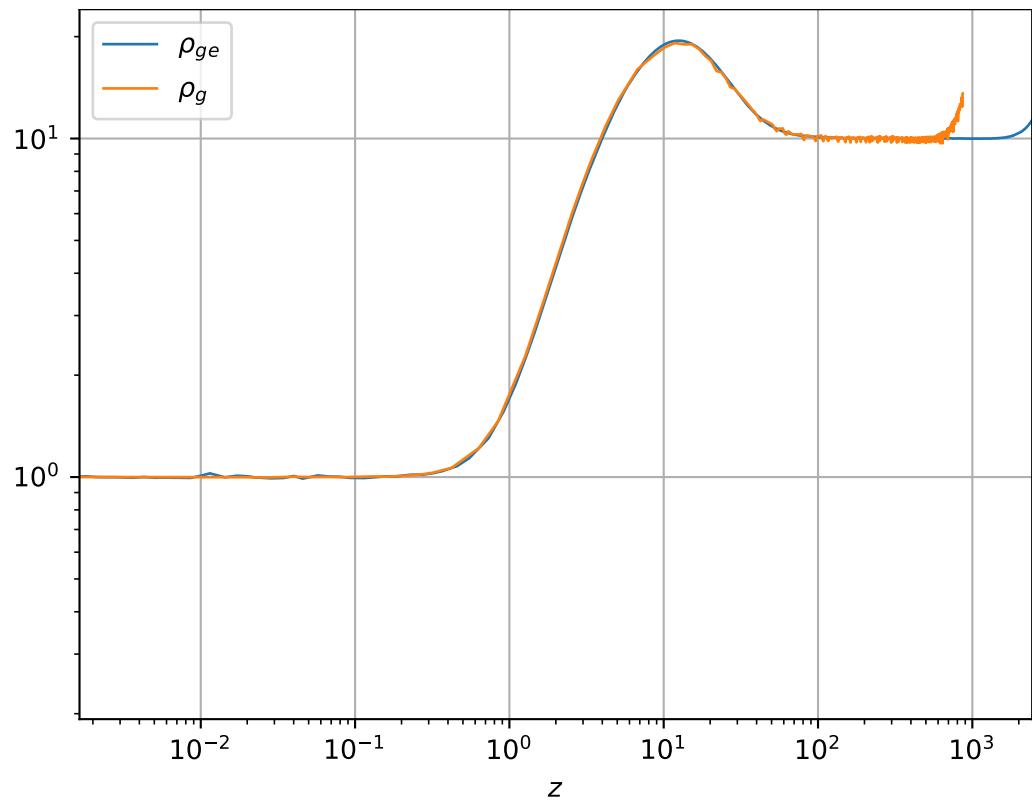


6. Использован точечный источник вместо внутреннего граничного усло-

вия.

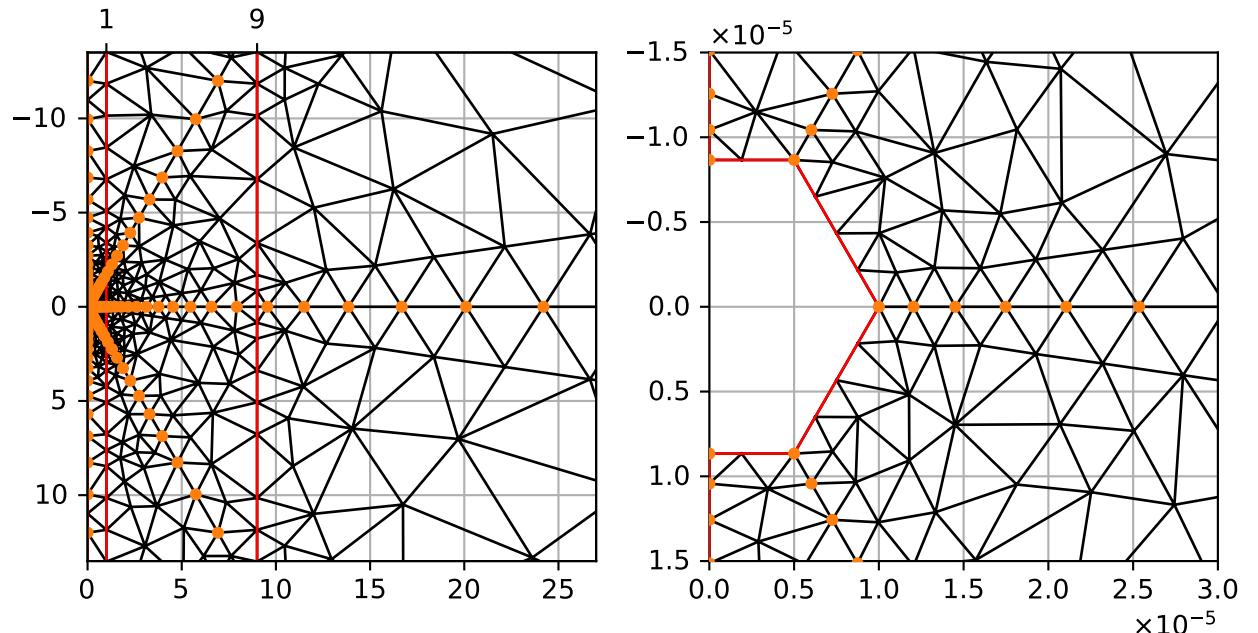
6659 узлов.

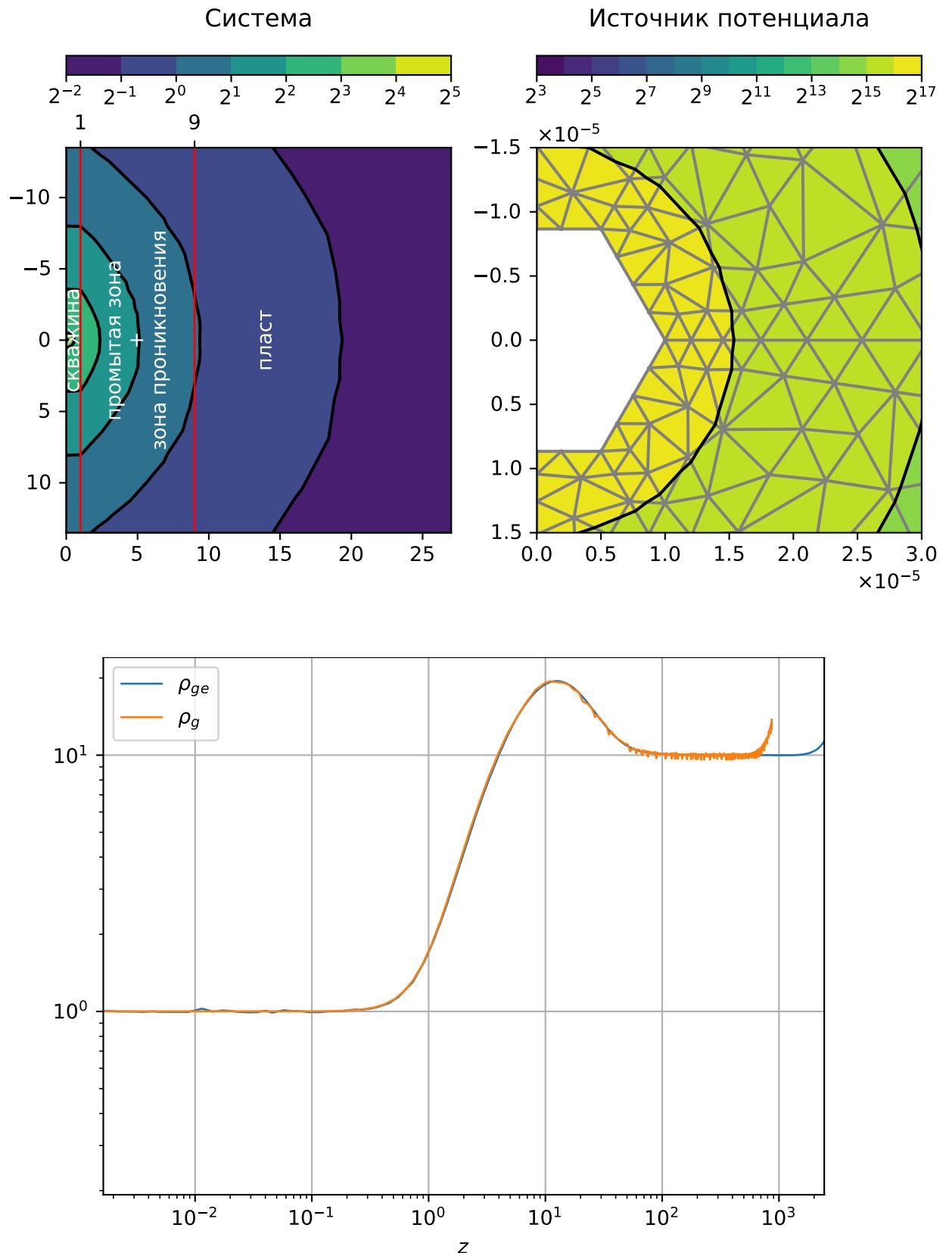




7. Добавлены на границе $r = 0$ узлы одномерной сетки.

