МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Физические методы контроля»

КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ В ПРИБОРОСТРОЕНИИ

Методические рекомендации к лабораторным работам для студентов направления подготовки 12.03.01 «Приборостроение» очной формы обучения

Могилев 2022

УДК 004.4:681.2
ББК 32.973.202:34.9
К63

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Физические методы контроля» «30» августа 2022 г., протокол № 1

Составитель канд. техн. наук, доц. А. В. Кушнер

Рецензент канд. техн. наук, доц. С. К. Крутолевич

В методических рекомендациях кратко изложены теоретические сведения, необходимые для выполнения лабораторных работ, и требования к оформлению. Составлены в соответствии с рабочей программой по дисциплине «Компьютерные технологии в приборостроении».

Учебно-методическое издание

КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ В ПРИБОРОСТРОЕНИИ

Ответственный за выпуск	С. С. Сергеев
Корректор	Т. А. Рыжикова
Компьютерная верстка	Е. В. Ковалевская

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 36 экз. Заказ №

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, г. Могилев.

Содержание

Введение

Целью преподавания дисциплины является ознакомление студентов с основными областями использования компьютерных технологий в неразрушающем контроле.

При изучении дисциплины студенты выполняют лабораторные работы, которые соответствуют темам лекций.

Варианты заданий выдаются студентам заранее с тем, чтобы они имели возможность подготовиться к выполнению лабораторной работы: просмотреть теоретический материал по теме работы и продумать алгоритмы решения задач.

Программы пишутся в среде MATLAB. Каждую программу в работающем виде (после отладки и тестирования) студент показывает преподавателю, после чего лабораторная работа подлежит защите.

К защите работы студент подготавливает отчет, включающий в себя титульный лист, формулировку задания, описание исходных данных, алгоритм решения задачи, результаты тестирования.

Защита лабораторной работы состоит из двух частей: практической и теоретической. В практической части студент объясняет алгоритмы работы представленной им программы, в теоретической – отвечает на вопросы по теме лабораторной работы.

При подготовке к защите студенту рекомендуется ответить на контрольные вопросы.

1 Лабораторная работа № 1. Знакомство с системой автоматизации решения инженерных и научных задач MATLAB

Цель работы: знакомство с системой **MATLAB**: освоение интерфейса системы; получение основных навыков по выполнению элементарных вычислений.

1.1 Теоретические сведения

Интерфейс системы MATLAB.

Окно **MATLAB** содержит меню, инструментальную панель с кнопками, реализующими наиболее часто выполняемые действия, и командное окно, в котором отображается строка запроса в виде двух знаков >>.

Стандартное выпадающее меню **File** содержит такие пункты, как New – для создания новых файлов, **Open** – открытие существующего файла-программы или файла-функции для редактирования, проверки текста или отладки, **Close Command Window** – закрытие командного окна, **Save Workspace As** – сохранение рабочей области и т. д.

Инструментальная панель командного окна позволяет выполнять требуемые действия простым нажатием на соответствующую кнопку. Большинство кнопок имеют стандартный вид и выполняют стандартные, подобные другим программам действия, – это копирование (**Copy**), открытие файла (**Open**), печать (**Print**) и т. д.

Команда **help**, набранная в ответ на запрос, завершаемая нажатием клавиши **Enter**, или кнопка инструментальной панели со знаком вопроса позволяет получить список функций, для которых доступна оперативная помощь. Команда **help** <**имя_функции**> позволяет получить на экране справку по конкретной функции.

Например, команда **help eig** позволяет получить оперативную справку по функции **eig** – функции вычисления собственных значений матрицы. С некоторыми возможностями системы **MATLAB** можно ознакомиться с помощью команды **demo**.

Основы работы.

Система представляет собой интерактивную среду для вычислений и моделирования, причем она может работать как в режиме непосредственных вычислений,так и в режиме интерпретации программ.

В командном окне после приглашения записываются команды **MATLAB**.

Например, набрав в ответ на приглашение оператор **>> y = sin(0.125)** и завершив набор нажатием клавиши **ENTER**, получаем ответ

y = 0.1247

>>

После ввода команды **MATLAB** непосредственно интерпретирует введенные инструкции, обрабатывает и выводит результат на экран.

Удобным свойством системы является возможность использовать клавиши-стрелки ↑↓ для доступа к стеку с ранее введенными командами. Таким образом, имеется возможность перевызвать ранее вызванную команду, отредактировать ее и снова выполнить.

Операторы MATLAB обычно имеют форму **переменная = выражение** или просто **выражение.**

Выражение, как правило, формируется из операторов, функций и имен переменных. После выполнения выражения генерируется матрица, которая выводится на экран и присваивается соответствующей переменной для последующего использования. Если имя переменной в левой части и знак = отсутствуют, автоматически генерируется переменная **ans** (answer – ответ), которой присваивается результат вычислений.

Команда **whos** выводит список всех переменных в рабочем пространстве. Все переменные в **MATLAB** представляют собой прямоугольные матрицы, элементы которых могут быть в общем случае комплексными числами. В приведенном примере **y** – это вырожденная матрица из одного элемента.

Для задания матриц в системе существует очень гибкая система инструментов. Например, для задания вектора, компоненты которого принимают значения от меньшего числа до большего, используется оператор « **:** **:** ». Например, **>> t=0:0.1:3.2;** задает вектор t, значения которого изменяются от 0 до 3,2 с шагом 0,1. Знак **«;»** подавляет вывод результата выполнения операции.

Оператор **x = 4:15** задает целочисленный вектор **x=[4 5 6 7 8 9 10 11 12 13 14 15]**.

Явное задание вектора (с помощью квадратных скобок) тоже допускается.

В приведенном ниже примере выполняется построение фигур Лиссажу, которые задаются параметрически:

$$\begin{cases} x = a1 \cdot \cos(\omega 1 \cdot t); \\ y = a2 \cdot \cos(\omega 2 \cdot t). \end{cases}$$

Если отношение ω1/ω2 – число рациональное, то такая кривая называется фигурой Лиссажу.

```
%Задание амплитуд и частот
>> a1=1.2; %Точка с запятой после оператора
>> a2=1.0; %обеспечивает отсутствие вывода
>> w1=1.5; %результата действия оператора
>> w2=1.0;
%Оператор whos позволяет в любой момент получить
%полную информацию о всех активных переменных.
%Переменные, введенные в данном сеансе, могут быть
%удалены с помощью оператора clear.

>>whos
Name Size Elements Bytes Density Complexa1 1 by 1 1 8 Full No
a2 1 by 1 1 8 Full No
w1 1 by 1 1 8 Full No
w2 1 by 1 1 8 Full No
Grand total is 4 elements using 32 bytes
```

```
>> t=0:0.1:3.2;
>> x=a1*cos(w1*t);
>> y=a2*cos(w2*t);
>> plot(x,y);
```

В системе **MATLAB** часть строки, следующая за знаком %, является комментарием.

Заканчивается пример вызовом команды **plot**, вызывающей окно построения графиков. Точки с координатами **(x(i), y(i)), (x(i+1), y(i+1))** соединяются отрезками прямых. При этом происходит автоматический подбор удобного масштаба. Масштабные метки и числа при них изображаются на границах рамки, а изображение осей координат не предусмотрено. Пример кривой Лиссажу представлен на рисунке 1.1.

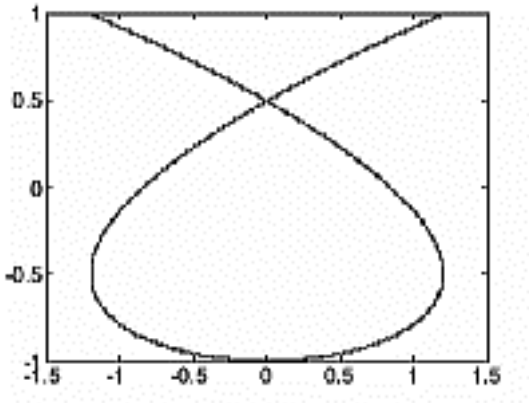


Рисунок 1.1 – Пример кривой Лиссажу

1.2 Практическое задание

Необходимо построить график функции. Вариант получить у преподавателя.

Контрольные вопросы

1 Что такое **MATLAB**?
2 Какие окна имеет интерфейс **MATLAB** и каково их назначение?
3 Какова структура окна Command Windows?
4 Как очистить окно Command Windows?
5 Как в **MATLAB** осуществляется ввод и выполнение команд?
6 Как удалить неиспользуемую более переменную из рабочего пространства MATLAB?
7 Как получить справку по выбранной команде **MATLAB**?

2 Лабораторная работа № 2. Использование m-файлов в системе MATLAB

Цель работы: получение основных навыков создания и использования m-файлов системы **MATLAB**; знакомство с языком программирования **MATLAB**.

2.1 Теоретические сведения

MATLAB может выполнять последовательность операторов, записанных в файл на диске. Такие файлы называются *m-файлами*, поэтому что имена этих файлов имеют вид **<имя>.m**. Большая часть работы в **MATLAB** будет состоять в создании, редактировании и выполнении таких m-файлов. Имеется два типа m-файлов: *файлы-программы*, или сценарии, и *файлы-функции*.

Файлы-программы содержат последовательность обычных операторов. Если файл с такой последовательностью имеет имя, например, **example.m**, то его имя (**example**), введенное в командной строке, вызовет выполнение последовательности операторов. Причем переменные в такой программе являются глобальными. Например, программы часто используются для ввода данных в большие матрицы.

Для того чтобы создать новый m-файл, необходимо в командной строке набрать **edit**, если нужно указать имя m-файла, то, например, пишется **edit example**. Файлы-программы состоят из последовательности обычных операторов **MATLAB**. Для того чтобы такая программа выполнилась, необходимо ввести **example** (то есть имя m-файла) в командной строке. Из такой программы также можно вызывать другие m-файлы, созданные разработчиком либо пользователем.

Наиболее простой способ выяснить синтаксис встроенных функций – это воспользоваться командой **help**. При использовании команды **help** без аргументов на экран будет выводиться список каталогов, которые имеются в системе с кратким описанием их содержимого. Если ввести команду **help** с именем каталога, например, **help elmat**, то результатом будет список математических функций. Ввод команды с именем определенной функции выдаст на экран описание данной функции. Следует обратить особое внимание на то, что в качестве ответа на запрос о помощи выводятся все строки комментариев, которые описаны в начале каждой функции как созданной разработчиками системы, так и собственными функциями пользователя.

Пример выполнения работы

В приведенном примере выполняется построение графика для двух различных функций, которые выбираются исходя из условия:

$$y = \cos^3 x^2 + \sin^2 x^3,$$

$$\begin{cases} 12 \cdot x + y, y > 2,6; \\ x \cdot e^y + \sin y \cdot \cos x \cdot \cos \pi y, y \le 2,6. \end{cases}$$

Переменная *x* изменяется от 0,9 до 1,8 через 0,05.

Для вычисления математических выражений используются обычные арифметические операции:

+ сложение;

– вычитание;

/ деление;

^ степень;

() определение порядка вычисления.

Особое внимание следует обратить на то, что **MATLAB** предназначен для работы с матрицами, поэтому написание **z=x*y** приведет к матричному перемножению, в то время как для данной задачи необходимы операции непосредственно с элементами матриц. Для этой цели перед арифметическими операциями добавляется **.** , например **z=x.*y**, в этом случае перемножение матриц будет осуществляться поэлементно.

Так как необходимо сравнить каждый элемент полученной матрицы **y** с числом 2,6, то используется возможность обращения к конкретным элементам массива с помощью индексирования. В этом случае обращение к элементам массива будет выглядеть следующим образом: **имя массива (номер элемента)**. Например, для того чтобы обратиться к пятому элементу массива **y**, необходимо записать **y(5)**; к элементам массива можно также обращаться через переменные (**y(i)**, где **i –** переменная).

Для того чтобы содержимое индекса менялось, используется цикл **for**. Синтаксис оператора **for** выглядит следующим образом:

for *переменная = начальное значение*: *шаг*: *конечное значение*

тело цикла;

end;

В тело цикла поместим оператор условия, синтаксис которого представлен далее:

if *условие*

выражение

else

выражение

end;

В результате m-файл строящегося графика по заданному выражению будет выглядеть следующим образом:

```
x=0.9:0.05:1.8;
y=(cos(x).^3).*(x.^2)+(sin(x).^2).*(x.^3);
for i=1:19
if (y(i)>2.6)
z(i)=12*x(i)+y(i);
else
z(i)=x(i)*exp(y(i))+sin(y(i))*cos(x(i))*cos(pi*x(i));
  end;
end;
plot(x,z);
```

В результате выполнения был получен график, представленный на рисунке 2.1.

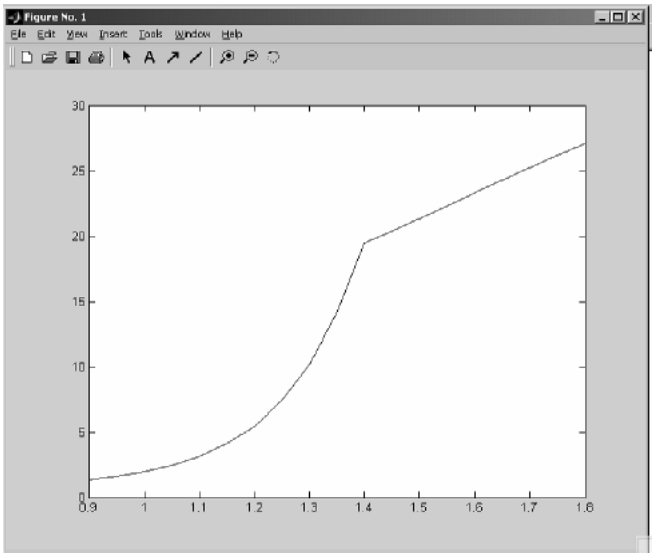


Рисунок 2.1 – Пример полученной кривой

2.2 Практическое задание

Необходимо построить график функции. Решение оформить в виде m-файла. Вариант получить у преподавателя.

Контрольные вопросы

1 Для чего служит редактор m-файлов?
2 Как вызвать редактор m-файлов и какие отладочные средства он имеет?
3 Как осуществляются цветовые выделения и синтаксический контроль при работе с пакетом **MATLAB**?
4 Как можно получить быструю справку по любой команде **MATLAB**?
5 Расскажите о **MATLAB** в роли суперкалькулятора.
6 Что принято называть сессией при работе с **MATLAB**?
7 Как изображается и для чего служит комментарий?
8 Что такое интерактивный режим?
9 Перечислите основные объекты **MATLAB**.

3 Лабораторная работа № 3. Матричные вычисления в MATLAB

Цель работы: получение основных навыков работы с матрицами в **MATLAB**.

