Федеральное агентство по образованию Российской Федерации

Московский инженерно-физический институт
(государственный университет)

Г.П. Аверьянов, В.А.Будин, В.В.Дмитриева

Автоматизация проектирования

Часть 1. Решение задач электрофизики в системе MATLAB

Компьютерный практикум

Москва 2009

УДК 004.9(075)
ББК 32.973.202-04я7
А19

Аверьянов Г.П., В.А.Будкин, Дмитриева В.В. **АВТОМАТИЗАЦИЯ ПРОЕКТИРОВАНИЯ. КОМПЬЮТЕРНЫЙ ПРАКТИКУМ. ЧАСТЬ 1. РЕШЕНИЕ ЗАДАЧ ЭЛЕКТРОФИЗИКИ В СИСТЕМЕ MATLAB:** *учебное пособие.* М.: МИФИ, 2009. — 111 с.

В учебном пособии рассматривается использование одного из наиболее известных и широко распространенных математических пакетов MatLab при проектировании электрофизических установок (ЭФУ) в рамках цикла занятий САПР ЭФУ.

В сжатой и компактной форме представлены необходимые данные по структуре пакета, правилам программирования в его среде, а также необходимый справочный материал по использованию встроенных математических библиотек, характерных для задач электрофизики, радиотехники и ускорительной техники.

Большое количество примеров проведения простейших расчетов, решения традиционных задач вычислительной математики значительно облегчает решение представленных во второй части пособия задач, характерных для электрофизики.

Тематика задач включает расчет электромагнитостатических электродинамических полей в различных структурах, динамики заряженных частиц в этих полях, их фокусировка и устойчивость.

Пособие предназначено для студентов дневного и вечернего отделений факультета «Автоматика и электроника» по специальностям «Физика пучков заряженных частиц и ускорительная техника», а также «Автоматика и электроника физических установок».

Рекомендовано к изданию редсоветом МИФИ *в качестве учебного пособия*

Рецензент канд. техн. наук, доц. В.М.Барбашов

ISBN

ОГЛАВЛЕНИЕ

Предисловие

Компьютерный практикум, связанный с изучением и возможностями использования ППП MATLAB для характерных задач электрофизики и ускорительной техники, входит в комплекс лабораторных практикумов по автоматизированному проектированию электрофизических установок. Он является естественным продолжением курса «Информатика и программирование» и практически осуществляет переход на более высокий уровень программирования математических задач.

Цель данного компьютерного практикума – дать студентам возможность применить в комплексе знания, полученные на младших курсах университета по общей физике и высшей математике для решения задач по электрофизике. Материалы по некоторым разделам электрофизики требуют самостоятельного изучения, так как соответствующие курсы читаются параллельно или будут прочитаны в дальнейшем. Такой подход способствует развитию интереса к своей специальности и проявлению творческой инициативы. Тематика задач в основном связана с рассмотрением статических электрических и магнитных полей, динамикой заряженных частиц в электромагнитных полях, а также с рассмотрением реальных элементов электрофизических установок.

Существует очень большое количество математических пакетов как численного, так и символьного подхода к решению подобных задач. Однако выбор системы для первоначального изучения работы с подобными пакетами не является принципиальным, так как их структура и возможности примерно одинаковы.

Язык MATLAB имеет ряд характерных особенностей. Так различие в понятии команд (выполняемых при вводе с клавиатуры) и операторов (выполняемых из программы) является достаточно условным. И команды и операторы как из программы, так и в режиме прямых вычислений выполняются однозначно. Под командами, в основном, понимаются средства управления периферийным оборудованием, под операторами – средства, выполняющие операции с операндами и данными.

MATLAB имеет удобный интерфейс, простые средства программирования, обширную библиотеку внутренних функций, возможность быстрого просмотра результатов как в цифровом, так и в

графическом виде и т.п. Все это создает студентам комфортные условия для работы и возможность сосредоточиться на физическом смысле задачи, исключив рутину традиционного программирования на императивных языках.

Практикум состоит из четырех разделов:

– изучение среды MATLAB, основных приемов работы и элементов программирования;

– работа в различных режимах, организация простых вычислений;

– использование внутренних библиотек для численного решения традиционных математических задач (численное дифференцирование, интегрирование, решение СЛАУ и ОДУ, использование матричного аппарата и т.д.);

– решение традиционных задач по электрофизике и ускорительной технике.

Характерной особенностью среды MATLAB является тот факт, что основным объектом записи и обработки информации является матрица. На первом этапе программирования это составляет основную сложность, так как студент чисто инстинктивно пытается использовать традиционные алгоритмы организации циклов. Здесь требуется, прежде всего, преодоление психологического барьера, что также немаловажно для развития у студентов алгоритмического мышления.

При выполнении заданий первого раздела практикума студент должен:

– научиться работать со справочным материалом:

– освоить приемы создания и редактирования матриц;

– освоить методику написания программ.

При выполнении второго раздела студент должен научиться строить простейшие арифметические и логические алгоритмы.

При выполнении задания третьего раздела студент должен научиться решать типичные задачи различных разделов вычислительной математики.

При выполнении четвертого раздела практикума студент должен научиться применять специфику среды MATLAB для решения задач электрофизики, правильно оформлять условия задачи, выбирать модель и адекватные методы и средства решения задачи.

В описании каждой задачи дается необходимый теоретический материал и ссылки на соответствующие литературные источники.

Набор предлагаемых задач относится к следующим разделам:

– теория колебаний,
– исследование фазового пространства,
– электростатика, исследование электростатических полей,
– магнитостатика, фокусировка пучков магнитным полем,
– динамика заряженных частиц в электромагнитном поле,
– расчет СВЧ–полей.

1. ОСНОВНЫЕ ПРИНЦИПЫ РАБОТЫ И ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ СИСТЕМЫ MATLAB

1.1. Структура операционной среды системы MATLAB

Система MATLAB является одновременно и операционной средой и языком программирования. Удобная операционная среда обеспечивает эффективное использование разнообразных возможностей системы и дает возможность получать решения задач в привычной для пользователя математической форме, не прибегая к рутинному программированию. Это взаимодействие пользователя и системы осуществляется через командную строку и графический интерфейс, редактор и отладчик М-файлов (файлов системы), просмотр рабочей области, доступ к справочной информации, работу с файлами и оболочкой DOS и еще через целый ряд других возможностей. Управление средой осуществляется традиционным образом с помощью системы последовательно разворачивающихся меню. Все эти интерфейсы реализуются в командном окне, редакторе/отладчике М-файлов, подсистеме просмотра рабочей области, а также с использованием инструментальной панели.

Пользовательский интерфейс системы изменяется от версии к версии и становится все более удобным и универсальным. В пособии рассматриваются наиболее общие и часто используемые свойства и функции (в зависимости от версии они могут располагаться в разных частях меню системы) среды MATLAB.

Окно системы MATLAB

После загрузки системы на экране появляется окно системы MATLAB (рис. 1.1), главное меню которого содержит опции *FILE, EDIT, WINDOW, HELP, DESKTOP* и некоторые другие. В этом окне также располагается инструментальная панель. Опция *DESKTOP* определяет рабочую конфигурацию окон системы MATLAB. Активизируются окна – *Командное окно* (Command window), *Помощь* (Help), *Рабочая область* (Workspase), *Текущая директория* (Current Directory). Рабочая область(Workspase) и Текущая директория (Current Directory) располагаются в окне системы MATLAB. Рассмотрим только некоторые основные, наиболее часто используемые, опции (методы и средства изменения текущего состояния операционной среды) меню.

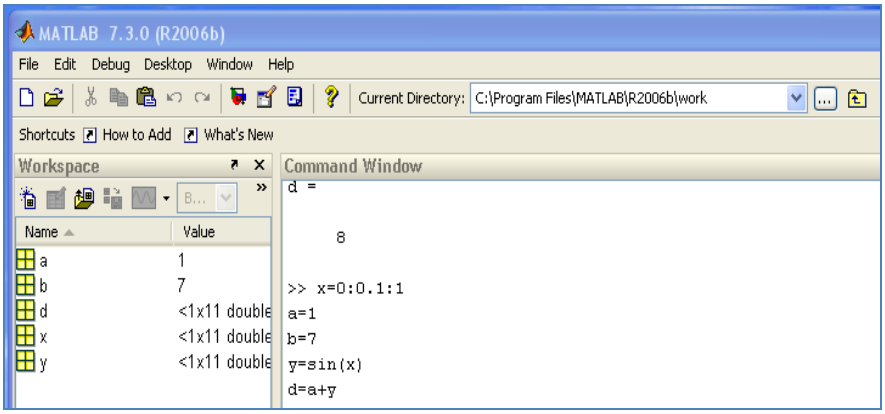


Рис. 1.1. Окно системы MATLAB

Командное окно.

Командное окно (рис. 1.2) содержит свое меню, позиции которого соответствуют главному меню системы, но с несколько меньшим набором опций. Командное окно является основным рабочим органом системы, так как в него заносятся все выполняемые команды и программы, а также выводятся результаты расчетов.

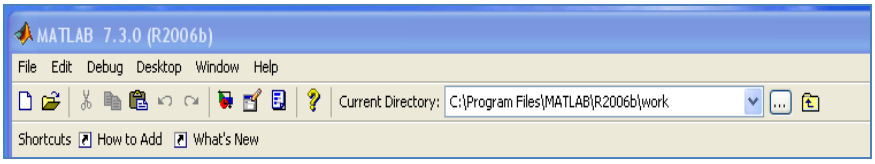


Рис. 1.2. Командное окно системы MATLAB

Прежде всего, необходимо иметь представление о том, как создать файл, вызвать его для редактирования, установить соответствующие форматы данных и прописать путь доступа к каталогу, содержащему нужные файлы. Для этого служат соответствующие опции меню *FILE – new, open, show workspase, set path, preferences, exit,* которые позволяют эффективно работать с файловой системой.