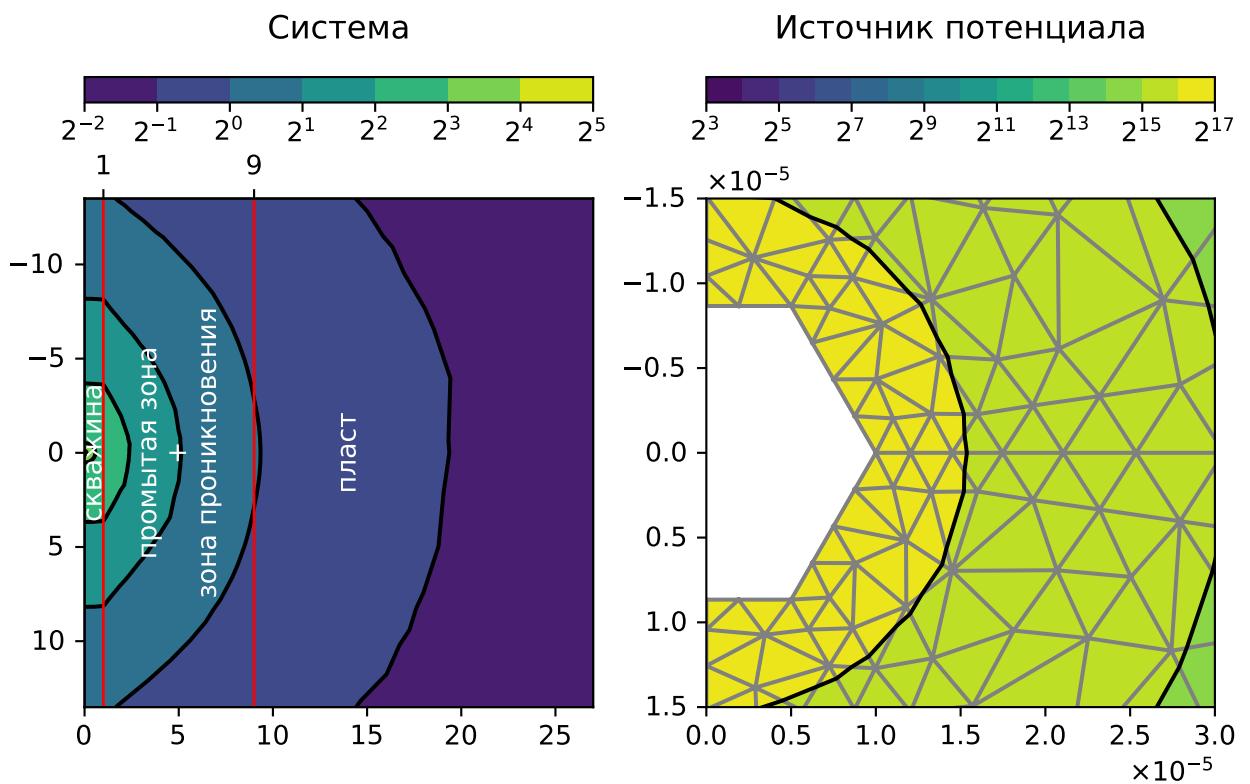
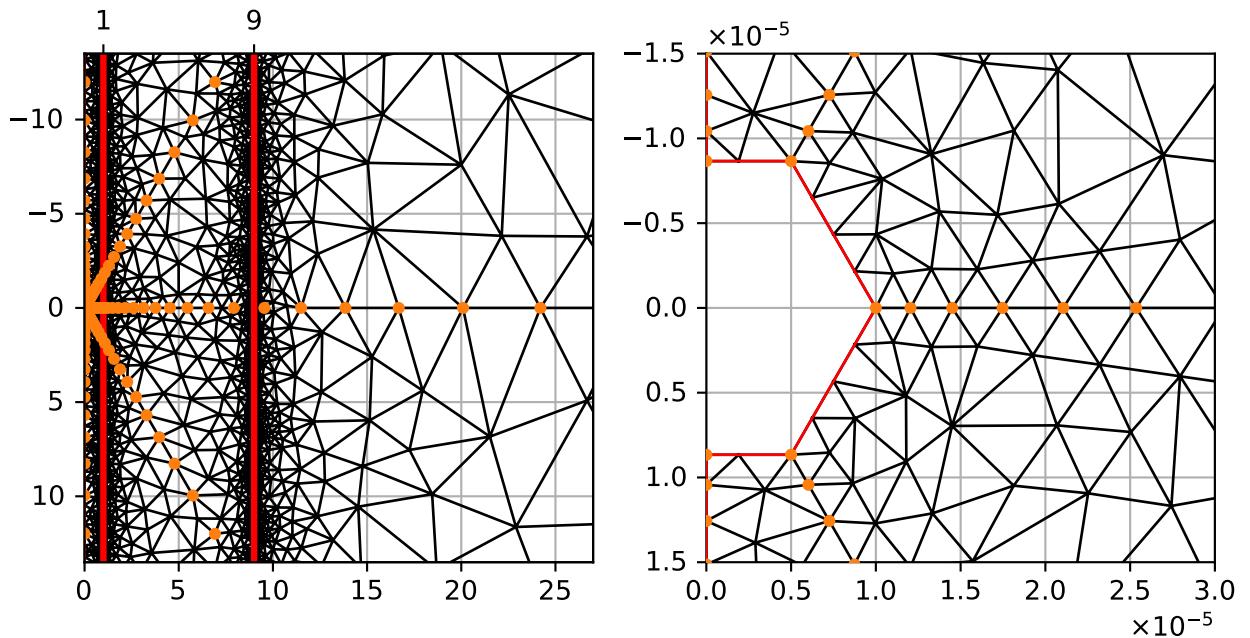
6845 узлов.

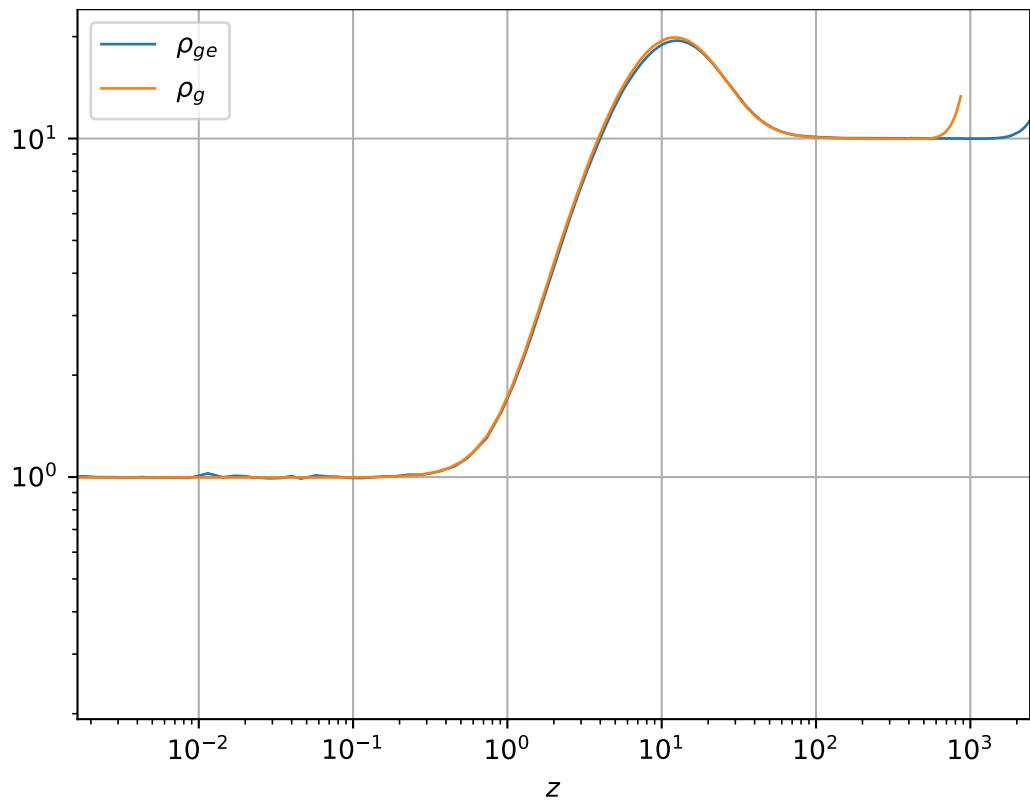




8. Добавлены сегменты возле мест разрыва коэффициента УЭС. Использованы лагранжевые элементы \mathcal{P}_2 .

156856 узлов.





9. Использованы узлы, равноотстоящие в логарифмическом масштабе; узлы лежащие в местах пересечений границ областей УЭС с концентрическими окружностями и лучами, отрезки графа на местах разрыва коэффициента УЭС; лагранжевые элементы \mathcal{P}_2 ; настройка 'p' библиотеки triangle.

2859 узлов.