3.1 Общие сведения

Название пакета **MATLAB** расшифровывается как **Matrix Laboratory**, т. е. первоначально он был разработан именно как пакет, реализующий математические методы матричной алгебры. Все переменные в **MATLAB** представляют собой матрицы. Матрицы могут быть заданы различными способами:

– введены явно с помощью списка элементов;
– сгенерированы встроенными операторами или функциями;
– созданы в m-файлах;
– загружены из внешнего файла данных.

Явное определение матриц выглядит следующим образом:

X=[1 2 3; 4 5 6; 7 8 9]

или

X=[1 2 3
4 5 6
7 8 9]

В результате выполнения данных команд создается матрица размерами 3 × 3 и ее значение присваивается переменной X. Ввод больших матриц обычно выполняется с помощью m-файлов, т. к. в них легче находить и исправлять ошибки.

В системе **MATLAB** существуют встроенные функции для создания матриц:

zeros – создает матрицу заполненную нулями;

ones – создает матрицу заполненную единицами;

eye – создает нулевую матрицу с диагональю, заполненную единицами;

rand – создает матрицу, заполненную случайными числами с равномерным распределением;

randn – создает матрицу заполненную случайными числами с нормальным распределением.

Векторы и подматрицы используются в системе **MATLAB** для получения компактной записи алгоритмов сложной обработки данных. Оператор двоеточие может быть использован для доступа к подматрицам. Например, **A(1:4, 3)** является вектор-столбцом, состоящим из четырех первых элементов третьего столбца матрицы **A**. Двоеточие само по себе обозначает всю строку или весь столбец. Например **A(:,5)** является пятым столбцом матрицы **A**, а **A(1:3,:)** представляет собой три строки матрицы.

В общем случае, выражение вида **начальное_значение: шаг: конечное_значение** позволяет получить последовательность чисел. Если **шаг** не задан, то он принимает значение, равное 1.

Выражение **1:4** фактически задает вектор-строку **[1 2 3 4]**.

Для работы с матрицами в системе **MATLAB** существует множество функций. Например:

det – определитель матрицы;

trace – суммирует диагональные элементы;

rank – вычисляет ранг матрицы.

Не следует забывать, что и все арифметические операторы (+, –, *, /, ^) являются матричными. Так, например, решением системы линейных алгебраических уравнений, заданной в матричной форме **X*A = B**, является столбец **X = B / A**. Существуют также специфично матричные операторы, например, транспонирование – ‘ , или обратное деление – \. Обратное деление позволяет найти решение матричного уравнения **A*X = B** как **X = A\B**.

Пример выполнения задания

По заданию необходимо решить систему линейных уравнений методом Крамера:

$$\begin{cases} -41{,}13 \cdot x1 - 78{,}41 \cdot x2 - 97{,}96 \cdot x3 - 2{,}88 \cdot x4 = -711{,}29; \\ 80{,}84 \cdot x1 + 12{,}03 \cdot x2 + 35{,}56 \cdot x3 + 64{,}97 \cdot x4 = 348{,}27; \\ -10{,}68 \cdot x1 + 39{,}43 \cdot x2 - 7{,}93 \cdot x3 - 83{,}21 \cdot x4 = 53{,}43; \\ 52{,}93 \cdot x1 + 72{,}18 \cdot x3 - 10{,}74 \cdot x3 - 30{,}46 \cdot x4 = 266{,}30. \end{cases}$$

Метод Крамера состоит в следующем. Сначала рассчитывается определитель матрицы коэффициентов системы линейных уравнений. Затем составляются дополнительные матрицы из исходной путем замены столбца коэффициентов, соответствующего одной из переменных, на столбец свободных членов. Рассчитываются определители каждой из дополнительных матриц. Корни находятся как частное от определителя каждой дополнительной матрицы и определителя исходной матрицы коэффициентов.

m-файл для расчета системы линейных уравнений методом Крамера будет выглядеть следующим образом:

```
A=[-41.14 -78.41 -97.96 -2.88; 80.84 12.03 35.56 64.97; -10.68 39.43 -7.93 -83.21; 52.93 72.18 -10.74 -30.46];
B=[-711.29;348.27;53.43;266.30];
dA=det(A);
A1=[B A(:,2) A(:,3) A(:,4)];
A2=[A(:,1) B A(:,3) A(:,4)];
A3=[A(:,1) A(:,2) B A(:,4)];
A4=[A(:,1) A(:,2) A(:,3) B];
X1=det(A1)/dA
X2=det(A2)/dA
X3=det(A3)/dA
X4=det(A4)/dA
```

Результатом выполнения данного m-файла является нахождение корней данной системы линейных уравнений:

```
X1 =
2.0272

X2 =
2.8388

X3 =
4.1360

X3 =
0.0488
```

3.2 Практическое задание

Необходимо найти корни системы линейных уравнений методом Крамера, используя операторы матричной алгебры. Вариант получить у преподавателя.

Контрольные вопросы

1 Как в **MATLAB** осуществляются операции с матрицами?

2 Как в **MATLAB** осуществляется вычисление элементарных функций для векторов и матриц?

3 Перечислите основные арифметические операторы и их синтаксис в системе **MATLAB**.

4 Что такое встроенная функция? Перечислите алгебраические и арифметические встроенные функции в **MATLAB**.

4 Лабораторная работа № 4. Разработка графического интерфейса пользователя

Цель работы: получение основных навыков по разработке графического интерфейса пользователя.

4.1 Общие сведения

При разработке прикладных программ представляется полезным создание графического интерфейса пользователя. Фактически это создание среды расчета задач определенного класса без программирования со стороны пользователя. Обычно такие интерфейсы имеет смысл разрабатывать для задач с несколькими параметрами, где предполагается их неоднократное решение. В таком случае целесообразно разработать графический интерфейс, помогающий пользователю получать результаты решения задачипри определенном выборе параметров.

Графический интерфейс для решения научных или учебных задач должен включать:

– одно или несколько окон для вывода графического результата;

– несколько редактируемых окон, с помощью которых задаются или изменяются значения параметров решаемой задачи;

– управляющие кнопки, которые позволяют запускать, останавливать процесс расчета и производить выход из задачи;

– поясняющие надписи.

Для вызова графического редактора необходимо в командном окне **MATLAB** набрать команду **guide**. В результате выполнения данной команды выдается следующее меню рисунок 4.1.

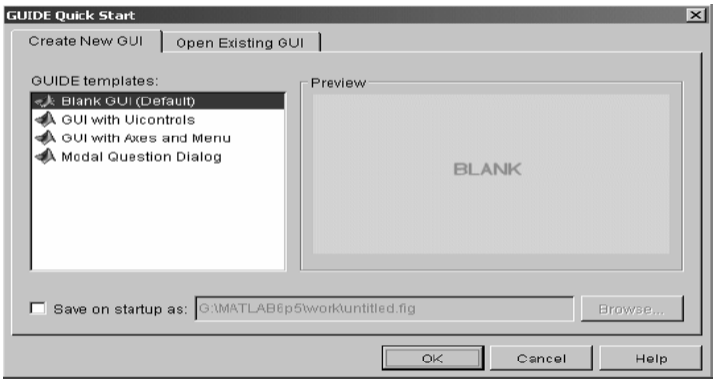


Рисунок 4.1 – Меню

Для создания чистого нового графического интерфейса выбираем **Blank GUI (Default)**. В результате на экране отображается следующее окно конструктора (рисунок 4.2).

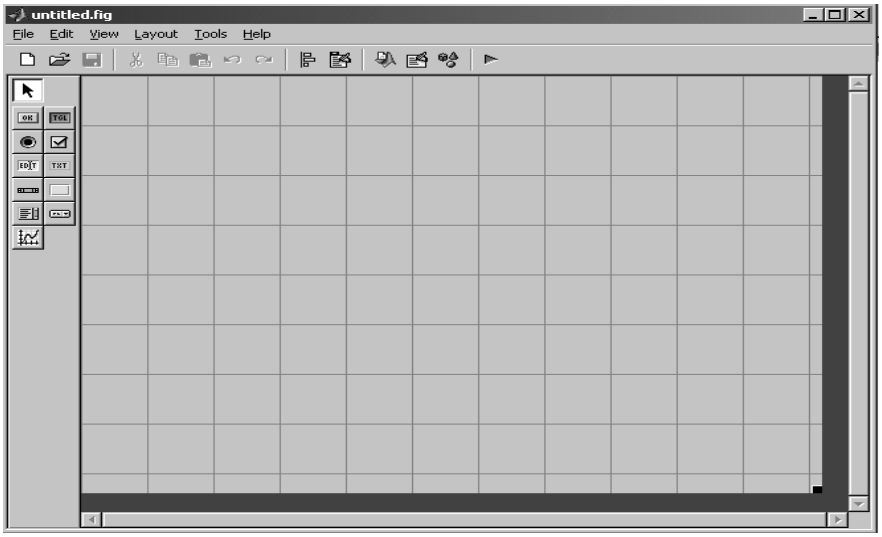


Рисунок 4.2 – Окно инструктора

Для создания подокон, в которые будут выводиться графики, используется кнопка  , показанная слева на рисунке 4.2.

Щелкнув по этому элементу на панели управления и переведя мышь на панельрисунка, следует поместить указатель в то место, где должен находиться левый верхний угол первого подокна. Нажав и удерживая левую кнопку мыши, необходимо вытянуть получающийся прямоугольник до нужных размеров. После этого процедура построения двух других окон повторяется аналогично.

Для создания редактируемых окон ввода используется кнопка **edit**  **.** Используется она так же, как при создании подокон с осями. Сначала появляется мышь, нагруженная крестиком, с помощью которой строится прямоугольник ввода.

Надписи на панели рисунка создаются с помощью кнопки **txt**  , которая переносится и выравнивается аналогично вышеописанному. Для того чтобы внутри области статического текста появилась какая-либо надпись, необходима работа с редактором свойств, который вызывается либо при помощи кнопки **Property editor**, либо с помощью двойного нажатия левой кнопкой мыши на соответствующем объекте на панели рисунка.

Для создания и размещения кнопок используется панель с надписью **Ок**  .

Способ размещения кнопки и выбора ее размера полностью совпадает с методом,описанным выше для окна редактирования и окна статического текста.

Построенные таким образом окна вывода и редактирования, окна статического текста и кнопки, а также другие объекты можно выровнять и установить определенные промежутки между ними с помощью панели выравнивания **Alignment Tools**  . Для этого необходимо на панели управления щелкнуть по соответствующей кнопке, и появится панель выравнивания. Для задания ряда объектов, с которыми будут выполняться какие-либо действия, необходимо их выделить, щелкая по каждому из них при нажатой клавише **Shift**. Выделенные

объекты отмечаются черными точками вокруг соответствующих объектов. При необходимости изменить размер какого-либо объекта (кнопки, окна и т. д.) следует щелкнуть по этому объекту с помощью левой кнопки мыши и с помощью мыши изменить требуемый размер так же, как и размер любого окна **Windows**.

Для размещения надписей на кнопках и в области статического текста необходимовыделить соответствующий объект (либо двойным щелчком прямо в области рисунка, либо в верхнем окне редактора свойств), в нижнем окне редактора свойств найти свойство **String** и после его выделения вписать между кавычками требуемый текст (например, ‘Пуск’ на соответствующей кнопке). Для задания надписей над каждым из окон вывода необходимо выделить соответствующее окно и вызвать редактор свойств, в нижнем окне которого надо найти свойство **Title**.

Поскольку это свойство само является объектом, то, выполнив двойной щелчок на этом свойстве, раскрывают его свойства. После этого следует найти свойство **String**, задать его значение (соответствующий текст, например, **X=**) в среднем окне и, найдя свойство **FontUnits**, задать его значение **normalized**.

Следует еще отметить свойство **Tag**, которое имеют все объекты. Особенно важно знать и/или задать это свойство для тех объектов, к которым потом в программе придется обращаться. Например, если на рисунке задано несколько окон вывода, то для вывода графика в требуемое окно проще всего его будет идентифицировать окно с помощью свойства **Tag**. Это свойство, как правило, имеет по умолчанию вполне осмысленное значение (**Axes1**, например), но при желании его можно задать так, как нравится.

Пример выполнения задания

Для примера воспользуемся m-файлом, разработанным для лабораторной работы № 2. Напишем для него графический интерфейс. Интерфейс для такой программы должен содержать кнопки **пуск** и **выход,** полей для ввода начального и конечного значения **X**с подписями, а также окна для вывода графической информации. В результате получим следующий графический интерфейс рисунок 4.3.

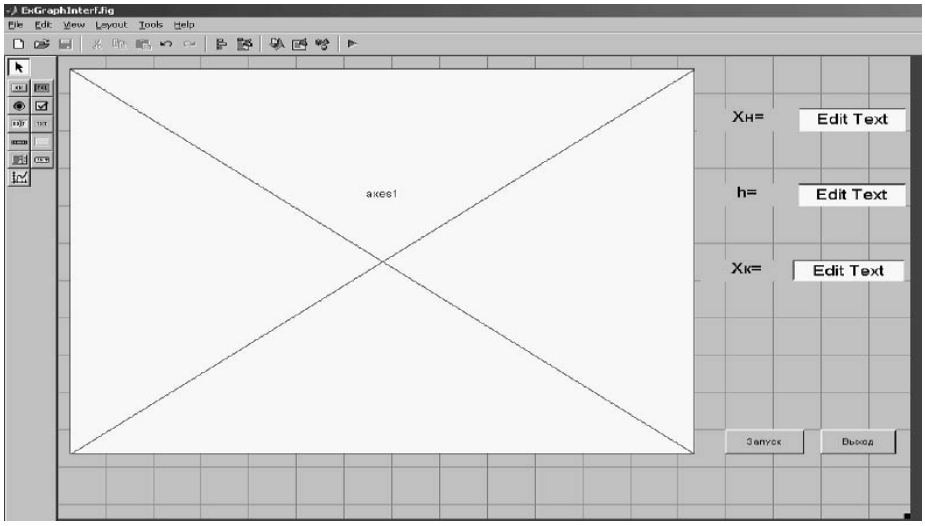


Рисунок 4.3 – Графический интерфейс

Затем сохраняем данный файл под желаемым именем, например **ExGraphInterf.**

В редакторе открывается m-файл, содержащий шаблон с описанием функций, вызываемых при работе графического интерфейса. Для придания требуемой функциональности, следует заполнить этот шаблон.