Опция *new* выполняет две функции:

открывает в редакторе/отладчике новый текстовый файл для написания новой программы (подопция *M-file*);

открывает графическое окно (подопция *Figure*) для просмотра результатов расчета в виде графиков.

Опция *open* открывает для редактирования и просмотра в редакторе/отладчике текстовый файл, хранящийся в каталоге пользователя. Для этого необходимо в открывшемся окне «Открытие документа» найти нужный файл и загрузить его в редактор нажатием клавиши «Открыть» или двойным щелчком мыши на имени файла.

Опция *exit matlab* завершает работу системы MATLAB.

Опция *show workspase* (или просто *workspase)* осуществляет просмотр рабочей области памяти системы MATLAB, в которой размещены все данные, участвующие в текущем сеансе работы. В рабочей области хранится информация обо всех именах переменных и констант, о размерности массивов, о размере занимаемой ими памяти (в байтах) и о типе переменных. Характерной особенностью системы MATLAB является то, что размерность и размер памяти, отводимой переменной, определяются при каждом новом обращении к ней в процессе выполнения программ (старое содержимое затирается). Иногда в данном сеансе работы при большом объеме вычислений рабочая область может быть забита различным «мусором» – пробными расчетами различных вариантов задач, которые

студенты создают в изобилии. Чтобы избежать подобных ситуаций, нужно периодически очищать рабочую область памяти полностью или частично. Для этого используется команда «*clear*» (рассматривается ниже). Открыв окно рабочей области, можно просмотреть содержимое ячеек любого элемента программы. Для этого необходимо в окне рабочей области (workspase) выделить мышью нужную переменную и щелкнуть мышью два раза. Подобную информацию можно получить, используя команды «*who*» и «*whos*». В появившейся двумерной таблице можно не только просматривать результаты расчетов, но и вносить напрямую необходимые изменения в содержание ячеек массива переменной. Удалить переменную или группу переменных из рабочей области можно, выделив их мышью и нажав клавишу «DEL». Просмотр содержимого рабочей области может помочь при отладке программы, хотя для этого существует в MATLAB традиционное средство отладки “Deburg”.

Опция *preferences* определяет выбор режимов и параметров работы командного окна, редактора/отладчика, форматы копирования и т.д. Здесь же можно установить числовые и текстовые форматы записи информации.

Опция *set path* устанавливает пути доступа к рабочим файлам. Файлы, созданные пользователем, сохраняются по умолчанию в каталоге “*work”* самой системы MATLAB (например, C:/MATLAB701/WORK). Для сохранения файлов в собственном рабочем каталоге (или их вызова) необходимо установить путь доступа к этому каталогу опцией *set path* или использовать проводник инструментальной панели.

Меню *EDIT* содержит стандартные функции редактирования текстов – копирование, удаление, занесение в буфер, извлечение из буфера. Меню содержит также три процедуры – очистка командного окна (*clear command window*), очистка рабочей области (*clear workspace*), очистка буфера ранее исполненных команд (*clear command history*). Полезно периодически в течение сеанса работы производить очистку командного окна от результатов предшествующих расчетов. Рабочую область можно очищать, используя команды:

Clear – удаляются все переменные из рабочей области;

Clear имя1 имя2 имя3 – удаляются из рабочей области все переменные списка;

Clear global имя1 имя2 имя3 – удаляются из рабочей области все глобальные переменные списка.

Меню *WINDOW* содержит перечень активных окон, а также перечень последних открываемых файлов.

Меню *HELP* (рис. 1.3) содержит опции, дающие справку по интересующему нас объекту.

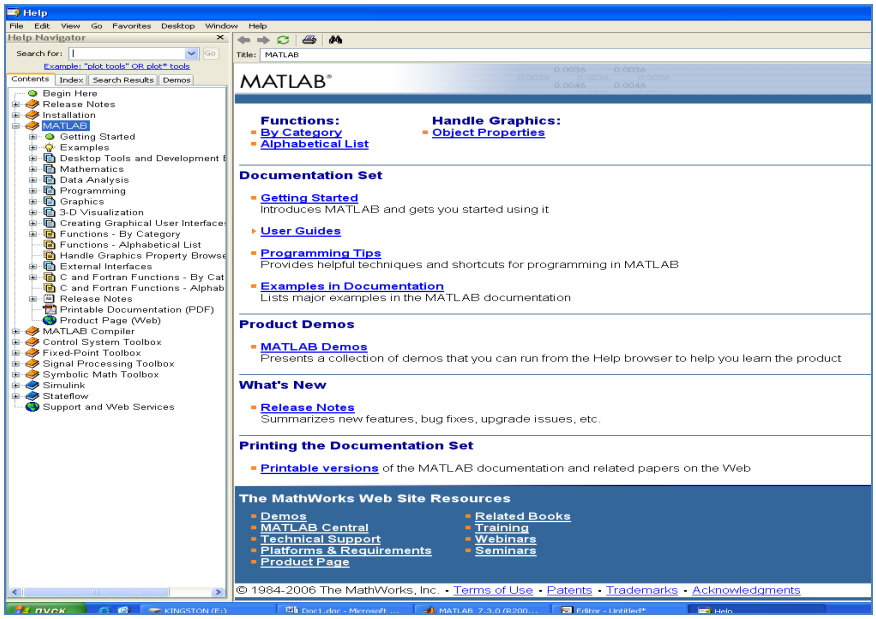


Рис. 1.3. Окно HELP системы MATLAB

Инструментальная панель

Инструментальная панель окна системы MATLAB, располагающаяся в верхней части окна, обеспечивает быстрый доступ к операциям над М-файлами: создание нового файла; открытие существующего файла; удаление, копирование, вставку фрагмента; просмотр рабочей области и путей доступа, а также обращение к текущей помощи. В целом она повторяет действия опций главного меню. Свои панели инструментов, с несколько меньшими возможностями, имеют также и другие окна – Редактор/отладчик М-файлов, Рабочая область и Помощь.

Редактор/отладчик М-файлов

Редактор/отладчик системы MATLAB (рис. 1.4) является основным средством написания программ и их отладки. Он может быть вызван из командной строки командного окна командами «edit», «edit имя_М-файла» или из главного меню системы, используя опции «*file – new – M-file*» (для открытия нового файла), «*file – open*» (для открытия существующего файла), а также через соответствующие им стандартные значки инструментальной панели. Функции редактора/отладчика типичны для программных инструментов такого класса. Созданная в редакторе/отладчике программа копируется в командное окно системы и посылается на исполнение (в режиме интерпретации). При наличии ошибок на стадиях трансляции, компоновки или исполнения программы в командное окно выводится необходимая диагностика ошибок. При отсутствии ошибок производится выполнение программы и вывод результатов на экран (или в файл) в текстовом или графическом виде. Отличительной чертой редактора/отладчика системы MATLAB является возможность выделять из текста программы отдельные фрагменты и выполнять их самостоятельно в командном окне. Быстро просмотреть полученные результаты расчета, в т.ч. и промежуточные, можно в рабочей области памяти.

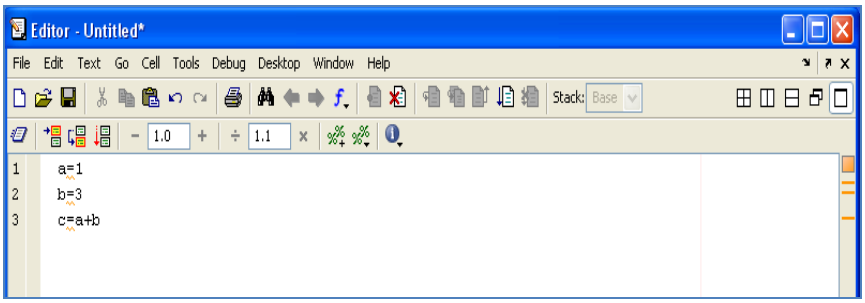


Рис. 1.4. Окно редактор/отладчик системы MATLAB

Практический совет. С чего начинается создание программы?

1. Из главного меню системы «*file – new – M-file*» вызывается редактор/отладчик.

2. В окне редактора/отладчика записывается текст программы.

3. Производится отладка и расчет программы путем копирования её (или её фрагментов) в командное окно.

4. Текущий каталог пользователя устанавливается из главного меню системы «*file – set path*» или с использованием проводника главной инструментальной панели MATLAB.

5. Готовая программа сохраняется в текущем каталоге (каталоге пользователя).

1.2. Справочная информация и способы доступа к основным используемым разделам

Прежде всего выделим разделы справочной информации, которые обязательно потребуются при написании любой, даже самой простой, расчетной программы:

matlab\general – список команд: доступ к информации, работа с рабочей областью памяти, установление путей доступа, работа с операционной системой, контроль рабочего окна и другие функции;

matlab\ops – арифметические, логические операторы и операторы отношений, специальные символы и другие функции;

matlab\lang – основные языковые конструкции (условные операторы, циклы и т.д.), вычислительные функции;

matlab\elmat – операции с матрицами и массивами, матрицы специального типа;

matlab\elfun – элементарные математические функции;

matlab\specfun – специальные математические функции;

matlab\matfun – матричные функции, линейная алгебра;

matlab\polyfun – интерполяция и полиномы;

matlab\funfun – дифференцирование, интегрирование, решение ОДУ, оптимизация;

matlab\graph2d – двухмерная графика;

matlab\graph3d – трехмерная графика;

matlab\specgraph – специальная графика.

Доступ к указанным разделам производится из командного окна набором команд:

> *help* – перечень всех разделов помощи;

> *help имя_раздела* – доступ к конкретному разделу;

> *help имя_раздела\имя_функции* – доступ к конкретной функции.

Более подробную справочную информацию по интересующему вопросу можно получить, используя меню “HELP”. В окне “HELP NAVIGATOR” - “SEARCH” нужно набрать имя соответствующего раздела или подраздела. (Организация “HELP” может несколько отличаться для различных версий MATLAB). На первом этапе работы нужно обратить внимание на следующие разделы и подразделы, в которых описаны операторы и операции над различными типами данных. Разделы *Operators* и *Operations* содержат подробную информацию по вопросам, приведенным ниже.

