**Министерство науки и высшего образования Российской федерации**

**Федеральное государственное автономное образовательное учреждение**

**Высшего профессионального образования**

**Казанский (Приволжский) Федеральный институт**

ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ

И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

кафедра вычислительной математики

Направление: 01.03.04 — Прикладная математика

Профиль: прикладная математика

КУРСОВАЯ РАБОТА

**Приближенное решение нелинейной стационарной краевой задачи**

|  |  |
| --- | --- |
| Студент 4 курса  группы 09-822  '' '' \_\_\_\_\_ 2021 г. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И. Ж. Шабаков |
|  |  |
| Научный руководитель  ассистент б. с  '' '' \_\_\_\_\_ 2021 г. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Г. О. Трифонова |

Казань — 2021

**СОДЕРЖАНИЕ**

[1. ВВЕДЕНИЕ И ПОСТАНОВКА ЗАДАЧИ 3](#_Toc91473426)

[*2.* ТЕОРЕТИЧЕСКИЕ АСПЕКТЫ ТЕМЫ 4](#_Toc91473427)

[2.1 Теория по нелинейным уравнениям 4](#_Toc91473428)

[2.2 Теория метода конечных элементов (МКЭ) 5](#_Toc91473429)

[2.3 Ознакомление с пакетом Partial Differential Equation Toolbox среды MATLAB 7](#_Toc91473430)

[3. ПРАКТИЧЕСКАЯ ЧАСТЬ 12](#_Toc91473431)

[3.1 Изучение нелинейной задачи Minimal Surface Problem на примере из пакета Partial Differential Equation Toolbox среды MATLAB. 12](#_Toc91473432)

[3.2 Решение модельной нелинейной стационарной краевой задачи с применением пакета Partial Differential Equation Toolbox среды MATLAB 17](#_Toc91473433)

[4 ЗАКЛЮЧЕНИЕ 32](#_Toc91473434)

[5 ЛИТЕРАТУРА 33](#_Toc91473435)

[6 ПРИЛОЖЕНИЕ 34](#_Toc91473436)

# **ВВЕДЕНИЕ И ПОСТАНОВКА ЗАДАЧИ**

Цель данной работы - исследование краевой нелинейной стационарной задачи.

|  |  |
| --- | --- |
|  | (1.1) |

Фильтрация (имеет нелинейный вид) происходит в области .

Проект состоит из нескольких глав, а именно: теоретическая часть, куда включена теория по нелинейным уравнениям, показывающая разрешимость уравнения (1.1), общая теория по методу конечных элементов, который лежит в основе алгоритмов решения пакета MATLAB PDE Toolbox. Представлено ознакомление с самим пакетом Partial Differential Equation Toolbox, раскрывающее способы триангуляции области и подходы решения дифференциальных уравнений в частных производных.

Практическая часть включает в себя решение тестового примера, представленного в Help’е MATLAB’а и исследование модельной задачи (1.1): проведение серии испытаний с различным параметром и оценка сходимости с точным решением.

Задача исследования заключается в изучении теории по направлению курсовой работы и исследование решения модельной задачи.

В приложении имеется код программы. Программа реализована в среде MATLAB. Курсовая работа оформлена в Word

# **ТЕОРЕТИЧЕСКИЕ АСПЕКТЫ ТЕМЫ**

# **Теория по нелинейным уравнениям**

Краевым задачам для эллиптических дифференциальных уравнений можно дать функционально-аналитическую формулировку, то есть представить в виде операторного уравнения. Рассмотрим неоднородную краевую задачу Дирихле:

|  |  |
| --- | --- |
|  | (2.1.1) |

Для уравнения (2.1.1) подходящей функционально-аналитической постановкой является:

|  |  |
| --- | --- |
|  | (2.1.2) |

Оператор (действует из рефлексивного банахова пространства в ) можно рассматривать как расширение дифференциального оператора E. Оператор представим в виде:

L – непрерывный линейный оператор, действующий из в

– нелинейный оператор

сопряженный к L оператор

Оператор A является энергетическим расширением оператора E. Согласно Лемме 2.1[1, 55с.] мы можем назвать (2.1.2) функционально-аналитической формулировкой краевой задачи (2.1.1) при условии, что А является энергетическим расширением оператора E. Для того, чтобы A являлось энергетическим расширением, необходимо соблюдение условий a -d), приведенных в [1, 55стр].

Данные условия выполняются для нашей задачи (2.1.1). Подробный процесс описан в [1, 63стр]. Так же стоит указать условия на коэффициенты эллиптического дифференциального оператора, которые обеспечивают у оператора A наличие свойств(непрерывность/монотонность/коэрцитивность). В [1, 87стр] Леммы 1.4-1.6 показывают, что интересующие нас свойства переносятся на , и устанавливают, при каких условиях обладают данными свойствами. Эти манипуляции отображают разрешимость уравнения с монотонным и коэрцитивным оператором, коим является (1.1) (см. [1, §2, Теорема 2.1])

# **Теория метода конечных элементов (МКЭ)**

В основе алгоритмов решения PDE Toolbox лежит метод конечных элементов (МКЭ) для решения дифференциальных уравнений в частных производных (или же системы дифф. уравнений). МКЭ представляет из себя аппроксимацию области в виде треугольников (для двухмерной области), то есть геометрия области задается путем триангуляции.

Рассмотрим модельную краевую задачу:

|  |  |
| --- | --- |
| , | (2.2.1) |

является решением задачи (2.2.1).. Нам потребуется перейти к интегральному тождеству.

Для того, чтобы сделать это, умножим наше уравнение (2.2.1) на функцию , принадлежавшей пространству пробных функций (, V является аффинным множеством). После чего интегрируем полученное выражение по , применяем формулу Гаусса- Остроградского и учитываем краевые условия. Получаем:

Разобьем область на конечные элементы одного типа . Обозначим за триангуляцию области. Тогда: . . Так же граничные точки условия Дирихле являются вершинами каких-либо элементов. – сетка узлов, - множество узлов интерполяции, принадлежащих .

Пространство конечных элементов имеет вид:

Аппроксимация и :

Задача сводится к поиску такой функции , что для любой пробной функции :

|  |  |
| --- | --- |
|  | (2.2.2) |

Интегральное тождество является основой МКЭ. Данное тождество эквивалентно задаче (2.2.1): если – есть решение (2.2.1), то оно удовлетворяет тождеству (2.2.2). Далее (2.2.2) сводится к системе алгебраических функций (выбором базиса в ). И посредством кусочно-полиномиальной интерполяции получают приближенное решение .

# **Ознакомление с пакетом Partial Differential Equation Toolbox среды MATLAB**

Для решения дифференциальных уравнений в частных производных в среде Matlab имеется специальный пакет Partial Differential Equation Toolbox (PDE Toolbox). С помощью которого возможно решение краевых задач для дифференциальных уравнений в частных производных методом КЭ.

Пакет имеет широкий функционал, позволяющий автоматизировать реализацию МКЭ для решения дифференциальных уравнений в частных производных. В состав PDE Toolbox входит pdetool(графическая приложение), имеющий понятный интерфейс, которая упрощает для пользователя осуществление решения дифф. уравнений. PDE Toolbox подходит для эллиптических, параболических и гиперболических уравнений.

Более подробнее хочется остановиться на функционале пакета PDE Toolbox, который позволяет определить геометрию области и построение треугольной сетки. Для этого в Matlab, к примеру, имеется (причем не единственная!) функция **generateMesh**, в качестве аргумента принимающая геометрию области.

Пример:

model = createpde;

geometryFromEdges(model,@circleg);

generateMesh(model);

figure

pdemesh(model);

axis equal

title('Триангуляция геометрии')

Сначала создается переменная модели, которой присваивается геометрия области (в нашем случае окружность, ) . После чего происходит триангуляция области вызовом метода generateMesh(), аргументом которой является наша окружность.