Например, чтобы при запуске программы в окнах ввода появлялись значения поумолчанию, следует дополнить функцию ExGraphInterf_OpeningFcn:

```
function ExGraphInterf_OpeningFcn(hObject, eventdata, handles, varargin)
…
handles.output = hObject;
guidata(hObject, handles);
set(handles.edit1,'string','0.8')
set(handles.edit2,'string','0.05')
set(handles.edit3,'string','1.8')
```

Кроме этого необходимо обозначить реакцию на нажатие кнопок (функции **pushbutton1_Callback и pushbutton2_Callback)**:

```
…
function pushbutton1_Callback(hObject, eventdata, handles)
xn=str2double(get(handles.edit1,'string'))
h=str2double(get(handles.edit2,'string'))
xk=str2double(get(handles.edit3,'string'))
x=xn:h:xk
y=(cos(x).^3).*(x.^2)+(sin(x).^2).*(x.^3);
for i=1:((xk-xn)/h)+1
if (y(i)>2.6)
z(i)=12*x(i)+y(i);
else
z(i)=x(i)*exp(y(i))+sin(y(i))*cos(x(i))*cos(pi*x(i));
end;
end;
axes(handles.axes1);plot(x,z);

…
function pushbutton2_Callback(hObject, eventdata, handles)
delete(ExGraphInterf);
…
```

Обратим внимание на выделенные области, как раз они и подвергаются модификации для того, чтобы графический интерфейс работал с определенной информацией. Разберем каждую из команд. Команда **set(handles.edit1,'string','0.8')** передает данные в объект edit1 в результате числовое значение 0,8 отображается в окне; **xn=str2double(get(handles.edit1,'string'))** – получает данные в текстовом виде из объекта edit1, преобразует их в численный формат и присваивает переменной **xn,** таким же образом задаются переменные **h** и **xk**. В строке **x=xn:h:xk** формируем вектор **x** из полученных **xn**, **h** и **xk**. Остальную часть функции берем из лабораторной работы № 2. Для того чтобы вывести полученный график в

окне, необходимо задать для этого окна оси координат. Они задаются с помощью команды **axes(handles.axes1);**,где **axes1** является именем объекта (окна), в котором производится вывод графика. В результате запускарасчета получается следующий результат (рисунок 4.4).

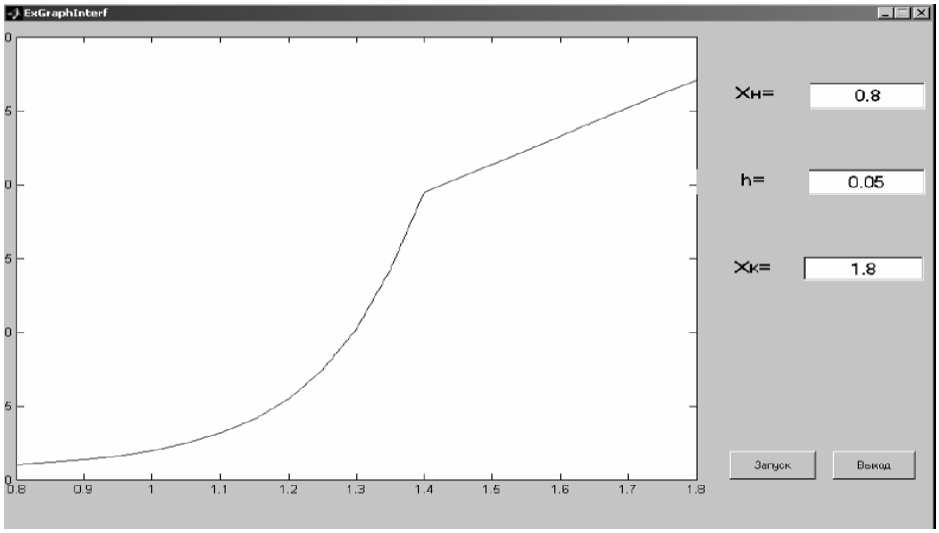


Рисунок 4.4 – Вывод графика

4.2 Практическое задание

Необходимо разработать графический интерфейс для задания из лабораторной работы № 2.

Контрольные вопросы

1 Как создать новый графический интерфейс?
2 Как настроить графический интерфейс?
3 Какие классы существуют в графическом интерфейсе и за что они отвечают?

5 Лабораторная работа № 5. Решение дифференциальных уравнений в системе MATLAB

Цель работы: получение основных навыков решения дифференциальных уравнений в **MATLAB** на примере расчета параметров электрических цепей.

5.1 Теоретические сведения

Решение обыкновенных дифференциальных уравнений.

Анализ поведения многих систем и устройств в динамике обычно базируется на решении систем обыкновенных дифференциальных уравнений (ОДУ). Их, как правило, представляют в виде системы из дифференциальных уравнений первого порядка в форме Коши

$$\frac{dy}{dt} = y',$$

с граничными условиями **y**(t_0) =**b**, где t_0 – начальная точка интервала. Параметр *t* необязательно означает время, хотя чаще всего решение дифференциальных уравнений ищется во временной области. Вектор **b** задает начальные условия.

Решатели ОДУ.

Для решения систем ОДУ в **MATLAB** реализованы различные численные методы.

Их реализации названы решателями ОДУ.

Обобщенное название **solver** (решатель) означает один из возможных численных методов решения ОДУ: **ode45**, **ode23**, о**de113**, **ode15s**, **ode23s**, **ode23t**, **ode23tb**, **bvp4c** или **pdepe**.

Решатели реализуют следующие методы решения систем дифференциальных уравнений, причем для решения жестких систем уравнений рекомендуется использовать только специальные решатели **ode15s**, **ode23s**, **ode23t**, **ode23tb**:

ode45 – одношаговые явные методы Рунге – Кутта 4-го и 5-го порядка. Это классический метод, рекомендуемый для начальной пробы решения. Во многих случаях он дает хорошие результаты;

ode23 – одношаговые явные методы Рунге – Кутта 2-го и 4-го порядка. При умеренной жесткости системы ОДУ и низких требованиях к точности этот метод может дать выигрыш в скорости решения;

ode113 – многошаговый метод Адамса – Башворта – Мултона переменного порядка. Это адаптивный метод, который может обеспечить высокую точность решения;

ode23tb – неявный метод Рунге – Кутта в начале решения и метод, использующий формулы обратного дифференцирования 2-го порядка в последующем. Несмотря на сравнительно низкую точность, этот метод может оказаться более эффективным, чем ode15s;

ode15s – многошаговый метод переменного порядка (от 1 до 5, по умолчанию 5), использующий формулы численного дифференцирования. Это адаптивный метод, его следует применять, если решатель ode45 не обеспечивает решения;

ode23s – одношаговый метод, использующий модифицированную формулу Розенброка 2-го порядка. Может обеспечить высокую скорость вычислений при низкой точности решения жесткой системы дифференциальных уравнений;

ode23t – метод трапеций с интерполяцией. Этот метод дает хорошие результаты при решении задач, описывающих колебательные системы с почти гармоническим выходным сигналом;

bvp4c – служит для проблемы граничных значений систем дифференциальных уравнений вида y '== f(t,y), F(y(a), y(b), p) = 0 (краевая задача);

pdepe – нужен для решения систем параболических и эллиптических дифференциальных уравнений в частных производных, введен в ядро системы для поддержки новых графических функций OpenGL, пакет расширения Partial Differential Equations Toolbox содержит более мощные средства.

Все решатели могут решать системы уравнений явного вида y' = F(£, y), **ode15s** – жестких дифференциальных и дифференциально-алгебраических уравнений, **ode23t** – умеренно жестких дифференциальных и дифференциально-алгебраических уравнений.

Использование решателей систем ОДУ.

В описанных далее функциях для решения систем дифференциальных уравнений приняты следующие обозначения и правила:

options – аргумент, создаваемый функцией **odeset** (еще одна функция – **odeget** или **bvpget** (только для **bvp4c**) позволяет вывести параметры, установленные по умолчанию или с помощью функции **odeset** /**bvpset**);

tspan – вектор, определяющий интервал интегрирования $[t_0\ t_{final}]$. Для получения решений в конкретные моменты времени t0, tl,..., tfinal (расположенные в порядке уменьшения или увеличения) нужно использовать **tspan** = $[t_0\ t_l\ ...\ t_{final}]$;

y0 – вектор начальных условий;

p1, p2,… – произвольные параметры, передаваемые в функцию F;

T, Y – матрица решений Y, где каждая строка соответствует времени, возвращенном в векторе-столбце T.

Описание функций для решения систем дифференциальных уравнений:

[T Y] = solver(@F,tspan,y0), где вместо solver подставляют имя конкретного решателя; интегрирует систему дифференциальных уравнений вида y'=F(t,y) на интервале tspan с начальными условиями y0. @F – дескриптор ODE-функции. Каждая строка в массиве решений Y соответствует значению времени, возвращаемому в векторе-столбце T;

[T Y] = solver(@F, tspan, y0. options) дает решение, подобное описанному выше, но с параметрами, определяемыми значениями аргумента options, созданного функцией odeset. Обычно используемые параметры включают допустимое значение относительной погрешности RelTol (по умолчанию le-3) и вектор допустимых значений абсолютной погрешности AbsTol (все компоненты по умолчанию равны 1e-6);

[T Y] = solver(@F,tspan,y0,options,pl,p2...) дает решение, подобное описанному выше, передавая дополнительные параметры pi, p2,... в m-файл F всякий раз, когда он вызывается. Следует использовать options=[], если никакие параметры не задаются.

Пример выполнения практического задания

В качестве примера рассмотрим простейшую электрическую цепь (рисунок 5.1).

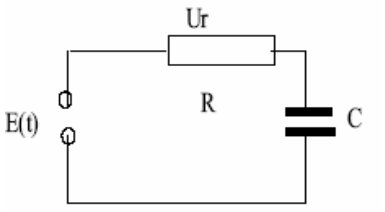


Рисунок 5.1 – Электрическая схема RC-цепочки

Поведение этой цепи описывается системой двух уравнений:

$$\begin{cases} i(t) = C\dfrac{du_c(t)}{dt}; \\ E(t) = i(t)\cdot R + u_c(t). \end{cases}$$

Запишем это решение в форме задачи Коши:

$$\frac{du_c(t)}{dt} = \frac{1}{R\cdot C}\left(E(t) - u_c(t)\right);$$
$$u_c(0) = U_0.$$

Для решения задачи применим простейший метод Эйлера первого порядка, покоторому для решения задачи в форме (1) применяется следующий алгоритм:

$$y_i = y_{i-1} + f\left(t_{i-1}, y_{i-1}\right)\cdot h,$$

где h – шаг решения.

Реализация этого метода может быть следующей:

```
clear;
%Задание параметров цепи
C=0.1;
R=1;
t0=0;
U0=0;
h=0.01;
%Начальные значения
t(1)=t0;
U(1)=U0;
t=0:h:2;
%Решение ОДУ
for k=2:length(t)
U(k)=U(k-1)+dudt(t(k-1),U(k-1), R, C).*h;
end;
plot(t,U);
```

В отдельных m-файлах необходимо разместить описание правой части уравнения и формы сигнала источника E(t).

```
%Правая часть ОДУ
function res = dudt(t,u, R, C)
 res=(E(t)-u)./R./C;
%Сигнал E(t) - ступенчатая функция
function res=E(t)
if t < 1
res=0;
 else
```

```
res=1;
end;
```

Для приведенного примера решение выглядит следующим образом (рисунок 5.2).

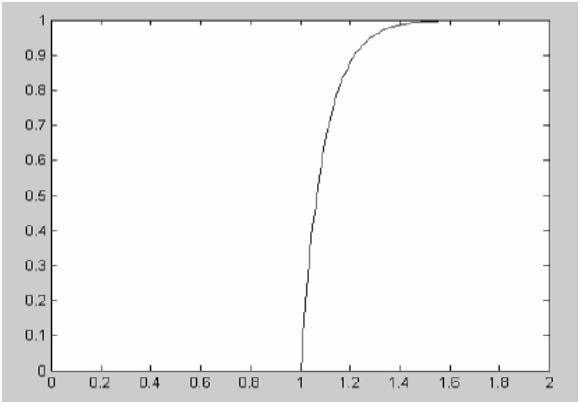


Рисунок 5.2 – Решение ОДУ

При помощи решателей **MATLAB** данную задачу можно решить следующим образом.

Сначала необходимо создать m-файл с описанием функции правых частей уравнения, руководствуясь следующими правилами:

– функция описания правых частей должна содержать не менее двух входных аргументов t и y, даже если какой-то из них не используется явно при вычислении правых частей;

– правые части системы, которые вычисляются этой функцией F(t, y), должны образовывать вектор-столбец;

– любые дополнительные параметры, которые необходимо передавать функции F(t, y), должны быть в конце списка параметров самой функции и в списке аргументов вызова решателя после специального параметра flag.

В данном случае изменений m-файла dudt.m не требуется и вызов решателя с выводом результатов в виде графика выглядит следующим образом:

```
[T,U] = ode23(@dudt, [0 2], 0, [], 1, 0.1);
plot(T,U);
```

5.2 Практическое задание

Провести расчет электрической схемы в динамике методом Эйлера 1-го порядка и с помощью решателей **MATLAB**. Задание получить у преподавателя.

Контрольные вопросы

1 Как записываются системы дифференциальных уравнений?
2 Как задать начальные условия?
3 Как задать временный интервал?
4 Как получить результат для дальнейшего использования?

6 Лабораторная работа № 6. Решение нелинейных дифференциальных уравнений в системе MATLAB

Цель работы: получение основных навыков решения дифференциальных уравнений в MATLAB на примере расчета параметров электрических цепей.

6.1 Общие сведения

Решение нелинейных алгебраических уравнений.

Для решения нелинейных алгебраических уравнений (поиска нулей нелинейных функций) в **MATLAB** используется функция **FZERO**.

Вызов функции осуществляется следующим образом:

X = FZERO(FUN,X0) или
X = FZERO(FUN,X0,OPTIONS,P1,P2,...)

где **FUN** – функция описывающая левую часть уравнения;
X0 – начальная точка поиска или диапазон поиска;
OPTIONS – необязательные параметры ([ ] – если не используются);
P1, P2,... – дополнительные параметры, передаваемые в функцию **FUN**.

Расчет динамических нелинейных электрических цепей.

Рассмотрим электрическую цепь, представленную на рисунке 6.1.

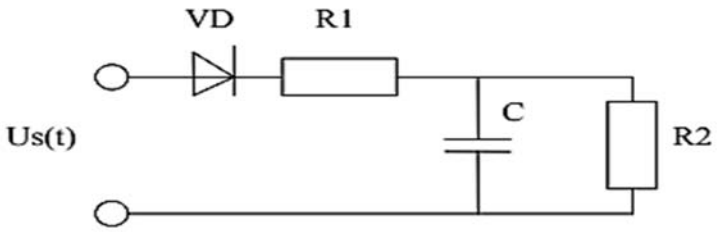


Рисунок 6.1 – Электрическая цепь

Введение нелинейного элемента диода затрудняет расчет таких цепей. Особенно сложен динамический анализ таких цепей.