Special Characters. В этом разделе помощи содержится полная информация о правилах формирования матриц и векторов, использования различных типов скобок, символов – «,», «.», «:», «;», «=», «’» ,«%» ,«!», «@» ,«..»

Arithmetic Operations. В этом разделе помощи содержится полная информация о правилах проведения арифметических операций над матрицами и векторами, а также модифицированных операций над массивами. Здесь располагается информация о правом и левом делении массивов, преобразовании их по схеме Холецкого, факторизации, вычислении детерминанта, инверсии матриц и других операций (*mldivide, mrdivide, chol, det, inv, lu*). Показано применение этих функций при решении систем линейных алгебраических уравнений.

Bit-wise Operations. Раздел содержит перечень операций и функций побитовой обработки информации (например, операции логического умножения или сложения).

Date and Time Operations. Раздел содержит перечень функций, оперирующих с текущим временем и датами.

Relational Operations. Раздел содержит перечень логических отношений (<, >, <=, >=, ==, ~=).

Logical Operations. Раздел содержит перечень логических операций над отдельными переменными и массивами (логические ***«и»*** и ***«или»***, анализ нулевых и ненулевых элементов массивов и другие операции), используемых при построении сложных логических выражений.

Set Operations. Раздел содержит перечень операций обработки множеств.

В процессе написания программы немаловажно освоить удобный способ получения помощи по различным разделам системы.

Быстрый способ нахождения справки по функциям – обращение к разделам:

Function Reference – перечень разделов помощи MATLAB (самый общий поход);

Functions - Alphabetical List – перечень в алфавитном порядке функций MATLAB по тематическим разделам;

Mathematics – перечень разделов математики с указанием вложенных подразделов с более детальным рассмотрением методов и функций.

В разделе *Mathematics* – содержатся часто используемые подразделы, связанные с нелинейными численными методами и матрицами.

Nonlinear Numerical Methods.

Ordinary Differential Equations – обыкновенные дифференциальные уравнения;

Equations Boundary – дифференциальные уравнения с граничными условиями;

Partial Differential Equations – дифференциальные уравнения в частных производных;

Optimization – методы оптимизации;

Numerical Integration (Quadrature) – численное интегрирование.

Matrix.

Справка по вопросам различных действий и преобразований матриц.

Практический совет. Как работать с помощью?

На первом этапе обучения более быстрым доступом к справочной информации может оказаться доступ через командную строку в командном окне.

В дальнейшем нужно более подробно изучить структуру построения раздела «help» данной версии MATLAB, запомнив расположение интересующих вас тематических разделов.

1.3. Программирование в системе MATLAB

Язык системы MATLAB можно считать языком программирования сверхвысокого уровня, предназначенного для решения математических задач. Программирование может осуществляться в двух режимах.

Режим прямых вычислений – *режим командной строки***.** В этом режиме можно производить расчет программы последовательно по операторам и командам, в том числе с использованием внутренних функций библиотеки MATLAB. Результаты расчетов представляются в табличном и графическом виде. Режим командной строки удобно использовать при оценочных расчетах. Последовательность выполняемых операций хранится в достаточно ёмком командном буфере, что позволяет, в случае необходимости, вернуться к выполнению любого шага в последовательности введенных команд.

Работа в *редакторе/отладчике*. Для решения сложных задач (сложных алгоритмов) система MATLAB предоставляет пользователю возможность заранее подготовить текст программы (последовательность команд и операторов) в окне *редактора/отладчика*, который записывается в файл. Затем этот файл загружается в командное окно для обработки и исполнения.

Система работает в режиме интерпретации команд и операторов – они вводятся в ходе сеанса работы в командной строке и система выполняет их обработку и выполнение. Результаты расчета (значения всех переменных и констант программы) хранятся в рабочей области, поэтому повторный расчет нового варианта выполненной программы можно начинать непосредственно с того места программы (команды), в котором вносятся новые данные относительно уже выполненного варианта.

Характерной особенностью программирования в системе MATLAB является наличие большого количества внутренних функций (системных и математических), обращение к которым происходит просто по их имени с указанием необходимых факти-

ческих параметров. Пользователь может создать свою личную библиотеку функций, которые оформляются так же как и внутренние функции. Имя функции пользователя должно совпадать с именем файла, в котором она хранится. Сохранение файла производится (по умолчанию) в рабочую папку MATLAB, например, *C:\MATLAB701\WORK*. Для сохранения или обращения к функциям, размещенным в личном каталоге, нужно сделать этот каталог текущим (используя меню *FILE* в командном окне, установить путь доступа к файлам *SET PATH*). Обращение к функциям личной библиотеки такое же, как к внутренним функциям.

1.3.1. Основные типы данных языка MATLAB

В данном разделе будут рассматриваться только основные принципы организации и приемы составления программ. В системе MATLAB имеется шесть базовых классов данных, которые являются тем или иным видом массива (рис. 1.5):

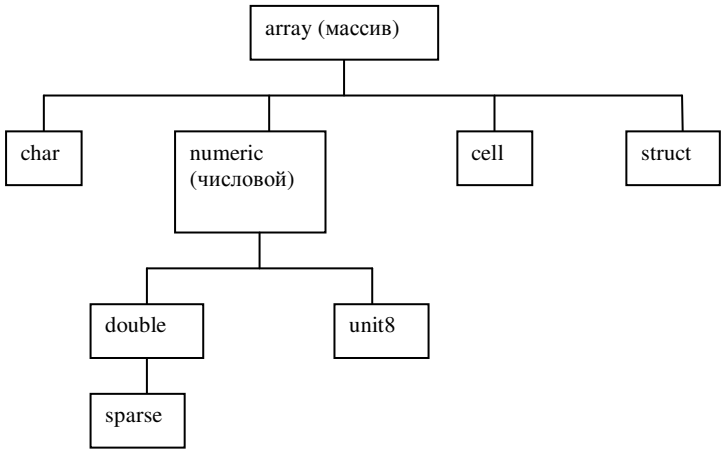


Рис. 1.5. Базовые классы данных

числовой массив удвоенной точности – Double;
массив символов – Char;
разреженная матрица – Sparse;
массив ячеек – Cell;
массив записей – Struct;
массив 8-разрядных целых чисел без знака – Unit8.

В большинстве случаев используются два класса – числовой массив удвоенной точности (Double) и массив символов (Char), или просто строка. Вышеназванные типы называются *встроенными*, для них в системе MATLAB определены свои методы обработки. Дочерние классы данных (расположены по схеме ниже родительского класса) поддерживаются также и методами родительского класса.

К массиву данных *array* применимы методы определения его размера, длины, размерности, транспонирования, объединения массивов, многомерной индексации, переопределения и перестановки размерностей многомерного массива.

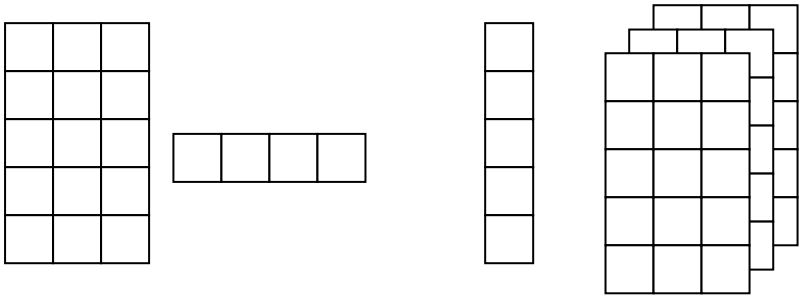
К дочернему классу данных *numeric* применимы методы формирования векторов, выделения строк и столбцов, выделения подблоков массива, обработки комплексных чисел. К дочернему классу данных *double* применимы методы арифметических и логических операций, различных математических функций и функций от матриц.

К дочернему классу данных *char* применимы методы работы со строковыми функциями и методы преобразования строк в тип *double.*

Поскольку основной элемент записи информации в системе MATLAB – массив, то нет необходимости описывать предварительно его тип и размерность. Любая встретившаяся в программе переменная – массив:

скалярная величина – массив размерности *1*1*;
вектор – массив размерности *N*1*;
матрица – массив размерности *M*N* и т.д.

Размерность массива определяется количеством индексов в имени переменной. Например, *A(3,4), DD(1,4), BB(5,1), CD(4,10,2).* В произвольной форме записи – *ABC(i, j)*, где *i* и *j* предварительно определены.



а) *б)* *в)* *г)*

Рис. 1.6. Графическая интерпретация типов массивов

$$\begin{vmatrix} 1 & 3 & 8 & 9 \\ 5 & 7 & 0 & 9 \\ 0 & 5 & 3 & 7 \end{vmatrix}, \quad \begin{vmatrix} 4 & 5 & 7 & 9 \end{vmatrix}, \quad \begin{vmatrix} 4 \\ 5 \\ 1 \end{vmatrix}, \quad \begin{vmatrix} a(1,1) & a(1,2) & a(1,3) \\ a(2,1) & a(2,2) & a(2,3) \\ a(3,1) & a(3,2) & a(3,3) \end{vmatrix}.$$

Каждая клетка – это элемент матрицы, который характеризуется своим набором индексов. В MATLAB существует несколько определений массивов: вектор-столбец (рис. 1.6,*в*) соответствует классическому представлению вектора, двухмерная матрица (рис. 1.6,*а*) соответствует классическому представлению матрицы, вектор-строка (рис. 1.6,*б*), многомерная матрица – многомерный массив (рис. 1.6,*г*). Нумерация и порядок следования индексов элементов *MAS(I,J,K,L,…)* следующие: первый индекс – номер строки, в которой располагается этот элемент, второй индекс – номер столбца, третий индекс – номер слоя и т.д. *Индексы могут принимать только целые положительные значения.*

В приведенном примере вектор-столбец (рис. 1.6,*в*) состоит из одного столбца (j=1) и четырех строк (i=1÷4). Матрица (рис. 1.6,*а*) состоит из четырех строк (i=1÷4), и трех столбцов (j=1÷3). Вектор-строка (рис. 1.6,*б*) состоит из одной строки (i=1) и четырех столбцов (j=1÷4). Массив (рис. 1.6,*г*) состоит из четырех строк (i=1÷4), трех столбцов (j=1÷3) и трех слоев (k=1÷3).