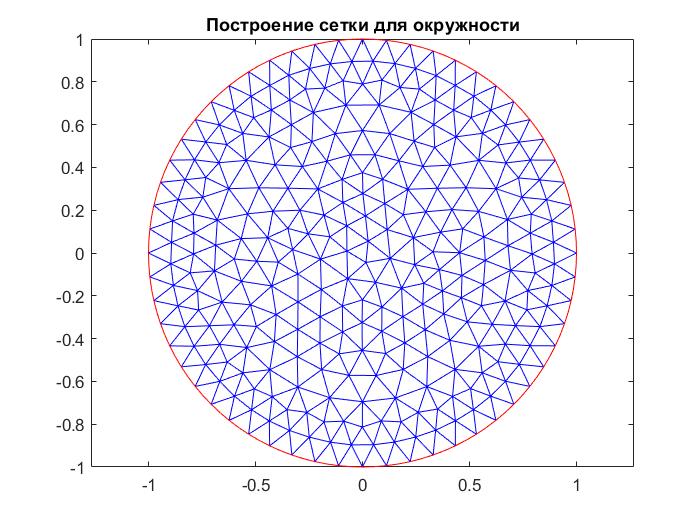


Рис 1. Сетка для окружности

Так же мы можем регулировать размер треугольников в нашей области. Для этого достаточно передать в функцию generateMesh параметр ‘Hmax’(‘Hmin’), регулирующий максимальный(минимальный) размер треугольников. Ниже пример триангуляции при различном параметре “Hmax”:

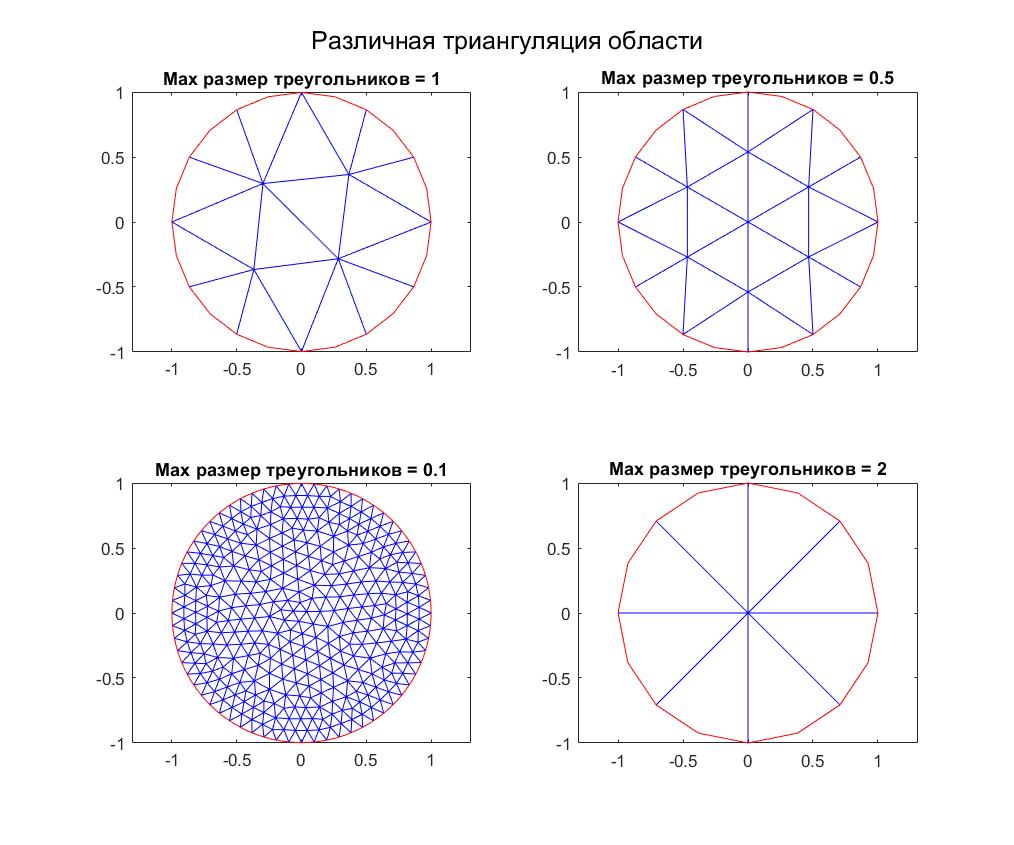


Рис 2. Триангуляция области при разных параметрах “Hmax”

Данная функция имеет еще пару полезных параметров, о которых можно узнать из Help’а внутри Matlab’а.

Кроме построения сетки нам будут полезно получить значения внутри узлов сетки. Для этого нужно обратиться к нашей модели:

p=model.Mesh.Nodes

В p хранятся координаты узлов сетки (вершин элементов): в p(1, i) (p(2, i)) хранится x (y) координата i-го узла.

Стоит упомянуть, что внутри PDE Toolbox можно задать любую интересующую нас геометрию для области. Имеются два способа:

1. Матрица
2. Функция

**1 Способ**

Генерацию геометрии можно задать с помощью матрицы, где каждый столбец определяет один сегмент. Первая строка отвечает за тип каждого элемента (2 – отрезок, 4- эллипс). Вторая и третья, четвертая и пятая строки отвечают за x(y) координаты начальной и конечной точки сегмента. Шестая и седьмая строки отображают находящиеся слева и справа области. Восьмая, девятая, десятая, одиннадцатая, двенадцатая строки нужны только для эллипса, определяя x и y координаты центра эллипса, его x и y полуоси (для отрезка прямой являются нулями) и угол поворота эллипса соответственно. Для отрезка данные строки равны нулю.

Пример:

Построим простую область, а именно квадрат. Матрица, определяющая геометрию области будет выглядеть так:

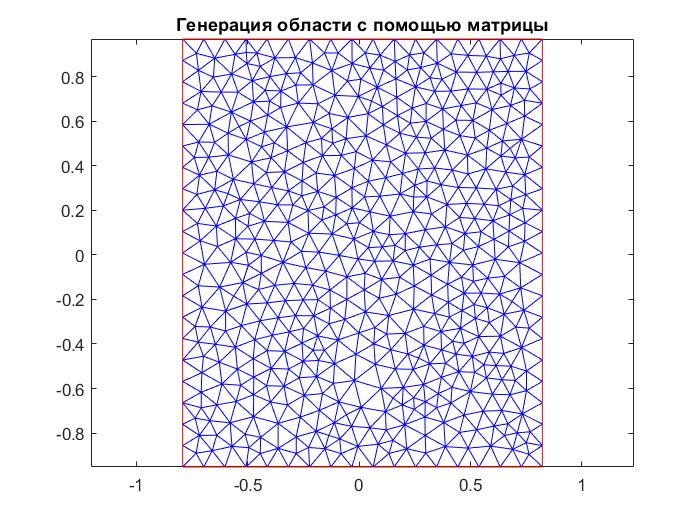


Рис 3. Генерация области с помощью матрицы

**2 Способ**

Для того, чтобы задать геометрию области с помощью функции, необходимо определить две матрицы: d и xy. В матрице d количество столбцов равняется количеству сегментов. В матрице xy задаются данные для прямых. Матрица d является обязательной. После объявления этих двух матриц следует определение сегментов в параметрическом виде. Для больших подробностей следует обратиться к Help в MATLAB’е.