Такая цепь описывается системой уравнений

$$i_{VD}(t) = C\frac{du_c(t)}{dt} + \frac{u_c(t)}{R2};$$
$$U_s(t) = u_{VD}(t) + i_{VD}(t)\cdot R1 + u_c(t).$$

Ток через диод приближенно описывается следующим выражением:

$$I_{VD} = I_0\left(e^{40\cdot U_{VD}} - 1\right),$$

где I_0 – тепловой ток.

Запишем систему в форме задачи Коши:

$$\frac{du_c(t)}{dt} = \frac{1}{R1 \cdot C}\left(U_s(t) - u_{VD}(t) - u_c(t) \cdot \left(1 + \frac{R1}{R2}\right)\right).$$

Напряжение на диоде в каждый момент времени можно найти исходя из предположения, что за момент времени, равный шагу интегрирования, оно существенно не изменится. Для этого необходимо решить нелинейное уравнение

$$U_s(t) = u_{VD}(t) + I_0\left(e^{40 \cdot U_{VD}(t)} - 1\right) \cdot R1 + u_c(t)$$

в каждом шаге интегрирования.

Реализация может быть следующей. Необходимо в m-файле оформить описание правой части ОДУ:

```
%Описание правой части ОДУ
function res=ducdt(t,uc, R1, R2, C)
%Вычисление значения напряжения на диоде
uvd=fzero(@uvdeq,[-15 15], [ ], R1, uc, us(t));
res=1/(R1*C)*(us(t)-uvd-uc*(1+R1/R2));
%Левая часть нелинейного уравнения
function res=uvdeq(Uvd, R1, Uc, Us)
res=Us-Uvd-1e-6*(exp(40*Uvd)-1)*R1-Uc;

%Форма сигнала Us
function res=us(t)
res=sin(2*pi*50*t);
```

Далее вызов решателя имеет следующий вид:

```
[T,U] = ode23(@ducdt, [0 0.15], 0, [], 1000, 10000, 10e-6);
plot(T,U)
```

Расчет проведен со следующими параметрами: t = 0 0,15 с; Uc(0) = 0 В; R1 = 1000 Ом; R2 = 10000 Ом; C = 10мкФ.

Результаты расчета приведены на рисунке 6.2.

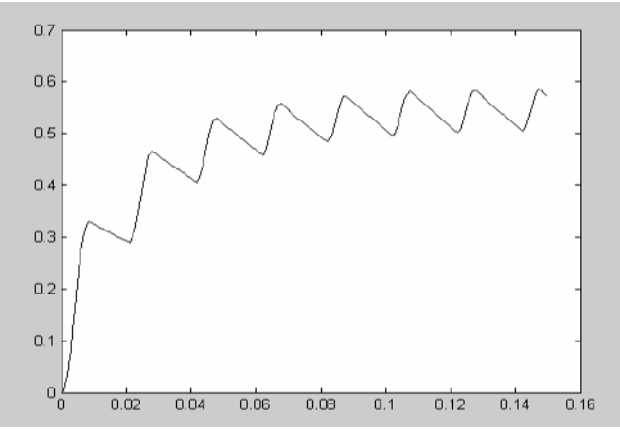


Рисунок 6.2 – Результаты расчетов

6.2 Практическое задание

Провести расчет электрической схемы в динамике. Задание получить упреподавателя.

Контрольные вопросы

1 Как записываются системы дифференциальных уравнений?
2 Как задать начальные условия?
3 Как задать временный интервал?
4 Как получить результат для дальнейшего использования?

7 Лабораторная работа № 7. Знакомство с набором инструментов PDE Toolbox для решения дифференциальных уравнений в частных производных в среде MATLAB

Цель работы: получение основных навыков решения дифференциальных уравнений в частныхпроизводных в среде **MATLAB** на примере решения задач теплопереноса и магнитостатики.

7.1 Общие сведения

Partial Differential Equation (PDE) Toolbox содержит средства для исследования и решения нестационарных дифференциальных уравнений второго порядка в частных производных. В пакете используется метод конечных элементов. Команды и графический интерфейс пакета могут быть использованы для математического моделирования PDE применительно к широкому классу инженерных и научных приложений, включая задачи сопротивления материалов, расчеты электромагнитных устройств, задачи тепломассопереноса и диффузии.

Большинство физических процессов, происходящих в природе и технике, описывается дифференциальными уравнениями. Различают обыкновенные дифференциальные уравнения (ОДУ, ODE) и дифференциальные уравнения в частных производных (PDE), которые иначе называются уравнениями математической физики. Первые, как правило, описывают процессы в устройствах и системах с сосредоточеннымипараметрами. Вторые описывают процессы в системах с пространственно-распределенными параметрами. Важнейшим частным случаем таких систем являются физические поля.

Основные свойства PDE **Toolbox MATLAB**:

– полноценный графический интерфейс для обработки PDE второго порядка;
– автоматический и адаптивный выбор конечно-элементной сетки;
– задание граничных условий: Дирихле, Неймана и смешанные;
– гибкая постановка задачи с использованием синтаксиса языка **MATLAB**;
– полностью автоматическое сеточное разбиение и выбор величины конечных элементов;

– нелинейные и адаптивные расчетные схемы;

– возможность визуализации полученного в ходе решения PDE распределения требуемых физических величин, демонстрация принятого разбиения и анимационные эффекты.

Для решения PDE задачи всю расчетную область представляют в виде совокупности неперекрывающихся геометрических фигур достаточно простой формы. Размеры таких фигур, как правило, бывают малы по сравнению с размерами расчетной области. Эти элементарные фигуры называют конечными элементами. Трехмерные расчетные области обычно разбивают на многогранники, а двумерные – на многоугольники. Простейшие многогранники (прямолинейные четырехузловые тетраэдры) и простейшие многоугольники (прямолинейные трехузловые треугольники) называют симплекс-элементами. Вся совокупность конечных элементов в расчетной области называется конечно-элементной сеткой. Вершины этих многогранников или многоугольников называют узлами конечно-элементной сетки.

Для запуска **PDE Toolbox** в окне **MATLAB** необходимо ввести команду **pdetool**.

После запуска приложения на экране появится окно, изображенное на рисунке 7.1. Как видно, окно приложения состоит из главного меню (первая строка), панели инструментов(вторая строка), строки ввода **Set formula**, объекта axes для отображения геометрии расчетной области, информационной строки Info и кнопки закрытия приложения Exit.

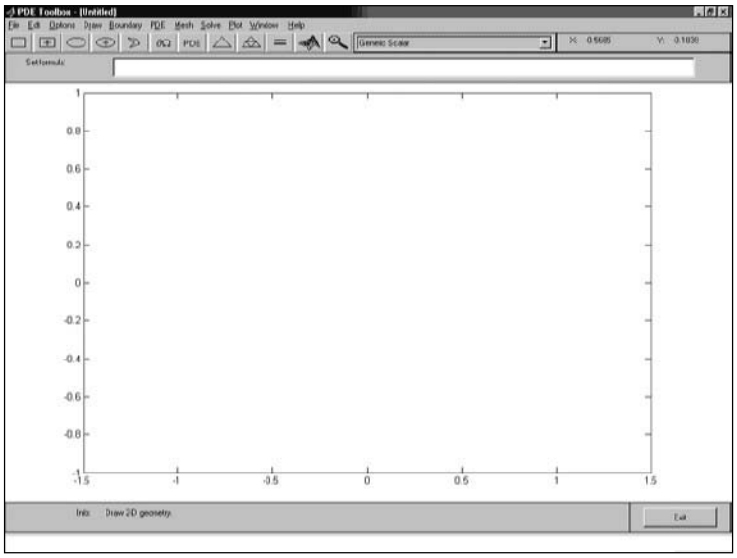


Рисунок 7.1 – Окно программы PDE Toolbox

Команды меню объединены в группы, которые представлены в виде пунктов в главном меню **PDETool**.

Группа команд **File** содержит файловые команды (открытие, сохранение и печать файлов) и команды общего назначения (создание новой модели и закрытие приложения PDETool).

Группа команд **Edit** содержит команды редактирования PDE-модели (в том числе с использованием системного буфера).

Группа команд **Options** содержит команды установки (изменения) режимов работы приложения **PDETool**.

Группа команд **Draw** содержит команды прорисовки и редактирования геометрических объектов в PDE-модели.

Группа команд **Boundary** содержит команды ввода и редактирования граничных условий, показа номеров граничных сегментов и зон расчетной области, а также команды удаления границ между зонами.

Группа команд **PDE** содержит команды ввода и редактирования параметров (коэффициентов PDE), показа номеров зон расчетной области.

Группа команд **Mesh** содержит команды работы с конечно-элементной сеткой (генерация, сгущение (переопределение), регуляризация, показ номеров объектов сетки и их параметров качества, экспорт).

Группа команд **Solve** содержит команды решения PDE, ввода и редактирования параметров решателя PDE.

Группа команд **Plot** содержит команды визуализации решения PDE.

Window – переключение между окнами **MATLAB**.

Help – команды работы со справочной системой **MATLAB**.

Меню **Options** содержит следующие команды:

Grid – показать/скрыть координатную сетку в объекте Axes;

Grid Spacing – установить лимиты и шаг координатной сетки;

Snap – округлять координаты указателя мыши при показе их значений пользователю;

Axes Limits – установить значения пределов координатных осей;

Axes Equal – установить на экране одинаковый масштаб по осям x и y;

Turn Off Toolbar Help – запретить/разрешить выдачу подсказок по кнопкаминструментальной панели **PDETool**;

Zoom – включить режим показа в увеличенном масштабе выделяемой прямоугольной области в PDE-модели;

Application – переключение типа PDE-задачи:

Generic Scalar – скалярная краевая задача;

Generic System – система PDE;

Structural Mechanics, **Plane Stress** – задача теории упругости на недеформируемой сетке;

Structural Mechanics, **Plane Strain** – задача теории упругости на деформируемой сетке;

Electrostatics PDE – электростатики относительно скалярного электрического потенциала;

Magnetostatics PDE – магнитостатики относительно векторного магнитного потенциала;

AC Power Electromagnetics – переменное электромагнитное поле;

Conductive Media DC – электрическое поле постоянного тока в проводящей среде;

Heat Transfer – уравнение теплопередачи;

Diffusion – уравнение диффузии;

Refresh – Обновить изображение PDE-модели в поле axes.

Меню **Draw** содержит следующие команды:

Draw Mode – переключение в режим ввода (прорисовки) геометрии. В этом режиме возможно создание, уничтожение и изменение геометрическихобъектов в расчетной области;

Rectangle/square – ввод прямоугольника или квадрата с помощью мыши. Начальной точкой прямоугольника является его верхняя левая вершина. Если привводе будет удерживаться клавиша Ctrl, то будет прорисовываться квадрат. Стороны прямоугольника всегда параллельны осям координат;

Rectangle/square (centered) – аналогично предыдущему, но начальной точкой прямоугольника является его центр;

Ellipse/circle – ввод эллипса или круга с помощью мыши. Начальной точкой эллипса является его верхняя левая точка. Если при вводе будетудерживаться клавиша Ctrl, то будет прорисовываться круг. Главные оси эллипса всегда параллельны осям координат;

Ellipse/circle (centered) – аналогично предыдущему, но начальной точкой круга или эллипса является его центр;

Polygo – прорисовка многоугольника с помощью мыши. Ввод производится отрезками ломаной линии, пока она не станет замкнутой;

Rotate – поворот выделенных геометрических объектов на некоторый угол вокругнекоторой точки;

Export Geometry Description, Set Formula, Labels – экспорт в базовую рабочую область переменных описания геометрии.

Непосредственно под панелью инструментов расположена строка ввода **Set Formula**. Она должна содержать формулу описания геометрии расчетной области. По синтаксису это должно быть выражение, операндами которого являются метки (идентификаторы) геометрических объектов. Операция объединения обозначается знаком плюс, операция исключения – знаком минус, операция пересечения – знаком умножения (звездочка). Эта строка ввода доступна для редактирования только в режиме **Draw Mode**.

Меню **Boundary** содержит следующие команды:

Boundary Mode (Ctrl+B) – переключение в режим ввода граничных условий (ГУ);

Specify Boundary Conditions – ввод параметров граничных условий (рисунок 7.2).

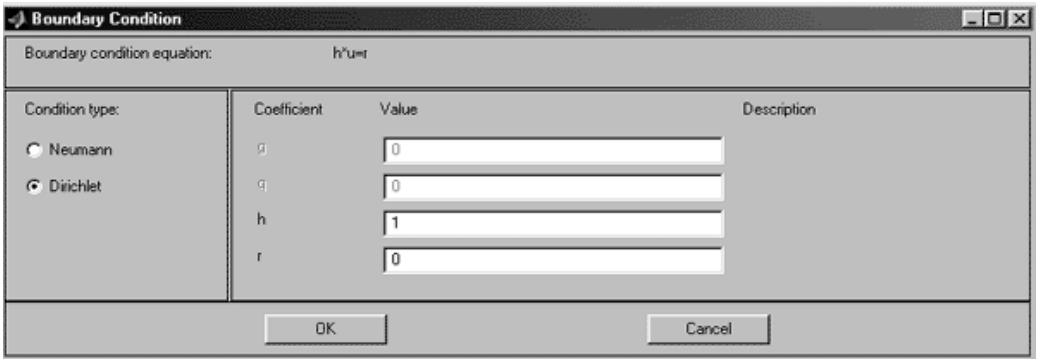


Рисунок 7.2 – Диалоговое окно ввода параметров граничных условий

По данной команде раскрывается диалоговое окно, предлагающее ввести нужные значения параметров для всех или для выделенных граничных сегментов;

Show Edge Labels – показать номера граничных сегментов;

Show Subdomain Labels – показать номера зон;

Remove Subdomain Border – удалить границу зон. По данной команде удаляется граница раздела зон, которой принадлежат выделенные граничные сегменты, и производится автоматическая перенумерация зон;

Remove All Subdomain Borders – удаление всех границ зон. По данной команде удаляются все границы зон вне зависимости от выделения. Вся расчетная область станет зоной № 1.

Export Decomposed Geometry, Boundary Cond's – экспорт в базовую рабочую область переменных описания граничных условий.

Меню PDE содержит следующие команды:

PDE Mode – переключение в режим ввода параметров (коэффициентов) PDE;

Show Subdomain Labels – показать номера зон;

PDE Specification – ввод параметров (коэффициентов) PDE. По данной команде раскрывается диалоговое окно (рисунок 7.3), предлагающее ввести нужные значения параметров.