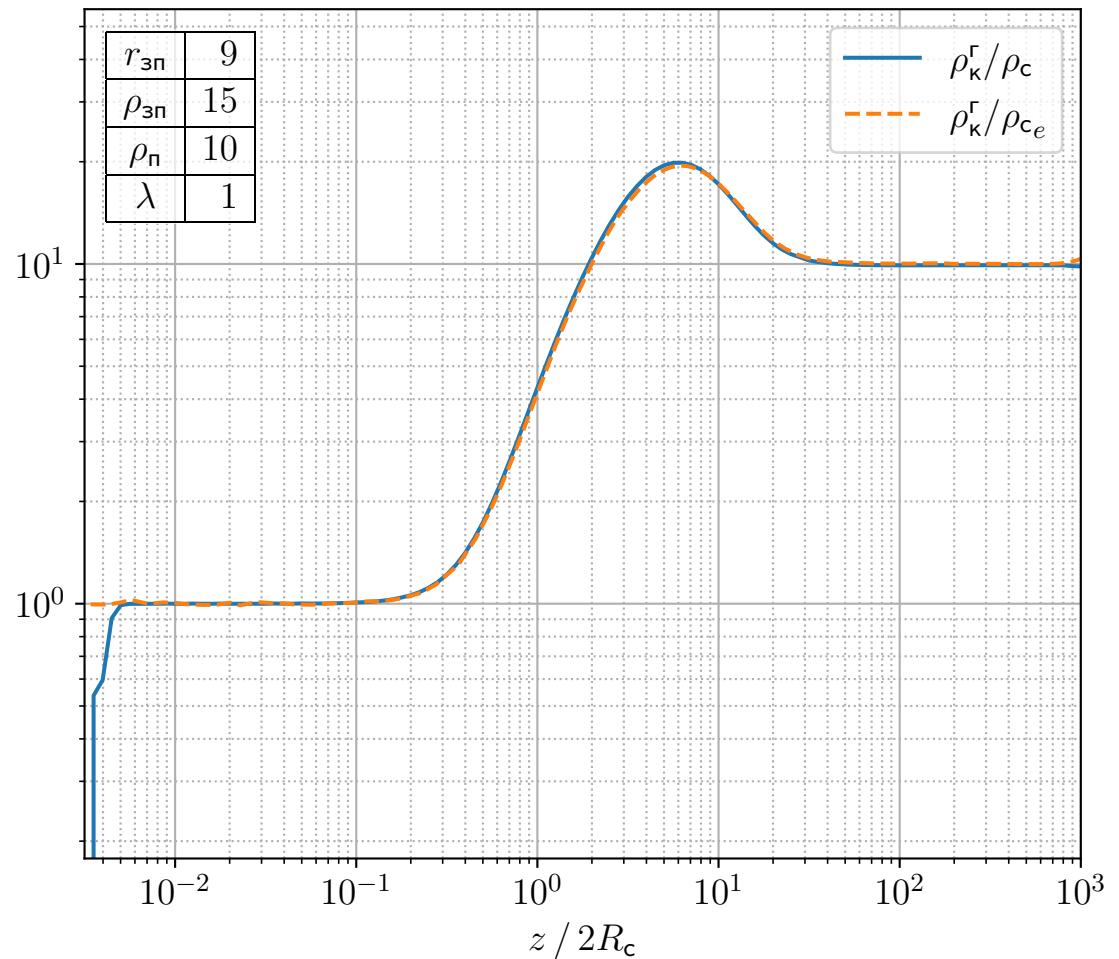


Рис. 3.2

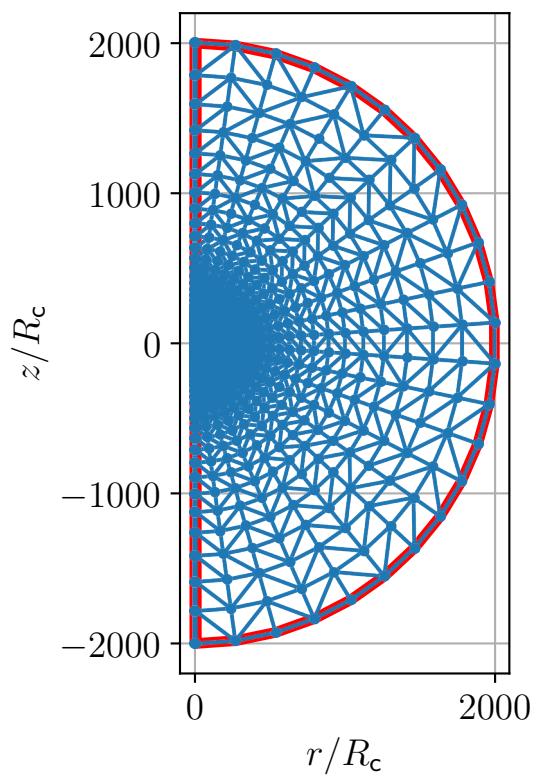


Рис. 3.3

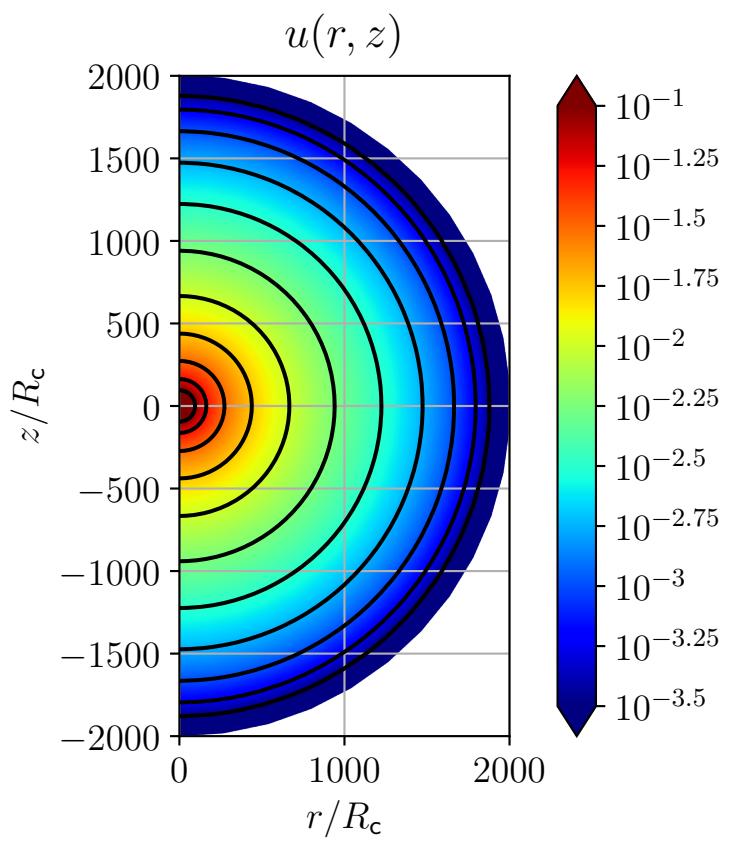


Рис. 3.4

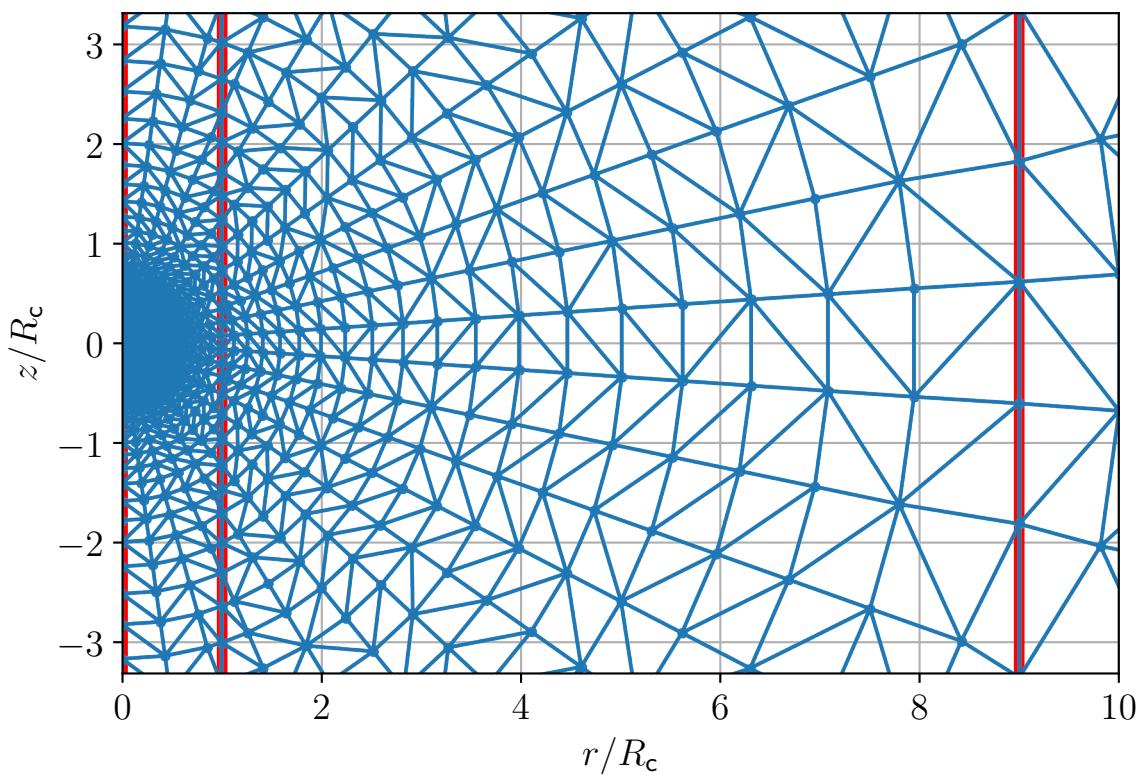


Рис. 3.5

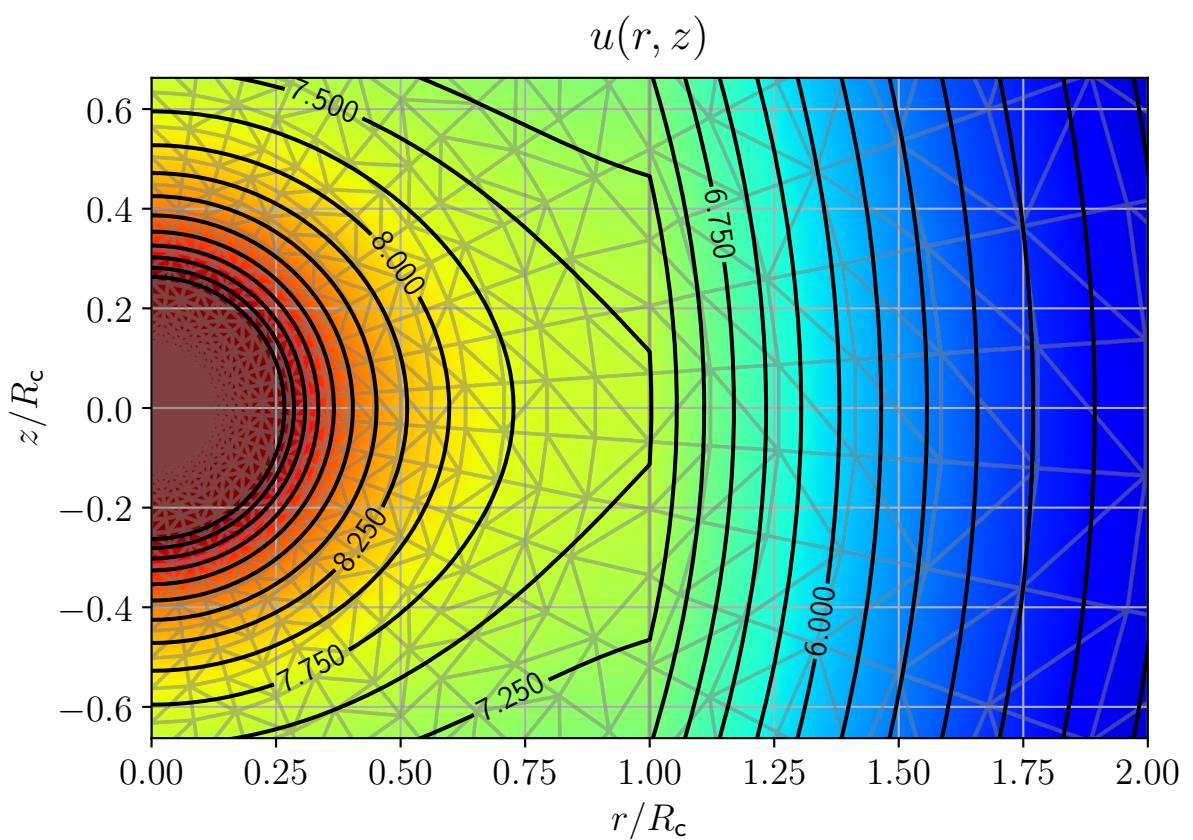


Рис. 3.6

3.4 Результаты

Лучший результат получен в эксперименте 9 за $538 \text{ мс} \pm 37.8 \text{ мс}$, 7 проходов (среднее значение \pm среднеквадратичное отклонение от 7 проходов, в каждом проходе 1 цикл). Использованы узлы, равноотстоящие в логарифмическом масштабе; узлы лежащие в местах пересечений границ областей УЭС с концентрическими окружностями и лучами, отрезки графа на местах разрыва коэффициента УЭС; лагранжевые элементы \mathcal{P}_2 ; настройка 'r' библиотеки triangle. Использован компьютер для вычисления: операционная система Ubuntu 18.04 LTS, процессор Intel Pentium 4415U 2.30 ГГц с 4 логическими процессорами (1 физический процессор \times 2 ядра в физическом процессоре \times 2 потока в каждом ядре)..

В результате проведённых численных экспериментов разработан метод построения оптимальных сеток для решения задачи БКЗ на сетке, позволяющий получить приемлемое численное решение с меньшими затратами ресурсов. В данном случае мы следили за двумя параметрами: гладкостью решения и сравнением "на глаз" с известной палеткой. Поскольку принятая методика решения обратной задачи БКЗ основано на использованием палеток на "сравнении на глаз", то такой критерий годится. Относительная ошибка примерно $< 10\%$.

4 РЕШЕНИЕ

4.1 Модель 1

Трёхслойная осесимметричная модель:

первый слой — скважина ($0 < r < 1$) с УЭС $\rho_c = 1$,

второй слой — зона проникновения ($1 < r < r_{зп} = 4$) с УЭС $\rho_{зп} = 4$,

третий слой — пласт ($r > r_{зп}$) с УЭС $\rho_{п}$.