Пример:

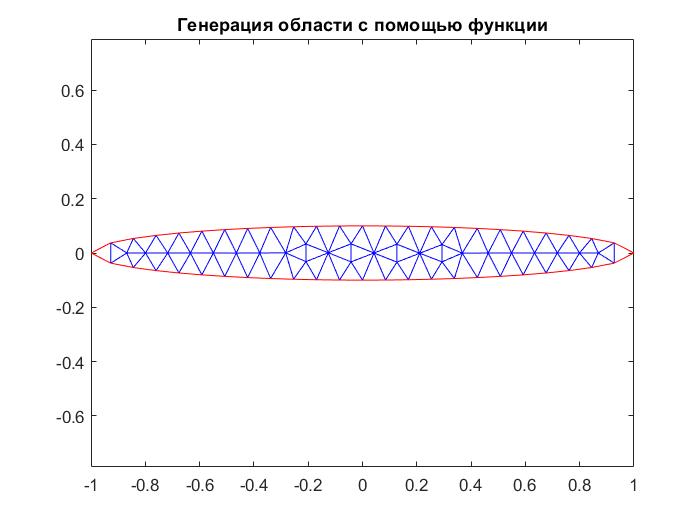


Рис 4. Генерация области с помощью функции

Задание геометрии с помощью функции позволяет строить более сложные области

# **ПРАКТИЧЕСКАЯ ЧАСТЬ**

# **Изучение нелинейной задачи Minimal Surface Problem на примере из пакета Partial Differential Equation Toolbox среды MATLAB.**

Прежде чем приступить к решению задачи мы должны убедиться, что среда MATLAB умеет решать уравнение этого типа. PDE Toolbox позволяет получить решение уравнений вида:

|  |  |
| --- | --- |
|  | (3.1.1) |

Порядок решений уравнения (системы уравнений):

1. Создать контейнер для дифференциального уравнения (либо же системы дифференциальных уравнений)
2. Задать геометрию области
3. Задать граничные условия
4. Задать коэффициенты уравнения(системы)
5. Собрать сетку
6. Решить встроенными в пакет PDE Toolbox специальными функциями полученную задачу

Немного полезной информации:

Так как в главе выше была описана генерация геометрии области и построение сетки вне графического приложения, стоит так же остановиться на таких вещах как задание граничных условий и коэффициентов уравнения.

Для того, чтобы задать граничные условия уравнения, нужно воспользоваться встроенной в PDE Toolbox функции applyBoundaryCondition, которая в качестве аргументов принимает тип граничного условия, ребра или грани, на котором оно (г.у) задается и коэффициенты. Например, задача условия Дирихле для двухмерной области подразумевает, что решение на определенном ребре или грани удовлетворяет уравнению . Для условия Нейманна нужно решить уравнение вида .

applyBoundaryCondition(model,'dirichlet','Edge',[e1,e2,e3],'r',3,'h',2);

задает граничное условие Дирихле на ребрах

applyBoundaryCondition(model,'neumann','Edge',[e1,e2,e3],'q',2,'g',3);

задает граничное условие Нейманна на ребрах . Так же можно задать смешанные граничные условия.

Для следующей задачи потребуется задать коэффициенты c и f. Они задаются похожим образом и могут быть либо константой, либо функцией. Особый интерес представляет из себя задача коэффициента функцией. Она имеет два обязательных аргумента, а именно **location** и **state**. location имеет поля location.x, location.y, location.z, которые являются координатами точек, для которых вычисляется значение коэффициента. state же имеет поля state.ux, state.uy, state.uz, которые являются частными производными решения в точке, а так же поле state.u – текущее значение решения u.

Примеры задания коэффициентов:

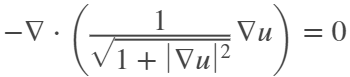
c = c\_coef(location,state)

specifyCoefficients(model,'c',@c\_coef,...)

f = f\_coef(location,state)

specifyCoefficients(model,'f',@f\_coef,...)

Разобравшись в пакете PDE Toolbox, попробуем решить нелинейную задачу минимальной поверхности. Имеется уравнения вида

, 

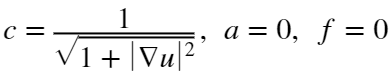
Граничное условие Дирихле:



Эллиптическое уравнение в PDE Toolbox имеет вид:



Для данной задачи вычисление коэффициента c, f и a равно:

.

Для начала создадим геометрию области. В нашем случае это будет окружность(задается с помощью функции @circleg):

model = createpde;

geometryFromEdges(model,@circleg);

Построим геометрию области и отобразим узлы. Окружность задается 4 узлами:

pdegplot(model,'EdgeLabels','on');

axis equal

title 'Геометрия области с отображением узлов';

Задаем коэффициенты для нашего уравнения:

a = 0;

f = 0;

cCoef = @(region,state) 1./sqrt(1+state.ux.^2 + state.uy.^2);

specifyCoefficients(model,'m',0,'d',0,'c',cCoef,'a',a,'f',f);

Задаем граничное условие  для нашего уравнения с помощью функции applyBoundaryCondition, которой передается в качестве аргумента матрица bcMatrix

bcMatrix = @(region,~)region.x.^2;

applyBoundaryCondition(model,'dirichlet',...

'Edge',1:model.Geometry.NumEdges,...

'u',bcMatrix);

Генерируем триангуляцию области и строим график сетки. Максимальный размер треугольников сделаем равным 0.1:

generateMesh(model,'Hmax',0.1);

figure;

pdemesh(model);

axis equal

Включаем отображение прогресса решения задачи. Решение задачи осуществляется методом solvepde

model.SolverOptions.ReportStatistics = 'on';

result = solvepde(model);

u = result.NodalSolution;

Строим график решения:

figure;

pdeplot(model,'XYData',u,'ZData',u);

xlabel 'x'

ylabel 'y'

zlabel 'u(x,y)'

title 'Minimal Surface'

Полученные графики:



Рис 5. Геометрия области для задачи минимальной поверхности

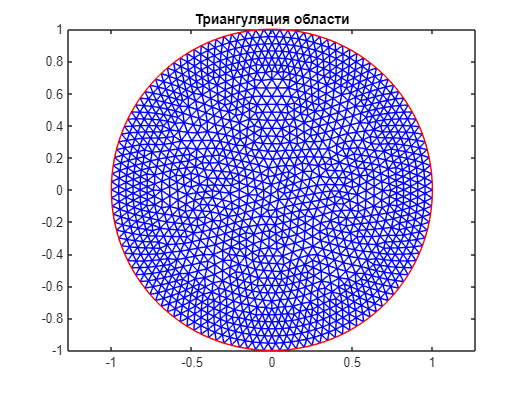


Рис 6. Триангуляция поверхности для задачи минимальной поверхности при Hmax = 0.05

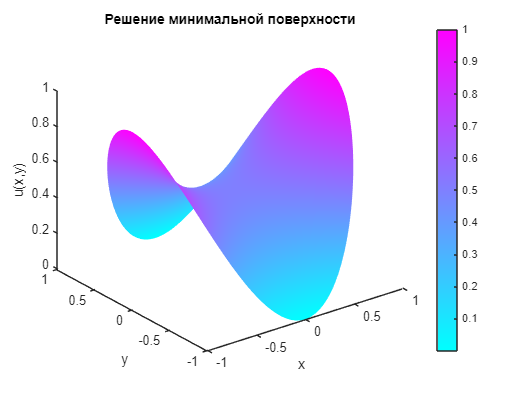


Рис 7. Решение задачи минимальной поверхности. Ось z есть u(x,y)

# **Решение модельной нелинейной стационарной краевой задачи с применением пакета** **Partial Differential Equation Toolbox среды MATLAB**

Перейдем к исследованию нашей модельной задачи (1.1). Решение реализуется подобно примеру, описанному выше. Отличия лишь в том, что мы знаем точное решение уравнения, поэтому можем сравнить на сколько приближенное решение близко к точному. Пакет PDE Toolbox позволяет нам решить уравнение (1.1), так как оно имеет вид (3.1.1)

Для решения уравнения необходимо задать геометрию области, определиться с коэффициентами a, c и f, граничными условиями. В качестве области так же рассматривается единичная окружность. Коэффициенты задаются следующим образом:

Коэффициент a:

Коэффициент c:

Коэффициент f:

На границе окружности задано условие:

Параметр

Проведем численный эксперимент, варьируя значения и сравнивая приближенное и точное решение. Для этого мы должны получить значения узлов нашей модели и подставить их:

p = model.Mesh.Nodes;

exact = u(r2(p(1,:),p(2,:)));

:

При получаем такую картину

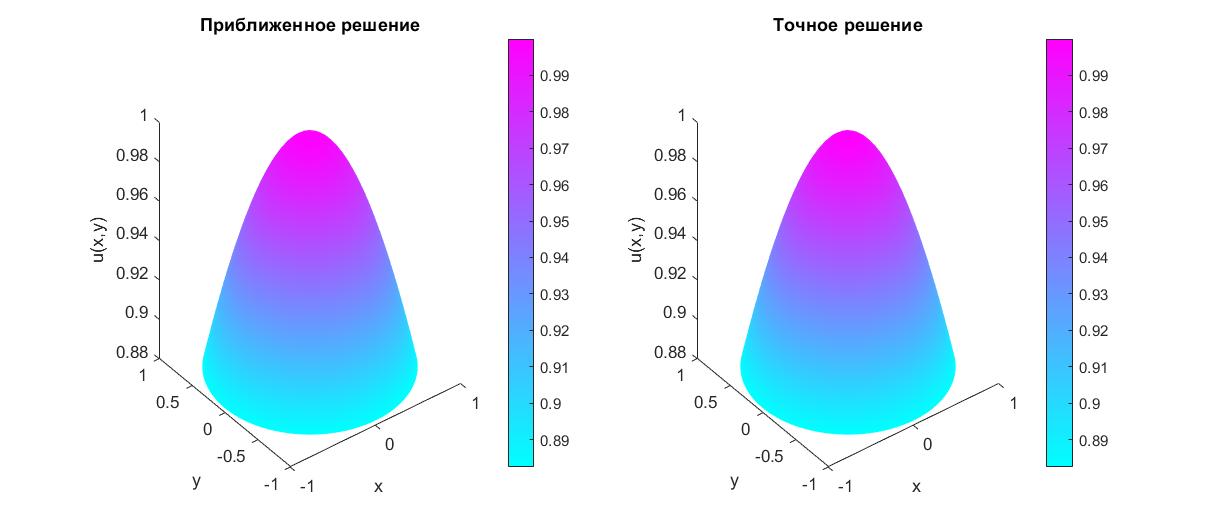


Рис 8. Приближенное и точное решение (= 2)

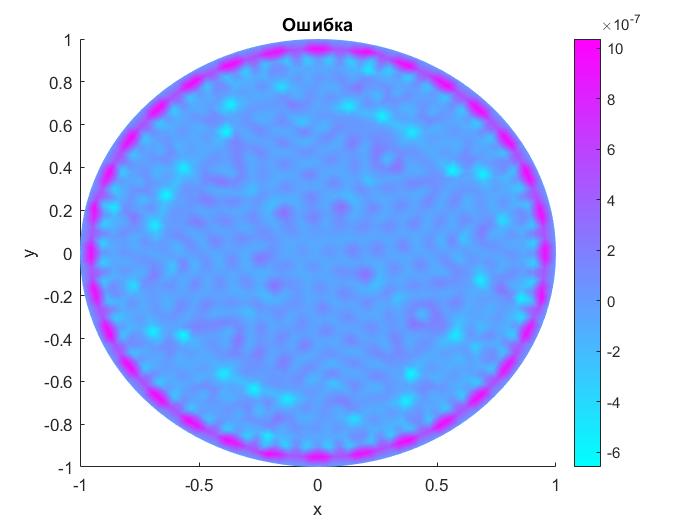


Рис 9. Ошибка вычислений (= 2)

Погрешность при = 2 составляет **1,0360e-06**

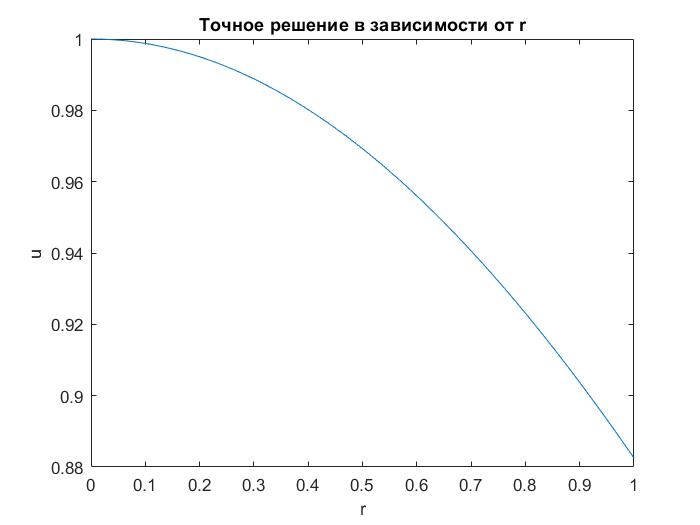
****

Рис 10. Точное решение (= 2)

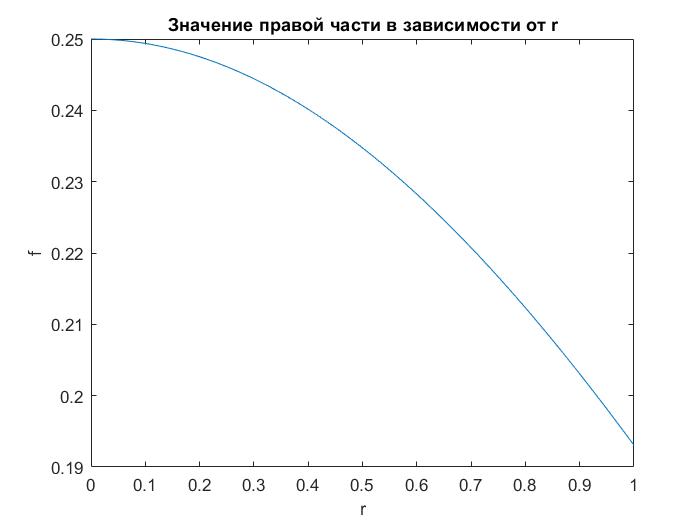


Рис 11. Правая часть (= 2)

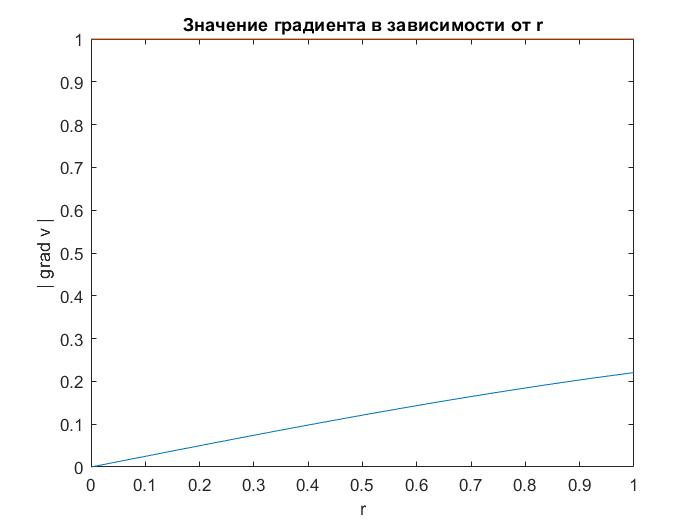


Рис 12. Значение градиента (= 2)

Так как значение градиента меньше единицы- закон фильтрации на всей области не меняется, а значит и скорость фильтрации постоянна:

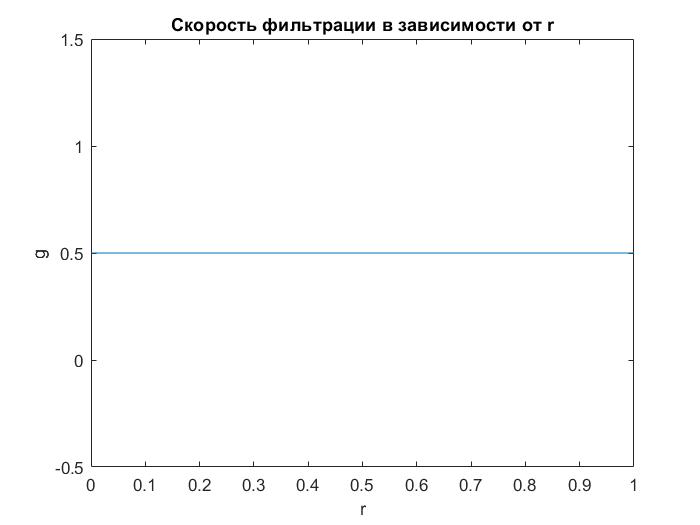


Рис 13. Скорость фильтрации (= 2)

:

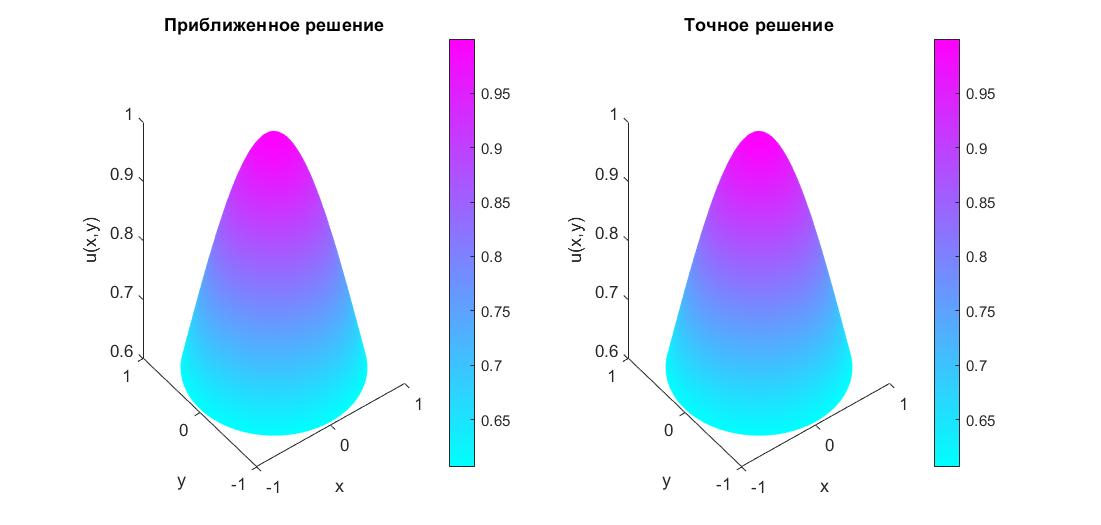
****

Рис 14. Приближенное и точное решение (= 1)

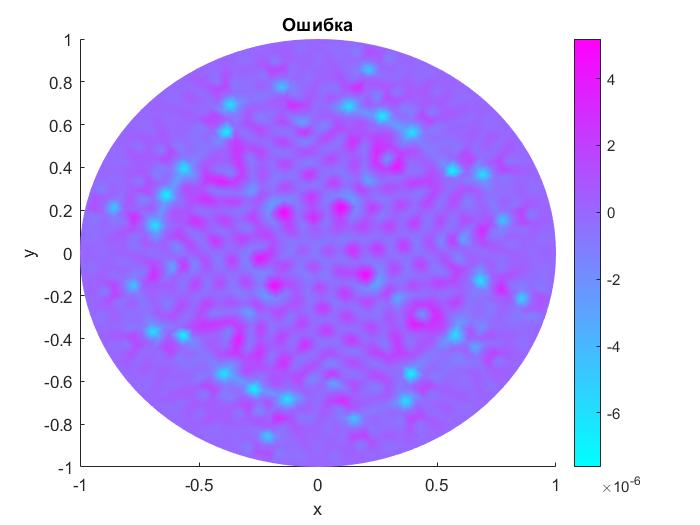


Рис 15. Ошибка вычислений (= 1)

Погрешность при =1 составляет **7,6170e-06**

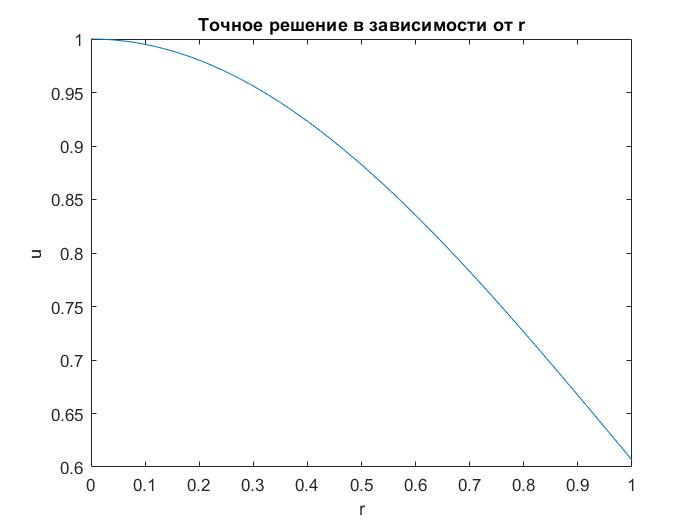
****

Рис 16. Точное решение (= 1)

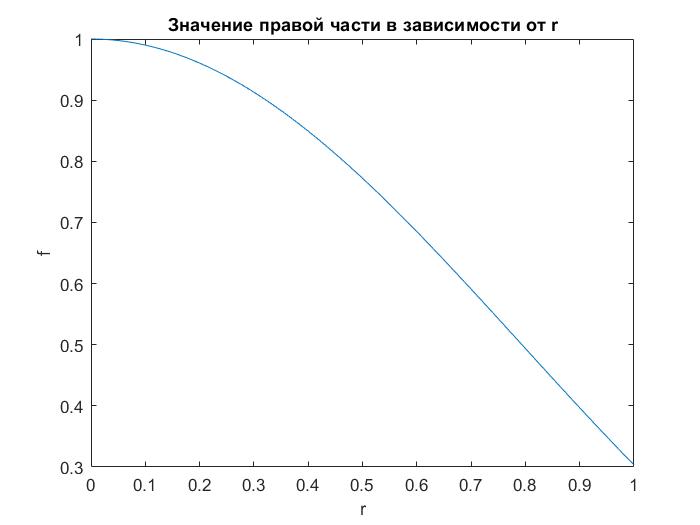


Рис 17. Правая часть (= 1)

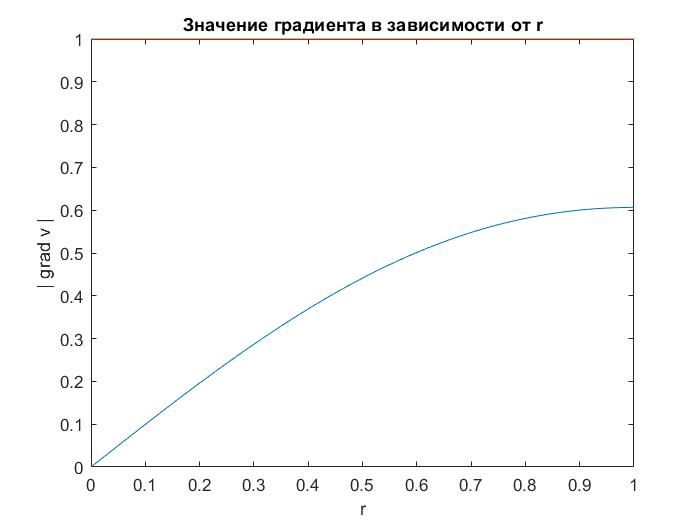


Рис 18. Значение градиента (= 1)

При так же видим, что на всей области значение градиента меньше единицы– закон фильтрации не меняется, а скорость фильтрации постоянна

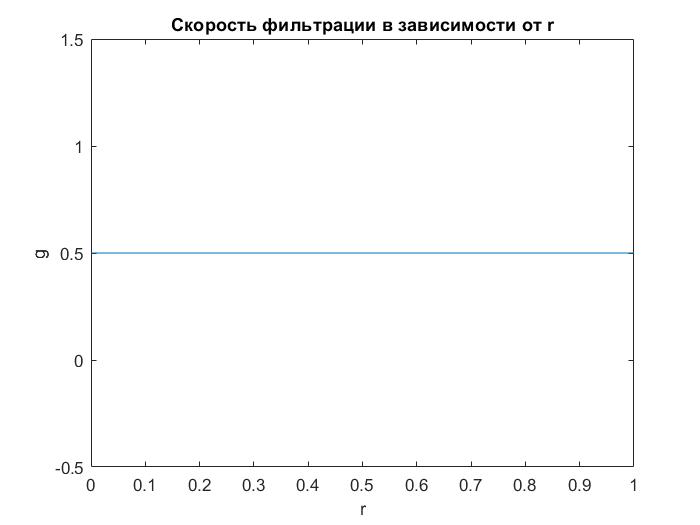


Рис 19. Скорость фильтрации(= 1)

:

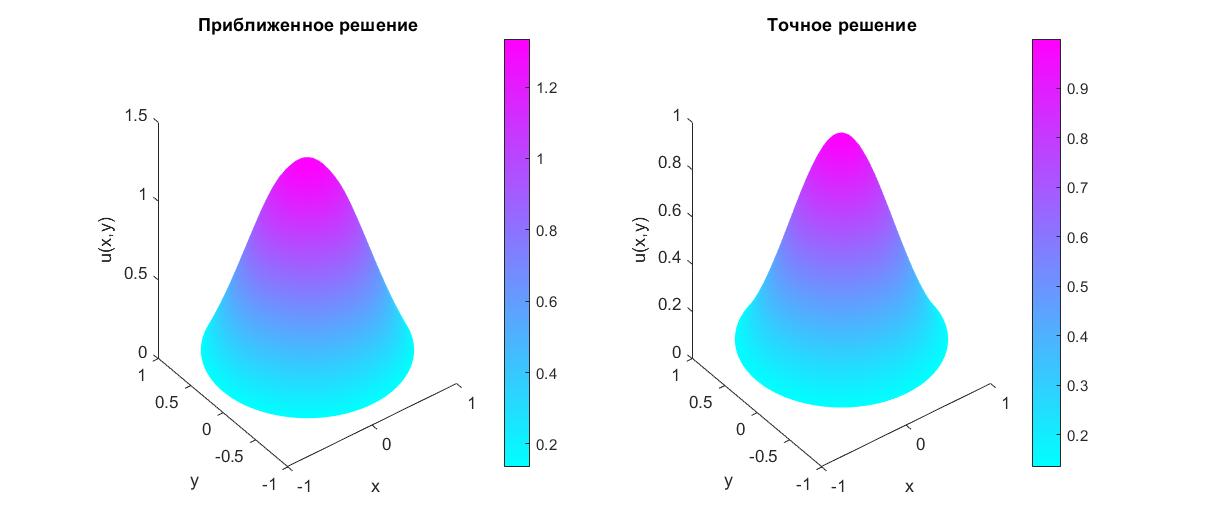
****

Рис 20. Приближенное и точное решение (= 0.5)

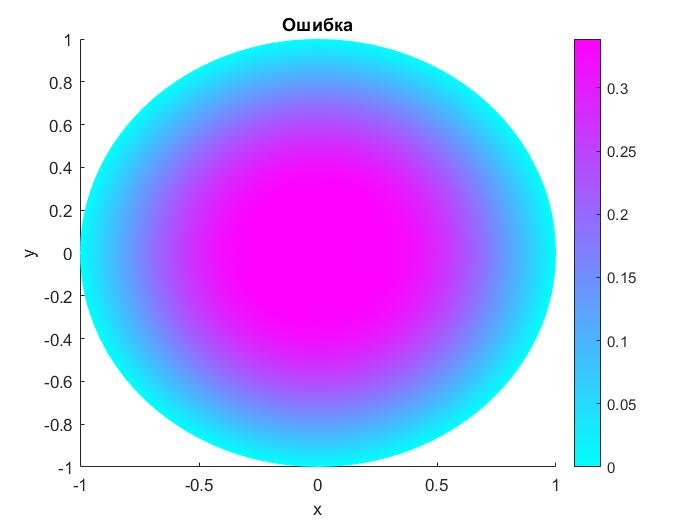


Рис 21. Ошибка вычислений (= 0.5)

Погрешность при =0.5 составляет **0,3388**

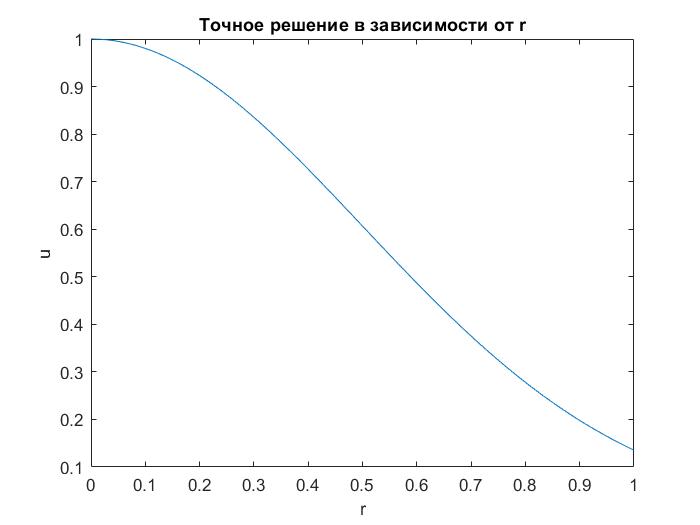
****

Рис 22. Точное решение (= 0.5)

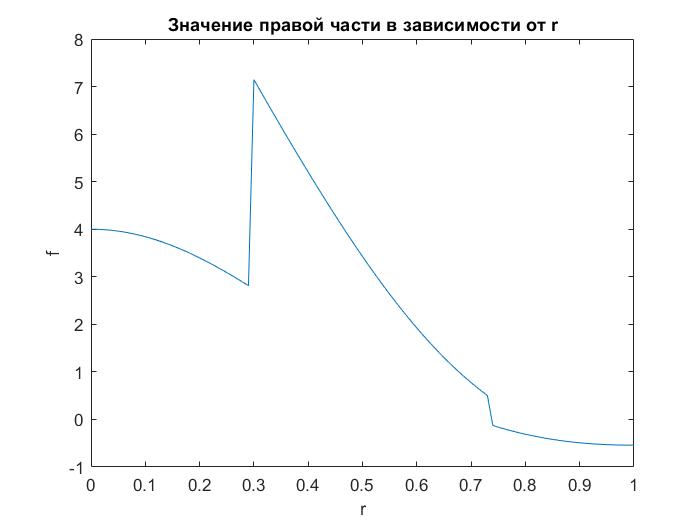


Рис 23. Правая часть (= 0.5)

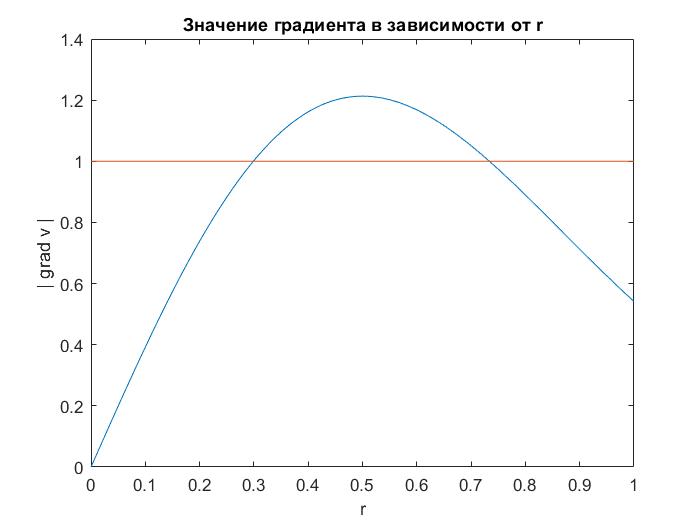


Рис 24. Значение градиента (= 0.5)

При картина такова: при r= 0.29 и r=0.73 закон фильтрации меняется, следовательно меняется и скорость фильтрации

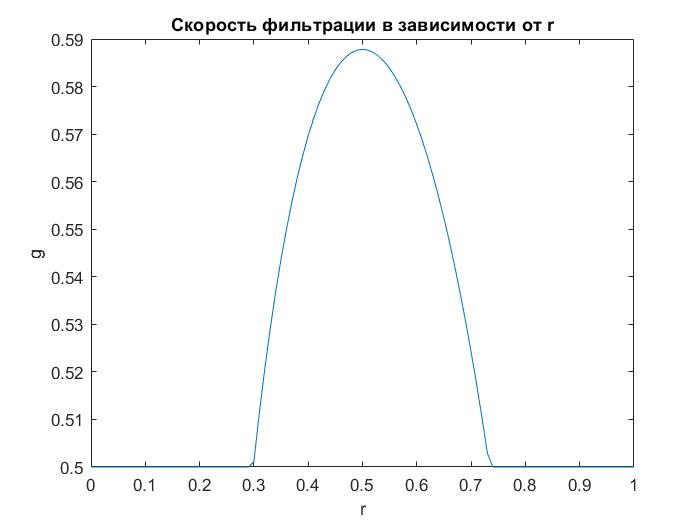


Рис 25. Скорость фильтрации (= 0.5)