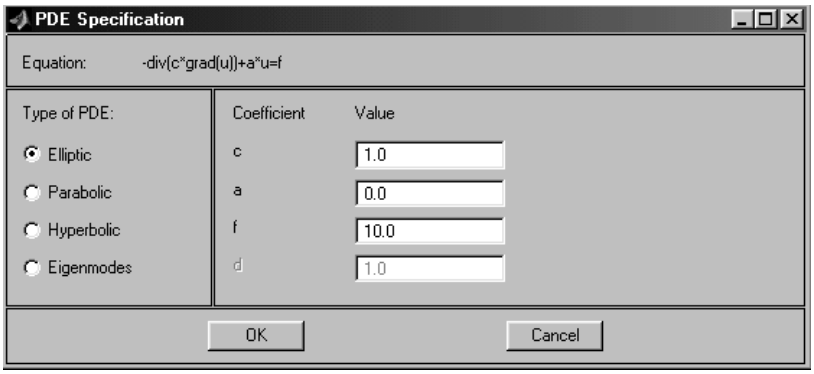


Рисунок 7.3 – Диалоговое окно ввода коэффициентов PDE

Вводимые параметры относятся только к выделенным зонам; если выделенных нет, то ко всем зонам;

Export PDE Coefficients – экспорт в базовую рабочую область переменных, описывающих распределение PDE коэффициентов в расчетной области.

Меню **Mesh** содержит следующие команды:

Mesh Mode – переключение в режим построения конечно-элементной сетки;

Initialize Mesh (Ctrl+I) – инициализация (генерация) конечно-элементной сетки. Начальный вариант сетки генерируется независимо от существования и текущего состояния сетки;

Refine Mesh (Ctrl+M) – переопределение (сгущение) конечно-элементной сетки вовсей расчетной области.

Jiggle Mesh – регуляризация конечно-элементной сетки. Конечно-элементная сетка перестраивается (узлы перемещаются) таким образом, чтобы показатель нерегулярности всех конечных элементов не превышал установленной в PDETool величины;

Undo Mesh Change – отменить последнее изменение конечно-элементной сетки;

Display Triangle Quality – отобразить в цвете значения показателя регулярности для всех конечных элементов (треугольников) сетки;

Show Node Labels – показать номера узлов конечно-элементной сетки;

Show Triangle Labels – показать номера конечных элементов (треугольников);

Parameters – установить параметры генератора сетки;

Export Mesh – экспорт конечно-элементной сетки в базовую рабочую область.

Меню **Solve** содержит следующие команды:

Solve PDE (Ctrl+E) – решить краевую задачу (вычислить узловое распределение искомой величины);

Parameters – установить параметры решателя PDE;

Export Solution – экспорт решения (узлового распределения) в базовую рабочую область.

Меню **Plot** содержит следующие команды:

Plot Solution (Ctrl+P) – отобразить решение PDE. По этой команде в объекте axes окна PDETool будет графически показано распределение решения PDE в расчетной области в соответствии с установленными значениями параметров визуализации решения;

Parameters – установить параметры отображения решения. По данной команде раскрывается диалоговое окно (рисунок 7.4), предлагающее ввести нужные значения параметров.

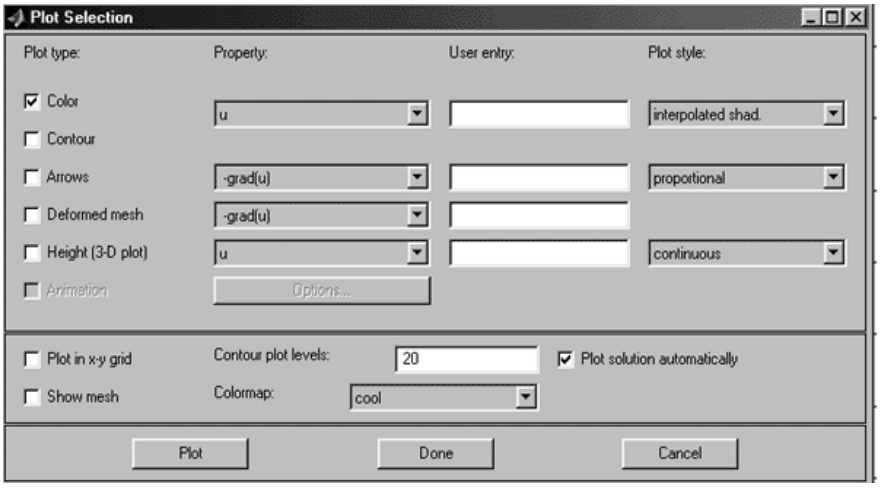


Рисунок 7.4 – Диалоговое окно ввода параметров визуализации решения PDE

Диалоговое окно состоит из двух панелей. Верхняя панель состоит из четырех колонок: **Plot type**, **Property**, **User entry**, **Plot style**. Колонка **Plot type** состоит окон контроля (check box), позволяющих указать, что требуется графически отображать: **Color** – цветной план рассчитанного поля **Contour** – изолинии поля; **Arrows** – показ векторного поля стрелками; **Deformed mesh** – отображение поля на деформированной сетке; **Height**(**3D plot**) – отображение скалярного поля в виде поверхностного графика в отдельной фигуре **MATLAB**; **Animation** – анимация решения нестационарной задачи. Колонка **Property** состоит из ниспадающих меню, позволяющих выбрать визуализируемое поле. Если в соответствующем ниспадающем меню нет требуемого поля, то можно воспользоваться строками ввода в колонке User entry в соответствии с правилами, действующими при подготовке входных параметров функций **assempde**, **parabolic**, **hyperbolic**. Колонка **Plot style** содержит ниспадающие меню, позволяющие выбрать стили визуализации соответствующих полей: **interpolated shad** – использование кусочно-линейной интерполяции при отображении; **flat shading** – отображение в кусочно-постоянном виде; **proportional** – каждая стрелка имеет длину, пропорциональную модулю вектора в соответствующей точке наблюдения; **normalized** – все стрелочки имеют одинаковую длину; **continuous** – непрерывная интерполяция при построении поверхностного графика; **discontinuous** – кусочно-постоянный поверхностный график.

Нижняя панель состоит из трех окон контроля (**check box**), строки ввода и ниспадающего меню. **Plot in x-y grid** – построение на x-y сетке; **Show mesh** – показать конечно-элементную сетку; **Plot solution automatically** – автоматическая визуализация решения, получаемого в результате выполнения команды меню **Solve PDE**. **Contour plotlevels** – число линий равного уровня отображаемого скалярного поля. **Colormap** – выбор палитры цветного плана скалярного поля. Внизу диалогового окна имеются три кнопки: **Plot**, **Done**, **Cancel**. **Plot** – отобразить решение PDE с учетом вновь указанных режимов; **Done** – принять введенные режимы; **Cancel** – отменить изменения режимов и закрыть диалоговое окно;

Export Movie – экспорт в базовую рабочую область переменной, содержащей информацию, необходимую для анимации решения нестационарной PDE-задачи.

Рассмотрим две типовые задачи, решение которых может быть найдено при помощи **PDE Toolbox**.

Магнитостатическая задача

Магнитостатическим называют магнитное поле, источниками которого являются постоянные электрические токи и остаточная намагниченность ферромагнитных тел.

Рассмотрим уравнения магнитостатического поля в дифференциальной форме. Одно из этих уравнений получило название «закон полного тока»:

$$rot\,H = \delta,$$

где H – пространственное распределение вектора напряженности магнитного поля;

δ – векторное поле плотности тока (при анализе магнитостатического поля можно считать, что все токи обусловлены действием сторонних источников тока).

Второе уравнение называется законом непрерывности линий магнитной индукции:

$$div\,B = 0,$$

где B – пространственное распределение вектора магнитной индукции.

Уравнения магнитостатики должны дополняться уравнением, описывающим магнитные свойства вещества и связывающим между собой векторы H и B. Такое уравнение называется уравнением материальной связи. В случае линейных изотропных диэлектрических свойств среды уравнение материальной связи между векторами H и B имеет вид:

$$B = \mu_a \cdot H + Br \text{ или } H = \vartheta_a\left(B - Br\right),$$

где μ_a – абсолютная магнитная проницаемость, $\mu_a = \mu_0\mu$;

μ_0 – абсолютная магнитная проницаемость вакуума (основная магнитная постоянная), $\mu_0 = 4\pi \cdot 10^{-7}$ Гн/м;

μ – относительная магнитная проницаемость среды;

Br – векторное поле остаточной магнитной индукции среды;

ϑ_a – удельное магнитное сопротивление среды, $\vartheta_a = \mu_a^{-1}$;

Векторные поля и Br являются пространственно-распределенными источниками магнитостатического поля. Векторное поле В можно представить в виде ротора некоторого векторного поля:

$$B = rot\,A\,.$$

Здесь векторное поле A называется векторным магнитным потенциалом.

Уравнение

$$rot\,\vartheta_a\left(rot A\right) = \delta + rot\,\vartheta_a\left(Br\right)$$

является уравнением магнитостатического поля относительно векторного магнитного потенциала.

Если ввести обозначение $A' = A/\mu_0$ – векторный магнитный потенциал, измеряемыйв единицах тока, то уравнение примет следующий вид

$$rot\left(\mu^{-1} rot A'\right) = \delta + rot\left(\mu^{-1} Mr\right),$$

где Mr – векторное поле остаточной намагниченности вещества, $Mr = Br/\mu_0$.

PDE Toolbox MATLAB не поддерживает наличие в расчетной области тел, обладающих остаточной намагниченностью. Значит, поддерживается только такая магнитостатическая задача, в которой $Mr = 0$.

В случае двумерного магнитостатического поля $\delta = \delta \cdot 1z$; $A' = A' \cdot 1z$, поэтому при $Mr = 0$ в соответствии с правилами векторного анализа уравнение примет следующий вид:

$$-div\mu^{-1}\left(grad A'\right) = \delta.$$

Такое преобразование корректно только в случае изотропных магнитных свойств вещества.

В **PDETool** краевая задача магнитостатики базируется на этом уравнении. На рисунке 7.5 показан вид диалогового окна ввода коэффициентов PDE магнитостатической задачи. Здесь приняты следующие обозначения переменных (полей): A – векторный магнитный потенциал A'; B – вектор магнитной индукции; mu – относительная магнитная проницаемость; J – плотность тока. По умолчанию mu = 1; J = 1. Коэффициенты PDE, показанные на рисунке 7.5, можно задавать в виде скалярных полей с помощью выражений.

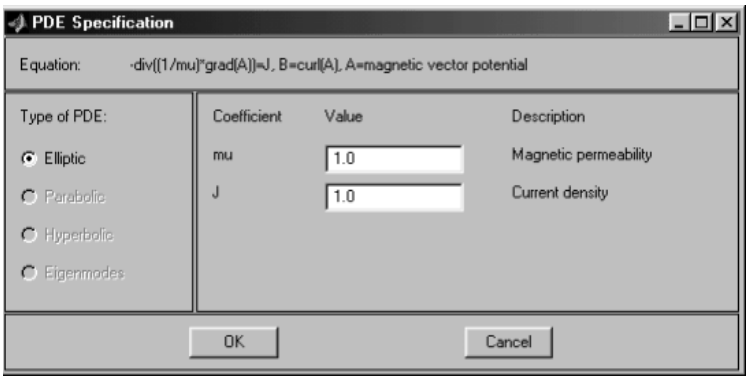


Рисунок 7.5 – Диалоговое окно ввода коэффициентов PDE магнитостатики

На рисунках 7.6 и 7.7 показан вид диалоговых окон ввода граничных условий для краевой задачи магнитостатики и значения соответствующих параметров по умолчанию.

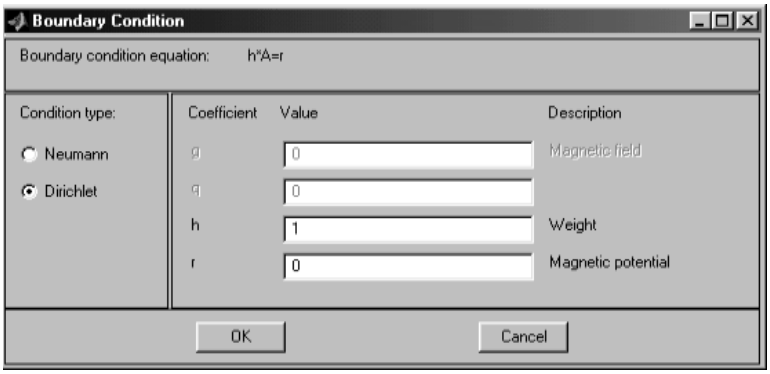


Рисунок 7.6 – Диалоговое окно ввода граничных условий Дирихле

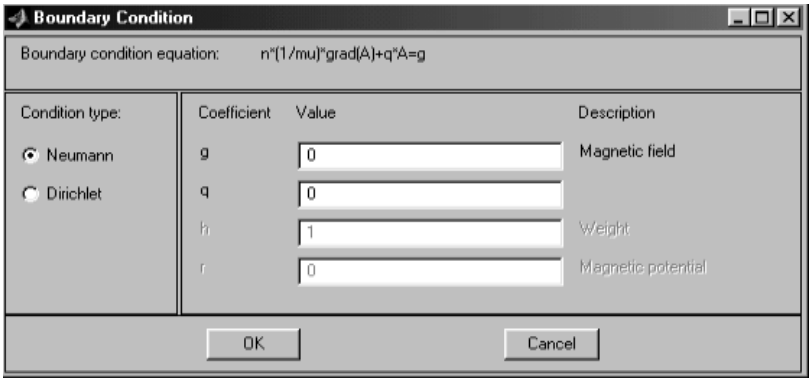


Рисунок 7.7 – Диалоговое окно ввода граничных условий Неймана

Краевая задача магнитостатики названа в PDETool Magnetostatics.

Теплопередача

Тепловое поле характеризуется пространственным распределением температуры и плотности потока тепловой мощности.

Уравнения теплового поля в дифференциальной форме имеют вид:

$$divpm = Q - \frac{dU}{dt};$$

$$pm = -k \cdot grad\, T;$$

$$\frac{dU}{dt} = \frac{d\left(\rho CT\right)}{dt} = \rho C \frac{dT}{dt},$$

где pm – плотность потока тепловой мощности, Вт/м^2;

Q – объемная плотность мощности сторонних источников тепла, Вт/м^3;

U – объемная плотность внутренней тепловой энергии вещества, Дж/м^3;

k – теплопроводность вещества, ВтК $^{-1}$ · м$^{-1}$;

T – температура, К;

ρ – плотность вещества, кг/м^3;

C – удельная теплоемкость вещества, Дж кг $^{-1}$ · К$^{-1}$.