Набор индексов однозначно определяет только *местоположение ячейки, а не ее содержимое.* Содержимое ячеек (их заполнение) определяется условиями решаемой задачи.

С вектором-столбцом, двухмерной матрицей, вектором-строкой можно проводить как обычные матричные операции, так и специальные операции над массивами. Для многомерных массивов допустимы только специальные операции.

Рассмотрим различие этих операций на примере умножения двух прямоугольных матриц *A* и *B*. Для обычного умножения матриц *C=A*B* необходимо, чтобы число строк матрицы *A* равнялось числу столбцов матрицы *B,* т.е.

$$C(i, j) = \sum_{i,j=1,N} A(i, j) * B(j, i) .$$

Размерности матриц – *A(M*N), B(N*M), C(M*M).*

Для специального умножения матриц *C=A .*B*, необходимо чтобы они были одинаковой размерности *(N*M).* Производится попарное (унарное) перемножение элементов массивов с одинаковыми индексами. Результат заносится в ячейку результирующего массива с теми же индексами. Признаком специальной операции над массивами является десятичная точка перед стандартным символом операции:

$$C(i, j) = A(i, j) * B(i, j) .$$

Если во многих других языках программирования подобные операции требуют организации циклов, то в MATLAB они реализуется записью простого оператора *C=A*B* для классического умножения матриц и оператора *C=A .* B* для унарного умножения.

Еще раз отметим, что понятие матрицы в системе MATLAB понимается значительно шире, чем в линейной алгебре. Под матрицей понимается массив, с которым можно осуществлять как «истинные» матричные операции, так унарные операции.

1.3.2. Операторы языка

1.3.2.1. Операторы повторения

Поскольку объектом обработки данных, в отличие от многих языков программирования, в системе MATLAB является матрица (массив), то при написании алгоритма в значительной степени уменьшается необходимость использовать циклы. Однако в ряде случаев реализовать алгоритм программы без применения циклов просто невозможно. Приведенные варианты организации циклов связаны с формированием матриц и векторов. Предварительно создаются матрицы нужного размера, заполненные нулями, или создаются пустые матрицы. В качестве цикловой переменной используются как целочисленные, так и вещественные векторы. Структура цикла обязательно заканчивается оператором *end*.

Цикл с фиксированным конечным значением целочисленной цикловой переменной:

```
k=5;
a = zeros(k,k);  % предварительная запись нулевой матрицы
for m = 1:k
   for   n = 1:k
      a(m,n) = 1/(m+n -1);
   end
end
```

Цикл с вещественной цикловой переменной в виде вектора (массив с шагом -0.1):

```
x=[];
for   s = 1.0: -0.1: 0.0
      x=[x sin(s)];
end
x
```

Цикл с использованием в качестве цикловой переменной матрицы специального вида:

```
n=4;
x=[];
for   e = eye(n)
```

```
 x=[x cos(e)];
end
x
```

Цикл с предусловием:

```
eps = 1;
while (1+eps) > 1
   eps = eps/2;
end
eps = eps*2
```

Цикл с вещественной цикловой переменной в виде произвольного вектора:

```
x=[];
for   v=[0  2  3  1]
   x=[x   2.^v];
end
x
```

1.3.2.2. Условный оператор

В структуре условного оператора возможны две формы продолжения вычислений в случае невыполнения логического условия:

else – просто выполняются соответствующие данной ситуации операции (т.е. алгоритм предполагает два варианта продолжения программы);

elseif – проверяется ещё одно логическое условие (т.е. алгоритм предполагает три варианта продолжения программы).

Допустимы несколько уровней вложенности условного оператора. Структура условного оператора обязательно заканчивается оператором *end*.

```
   for  n = 1:k
      if  m == n
         a(m,n) = 2;
      elseif  abs(m-n) == 2
         a(m,n) = 1;
      else
         a(m,n) = 0;
```

```
      end
   end
end
```

1.3.2.3. Оператор выбора

Оператор переключения

switch <*выражение*> – скаляр или строка;
case <*значение_1*> – выполняются операции соответствующие равенству <выражение>=<значение_1>;
case <*значение_2*> – выполняются операции соответствующие равенству <выражения> одному из возможных его значений;
otherwise – выполняются операции при всех прочих значений <выражения>;
end

Структура оператора выбора обязательно заканчивается оператором *end.*

```
v = [2 4 1 5 3]
for j = 1:5
   i = v(j);
  switch i
    case 1
     t(i,:) = 'variant_1';
    case 2
     t(i,:) = 'variant_2';
    case 3
     t(i,:) = 'variant_3';
    case 4
     t(i,:) = 'variant_4';
    case 5
     t(i,:) = 'variant_5';
  end
end
t
```

1.3.3. Сценарии и функции

Язык программирования системы MATLAB ориентирован на модульное программирование. Файлы, содержащие последовательность команд и операторов MATLAB, называются *М-файлами.* Они подразделяются на два типа:

М-сценарии;

М-функции.

Можно провести сопоставление программ, написанных на языке MATLAB и программ, написанных, например, на Fortran, Pascal или Си. В этих языках существует понятие основной программы (main), подпрограмм (Procedure или Subroutine) и функций (Function). Аналогом основной программы (main) в MATLAB является М-сценарий, аналогом подпрограмм и функций является М-функция MATLAB.

Возможны два варианта сохранения основной программы в исходном коде:

программа сохраняется в файле (М-сценарии) с произвольным именем с расширением «*.m*». Например, «*proba.m*»;

Script
% M-file proba1 – сценарий пробной программы
[Последовательность операторов и функций]
Конец программы

Команда *Script* задает заголовок сценария и записывается в первой строке. Во второй и последующих строках, начинающихся с символа **%**, записываются комментарии к программе. В комментариях пользователь может записать всю необходимую ему справочную информацию, содержащую сведения физической и математической постановке задачи, и о методе ее решения. При вызове в командной строке помощи «**>>***help имя_ Script_файла*» пользователь может просмотреть всю эту справку. Вызов программы на исполнение производится в командном окне набором в командной строке имени файла.

Программа сохраняется в файле с произвольным именем с расширением «.m», например, «*proba1.m*». В тексте программы отсут-

ствует команда *Script.* По умолчанию она сохраняется как М-сценарий.

М-функции являются инструментом создания библиотек функций и пакетов прикладных программ. Они используются так же, как обычные встроенные функции системы MATLAB. Новые функции, написанные пользователем, добавляются к словарю системы MATLAB и становятся доступными наряду со встроенными функциями. Они оформляются в виде текстовых файлов с расширением «*.m*». Функция в системе MATLAB не имеет никаких отличительных признаков, и при обращении к ней MATLAB просто ищет файл с соответствующим именем. После того как файл найден, он проходит процедуру компиляции, размещается в оперативной памяти и только потом выполняется. В этом он отличается от Script-файла, который выполняется только в режиме интерпретации. Если при выполнении функции включен режим *echo*, то функция тоже будет выполняться в режиме интерпретации и на терминал выводится результат выполнения каждого оператора.

М-функции являются файлами, которые допускают наличие входных и выходных аргументов (в отличии от М-сценариев). Они работают с переменными в *пределах собственной рабочей области, отличной от рабочей области системы MATLAB.*

М-функция имеет следующую структуру записи:

Function[<*список выходных переменных*>]=<*имя_функции* >
(< список входных переменных >)
% комментарии
……
% комментарии
Тело функции

Первая строка задает имя, количество и порядок следования входных и выходных аргументов.

Во второй и последующих строках, начинающихся с символа %, записываются комментарии к функции, в которых пользователь может записать всю необходимую ему справочную информацию

Тело функции – программный код, реализующий вычисления и присваивающий выходным параметрам вычисленные значения, которые передаются в вызывающую программу.

М-функция сохраняется в *файле, имя которого должно полностью совпадать с именем функции*. При обращении в вызывающей программе к функции, операционная система MATLAB отыскивает в списке пользовательских функций М-файл с соответствующим именем и производит необходимые действия в обработке этого файла.

Сама М-функция может внутри себя содержать вложенные подфункции, которые строятся по тому же принципу, что и она сама. Имя файла, в котором сохраняется эта структура, должно соответствовать только имени М-функции.

Таким образом, программа, написанная на языке MATLAB, представляет собой комплекс М-файлов (сценарий и соответствующие функции), т.е. модули, взаимодействие которых и порядок выполнения прописаны в сценарии.

1.3.4. Глобальные переменные

Как правило, каждая М-функция использует свои локальные переменные, которые изолированы от переменных других функций и рабочей области. Однако нередкими являются ситуации, когда одни и те же переменные необходимо использовать в сценарии и различных функциях. В этом случае переменные объявляются как *глобальные.* Присваивание значений глобальной переменной возможно из любой функции, в которой она объявлена. При объявлении глобальной переменной нужно учитывать следующее:

при использовании глобальной переменной следует соблюдать осторожность, так как ее значение может быть изменено в результате выполнения какой-либо М-функции, в которой эта переменная объявлена;

если глобальная переменная на момент объявления не существует, то ей присваивается значение пустого массива;

если при объявлении глобальной переменной оказывается, что в рабочей области уже существует переменная с таким именем, то возникает ошибка.

Объявление глобальных переменных производится следующим образом:

*global имя*1 *имя*2 .. *имяN*

Имена объявляемых переменных разделяются пробелом.

Практический совет.

Если Ваша программа достаточно объёмна, то разделите её на сценарий и функции. Функции должны представлять собой логически законченные фрагменты программы.

Если вновь создаваемая функция может быть использована и в других программах, то её следует внести в свою библиотеку М-функций.

Если вновь создаваемая функция используется только в данной программе и по логике алгоритма должна обращаться к другим функциям, тогда имеет смысл оформить её как вложенную структуру «функция + подфункции».