:

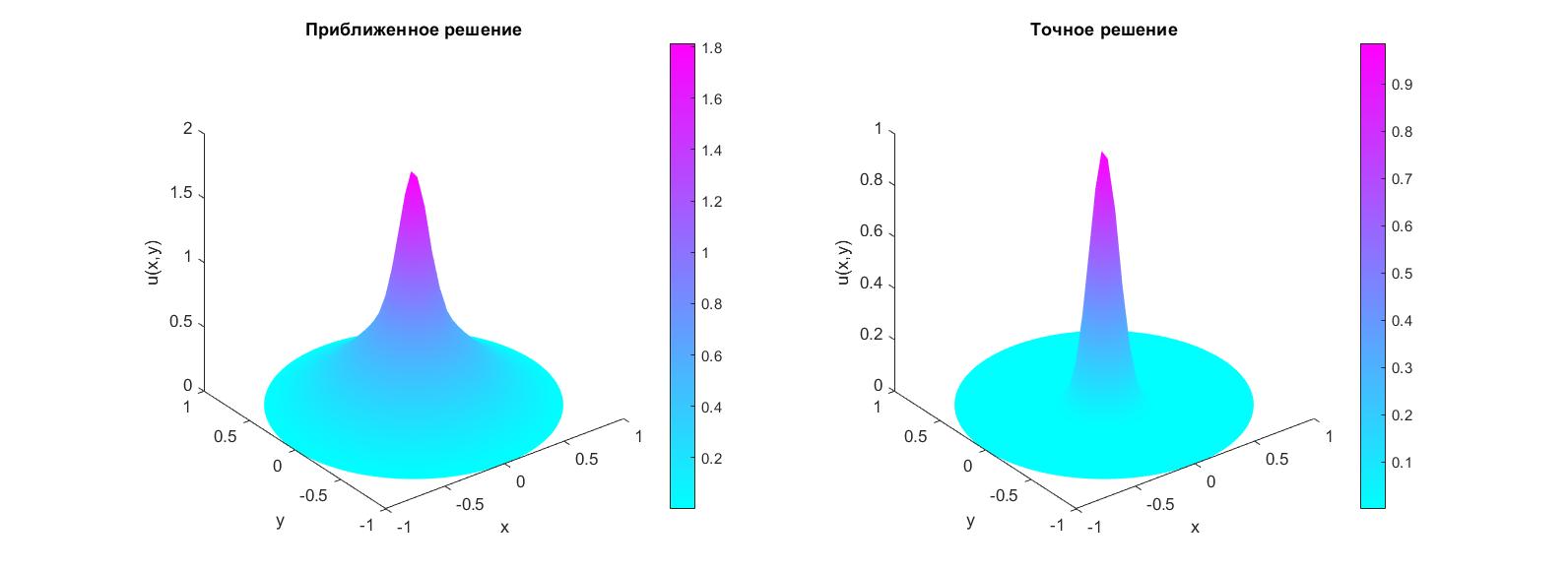
****

Рис 26. Приближенное и точное решение (= 0.1)

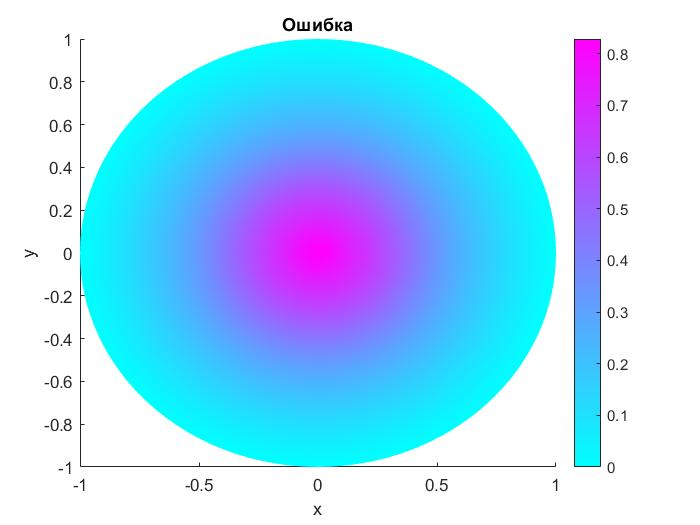


Рис 27. Ошибка вычислений (= 0.1)

Погрешность при =0.1 составляет **0,8284**

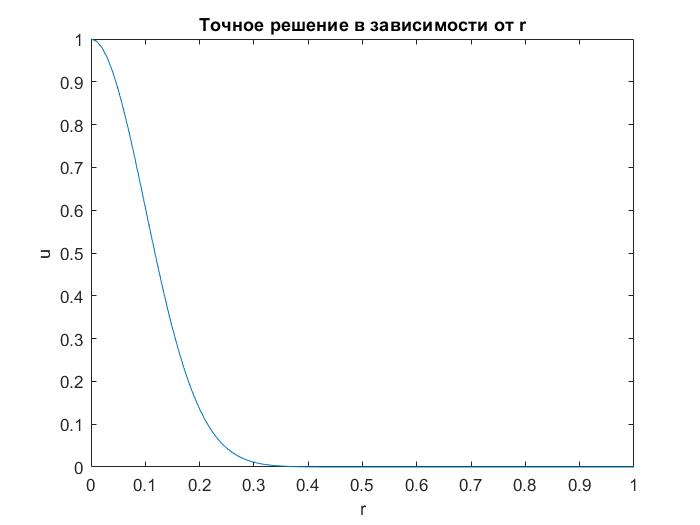
****

Рис 28. Точное решение (= 0.1)

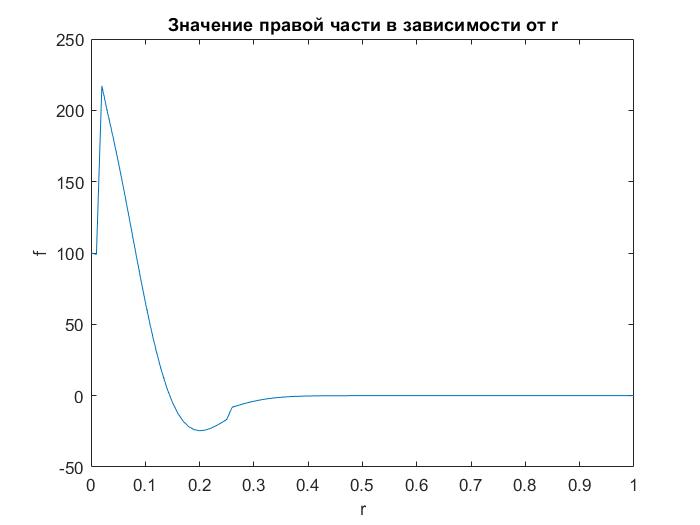


Рис 29. Правая часть (= 0.1)

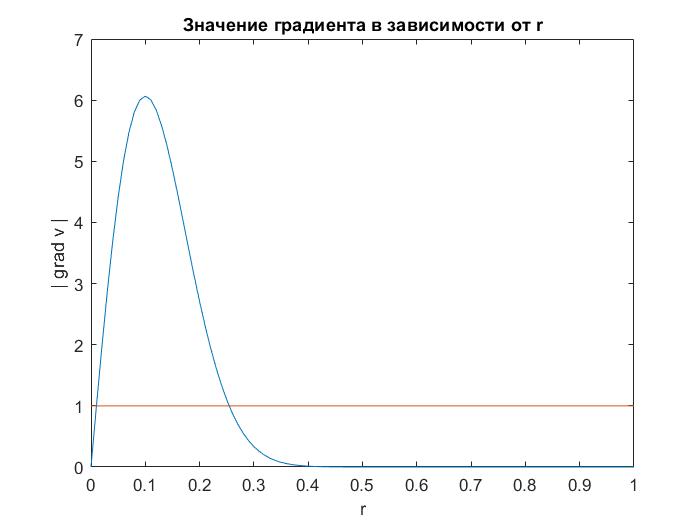


Рис 30. Значение градиента (= 0.1)