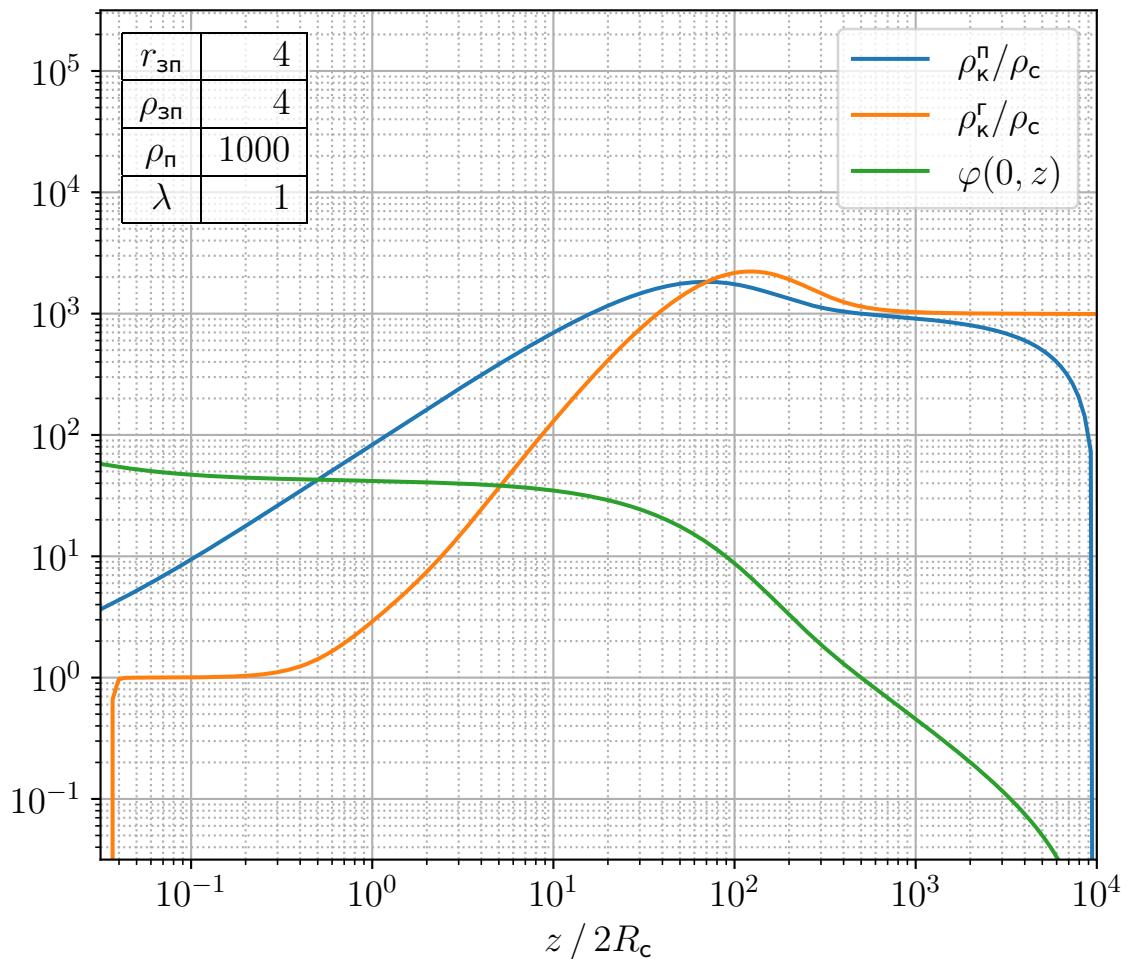


Рис. 4.1

Время вычисления решения (рис. 4.1) 1.24 с \pm 38.3 мс, 7 проходов (среднее значение \pm среднеквадратичное отклонение от 7 проходов, в каждом проходе 1 цикл).

6409 узлов расчетной сетки.

Далее изображены поле, расчетная сетка и решение для палетки с этой расчетной сеткой.