Поскольку MATLAB различает строчные и прописные буквы, глобальные переменные можно рекомендовать записывать прописными буквами для удобного зрительного восприятия.

В сценариях и функциях желательно (в комментариях) описывать постановку и метод решения задачи.

Во избежание трудно определяемых ошибок не пользуйтесь без особой необходимости глобальными переменными. Они требуют достаточно большого опыта программирования в системе MATLAB.

1.3.5. Класс объектов inline

Введение пользователем новых типов данных и операций является этапом дальнейшего развития принципов построения программ – объектно-ориентированным программированием. Классы и объекты позволяют добавлять новые типы данных и новые операции. Класс описывает тип переменной и определяет, какие операции и функции могут быть применены к переменной этого типа.

Объект – это образец или структура некоторого класса.

В рамках данного учебного пособия рассмотрим только некоторые свойства объектов класса inline. Этот класс предназначен для описания функций в форме *f(x,p1,p2,…),* соответствующей их математическому описанию. Например, вычислим функцию *f = (a^2+b^2).*

Для этого произведем следующие операции:

ss = inline('a^2+b^2') – запись вычисляемой функции в символьном виде;

для того, чтобы произвести вычисление этой функции с произвольными значениями *a* и *b* (например, *a = 3* и *b = 4*), достаточно выполнить команду - *ss(3,4).* Результат *ss = 25.*

Для объектов класса inline в системе MATLAB определен целый ряд методов. Рассмотрим только функцию *inline*

g = inline('<выражение>')

g = inline('<выражение>','имя_1','имя_2',…)

g = inline('<выражение>', n)

Первая форма обращения формирует объект – вычисляемую функцию в символьном виде. Аргументы функции определяются автоматически.

Вторая форма обращения формирует объект – вычисляемую функцию в символьном виде, аргументы которой имеют имена *'имя_1','имя_2',…*

Третья форма обращения формирует объект – вычисляемую функцию в символьном виде, аргументы которой имеют фиксированные имена *'x', 'p1', 'p2' ,… 'pn'.*

1.4. Примеры организации вычислений

1.4.1. Редактирование массивов

Текст программы, написанной на языке MATLAB, характеризуется простым синтаксисом. Основным объектом записи и обработки информации является массив (матрица). Как его создать?

Формирование записей массива должно удовлетворять требованиям:

элементы массива должны быть однотипными;

строки должны быть одинаковой длины (одинаковое число элементов);

запись элементов массива производится построчно;

элементы строки разделяются либо пробелами, либо запятыми;

записи строк разделяются точкой с запятой (**;**);

запись элементов массива должна быть заключена в квадратные скобки;

матрицы на экране могут быть записаны построчно в виде привычной прямоугольной таблицы (для наглядности);

одномерные массивы, элементы которых отличаются друг от друга на некоторую постоянную величину (шаг), записываются с использованием оператора (**:**):

<начальное значение> : <шаг> : <конечное значение>.

Для целочисленного вектора с шагом равным «+1» (только для
него!) запись сокращается:

<начальное значение> : <конечное значение>.

Квадратные скобки могут быть опущены:

двумерные (и более) массивы могут быть построены из строк, разделены строки символом «;» и объединены в единую запись квадратными скобками;

непосредственное заполнение элементов массива и подтверждение ввода данных нажатием клавиши «ENTER» автоматически устанавливает размерность массива и тип данных, которые размещаются соответствующим образом в рабочей области памяти.

Примеры записи массивов:

a = [1,4,6,2]; или a1 = [1 4 6 2];
b = [1 3 2 4; 6 5 3 7]; или b1 = [1 3 2 4; 6 5 3 7]
c = 1:2:8; c1 = 3:12; c2 = 8:-1:1; d = 0.1:0.05:1.0;

При записи векторов и массивов необходимо помнить следующее:

элементы строки могут быть числами, символами и арифметическими выражениями;

если элементами строки являются арифметическими выражениями, то в записи этих выражений не должно быть пробелов;

элементы строк векторов и матриц могут сами являться векторами и матрицами меньшей размерности.

После записи каждый элемент массива однозначно определяется набором своих индексов, т.е. обращение к вышеприведенным мас-

сивам даст соответственно следующие результаты: *a(2) = 4, b(1,3) = 2, b1(2,2)= 5*. Поскольку изначально система MATLAB была ориентирована на решение задач линейной алгебры с использованием матричного формализма, то индексы элементов матрицы не могут принимать нулевые и отрицательные значения.

Для резервирования места в рабочей области памяти для массива, с заранее известной размерностью, можно произвести запись только одного последнего элемента главной диагонали матрицы s(*m*,*n*) = <значение элемента>, где *m* – число строк, а *n* – число столбцов массива. В этом случае создается массив нужной размерности, все элементы которого, кроме (*m*,*n*)-го, имеют нулевые значения. Подобное же резервирование области памяти можно осуществлять, используя специальные функции создания нулевых или единичных матриц (хотя это и не является их основным предназначением). В дальнейших расчетах нулевые или единичные значения матричных элементов заменяются их фактическими значениями.

Остановимся более подробно на форме записи вектора с использованием оператора двоеточие – (**:**):

aa = 1:5 dd = 1:2:10 bb = 10:-1:0 cc = 0.25:0.05:0.75

Квадратные скобки в такой форме записи векторов опускаются:

vv = [1:5; 2:6]

В этих примерах создаются вектор-строки: «*aa*» с шагом (+1), который действует по умолчанию, «*dd*» с шагом (2), «*bb*» с шагом (-1), «*cc*» с шагом (0.05), «*vv*» – двумерная матрица размером 5*5.

Первые три вектора – целочисленные. Следовательно, подобную запись можно применять для задания индексов элементов массивов. Форма записи целочисленных массивов с использованием оператора (**:**) является очень удобным инструментом для различных операций с матрицами, где они используются в качестве индексов. Например, существует матрица «*mas*» размером 10*10. Нужно сформировать новую матрицу, элементами которой являются элементы исходной матрицы с четными индексами.

masnew = mas(2:2:10, 2:2:10)

В этом примере используется понятие массива (матрицы, вектора) как для идентификации исходного и формируемого массивов данных, так и для обозначения индексов выбираемых элементов. Для задания векторов индексов могут быть использованы любые разрешенные формы записи векторов. Можно записать:

masnew = mas(2:2:10, [2 4 6 8 10])
masnew = mas([2 4 6 8 10], 2:2:10)
masnew = mas([2 4 6 8 10], [2 4 6 8 10])

Все четыре формы записи одинаково правомерны.

Кроме привычной записи матрицы, содержащей какие-либо элементы, можно создавать пустые матрицы, не содержащие никаких элементов. Пустые матрицы используются при редактировании матриц и некоторых других случаях. Запись *x* = [ ] создает пустую матрицу *x*.

Массивы трех и более размерностей создаются подобным же образом или формируются из блоков с использованием специальных функций.

Работа с матрицами требует, прежде всего, освоение приемов редактирования:

добавление или удаление строк и столбцов;

перестановка строк и столбцов;

выборка строк, столбцов и блоков, формирование матриц из блоков;

индексирование матриц;

транспонирование матрицы и вектора.

Все эти действия основаны на умении правильно задать векторы индексов (строк и столбцов) в преобразуемой (или создаваемой) матрице. Элементами строки могут быть числа (скаляры), векторы, матрицы или текстовые выражения. Главное, чтобы они соответствовали *принципу однотипности элементов и правилам формирования матриц из блоков.* Например, блоки, формирующие строку, могут иметь различное число столбцов, но одинаковое число строк. Блоки, формирующие столбец, могут иметь различное число строк, но одинаковое число столбцов. Блоки, формирующие матрицу, должны быть согласованы как по строкам, так и по столбцам.

Результатом редактирования матрицы может служить как сама исходная матрица, так и вновь созданная.

Добавление строк и столбцов осуществляется по правилам формирования столбцов и строк. Например, добавление строки *b* («единиц») к матрице *a* выполняется следующим образом. Результатом является матрица *a1*.

```
a = [1 2 3 4; 5 6 7 8]
b = [1 1 1 1]
a1 = [a; b]
```

Результат

```
a1 = [1 2 3 4;  5 6 7 8;  1 1 1 1]
```

Добавление столбца **c** («нулевой») приводит к созданию матрицы *a2*.

```
c = [0 0]
a2 = [a c]
```

Результат

```
a2 = [1 2 3 4 0; 5 6 7 8 0]
```

Перестановка строк и столбцов осуществляется соответствующим индексированием. Создадим матрицы *aa, bb, cc* и произведем в них перестановки строк *(aa),* столбцов *(bb)* и строк и столбцов одновременно *(cc):*

```
aa = round(rand(5,6)*10)
bb = aa;
cc = aa;
aa(:,[2 3 4]) = aa(:,[3 4 2])
bb([1 3],:) = bb([3 1],:)
cc([1 3],[2 3 4]) = cc([3 1],[3 4 2])
```

Результатом этих действий будут новые матрицы

Исходные *aa, bb, cc*

```
2   4   7   2   5   4
6   7   2   6   0   7
```

```
   3    3    8    6    0    1
  10    4    6    4    3    0
   7    9    1    6    0    6
```

Конечные

```
aa =
   2    7    2    4    5    4
   6    2    6    7    0    7
   3    8    6    3    0    1
  10    6    4    4    3    0
   7    1    6    9    0    6
bb =
   3    3    8    6    0    1
   6    7    2    6    0    7
   2    4    7    2    5    4
  10    4    6    4    3    0
   7    9    1    6    0    6
cc =
   2    8    6    3    5    4
   6    7    2    6    0    7
   3    7    2    4    0    1
  10    4    6    4    3    0
   7    9    1    6    0    6
```

Удаление строк и столбцов осуществляется включением в векторы индексов строк и столбцов индексов тех строк и столбцов исходной матрицы, которые *должны остаться* после редактирования:

```
dd = cc(:,[1 2 3])
ee = cc([1 3 5],:)
ff = cc([1 3 5],[1 2 3])
```

Результат

```
dd =
     2    8    6
     6    7    2
     3    7    2
```

```
    10   4   6
     7   9   1
ee =
     2   8   6   3   5   4
     3   7   2   4   0   1
     7   9   1   6   0   6
ff =
     2   8   6
     3   7   2
     7   9   1
```

Другим вариантом удаления строк или столбцов, без сохранения исходной матрицы, является использование пустой матрицы (*[]*):

```
matr = round(rand(5,6)*10)
matr(2:4,:) = []

matr =
   2   2   4   6   9   6
   2   8   3   1   3  10
   0   2   3   0   2   7
   1   2   4   5   9   9
   6  10   4   9   2   0
```

Результат

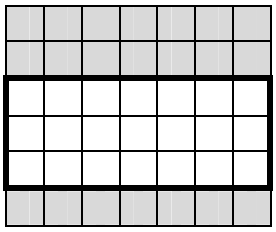
```
matr =
   2   2   4   6   9   6
   6  10   4   9   2   0
```

В этом случае в левой части оператора векторы индексов строк и столбцов формируются из индексов тех строк и столбцов, которые *должны быть удалены из исходной матрицы*.