При картина такова: при r= 0.01 и r=0.255 закон фильтрации меняется, следовательно меняется и скорость фильтрации

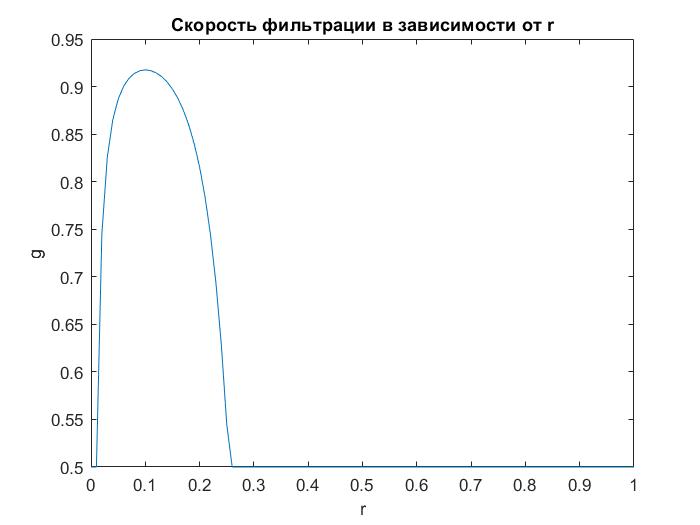


Рис 31. Скорость фильтрации (= 0.1)

В результате исследования получаем, что чем больше параметр , тем точнее получается приближенное решение, выполненное средством пакета MATLAB PDE Toolbox. Представим результаты в виде таблицы:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | =2 | =1 | =0.5 | =0.1 |
| Погрешность | 1,0360e-06 | 7,6170e-06 | 0,3388 | 0,8284 |

Таблица 1. Погрешность вычислений при разных значениях

Уменьшение размера сетки не дает видимого улучшения результата: погрешность вычисления так же остается большой для небольших значений .

|  |  |  |  |
| --- | --- | --- | --- |
|  | Hmax=0.1 | Hmax=0.05 | Hmax=0.025 |
|  | 0.3387 | 0.3342 | 0.3320 |
| =0.1 | 0.8284 | 0.8523 | 0.8086 |

Таблица 2. Погрешность вычислений при разных значениях Hmax

# **ЗАКЛЮЧЕНИЕ**

Работа по исследования решения нелинейной стационарной задачи фильтрации была выполнена. В ходе работы изучена литература по теме нелинейных уравнений, сформировано представление о том, как они разрешаются: в частности, методом конечных элементов, теоретические аспекты которого разбираются в одном из параграфов, и реализована программа, позволяющая решить краевую задачу дифференциального уравнения

В практической части используется пакет среды MATLAB PDE Toolbox, использование которого подробно поясняется в одном из параграфов. Для модельной задачи (1.1) были проведены численные эксперименты и получены соответствующие результат: чем меньше параметр , тем выше погрешность вычисления, а уменьшение размера сетки все равно не позволяет достичь приемлемой погрешности

# **ЛИТЕРАТУРА**

[1] Гаевский Х., Грегер К., Захариас К. Нелинейные операторные уравнения и операторные дифференциальные уравнения / М.: Мир, 1978. 336 с.

[2] Задворнов О. А. Существование решения квазилинейной эллиптической краевой задачи при наличии точечных источников // Учен. зап. Казан. ун-та. Сер. Физ.-матем. науки. 2010. Т. 152. № 1. С. 155–163.

[3] Даутов Р.З., Карчевский М.М. Введение в теорию метода конечных элементов. Учебное пособие. — Казань: Казанский государственный университет им. В.И. Ульянова–Ленина, 2004. — 239 с.– Режим доступа: <https://repository.kpfu.ru/?p_id=45616>

[4] Даутов Р.З Программная реализация метода конечных элементов в МATLAB. Учебное пособие. — Казань: Казанский федеральный университет, 2014. —106c.

[5] Help Matlab раздел Partial Differential Equation Toolbox — Examples.

[6] Использование MATLAB.Решение дифференциальных уравнений в частных производных. PDE Toolbox. :[Электронный ресурс] // Харьковский Национальный Университет механико – математический факультет кафедра геометрии им. А.В. Погорелова. URL: <http://geometry.karazin.ua/resources/documents/20140425101823_05e9c091f50f.pdf>

# **ПРИЛОЖЕНИЕ**

close all; clear; clear all

sig =0.5;

sig2= sig\*sig;

sig4=sig2\*sig2;

k=0.5;

%Функции u- точное решение, grad\_u- градиент u, phi- вторая производная, g-

%скорость фильтрации, f- правая часть

r2 = @(x,y) x.^2+y.^2;

r=@(r2) sqrt(r2);

u=@(r2) exp(-r2./(2\*sig2));

grad\_u=@(r2) r(r2).\*u(r2)/sig2;

phi=@(r2) (r2-2\*sig2)./sig4.\*u(r2);

g=@(s) k + (1-k)\*(s>1).\*(1-1./(s+(s<0.0000001)));

f\_func=@(r2) k\*phi(r2) + (1-k)\*(grad\_u(r2)>1).\*(phi(r2)-1./(r(r2)+(r(r2)<0.0000001)));

model = createpde;

geometryFromEdges(model,@circleg);

pdegplot(model,'EdgeLabels','on');

axis equal

title 'Geometry with Edge Labels';

%Коэффиценты a, f, c

a = 0;

f= @(region, ~) -f\_func(r2(region.x, region.y));

c = @(~, state) g(r(r2(state.ux, state.uy)));

specifyCoefficients(model,'m',0,'d',0,'c',c,'a',a,'f',f);

%Граничные условия

bcMatrix = @(location,~) u(r2(location.x, location.y));

applyBoundaryCondition(model,'dirichlet',...

'Edge',1:model.Geometry.NumEdges,...

'u',bcMatrix);

generateMesh(model,'Hmax',0.05);

figure;

pdemesh(model);

axis equal

model.SolverOptions.ReportStatistics = 'on';

result = solvepde(model);

u\_result = result.NodalSolution;

subplot(1,2,1)

pdeplot(model,'XYData',u\_result,'ZData',u\_result);

xlabel 'x'

ylabel 'y'

zlabel 'u(x,y)'

title 'Приближенное решение'

p = model.Mesh.Nodes;

exact = u(r2(p(1,:),p(2,:)));

erU=u\_result - exact';

subplot(1,2,2)

pdeplot(model,'XYData',exact,'ZData',exact);

xlabel 'x'

ylabel 'y'

zlabel 'u(x,y)'

title 'Точное решение'

figure;

pdeplot(model,'XYData',erU)

title('Ошибка');

xlabel('x')

ylabel('y')

err = norm(erU, inf);

figure;

R=(0:0.01:1);

U=u(R.^2);

plot(R,U);

xlabel 'r'

ylabel 'u'

title('Точное решение в зависимости от r')

figure;

R=(0:0.01:1);

U=-f\_func(R.^2);

plot(R,U);

xlabel 'r'

ylabel 'f'

title('Значение правой части в зависимости от r')

figure;

R=(0:0.01:1);

UG=grad\_u(R.^2);

plot(R,UG,R,R\*0+1);

xlabel 'r'

ylabel '| grad v |'

title('Значение градиента в зависимости от r')

figure;

R=(0:0.01:1);

U=g(grad\_u(R.^2));

plot(R,U);

xlabel 'r'

ylabel 'g'

title('Скорость фильтрации в зависимости от r')

hmax = 0.1;

error = [];

err = 1;

count=0

while count<3 % run until error <= 5e-7

count=count+1;

generateMesh(model,'Hmax',hmax); % refine mesh

results = solvepde(model);

u\_result = results.NodalSolution;

p = model.Mesh.Nodes;

exact = u(r2(p(1,:),p(2,:)));

err = norm(u\_result - exact',inf); % compare with exact solution

error = [error err]; % keep history of err

hmax = hmax/2

end