Можно получить следующее уравнение:

$$\rho C \frac{dT}{dt} - div\left(k \cdot grad\, T\right) = Q\,.$$

Это уравнение теплопроводности относительно поля температур.

Уравнение описывает динамический режим теплового поля. В статистическом режиме $dT/dt = 0$, поэтому уравнение теплопроводности приобретает следующий вид:

$$-div\left(k \cdot grad\, T\right) = Q\,.$$

В **PDETool** краевая задача теплопроводности базируется на этих уравнениях.

На рисунке 7.8 показан вид диалогового окна ввода коэффициентов PDE этой краевой задачи. Здесь приняты следующие обозначения переменных (полей):

rho – плотность вещества ρ;

C – удельная теплоемкость вещества;

k – теплопроводность вещества;

Q – объемная плотность мощности сторонних источников тепла.

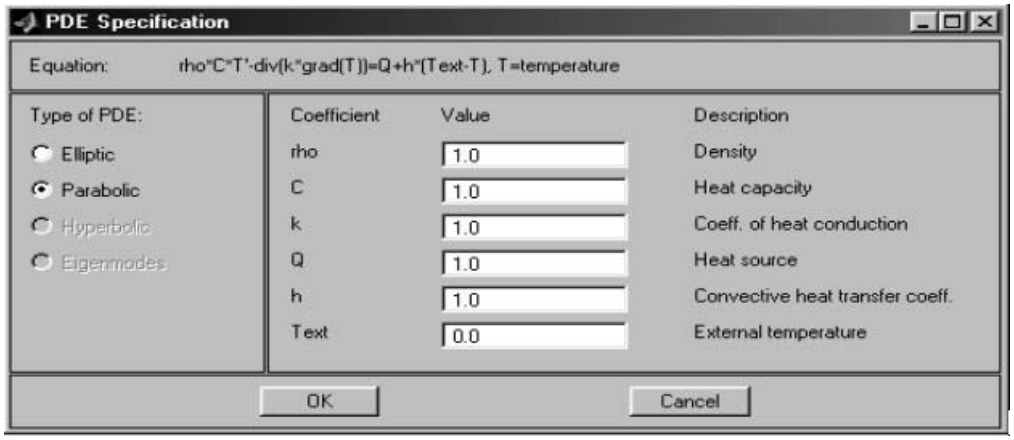


Рисунок 7.8 – Диалоговое окно ввода коэффициентов PDE теплопроводности

В поле **Equation** этого окна показан вид уравнения теплопроводности. Второе слагаемое правой части уравнения в этом окне записано некорректно. h*(Text-T) – это не объемно распределенный источник тепла, а составляющая плотности потока тепловой мощности через внешнюю границу расчетной области, пропорциональная разности температур окружающей среды и границы расчетной области. На самом деле это слагаемое относится к граничным

условиям, которые в задачах электромагнетизма называются «импедансными». Здесь h – коэффициент конвективного или кондуктивного теплообмена расчетной области с окружающей средой, Вт м $^{-2}$ К$^{-1}$; Text – температура окружающей среды при бесконечном удалении от расчетной области (здесь предполагается, что расчетная область окружена бесконечной однородной средой без источников тепла); T – температура границы расчетной области.

По умолчанию *rho* = 1, C = 1, k = 1, Q = 1, h = 1, Text = 0.

В режиме **Elliptic** ввод коэффициентов *rho* и C запрещен.

На рисунках 7.9 и 7.10 показан вид диалоговых окон ввода граничных условий для краевой задачи теплопроводности и значения соответствующих параметров по умолчанию.

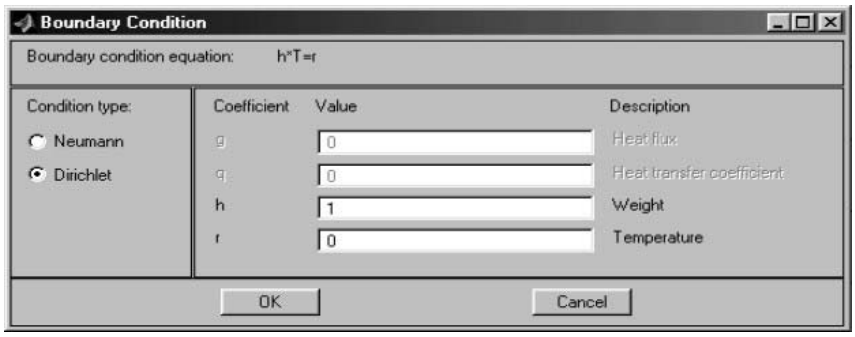


Рисунок 7.9 – Диалоговое окно ввода граничных условий Дирихле

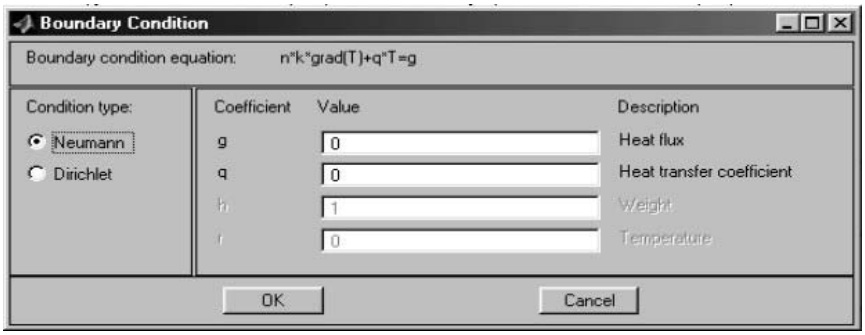


Рисунок 7.10 – Диалоговое окно ввода граничных условий Неймана

Краевая задача анализа теплового поля названа в **PDETool Heat Transfer**.

Пример выполнения задания

В качестве примера рассмотрим задачу моделирования поля рассеяния над несплошностью (трещиной) в ферромагнитном образце. Задание геометрии приведено на рисунке 7.11.

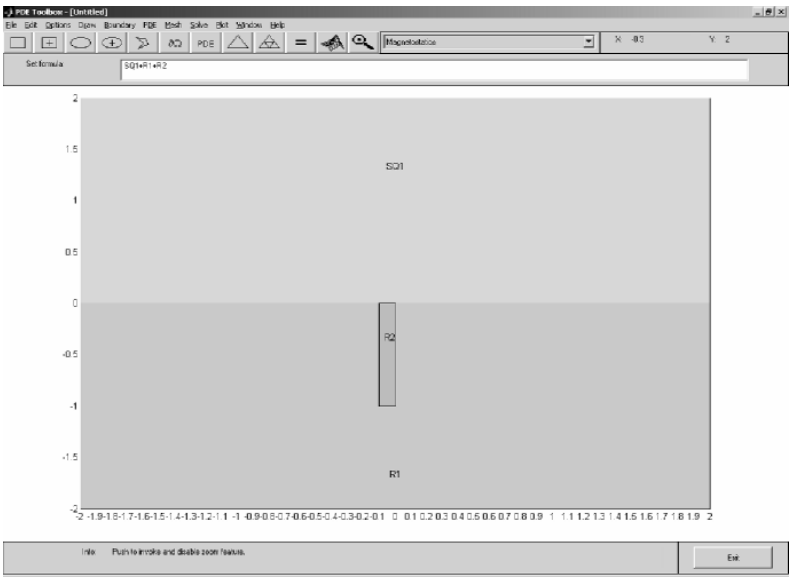


Рисунок 7.11 – Задание геометрии для решения задачи

Геометрия описывается квадратом *SQ1* и двумя прямоугольниками *R*1 и *R*2. *R*1 описывает сам образец из ферромагнитного материала, а *R*2 – трещина раскрытием 0.1 мм и глубиной 1 мм.

На рисунке 7.12 приведено описание граничных условий.

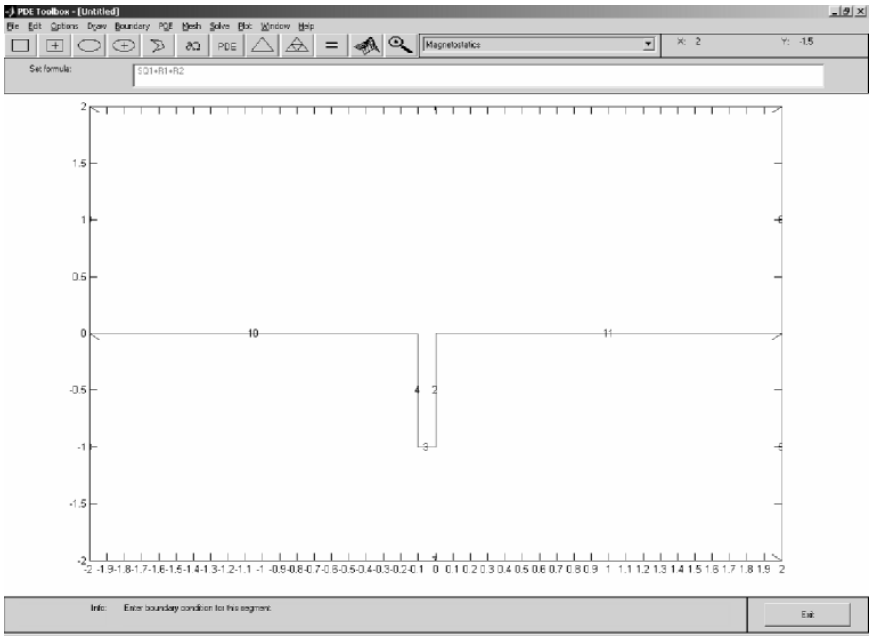


Рисунок 7.12 – Описание граничных условий

Одна граница здесь удалена (верхняя грань *R*1). Для левой и правой границы выбираем нулевые граничные условия Неймана, что фактически обозначает отсутствие тангенциальных составляющих магнитного поля. Для верхней границы выбираем нулевые граничные условия Дирихле – отсутствие внешнего поля. Для нижней границы условие Неймана $g = 3$, что соответствует наличию тангенциальной составляющей напряженностимагнитного поля $H_\tau = 3$А/мм.

На рисунке 7.13 приведено описание коэффициентов уравнений.

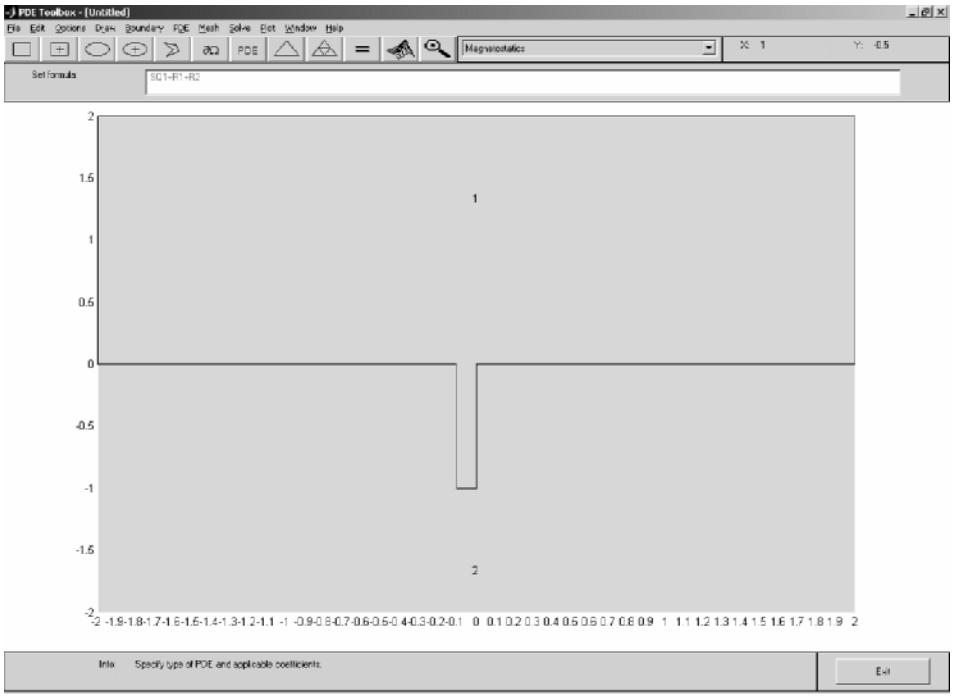


Рисунок 7.13 – Описание уравнений

Для верхней области задаем *mu* = 1, *J* = 0 (воздух, ток отсутствует). Для нижнейобласти *mu* = 500, *J* = 0 (сталь, ток отсутствует).

После инициализации сетки, выбора соответствующих параметров визуализации и решения задачи получен следующий результат (рисунок 7.14).

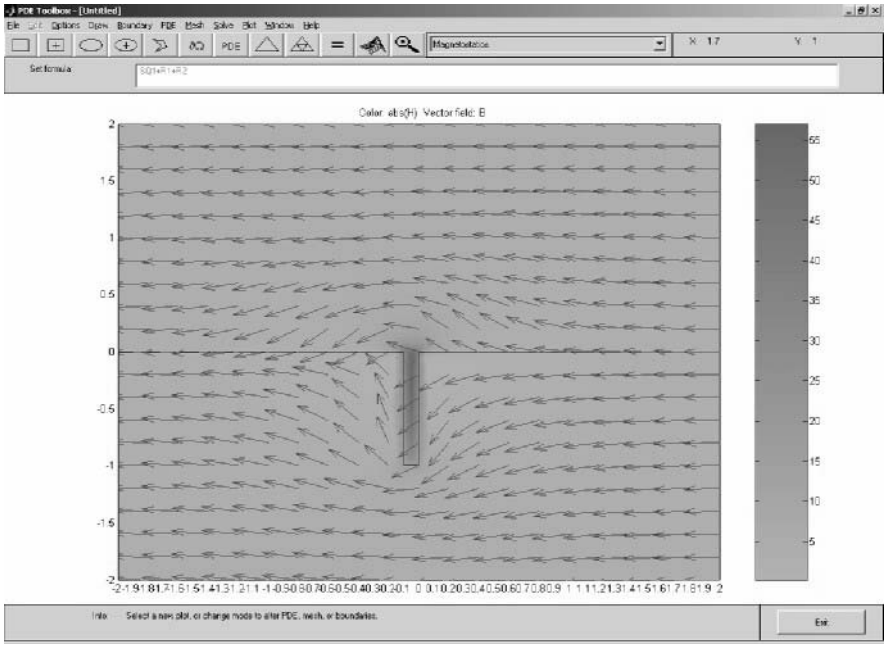


Рисунок 7.14 – Результат решения задачи магнитостатики

Теплоперенос

Рассмотрим решение задачи нагрева стенки с дефектом. Геометрия приведена на рисунке 7.15.