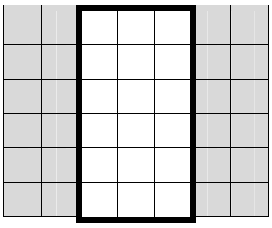
Выборка строк, столбцов и блоков (строковое, столбцовое и блочное сечение матриц) осуществляется заданием векторов индексов строк (*m*), и столбцов (*n*), содержащих нужные индексы исходной матрицы. Символ "**:**" в записях индексов матриц, например, *A(:,n)* и *A(m,:)*, означает «все строки» и «все столбцы». Запись

$A(m_1: m_2 , n_1 : n_2)$ означает выделения диапазона индексов строк и столбцов, определяющих размеры выделенного блока.

Строковое сечение матриц



Столбцовое сечение матриц



Блочное сечение матриц

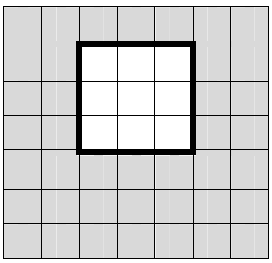
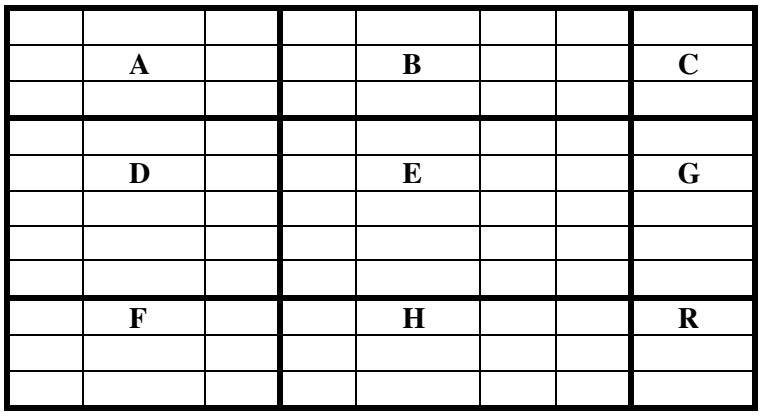


Рис. 1.7. Сечение матриц

Формирование матриц из блоков достигается согласованием их соответствующих размеров. Результирующая матрица формируется следующим образом:

M = [A B C; D E G; F H R]



Приведенная ниже конфигурация объединенной матрицы недопустима, несмотря на одинаковую размерность с предшествующей результирующей матрицей.

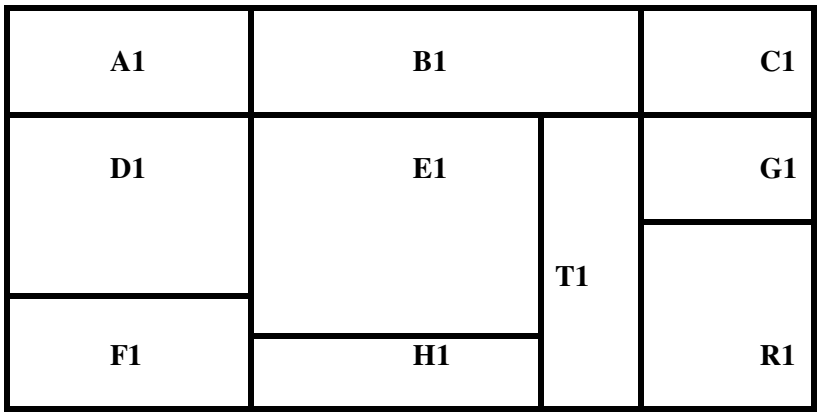


Рис. 1.8. Блочное построение матриц

Индексирование матриц осуществляется путем создания вектора индексов строк, вектора индексов столбцов или векторов индексов строк и столбцов. Эти векторы в произвольной последовательности содержат наборы индексов в пределах значений соответствующих индексов исходной матрицы:

```
ss = round(rand(5,6)*10)
r = [4 5 2 3 1 2 2]
c = [6 3 1 1]
pp = ss(r,c)
```

Результат

```
ss =
    1    7    4    5    3    6
    8    3    9    5    0    4
    4    2    5    4    7    2
    9    2    8    9    7    6
    7    2    5    0   10    7
r =
    4    5    2    3    1    2    2
c =
    6    3    1    1

pp =
    6    8    9    9
    7    5    7    7
    4    9    8    8
    2    5    4    4
    6    4    1    1
    4    9    8    8
    4    9    8    8
```

В этом примере *r*-индексы строк, *c*-индексы столбцов матрицы *ss*. Рассмотрим элемент *pp* (1,1) матрицы *pp*. Так как индексы его строки – 1 и столбца – 1 , следовательно выбираем из *r* и *c* первые элементы (т.е. из *r* – 4, а из *c* – 6). Это будет номер строки (4) и столбца (6) в матрице ss. На пересечении четвертой строчки и шестого столбца расположен элемент 6. Он и будет записан на место элемента *pp* (1,1) в матрицу *pp*. По аналогии заполняется вся матрица.

*Индексирование матриц 0-1 векторами***.** Если в исходной матрице нужно выделить элементы строк и столбцов по какому-нибудь признаку, то используются булевы векторы, состоящие из 0

и 1. Например, нужно исключить из матрицы, содержащей числа от 0 до 50, строки, элементы третьего столбца которых меньше 25:

```
mat = round(rand(5,6)*50)
l = mat(:,3)>25
mat1 = mat(l,:)
```

Результат

```
mat =
   19   46   10   27   31   42
    0   42   45   47   35   19
   21   18   28   17   20   21
   38   31   32   33   21   30
   40   37   12   20   33   28

l =
    0
    1
    1
    1
    0

mat1 =
    0   42   45   47   35   19
   21   18   28   17   20   21
   38   31   32   33   21   30
```

Операция *транспонирования* вектора *x* или матрицы *mat* производится с использованием символа « ‘ », т.е. - *x’, mat’*

*Работа с текстовыми переменными***.** Символьные строки в *MATLAB* вводятся в апострофах:

a = 'laboratory work'

Запись текстовой переменной производится в форме вектора, число элементов которого равно числу введенных символов. Текстовые переменные можно объединять. Например,

s = [a, 'number'].

Результатом этой операции будет текст – *laboratory work number.*

Текстовую переменную можно использовать как оператор, команду или часть выражения. Это осуществляется с помощью встроенной функции *EVAL.*

1.4.2. Генератор случайных чисел

Генератор случайных чисел RAND формирует случайное вещественное число в интервале [0 ÷1).

x=rand(4)

матрица размерности 4*4 из случайных вещественных чисел в интервале [0 ÷1).

Чтобы изменить интервал случайных чисел, нужно «растянуть» числовую шкалу с помощью числового коэффициента

*x1= rand(4)*10*

матрица размерности 4*4 из случайных вещественных чисел в интервале [0 ÷10).

Чтобы получить положительные и отрицательные числа, нужно сместить начало числовой оси на нужную величину

*x1= rand(4)*10-5*

матрица размерности 4*4 из случайных вещественных чисел в интервале [-5 ÷5).

1.4.3. Численное интегрирование

Численное интегрирование функции $y = f(x)$ сводится к разбиению интервала $[a,b]$, на котором определена функция, на множество малых отрезков, и нахождению искомой площади как совокупности элементарных площадей, полученных на каждом частичном промежутке разбиения. В зависимости от использованной аппроксимации получаются различные формулы численного интегрирования, имеющие различную точность.