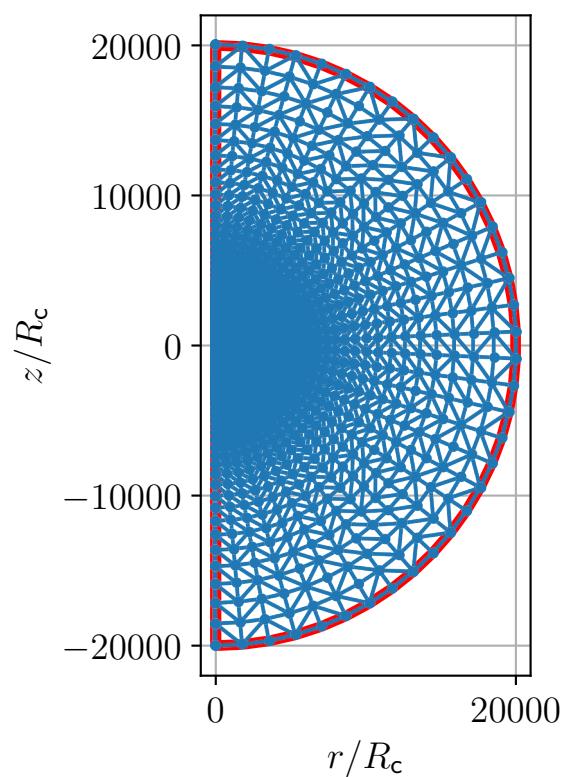


Рис. 4.2

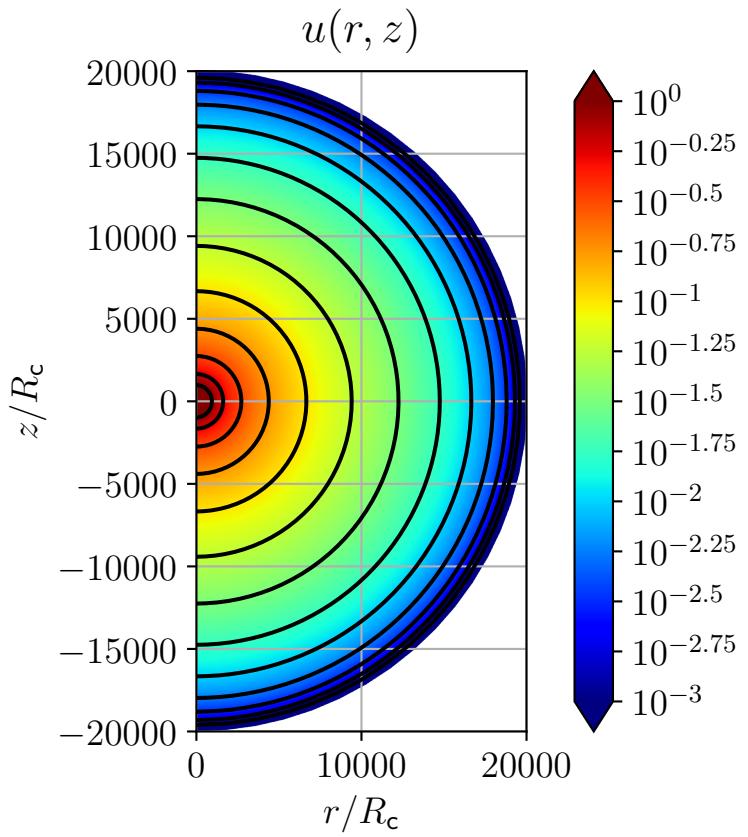


Рис. 4.3

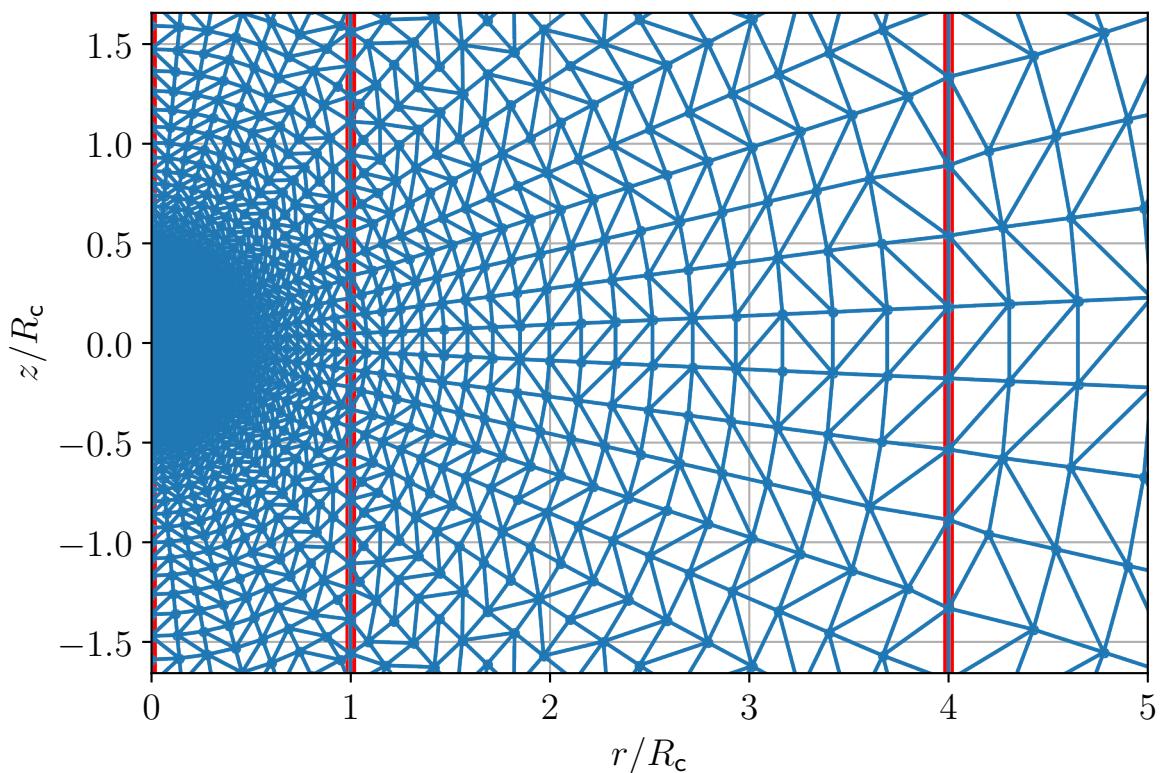


Рис. 4.4

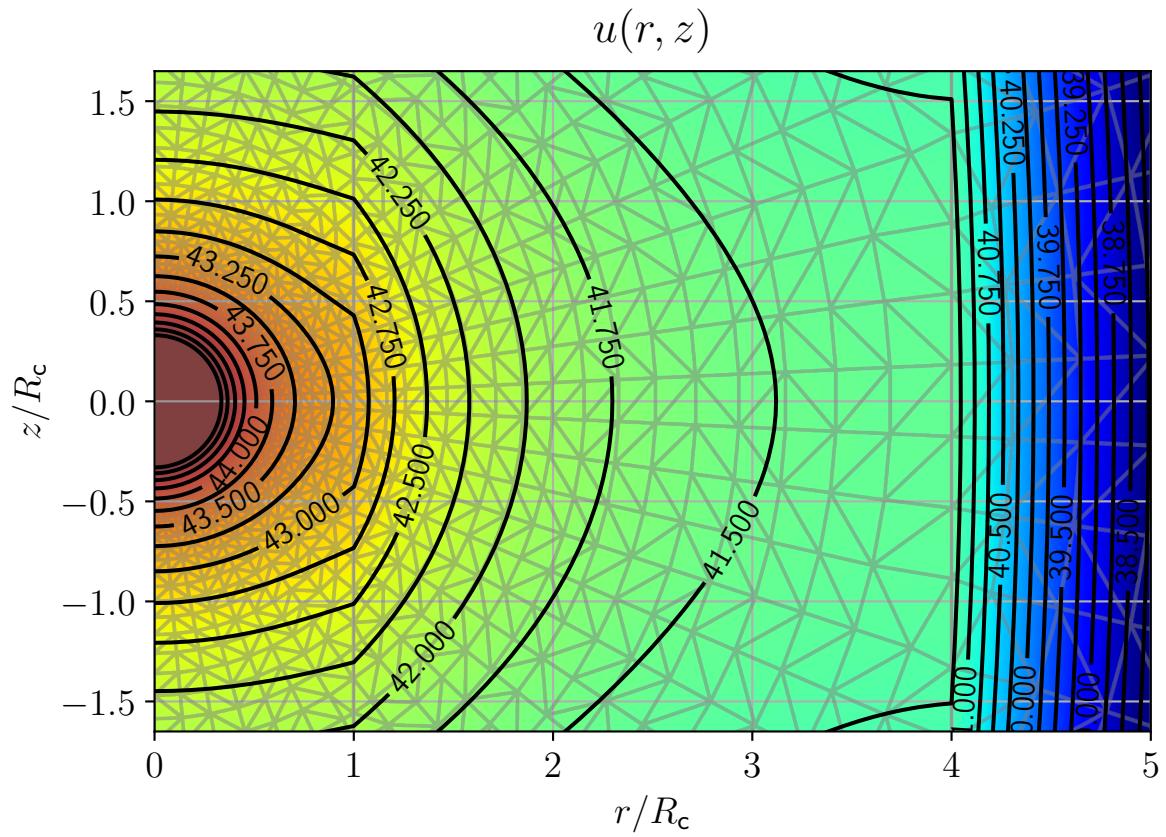


Рис. 4.5

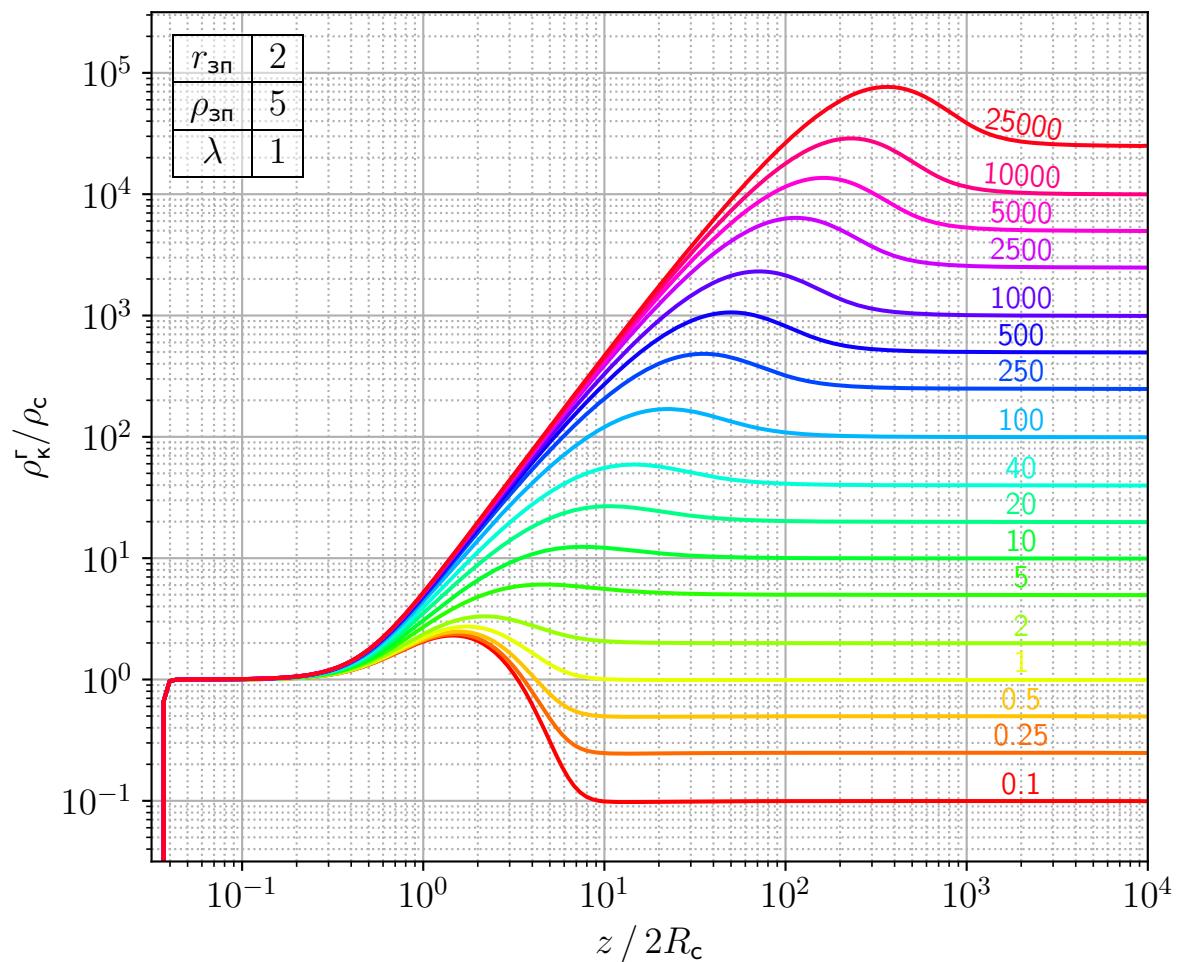


Рис. 4.6

Время вычисления палетки 18 с ± 159 мс, 7 проходов.

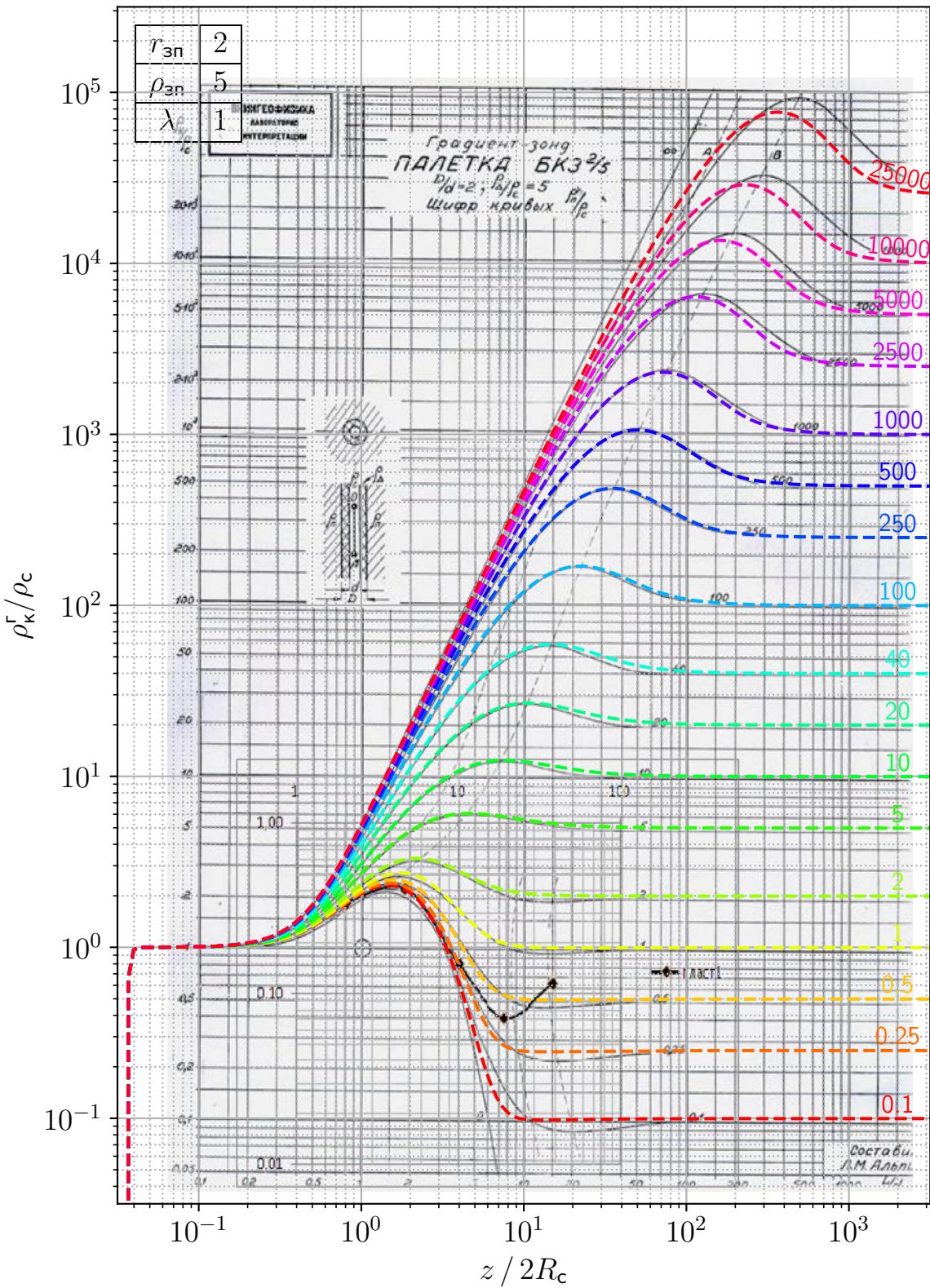


Рис. 4.7

4.2 Модель 2

Четырехслойная осесимметричная модель с анизотропией:
первый слой — скважина ($0 < r < 1$) с УЭС $\rho_c = 1$ с коэффициентом анизотропии $\lambda_c = 1$,
второй слой — промытая зона ($1 < r < r_{пз} = 2.2$) с УЭС $\rho_{пз} = 4$ с коэффициентом анизотропии $\lambda_{пз} = \lambda = 1.1$,
третий слой — зона проникновения ($r_{пз} < r < r_{3п} = 5.8$) с УЭС линейная функция от r со значениями $\rho_{3п}(r_{пз}) = \rho_{пз}$ и $\rho_{3п}(r_{3п}) = \rho_{п}$ с коэффициентом анизотропии $\lambda_{пз} = \lambda = 1.1$,
четвертый слой — пласт ($r > r_{3п}$) с УЭС $\rho_{п}$ с коэффициентом анизотропии $\lambda_{п} = \lambda = 1.1$.