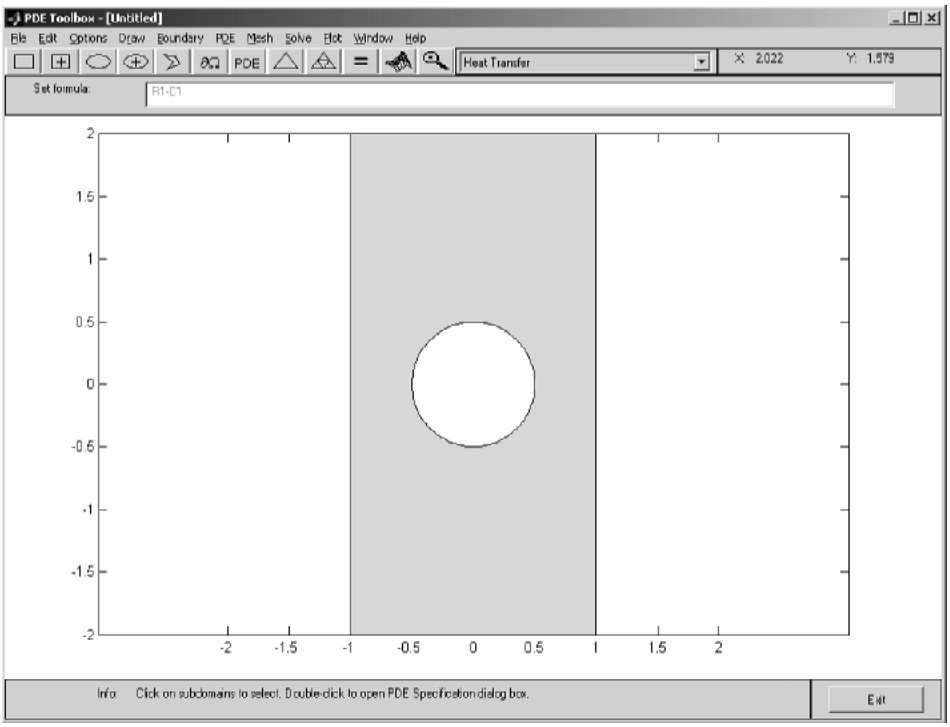


Рисунок 7.15 – Геометрия задачи

Граничные условия для левой стенки зададим как условия Дирихле. Стенка подогревается до температуры 100 °С. Правая граница – условие Неймана тепловой поток, –10 °С – стенка остывает. Остальные границы – нулевые условия Неймана (отсутствие теплопереноса).

При задании параметров уравнения выберем параболическое уравнение (учет времени) и зададим плотность $\rho = 1$, теплоемкость $C = 1$, коэффициент теплопроводности $k = 1$, остальные коэффициенты равны 0.

В параметрах решателя (Solve – Parameters) зададим временной интервал для поиска решения Time: [0:3]. Это значит, что решение будет получено для точек 1…3 с.

После инициализации сетки, выбора соответствующих параметров визуализации и решения задачи получен следующий результат (рисунок 7.16).

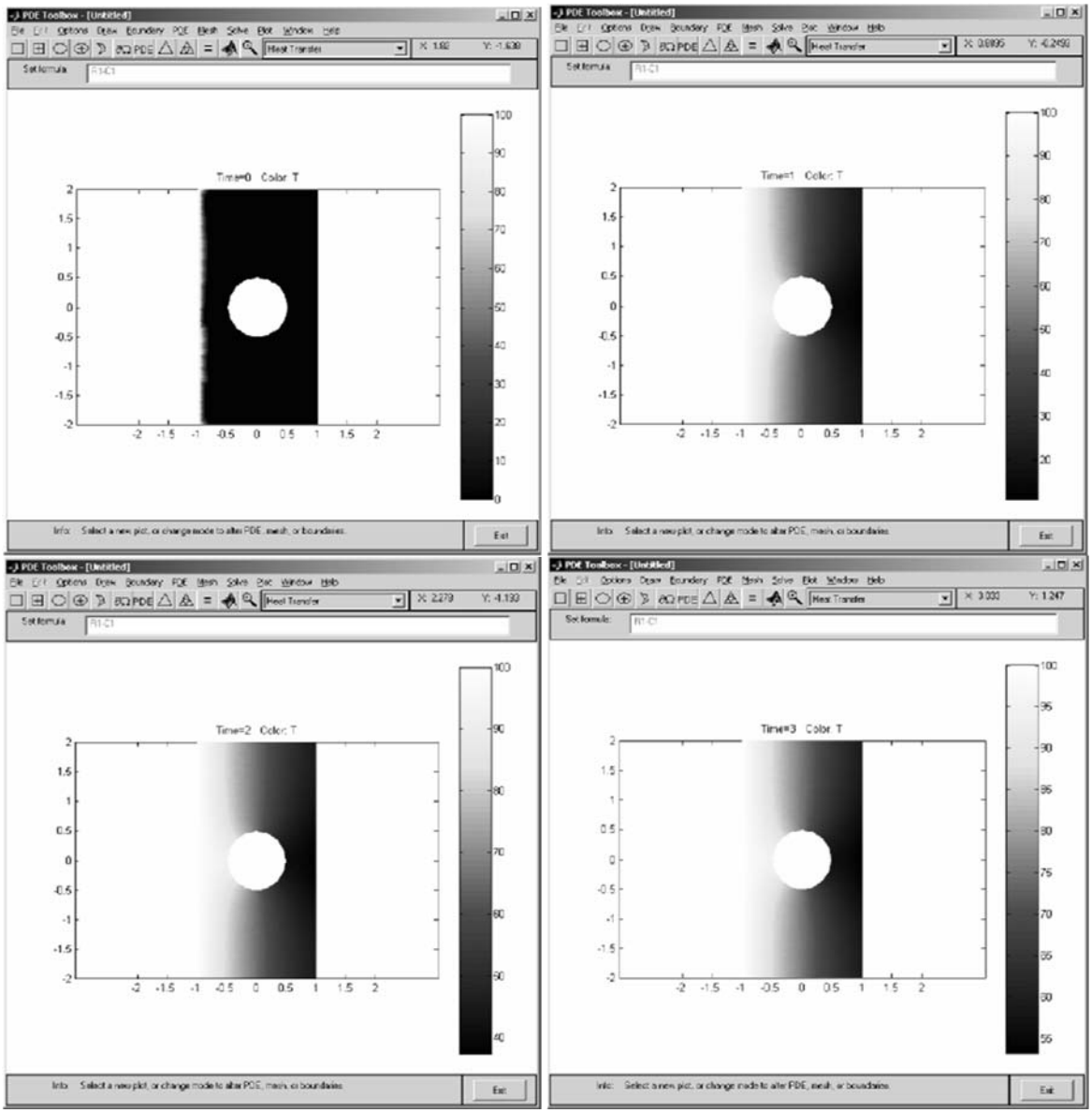


Рисунок 7.16 – Результат решения задачи теплопереноса

7.2 Практическое задание

Решить задачу теплопереноса и магнитостатики. Вариант получить упреподавателя.

Контрольные вопросы

1 Что такое Partial Differential Equation (PDE) Toolbox?
2 Как решить задачу по магнитостатике, используя **MATLAB**?
3 Как решить задачу по теплопередаче, используя **MATLAB**?

8 Лабораторная работа № 8. Знакомство с пакетом визуального моделирования SIMULINK

Цель работы: получение основных навыков работы с пакетом визуального моделирования **SIMULINK** на примере построения модели тракта обработки сигналов.

8.1 Общие сведения

Программа **Simulink** является приложением к пакету **MATLAB**. При моделировании с использованием **Simulink** реализуется принцип визуального программирования, в соответствии с которым пользователь на экране из библиотеки стандартных блоков создает модель устройства и осуществляет расчеты. При этом в отличие от классических способов моделирования пользователю не нужно досконально изучать язык программирования и численные методы математики, а достаточно общих знаний, требующихся при работе на компьютере, и знаний той предметной области, в которой он работает.

Simulink является достаточно самостоятельным инструментом **MATLAB,** и при работе с ним совсем не требуется знать сам **MATLAB** и остальные его приложения. С другой стороны, доступ к функциям **MATLAB** и другим его инструментам остается открытым, и их можно использовать в **Simulink**. Часть входящих в состав пакетов имеет инструменты, встраиваемые в **Simulink** (например, **LTI-Viewer** приложения **Control System Toolbox** – пакета для разработки систем управления). Имеются также дополнительные библиотеки блоков для разных областей применения (например, **Power System Blockset** – моделирование электротехнических устройств, **Digital Signal Processing Blockset** – набор блоков для разработки цифровых устройств и т. д).

Для запуска программы необходимо предварительно запустить пакет **MATLAB**. Основное окно пакета **MATLAB** показано на рисунке 8.1. Там же показана подсказка появляющаяся в окне при наведении указателя мыши на ярлык **Simulink** в панели инструментов.

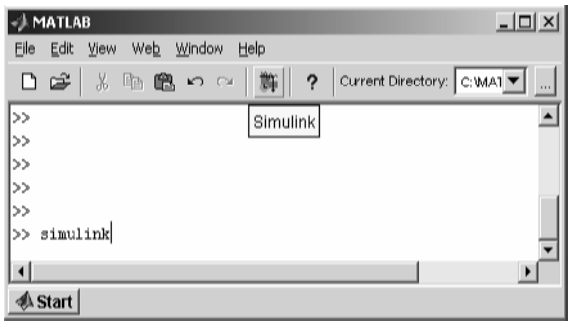


Рисунок 8.1 – Основное окно программы MATLAB

После открытия основного окна программы **MATLAB** нужно запустить программу **Simulink**. Это можно сделать одним из трех способов:

1) нажать кнопку (**Simulink**) на панели инструментов командного окна **MATLAB**;

2) в командной строке главного окна **MATLAB** напечатать **Simulink** и нажатьклавишу **Enter** на клавиатуре;

3) выполнить команду **Open** в меню **File** и открыть файл модели (mdl-файл).

Использование первого и второго способов приводит к открытию окна обозревателя разделов библиотеки **Simulink** (рисунок 8.2).

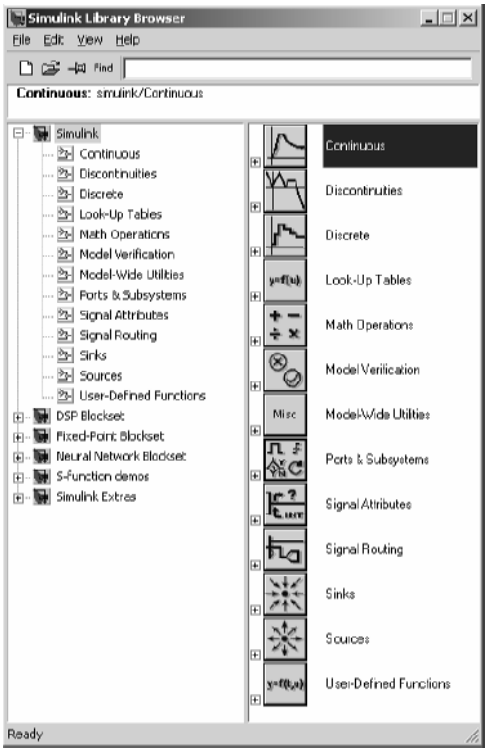


Рисунок 8.2 – Окно обозревателя разделов библиотеки **Simulink**

Библиотека **Simulink** содержит следующие основные разделы:

1 **Continuous** – линейные блоки.

2 **Discrete** – дискретные блоки.

3 **Functions** &**Tables** – функции и таблицы.

4 **Math** – блоки математических операций.

5 **Nonlinear** – нелинейные блоки.

6 **Signals** &**Systems** – сигналы и системы.

7 **Sinks** – регистрирующие устройства.

8 **Sources** – источники сигналов и воздействий.

9 **Subsystems** – блоки подсистем.

Для создания модели в среде **SIMULINK** необходимо последовательно выполнитьряд действий.

Создать новый файл модели с помощью команды File/New/Model или используя кнопку на панели инструментов.

Расположить блоки в окне модели. Для этого необходимо открыть соответствующий раздел библиотеки (например, **Sources** – Источники). Далее, указав курсором на требуемый блок и нажав на левую клавишу мыши, перетащить блок в созданное окно. Клавишу мыши нужно держать нажатой. На рисунке 8.3 показано окно модели, содержащее блоки.

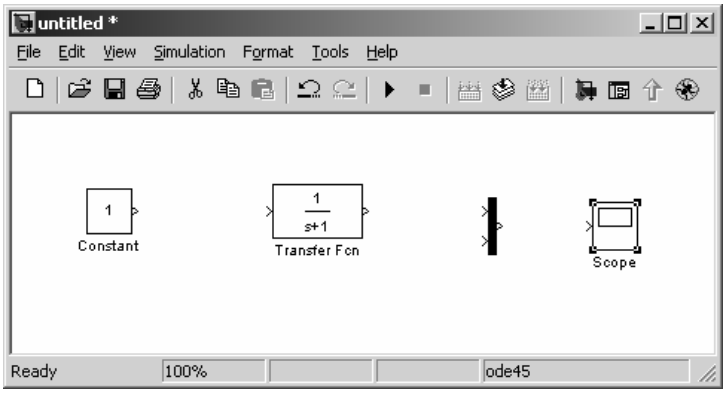


Рисунок 8.3 – Окно модели, содержащее блоки

Для удаления блока необходимо выбрать блок (указать курсором на его изображение и нажать левую клавишу мыши), а затем нажать клавишу Delete на клавиатуре.

Далее, если это требуется, нужно изменить параметры блока, установленные программой по умолчанию. Для этого необходимо дважды щелкнуть левой клавишей мыши, указав курсором на изображение блока. Откроется окно редактирования параметров данного блока. При задании численных параметров следует иметь в виду, что в качестве десятичного разделителя должна использоваться точка, а не запятая. После внесения изменений нужно закрыть окно кнопкой OK. На рисунке 8.4 в качестве примера показаны блок, моделирующий передаточную функцию, и окно редактирования параметров данного блока.