Метод трапеций. В этом методе используется линейная аппроксимация, т.е. график функции $y = f(x)$ представляется в виде ломаной, соединяющей точки y_i. Формула трапеций при постоянном шаге разбиения $h=\frac{b-a}{n}$, где n-число участков, имеет вид

$$\int_a^b f(x)dx = h(\frac{y_0 + y_n}{2} + \sum_{i=1}^{n-1} y_i) .$$

В MATLAB данный метод реализует функция *trapz(x,y).*

Метод Симпсона. Если подынтегральную функцию аппроксимировать параболой, то формула Симпсона с постоянным шагом интегрирования имеет вид

$$\int_a^b f(x)dx = \frac{h}{3}[y_0 + 4(y_1 + y_3 + ... + y_{n-1}) + 2(y_2 + y_4 + ... + y_{n-2}) + y_n] .$$

В MATLAB формула Симпсона реализуется функция *quad.*

Возможны три варианта решения задачи:

1) подынтегральное выражение записывается в виде текстовой переменной

```
quad('log(x)',0.00001,1);
```

2) подынтегральное выражение записывается в виде текстовой переменной с использованием функции *inline* (объект класса *inline)*

```
a = inline('log(x)')
quad(a,0.00001,1);
```

3) подынтегральное выражение записывается в отдельном файле (как *Function*)**.** Параметры, в данном примере – **c**, объявляются как *global*. Функция записывается в файле с тем же именем, что и имя функции:

```
function aa = v_int(x)
global c
aa = sqrt(c.^2+x.^2).
```

В *SCRIPT*-файле записывается процедура вычисления интеграла с использованием имени файла, в котором хранится подынтегральное выражение:

```
global c
c = 2;
quad(@v_int,0,1)
```

1.4.4. Решение обыкновенных дифференциальных уравнений (задача Коши)

Решение системы уравнений основано на методе Рунге–Кутта различных порядков – *ode23, ode45*. Ниже приведены примеры записи правых частей для нескольких различных уравнений: *r1(x,y), r1(x,y), r8(x,y), r19(x,y), r19m(x,y)* , где *x* – переменная интегрирования, *y* – интегрируемые функции.

```
function dydx = r1(x,y);
dydx(1) =  - x/y(1);

function dydx = r8(x,y);
dydx(1) =  - x*(y(1).^2 - 1)/y(1)/(x^2 - 1);


function dydt = r19m(t,y);
dydt(1) = sqrt(2*y(1));

function dydt = r19(t,y);
dydt(1) =  - sqrt(2*y(1));
```

В сценарии обращение к функции *ode45*, которая имеет следующие параметры:

```
[X,Y] = ode45(@r8,[0.1 0.9],9);
plot(X,Y)
```

% Дескриптор @ обеспечивает связь с файлом-функцией правой части системы дифференциальных уравнений.

% ,[0.1 0.9] – интервал, на котором необходимо получить решение.

%9 – начальное значение решения (y(0) = 9).

1.4.5. Графический вывод результатов расчетов

Семейство кривых (совмещение нескольких графиков)

Действие оператора *hold on* заключается в реализации графического режима, при котором можно последовательно выводить в графическое окно несколько графиков (режим «наложения) (рис. 1.9). Оператор *hold off* этот режим отменяет.

```
x = 0:0.1:5;
for   i = 1:5
y = (1-exp( -x/(i/2)));
plot(x,y)
hold on
end
hold off
```

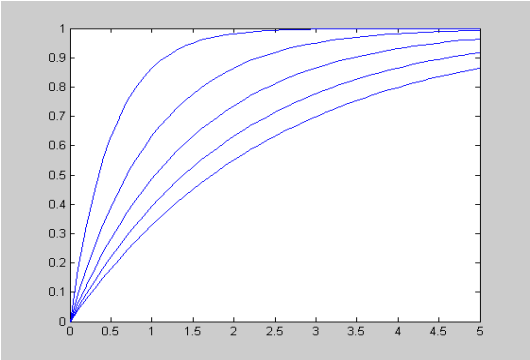


Рис. 1.9. Вывод нескольких графиков в одном графическом окне

Вывод графической информации в нескольких графических окнах

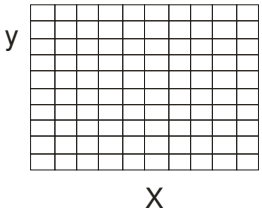
В случае необходимости вывода графической информации в различных графических окнах (рис. 1.10) необходимо разделить экран на несколько частей. Эта операция осуществляется функцией *subplot(i,j,k)*. Первый индекс означает количество окон по горизонтали, второй – по вертикали, третий – номер, куда выводится очередной график.

Рис. 1.10. Разделение экрана на графические окна

Функция *subplot* имеет еще ряд параметров управляющих работой по выводу информации на экран.

Построение линий уровня

При решении многих задач нередко возникает необходимость отобразить результаты расчетов в виде уровней исследуемой функции $z = f(x,y)$ на двумерной плоскости (x,y). Для этого необходимо в пределах рассматриваемой области $[x_{min}<x<x_{max}, y_{min}<y<y_{max}]$ определения построить двумерную сетку с достаточно мелким шагом.



В узлах этой сетки вычисляется значения функции $z = f(x,y)$ и графически соединяются узлы с одинаковыми значениями функции. Эти операции в MATLAB реализуются функциями *meshgrid* и *contour*.

Функция $[X,Y] = meshgrid(x,y)$ задает сетку, в узлах которой записываются координаты x и y (массивы X и Y).

Пример: Построить линии уровня $f(x,y) = x * \exp(-x^2-y^2)$

```
x = -2:0.5:2;
y = -2:0.5:3;
[X,Y] = meshgrid(x,y);
Z = X.*exp(-X.^2-Y.^2);
contour(X,Y,Z,'k')
```

X =

-2.0	-1.5	-1.0	-.5	0	.5	1.0	1.5	2.0
-2.0	-1.5	-1.0	-.5	0	.5	1.0	1.5	2.0
-2.0	-1.5	-1.0	-.5	0	.5	1.0	1.5	2.0
-2.0	-1.5	-1.0	-.5	0	.5	1.0	1.5	2.0
-2.0	-1.5	-1.0	-.5	0	.5	1.0	1.5	2.0
-2.0	-1.5	-1.0	-.5	0	.5	1.0	1.5	2.0
-2.0	-1.5	-1.0	-.5	0	.5	1.0	1.5	2.0
-2.0	-1.5	-1.0	-.5	0	.5	1.0	1.5	2.0
-2.0	-1.5	-1.0	-.5	0	.5	1.0	1.5	2.0
-2.0	-1.5	-1.0	-.5	0	.5	1.0	1.5	2.0
-2.0	-1.5	-1.0	-.5	0	.5	1.0	1.5	2.0

Y =

-2.0	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0
-1.5	-1.5	-1.5	-1.5	-1.5	-1.5	-1.5	-1.5	-1.5
-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0
-.5	-.5	-.5	-.5	. -.5	-.5	-.5	-.5	-.5
0	0	0	0	0	0	0	0	0
.5	.5	.5	.5	.5	.5	.5	.5	.5
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5
3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0

Функция *[X,Y] = meshgrid(x,y)* задает сетку, в узлах которой определены массивы *X* и *Y*. Строки массива *X* – повторяют запись

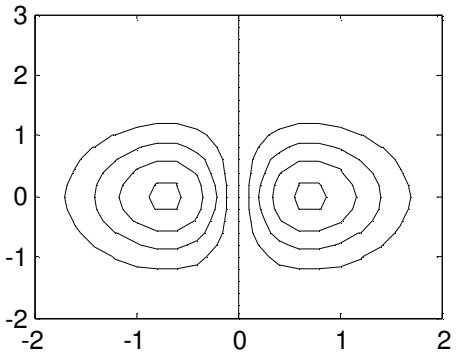


Рис. 1.11. Построение линий уровня

вектора *x*, Столбцы массива *Y* – повторяют запись вектора *y'*. Массив содержит значения исследуемой функции в узлах сетки.

Функция *contour(X,Y,Z,'k')* – строит линии уровня (рис. 1.11).

1.4.6. Примеры организации вычислений с учетом особенностей MATLAB

Рассмотрим некоторые примеры, иллюстрирующие особенности языка программирования MATLAB.

Построение результирующих массивов данных на основе исходных массивов меньшей размерности. Используется только механизм соответствующей индексации.

Построение фигуры, изображенной на рис.1.12. Фигура обладает плоскостной симметрией, поэтому построение произведем методом зеркальных отображений. Ветвь графика в первом квадранте декартовой системы координат описывается участком гиперболы

$$y = 1/(x + 0.5) - 0.5$$

при изменение *x* в диапазоне [0÷1.5].

Построим ветвь графика в первом квадранте. На участке изменения *x* от 1.5 до 0 вычисляем значения функции *y* в *N* точках (шаг

таблицы $h = a/(N-1)$). Таким образом, формируем два массива – аргумента *x* и функции *y*. Затем проходим участок изменения *x* от 0 до -1.5 с шагом *h*. При этом нет необходимости производить вычисления функции *y*, достаточно произвести зеркальное отображение первого квадранта во второй, совместив последнюю точку графика первого квадранта с первой точкой графика во втором квадранте. На этом участке

$$x_{i+N} = -x_{N-j}, \text{ а } y_{i+N} = y_{N-j}$$

для $i, j = 1, 2, ..N-1$. Индекс последней точки графика – (2N-1).

Чтобы завершить построение фигуры, необходимо пройти участок изменения аргумента *x* в обратном направлении от -1.5 до 1.5. Графически этот процесс выражается в зеркальном отображении верхней части графика в нижнюю половину (3-й и 4-й квадранты системы координат).

В третьем квадранте $x_{2N-1+i} = -x_j$, а $y_{2N-1+i} = -y_j$

для $i = 1, 2..N-1; \quad j = 2, 3..N$

В четвертом квадранте $x_{3N-2+i} = -x_j$, а $y_{3N-2+i} = -y_{N-j}$

для $i, j = 1, 2..N-1$.

N = 11; – число точек в квадранте;

n = N-1; – число интервалов;

a = 1.5; – размер по оси;

x = 1.5:-a/n:0; – вектор аргумента в первом квадранте;

y = 1./(x+0.5)-0.5; – вектор функции в первом квадранте;

xx = [x -x(n:-1:1) -x(2:n+1) x(n:-1:1)]'; – вектор аргумента (полный);

yy = [y y(n:-1:1) -y(2:n+1) -y(n:-1:1)]'; – вектор функции (полный);

plot (xx,yy) – построение графика;

hold on – построение осей;

plot ([-1.5 1.5],[0 0]);

plot ([0 0],[-1.5 1.5]).