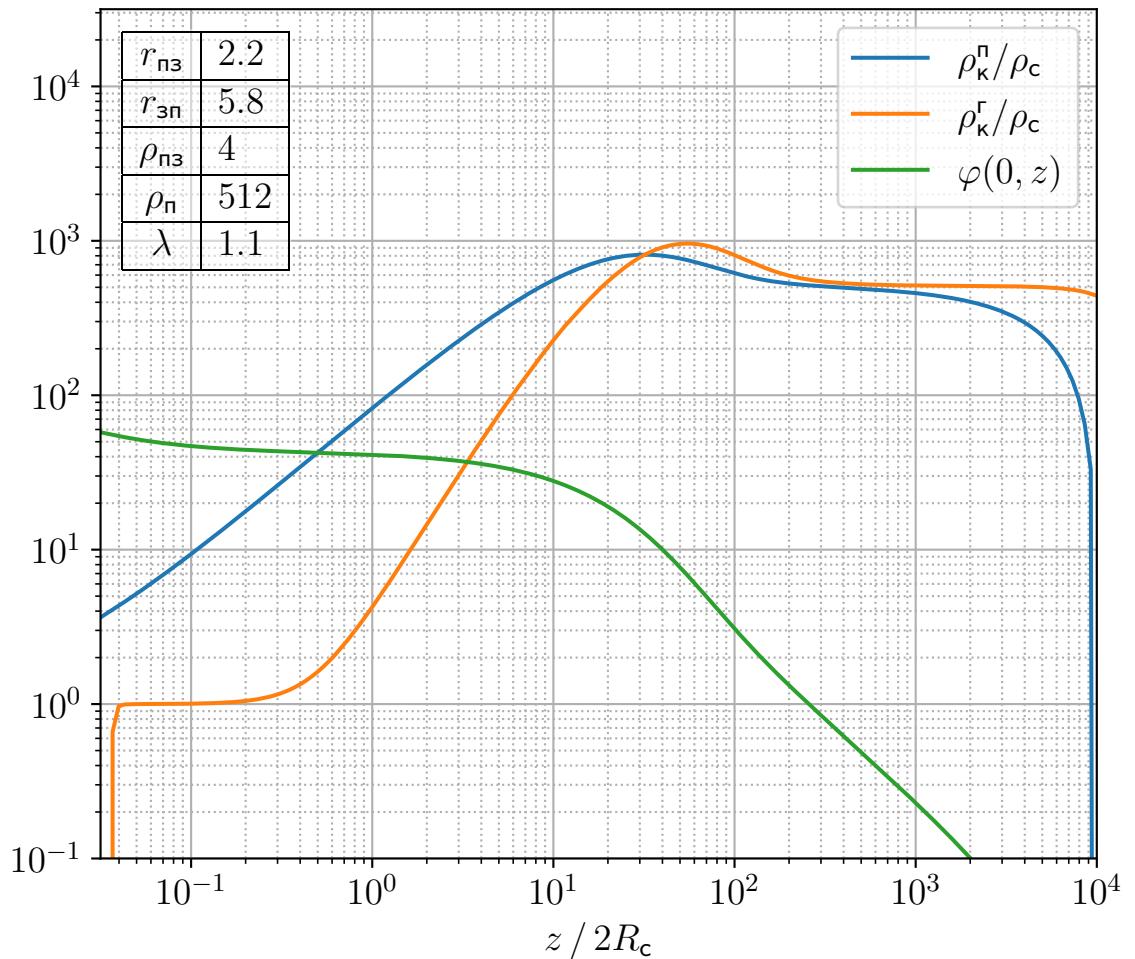


Рис. 4.8

Время вычисления решения 1.23 с ± 16.3 мс, 7 проходов.
6613 узлов расчетной сетки.
Далее изображены поле, расчетная сетка и решение для палетки с этой
расчетной сеткой.