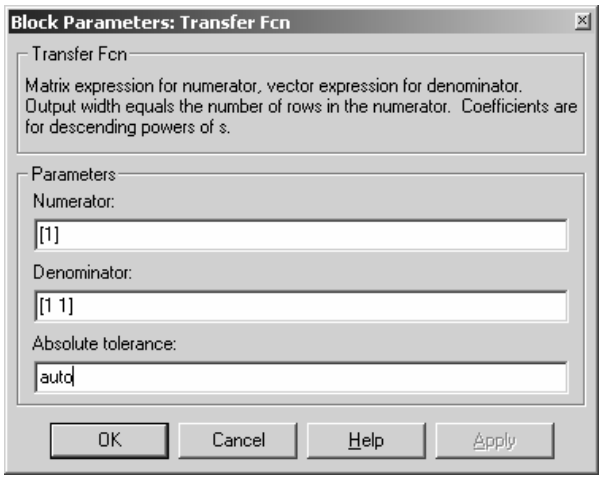


Рисунок 8.4 – Блок, моделирующий передаточную функцию, и окно редактирования параметров блока

После установки на схеме всех блоков из требуемых библиотек нужно выполнить соединение элементов схемы. Для соединения блоков необходимо указать курсором на выход блока, а затем нажать и, не отпуская левую клавишу мыши, провести линию к входу другого блока. После этого отпустить клавишу. В случае правильного соединения изображение стрелки на входе блока изменяет цвет. Для создания точки разветвления в соединительной линии нужно подвести курсор к предполагаемому узлу и, нажав правую клавишу мыши, протянуть линию. Для удаления линии требуется выбрать линию (так же, как это выполняется для блока), а затем нажать клавишу Delete на клавиатуре. Схема модели, в которой выполнены соединения между блоками, показана на рисунке 8.5.

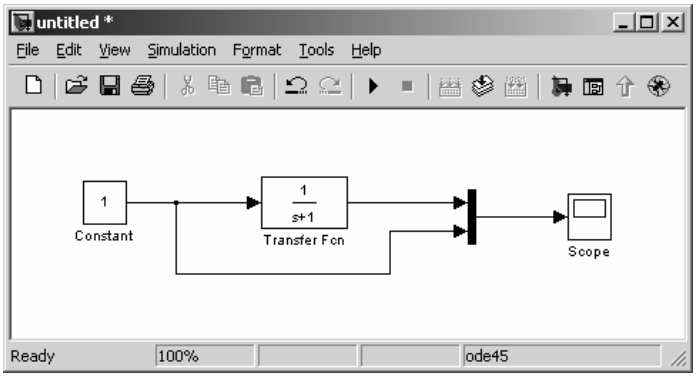


Рисунок 8.5 – Схема модели

После составления расчетной схемы необходимо сохранить ее в виде файла на диске, выбрав пункт меню File/Save As... в окне схемы и указав папку и имя файла.

Перед выполнением расчетов необходимо предварительно задать параметры расчета. Задание параметров расчета выполняется в панели управления меню Simulation/Parameters. Вид панели управления приведен на рисунке 8.6.

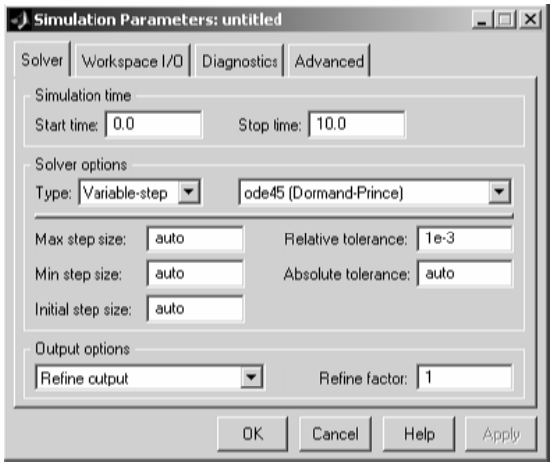


Рисунок 8.6 – Панель управления

Окно настройки параметров расчета имеет четыре вкладки:

1) Solver (Расчет) – установка параметров расчета модели;

2) Workspace I/O (ввод/вывод данных в рабочую область) – установка параметров обмена данными с рабочей областью **MATLAB**;

3) Diagnostics (Диагностика) – выбор параметров диагностического режима;

4) Advanced (Дополнительно) – установка дополнительных параметров.

Установка параметров расчета модели выполняется с помощью элементов управления, размещенных на вкладке Solver. Эти элементы разделены на три группы: Simulation time (интервал моделирования или, иными словами, время расчета), Solver options (параметры расчета), Output options (параметры вывода).

При выборе параметров расчета необходимо указать способ моделирования Type и метод расчета нового состояния системы. Для параметра Type доступны два варианта – с фиксированным Fixed-step или с переменным Variable-step шагом. Как правило, Variable-step используется для моделирования непрерывных систем, а Fixed-step – для дискретных.

Список методов расчета нового состояния системы содержит несколько вариантов.

Первый вариант (discrete) используется для расчета дискретных систем. Остальные методы используются для расчета непрерывных систем. Эти методы различны для переменного Variable-step и для фиксированного Fixed-step шага времени, но, по сути, представляют собой процедуры решения систем дифференциальных уравнений.

Ниже двух раскрывающихся списков Type находится область, содержимое которой меняется зависимости от выбранного способа изменения модельного времени. При выборе Fixed-step в данной области появляется текстовое поле Fixed-step size (величина фиксированного шага), позволяющее указывать величину шага моделирования. Величина шага моделирования по умолчанию устанавливается системой автоматически (auto).

Требуемая величина шага может быть введена вместо значения auto либо в форме числа, либо в виде вычисляемого выражения (то же самое относится и ко всем параметрам, устанавливаемым системой автоматически).

При выборе Fixed-step необходимо также задать режим расчета (Mode). Для параметра Mode доступны три варианта:

1) MultiTasking (многозадачный) – необходимо использовать, если в модели присутствуют параллельно работающие подсистемы и результат работы модели зависит от временных параметров этих подсистем. Режим позволяет выявить несоответствие скорости и дискретности сигналов, пересылаемых блоками друг другу;

2) SingleTasking (однозадачный) – используется для тех моделей, в которых недостаточно строгая синхронизация работы отдельных составляющих не влияет на конечный результат моделирования;

3) Auto (автоматический выбор режима) – позволяет Simulink автоматически устанавливать режим MultiTasking для тех моделей, в которых используются блоки с различными скоростями передачи сигналов и режим SingleTasking для моделей, в которых содержатся блоки, оперирующие одинаковыми скоростями.

При выборе Variable-step в области появляются поля для установки трех параметров:

1) Max step size – максимальный шаг расчета. По умолчанию он устанавливается автоматически (auto), и его значение в этом случае равно (StopTime – StartTime)/50. Довольно часто это значение оказывается слишком большим, и наблюдаемые графикипредставляют собой ломаные (а не плавные) линии. В этом случае величину максимального шага расчета необходимо задавать явным образом;

2) Min step size – минимальный шаг расчета;

3) Initial step size – начальное значение шага моделирования.

При моделировании непрерывных систем с использованием переменного шага необходимо указать точность вычислений: относительную (Relative tolerance) и абсолютную (Absolute tolerance). По умолчанию они равны соответственно 10^{-3} и auto.

В нижней части вкладки Solver задаются настройки параметров вывода выходных сигналов моделируемой системы (Output options). Для данного параметра возможен выбор одного из трех вариантов:

1) Refine output (скорректированный вывод) – позволяет изменять дискретность регистрации модельного времени и тех сигналов, которые сохраняются в рабочей области MATLAB с помощью блока To Workspace. Установка величины дискретности выполняется в строке редактирования Refine factor, расположенной справа. По умолчанию значение Refine factor равно 1, это означает, что регистрация производится с шагом $\Delta t = 1$ (то есть для каждого значения модельного времени). Если задать Refine factor равным 2, это означает, что будет регистрироваться каждое второе значение сигналов, 3 – каждое третье и т. д. Параметр Refine factor может принимать только целые положительные значения;

2) Produce additional output (дополнительный вывод) – обеспечивает дополнительную регистрацию параметров модели в заданные моменты времени; их значения вводятся в строке редактирования (в этом случае она называется Output times) в виде списка, заключенного в квадратные скобки. При использовании этого варианта базовый шаг регистрации Δt равен 1. Значения времени в списке Output times могут быть дробными числами и иметь любую точность;

3) Produce specified output only (формировать только заданный вывод) – устанавливает вывод параметров модели только в заданные моменты времени, которые указываются в поле Output times (моменты времени вывода).

Элементы, позволяющие управлять вводом и выводом в рабочую область MATLAB промежуточных данных и результатов моделирования, расположены на вкладкеWorkspace I/O.

Элементы вкладки разделены на три поля:

1) Load from workspace (загрузить из рабочей области). Если флажок Input (входные данные) установлен, то в расположенном справа текстовом поле можно ввести формат данных, которые будут считываться из рабочей области **MATLAB**. Установка флажка Initial State (начальное состояние) позволяет ввести в связанном с ним текстовом поле имя переменной, содержащей параметры начального состояния модели. Данные, указанные в полях Input и Initial State, передаются в исполняемую модель посредством одного или более блоков In (из раздела библиотеки Sources);

2) Save to workspace (записать в рабочую область) – позволяет установить режим вывода значений сигналов в рабочую область **MATLAB** и задать их имена;

3) Save options (параметры записи) – задает количество строк при передаче переменных в рабочую область. Если флажок Limit rows to last установлен, то в поле ввода можно указать количество передаваемых строк (отсчет строк производится от момента завершения расчета). Если флажок не установлен, то передаются все данные. Параметр Decimation (исключение) задает шаг записи переменных в рабочую область (аналогично параметру Refine factor вкладки Solver). Параметр Format (формат данных) задает формат передаваемых в рабочую область данных. Доступные форматы Array (массив), Structure (структура), Structure With Time (структура с дополнительным полем – время).

Вкладка Diagnostics позволяет изменять перечень диагностических сообщений, выводимых **Simulink** в командном окне **MATLAB**, а также устанавливать дополнительные параметры диагностики модели.

Запуск расчета выполняется с помощью выбора пункта меню Simulation/Start или кнопки на панели инструментов. Процесс расчета можно завершить досрочно, выбрав пункт меню Simulation/Stopили кнопку на панели инструментов. Расчет также можно приостановить (Simulation/Pause) и затем продолжить (Simulation/Continue).

Пример выполнения работы

В качестве примера рассмотрим моделирование оптимального фильтра для обнаружения прямоугольного сигнала на фоне помех. Из теории обработки сигналов известно, что передаточная функция такого фильтра имеет следующий вид:

$$H(\omega) = \frac{S^*(\omega)}{W_q(\omega)},$$

где $S^*(\omega)$ – сопряженный спектр сигнала;

$W_q(\omega)$ – спектральная плотность мощности шума.

Для прямоугольного сигнала амплитудой A и длительностью τ спектр будет иметь следующий вид:

$$S(\omega) = A\left(1 - \exp\left(-j\omega\tau\right)\right) / \; j\omega\,.$$

Примем, что на входе системы действует белый шум, т. е.

$$W_q(\omega) = \text{const}\,.$$

Тогда передаточная функция такого фильтра имеет следующий вид:

$$H(\omega) = \alpha\left(1 - \exp\left(-j\omega\tau\right)\right) / \; j\omega\,,$$

где α – постоянный коэффициент пропорциональности.

Анализ полученного выражения позволяет синтезировать фильтр с соответствующей характеристикой. Выражение $exp(-j\omega\tau)$ реализуется в виде задержки Transport Delay на величину длительности импульса τ. На рисунке 8.7 приведен пример модели, описывающей такой фильтр.

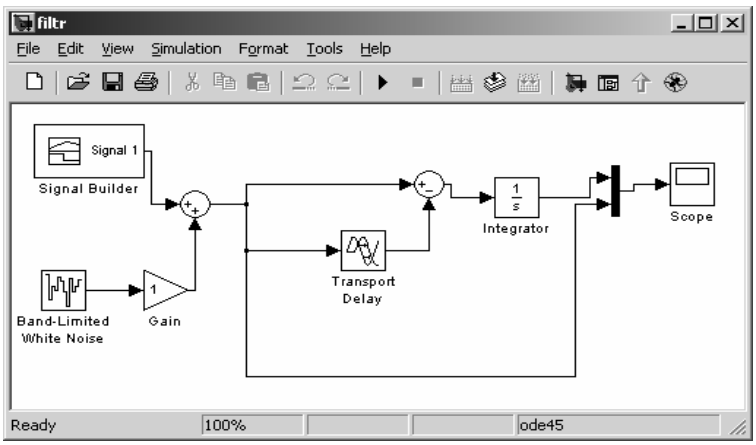


Рисунок 8.7– Модель оптимального фильтра

В модели использованы следующие блоки (рисунок 8.8):

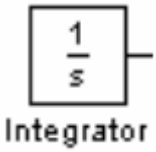
– блок из раздела **Sources** для построения сигналов произвольной формы;

– генератор белого шума из раздела **Sources;**

– усилитель сигналов из раздела **Math Operations**;

– арифметическое суммирование двух сигналов из раздела **Math Operations**;

– линия задержки сигнала из раздела **Continues**;



– блок накопления (интегрирования) из раздела **Continues**;

Рисунок 8.8 – Блоки, используемые в модели

– мультиплексор из раздела **Signal Routing**;

– осциллограф из раздела **Sinks**.

Окончание рисунка 8.8

Результат моделирования приведен на рисунке 8.9.

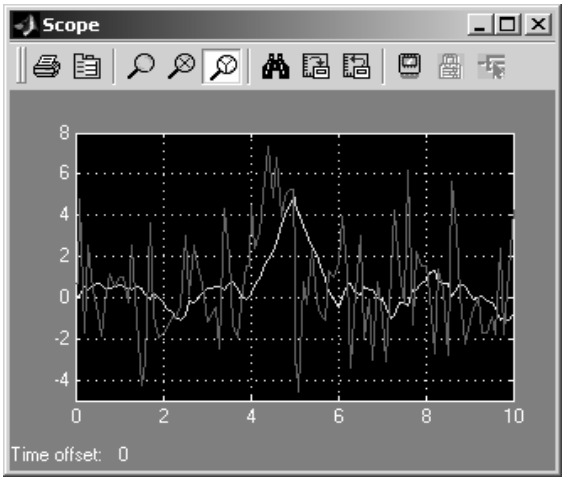


Рисунок 8.9 – Результаты моделирования

8.2 Практическое задание

Синтезировать фильтр по заданию преподавателя и выполнить моделирование всреде **Simulink.**

Контрольные вопросы

1 Что такое среда Simulink?
2 Для чего может использоваться среда Simulink?
3 Какие разделы содержит библиотека Simulink?

Список литературы

1 **Потемкин, В. Г.** MATLAB 6: среда проектирования инженерных приложений / В. Г. Потемкин. – Москва: ДИАЛОГ-МИФИ, 2003. – 448 с.

2 **Гультяев, А.** Визуальное моделирование в среде MATLAB: учебный курс / А. Гультяев. – Санкт-Петербург: Питер, 2002. – 432 с.