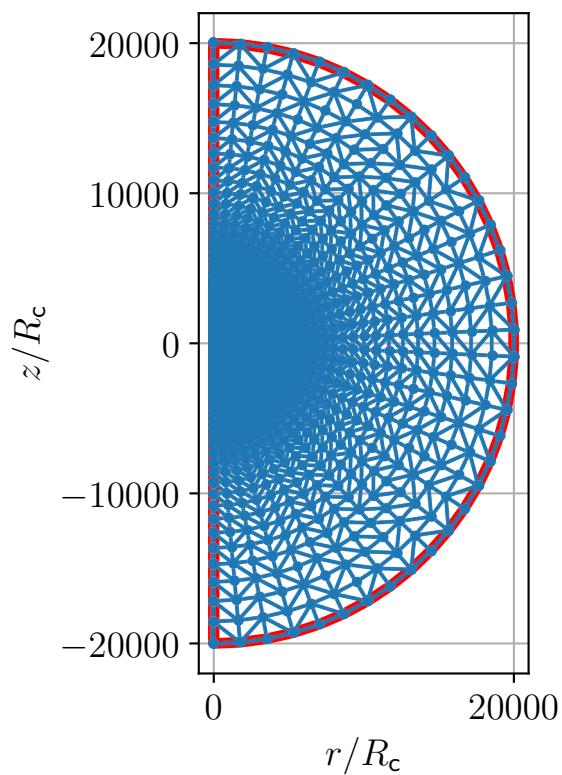


Рис. 4.9

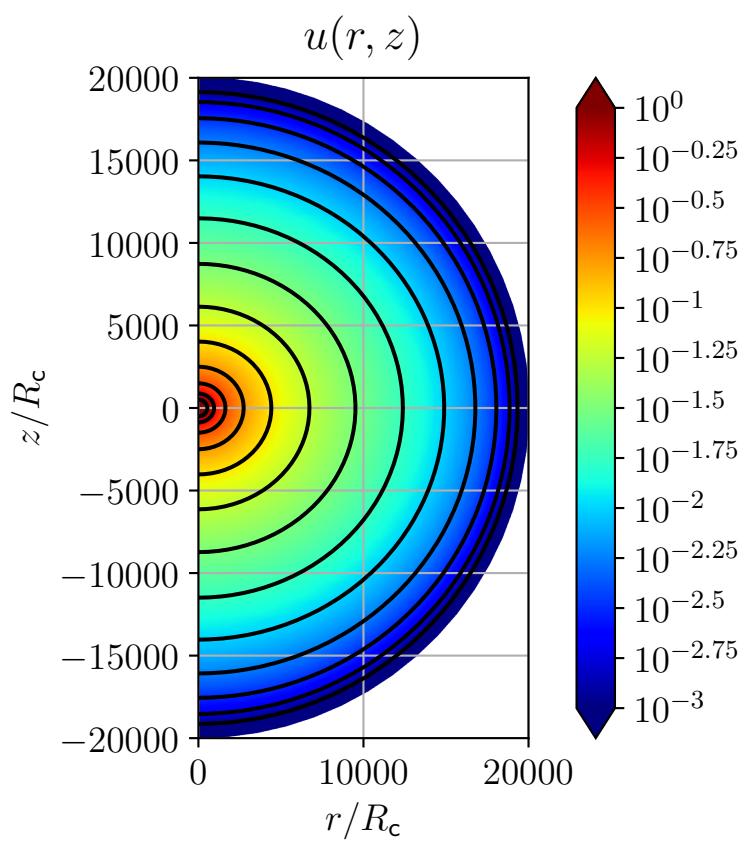


Рис. 4.10

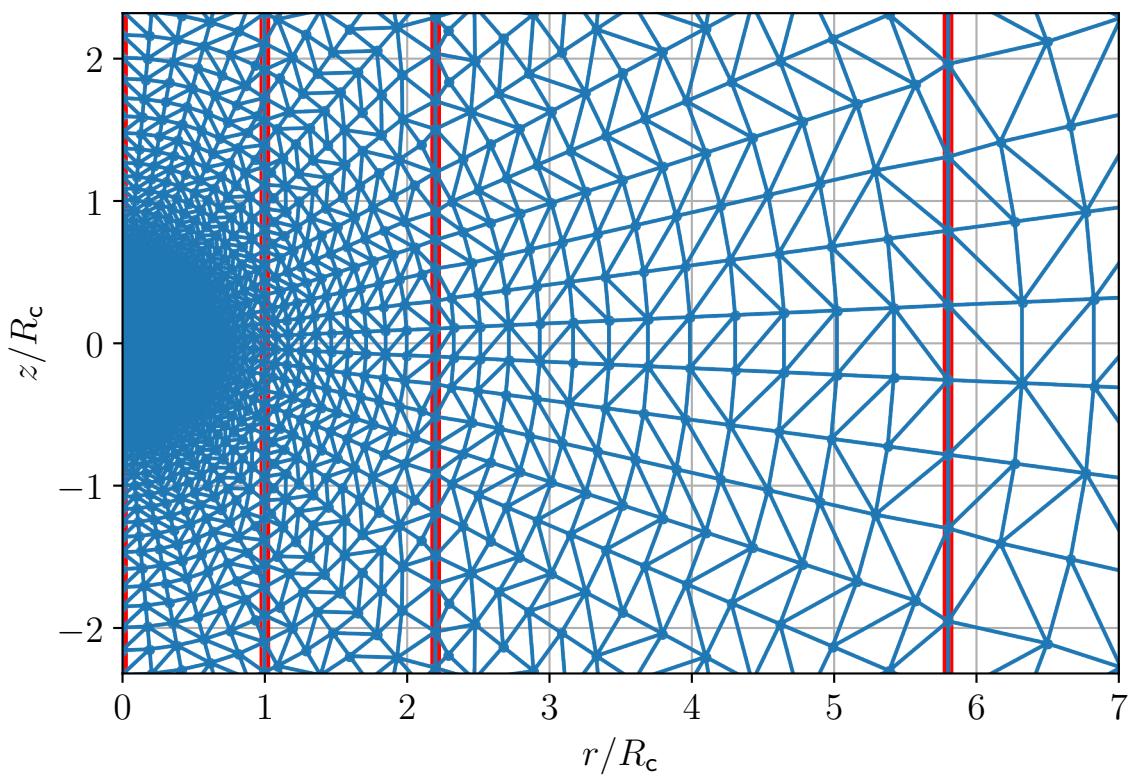


Рис. 4.11

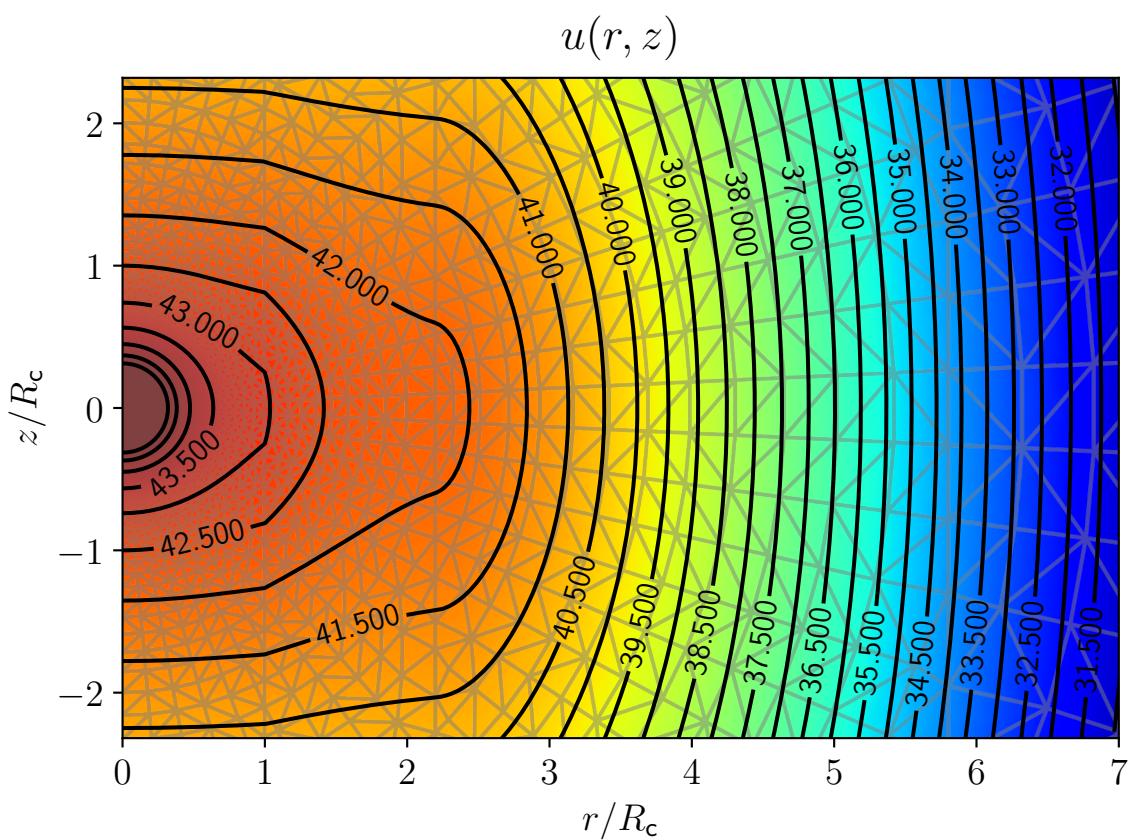


Рис. 4.12

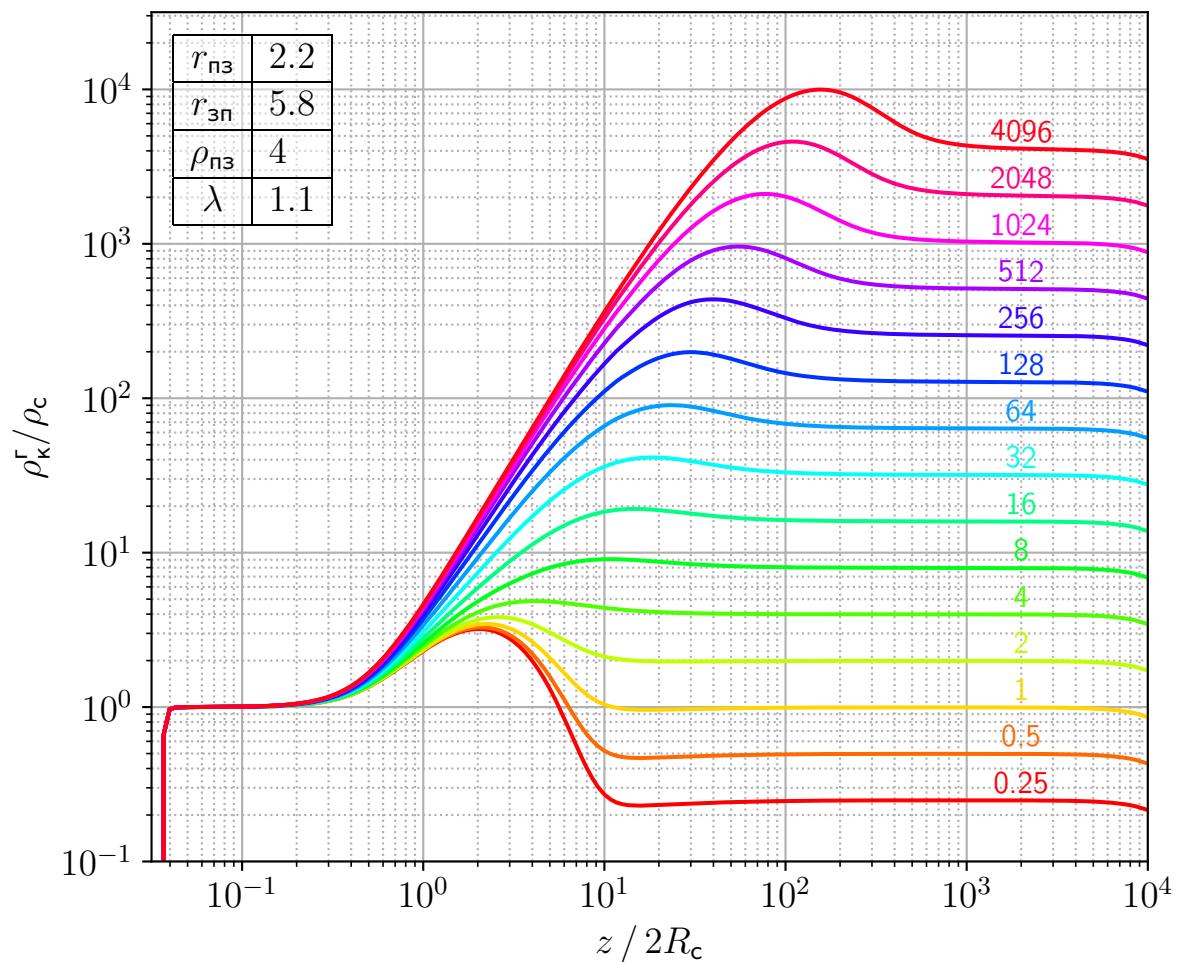


Рис. 4.13

Время вычисления палетки 16.5 с \pm 176 мс, 7 проходов.

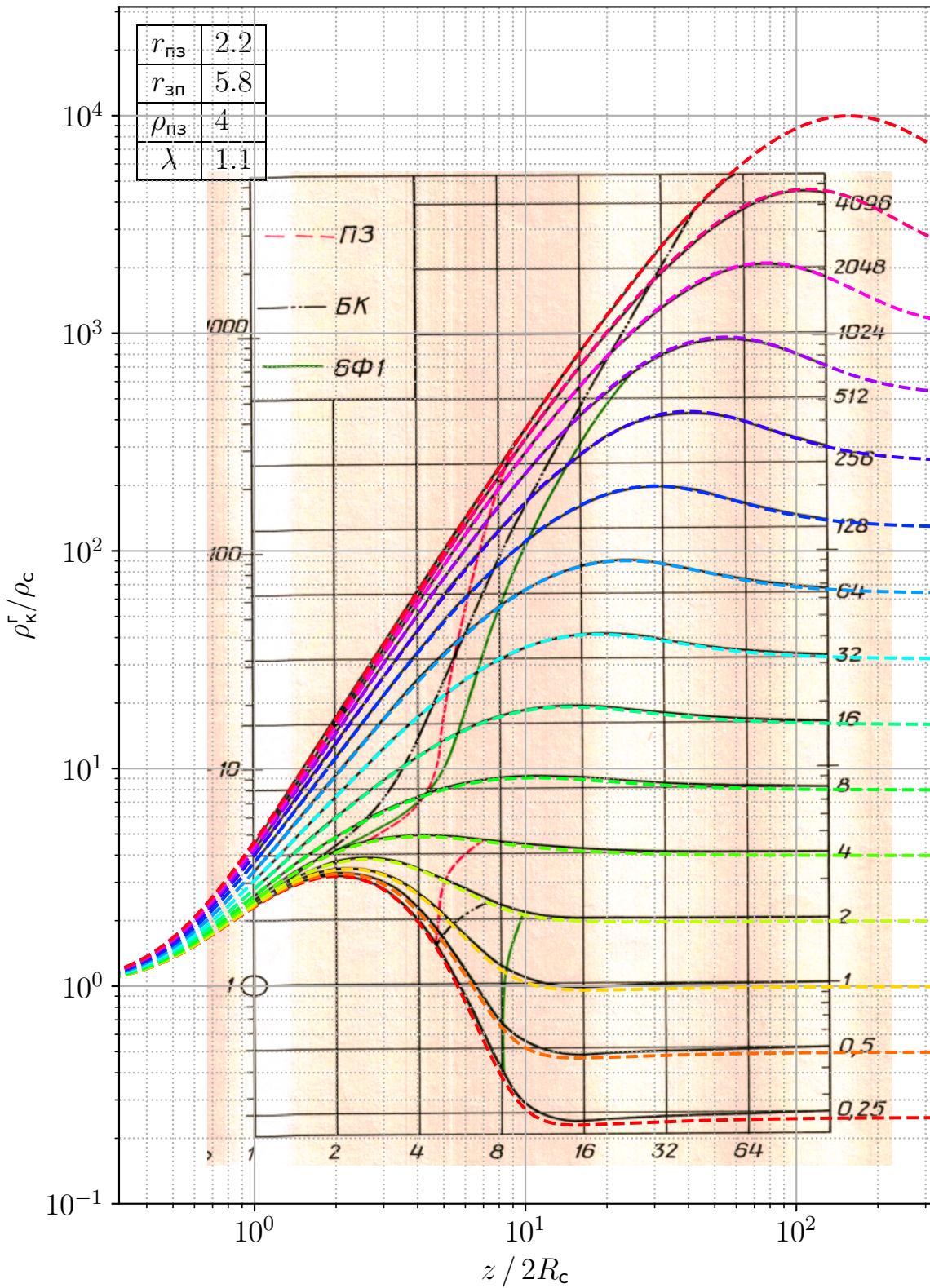


Рис. 4.14

ЗАКЛЮЧЕНИЕ

Освоены пакеты программ для параллельных вычислений: вычислительная платформа для автоматического решения ДУЧП FEniCS, библиотека для векторных вычислений NumPy.

Разработаны прототипы программ для численного решения прямой задачи БКЗ для двух известных моделей прискважинной зоны.

Проведены расчеты и сопоставление с известными результатами в литературе. Показана применимость метода решения прямой задачи БКЗ с приемлимой точностью. В данном случае мы следили за двумя параметрами: гладкостью решения и сравнением "на глаз" с известной палеткой. Поскольку принятая методика решения обратной задачи БКЗ основано на использованием палеток на "сравнении на глаз", то такой критерий годится. Относительная ошибка примерно < 10 %.

Все вычисления произведены на компьютере: операционная система Ubuntu 18.04 LTS, процессор Intel Pentium 4415U 2.30 ГГц с 4 логическими процессорами (1 физический процессор \times 2 ядра в физическом процессоре \times 2 потока в каждом ядре).

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Электрическое зондирование геологической среды. Часть 1. Прямые задачи и методика работ / И. Н. Модин, В. А. Шевнин, В. К. Хмелевской [и др.] — МГУ Москва, 1988. — 176 с. — URL: http://geophys.geol.msu.ru/STUDY/5KURS/Book1_1988MSU.pdf
2. Введение в теорию метода конечных элементов. Учебное пособие / Р.З. Даутов, М.М. Карчевский. — Казань: Казанский государственный университет им. В.И. Ульянова–Ленина, 2004. — 239 с. — URL: https://kpfu.ru/staff_files/F1229619272/MKEbookDRZ.pdf
3. Технология создания многопараметрических палеток для решения прямых и обратных задач скважинной геоэлектрики / К. С. Сердюк [и др.] Каротажник. Результаты исследований и работ ученых и конструкторов, 2014. — Вып. 241. — С. 32-41.
4. Геофизические исследования и работы в скважинах: в 7 т. Т. 1. Промысловая геофизика / Р.А. Валиуллин, Л.Е. Кнеллер. — Уфа: Информреклама, 2010. — 172 с.
5. Методические указания по комплексной интерпретации данных БКЗ, БК, ИК (с комплектом палеток) / Науч.-произв. об-ние "Союзпромгеофизика", Всесоюз. н.-и. и проект.-конструкт. ин-т геофиз. методов исслед., испытания и контроля нефтегазоразведоч. скважин. Е. В. Чаадаев, И. П. Бриченко, А. А. Левченко [и др.] — Калинин: НПО "Союзпромгеофизика", 1990. — 85 с.
6. Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics (2nd edition) / H. Elman, D. Silvester, A. Wathen. — OUP, Oxford, 2014. — 480 с. — URL: <https://books.google.ru/books?id=Ly-TAwAAQBAJ>
7. Solving PDEs in Python –The FEniCS Tutorial Volume I / H. P. Langtangen, A. Logg. — Springer, 2017. — 153 с. — URL: <https://fenicsproject.org/pub/tutorial/pdf/fenics-tutorial-vol1.pdf>

ПРИЛОЖЕНИЕ

ПРИЛОЖЕНИЕ А

ВАРИАНТ ПРОГРАММЫ НА PYTHON ДЛЯ РАСЧЕТА СЕТОК

```
1 # -*- coding: utf-8 -*-
2 import numpy as np
3
4
5 def build_line_2(r_x, R, Theta, graph):
6     # Функция добавления учета границы области в граф.
7     #
8     # Аргументы:
9     #     r_x -- радиус границы области
10    #     R -- радиусы концентрических окружностей в пространстве (
11        r, z)
12
13    """
14    x^2 + y^2 = r^2
15    y = sqrt(r^2 - x^2)
16    """
17    R_circ_x_idx = np.min((R >= r_x).nonzero())
18
19    # радиусы окружностей, которые пересекаются с границей
20    R_circ_x = R[R_circ_x_idx:]
21
22    """
23    y = tan theta * x
24
25    r <= R -> r^2 <= R^2
26
27    r^2 = x^2 + y^2
28    """
29    z = np.tan(Theta) * r_x
30    R__2 = r_x**2 + z**2
31    Theta_line_x_idx, = (R__2 <= R[-1]**2).nonzero()
32
```

```

33     # углы лучей, которые пересекаются с границей
34     Theta_line_x = Theta[Theta_line_x_idx]
35
36     '''
37     cos(theta) = r / R
38     R = r / cos(theta)
39     '''
40
41     R_ = R[R_circ_x_idx-1:]
42
43     # радиусы точек пересечений границы с лучами
44     R_line_x = r_x / np.cos(Theta_line_x)
45
46     # радиусы ближайших к границе точек сетки
47     R_circ_x_idx_near_idx = \
48         np.argmin(np.abs(np.repeat([R_], len(R_line_x), axis=0)
49                     - R_line_x.reshape(len(R_line_x), 1)),
50                  axis=1)
51
52     # индекс ближайших к границе узлов
53     verts_near_idx = (R_circ_x_idx_near_idx + R_circ_x_idx-1) \
54                         + len(R) * Theta_line_x_idx
55
56
57     verts = graph['vertices']
58     r, z = verts[:, 0], verts[:, 1]
59
60     # поместить ближайшие к границе узлы на границу
61     r[verts_near_idx] = r_x
62
63     R_circ1_x_idx = np.delete(np.arange(R_circ_x_idx-1, len(R)),
64                               np.append(R_circ_x_idx_near_idx, 0))
65     R_circ1_x = R[R_circ1_x_idx]
66
67     # пересечения границы с верхними частями окружностей
68     z_circ1_x_positive = np.sqrt(R_circ1_x**2 - r_x**2)

```

```

68     # пересечения границы с верхними частями окружностей
69     # возвращает( view)
70
71     z_circ1_x_negative = np.flip(-z_circ1_x_positive, axis=0)
72
73     # пересечения границы с окружностями
74     z_circ1_x = np.concatenate([z_circ1_x_negative,
75                                z_circ1_x_positive])
76
77     r_x_ = np.full(z_circ1_x.shape, r_x)
78     verts_circ1_x = np.stack([r_x_, z_circ1_x], axis=1)
79
80     verts_x = np.concatenate([verts, verts_circ1_x])
81     verts_circ1_x_idx = np.arange(len(verts_circ1_x)) + len(verts)
82
83     z_x = np.concatenate([z[verts_near_idx], z_circ1_x], axis=0)
84     idx = np.concatenate([verts_near_idx, verts_circ1_x_idx], axis=0)
85
86     dtype = [('z_x', float), ('idx', int)]
87
88     z_x_idx = np.empty(len(z_x), dtype=dtype)
89     z_x_idx['z_x'] = z_x
90     z_x_idx['idx'] = idx
91
92     verts_x_idx = np.sort(z_x_idx, order='z_x')['idx']
93
94     segs_line_x = np.stack([verts_x_idx[:-1], verts_x_idx[1:]], axis=1)
95
96     segs_x = np.concatenate([graph['segments'], segs_line_x], axis=0)
97
98     graph_x = dict(vertices=verts_x, segments=segs_x)
99

```

```

100 def calc_mesh(r, R, Theta):
101     # Функция расчета сетки
102     # с учетом границ областей в системе скважинапласт--.
103     #
104     # Аргументы:
105     #      r -- радиусы границы области
106     #      R -- радиусы концентрических окружностей в пространстве (
r, z)
107     # Theta -- углы лучей, исходящих из начала координат
108
109     import triangle as tr
110     from others import (cartprod, cylinder_verts,
111                           tri2mesh, concat_graphs)
112
113
114     # точка в начале координат
115
116     verts0 = [[0, 0]]
117     graph0 = dict(vertices=verts0)
118
119
120     # граф с точками, регулярно структурировано() расположеными в
пространстве
121
122     RT_verts1 = cartprod(R, Theta, order='F')    # декартово
произведение
123     verts1 = cylinder_verts(RT_verts1)    # перевод из полярных
координат в декартовые
124     graph1 = dict(vertices=verts1, segments=np.array([]).reshape(0,
2))
125
126
127     if r is None or len(r) == 0:
128         # в случае если не заданы границы областей
129         graph1 = dict(vertices=np.array([]).reshape(0, 2),
130                         segments=np.array([]).reshape(0, 2))
131     else:

```

```

132         # добавление учета границ областей в графе
133         for r_ in r:
134             graph1 = build_line_2(r_, R, Theta, graph1)
135
136     # объединение графов
137     graph = concat_graphs(graph0, graph1)
138
139     # добавление в граф границы выпуклой оболочки множества точек
140     # триангуляции,
141     # служит границей расчетной сетки
142     hull_segments = tr.convex_hull(graph['vertices'])
143     graph['segments'] = np.append(graph['segments'], hull_segments,
144                                   axis=0)
145
146
147     # триангуляция Делоне по заданному графу
148     tri = tr.triangulate(graph, 'p')
149
150
151
152
153 def main():
154     from others import crange
155
156
157     # r_s -- радиус скважины
158     # r_pz -- радиус промытой зоны
159     # r_zp -- радиус зоны проникновения
160
161     r_s, r_pz, r_zp = 1, 2.2, 5.8
162
163
164     # n_R_2 -- колво-- радиусов концентрических окружностей
165     # в единицу логарифма длины в пространстве (r /
166     # 2, z / 2)

```

```

166      #           n_Theta -- колво— углов лучей, исходящих из начала
координат
167      #           в правом полупространстве (r / 2, z / 2)
168      # log_R_2_min,
169      # log_R_2_max -- логарифмы значений границ радиуса
170      #           для концентрических окружностей
171
172      n_R_2, n_Theta = 2*10, 2*12
173      log_R_2_min, log_R_2_max = -1.5, 4
174
175
176      R_2 = 10 ** crange(log_R_2_min, log_R_2_max, 1/n_R_2)
177      R = 2 * R_2
178
179      Theta = np.linspace(-np.pi/2, np.pi/2, n_Theta)
180
181      # построение расчетной сетки
182      graph, tri, mesh = calc_mesh([r_s, r_pz, r_zp], R, Theta)
183
184      # вывод колва— узлов расчетной сетки
185      print(f'{mesh.num_vertices():d} узлов\n')
186
187
188      # вывод графиков графа и триангуляции
189
190      import matplotlib.pyplot as plt
191      from others import mpl_init, plot_mesh
192      mpl_init()
193
194
195      plt.figure()
196      plt.subplot(
197          aspect='equal',
198          adjustable='box',
199          xlabel=r'$r / R_{\text{}}$',
200          ylabel=r'$z / R_{\text{}}$'
201      )

```

```
202
203     plt.grid()
204     plot_mesh(graph, tri)
205
206
207     plt.figure()
208     plt.subplot(
209         aspect='equal',
210         adjustable='datalim',
211         xlabel=r'$r / R_{\text{}}$',
212         ylabel=r'$z / R_{\text{}}$',
213         xlim=(0, 7),
214         ylim=(-1, 1),
215     )
216
217     plt.grid()
218     plot_mesh(graph, tri)
219
220
221     plt.show()
222
223
224 if __name__ == "__main__":
225     main()
```