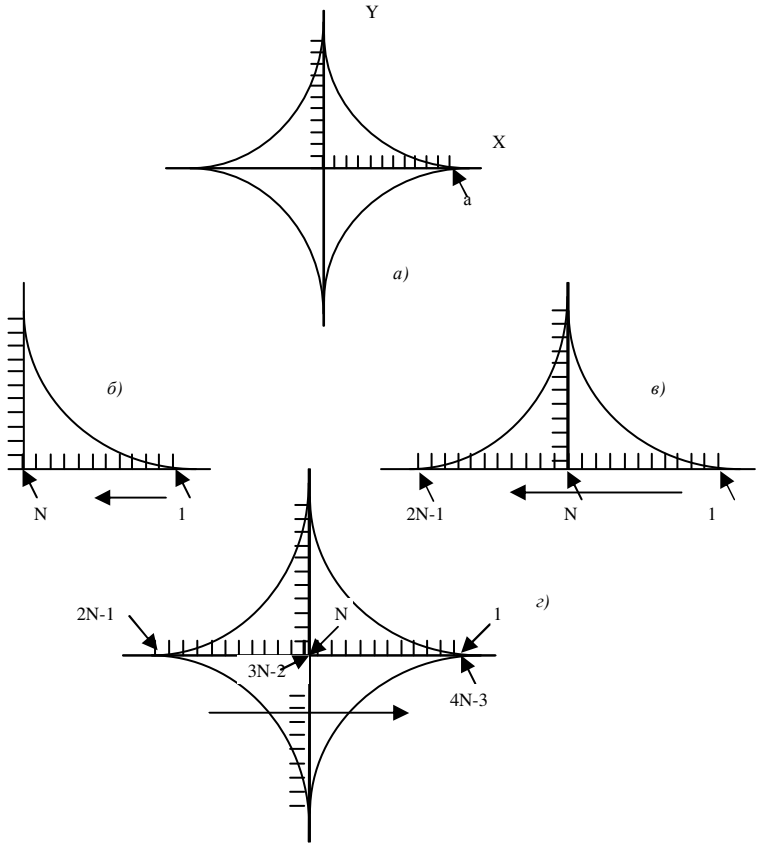


Рис.1.12. Фигура с плоскостной симметрией

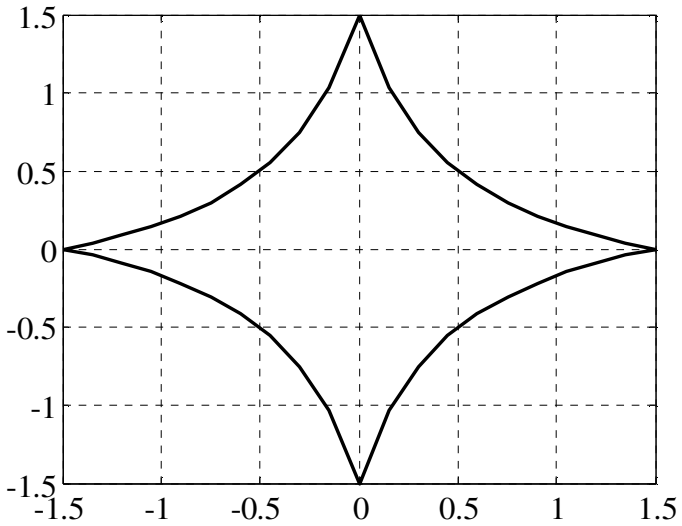


Рис. 1.13. Результат работы программы

Если написать эту же программу на Паскале или Фортране, то текст программы будет значительно объемнее, а алгоритм программы не будет столь наглядным как в данной программе. Кроме того, в системе MATLAB нет необходимости специально инициировать переход в графический режим, достаточно просто обратиться к соответствующим функциям графической подсистемы.

Использование *function.* Для решения задач нередко полезно использовать инструмент *function,* позволяющий структурировать и логически упростить программу, а также использовать внутренние функции MATLAB, использующие механизм *function.*

В качестве примера рассмотрим вычисления среднеарифметического и среднеквадратичного значений массива **x.** В первом варианте эти значения вычисляются внутри одной функции *stat(x)*, во втором варианте используется механизм вложенных функций – функция avg(x,n) вызывается из *stat(x)*. В первом варианте создается М-файл с именем *stat,* во втором случае создаются два М-файла с именами *stat* и *avg*.

Пример 1

```
function [mean,stdev] = stat(x)
n = length(x);
mean = sum(x)/n;
stdev = sqrt(sum((x-mean).^2/n)).
```

Пример 2

```
function [mean,stdev] = stat(x)
n = length(x);
mean = avg(x,n);
stdev = sqrt(sum((x-avg(x,n)).^2)/n);
function mean = avg(x,n);
mean = sum(x)/n.
```

Сценарий может иметь самый простой вид:

```
x = rand(1,100);
stat(x).
```

При использовании стандартных внутренних функций может возникнуть ситуация, когда вычисляемая функция (записанная в М-файле и являющаяся подфункцией вызывающей функции) зависит от параметра. Специфика такой ситуации заключается в том, что стандартная функция имеет фиксированный набор формальных параметров, в котором не предусмотрено наличие у вызываемой подфункции каких-либо дополнительных параметров. Рассмотрим вычисление минимума функций одной переменной

$$f1 = (x-a)^2 \quad и \quad f2 = (x-2)^2$$

Пример 3

```
function f1 = myfun1(x,a){запись функции, a – параметр}
f1 = (x - a)^2;
```

Сценарий использует стандартную внутреннюю функцию вычисления минимума – *fminbnd*. Параметр *a* объявляется перед обращением к этой функции

```
a = 1.5
x = fminbnd(@(x) myfun1(x,a),0,1).
```

Для второй функции

```
function f2 = myfun2(x){запись функции}
f2 = (x - 2)^2.
```

Сценарий имеет более простую форму

```
x = fminbnd(@myfun2(x),0,1).
```

Функция вычисления минимума *fminbnd* имеет три формальных параметра параметра:

имя М-файла, содержащего вычисляемую функцию;

значение левой границы интервала поиска минимума;

значение правой границы интервала поиска минимума.

Функция *myfun2(x)* зависит только от переменной ***x.*** Обратите внимание на запись сценария в случае, когда *myfun1(x,a)* зависит от параметра. Запись `@(x)` в списке фактических параметров подтверждает, что определяется минимум функции по одной переменной, а далее следует запись имени конкретной функции, содержащей параметры. В принципе, эта функция может содержать несколько параметров.

Ещё одним вариантом решения подобных задач является использование глобальных переменных. В этом случае параметры подфункции объявляются в ней и в вызывающей функции как глобальные. Взаимосвязь вызывающей и вызываемой функций имеет вид (смотри примеры, приведенные ниже):

Пример 4

```
function f3 = myfun3(x){запись функции, a -параметр}
global a
```

f1 = (x - a)^2.

Параметр *a* объявляется перед обращением к этой функции
Сценарий имеет следующую форму:

```
global a
a = 1.5
x = fminbnd(@myfun3(x),0,1).
```

Контрольные вопросы к главе 1

1. Что является объектом записи и обработки информации в MATLAB?
2. В чем различие операций над массивами и матрицами?
3. В какой форме можно задавать индексы массивов?
4. Как осуществляется перестановка строк и столбцов в массивах?
5. Как осуществляется выборка элементов строк и столбцов по условию?
6. Для каких целей используется формат данных « + »?
7. Какие внутренние функции формируют наиболее часто используемые числовые матрицы специального вида?
8. Возможно ли использование строковых выражений для численных вычислений?
9. Каким образом отображается и оформляется графическая информация?
10. Как отображаются линии постоянных значений функции (линии уровня)?
11. Цветовое оформление графики.
12. Дайте определение *M*-сценариев и *M*-функций.
13. Правила написания *M*-функций пользователя.
14. В каких случаях используются глобальные переменные?
15. Какие функции используются для интерполяции табличных функций, в чем их различие?
16. Укажите возможные варианты вычисления определенных интегралов.

17. Объясните значения формальных параметров функций решения обыкновенных дифференциальных уравнений ODE23 ODE45.
18. Как формируется запись правых частей системы обыкновенных дифференциальных уравнений?

2. РЕШЕНИЕ ЗАДАЧ В СИСТЕМЕ MATLAB

Решение задач в системе MATLAB связано с необходимостью изучения особенностей программирования в этой среде, что определяется спецификой реализации алгоритмов и обработки данных. MATLAB имеет много общего со структурой традиционных (императивных) языков программирования, поэтому программист, имеющий опыт работы с такими языками как FORTRAN, Pascal или C, достаточно просто адаптируется к работе в системе MATLAB.

Выполнение практических заданий в системе MATLAB включает в себя упражнения на построение простейших арифметических, логических алгоритмов, а также типичные задачи вычислительной математики и индивидуальные задания для студентов по электрофизике. Первая часть практикума предназначена для изучения и освоения приёмов создания и редактирования матриц различных типов. Работа проводится в двух режимах – режиме прямых вычислений и режиме М-файлов. Вторая часть практикума связана с вычислением интегралов, поиска минимума функций, решения систем обыкновенных уравнений и т.д. Третья часть практикума содержит индивидуальные задания студентам, связанные с расчетом полей и динамики заряженных частиц в различных элементах и устройствах электрофизических установок.

2.1. Режим прямых вычислений (командная строка)

2.1.1. Программирование простейших конструкций языка, различные типы данных

Задание 1. Запись и вычисление арифметических выражений.

Обратить внимание на форму записи внутренних математических функций (см. соответствующий раздел помощи!).

1. Записать арифметические выражения:
   - $a=e^{2.5}+9^{0.5}+8^{1/3}$.
   - $b=\sin(\pi/4)+\cos(\pi/4)$.
   - $c=\ln(e^4)+\mathrm{arctg}(1)$.

Самостоятельно сформулируйте и вычислите ряд выражений, содержащих различные внутренние функции.

2. Вычислить значения функций в заданных диапазонах изменения аргументов, используя матричную запись. Определить, какие виды операций применимы к данным выражениям – обычные матричные или унарные.

- $y = \sin(x)$ для $x=[0 \div 2\pi]$ с шагом $\pi/8$;
- $z = (a-2b)/(2a^2+4b^2)$ для $a = [1 \div 4], b = [1 \div 4]$ с шагом 1;
- $s = (-1)^n/n$ для $n = [0 \div 10]$ с шагом 1.
- Самостоятельно сформулируйте и вычислите ряд выражений, к которым применимы обычные матричные или унарные операции.

Задание 2. Формирование векторов и матриц (различные способы записи).

1. Сформировать ряд векторов и матриц произвольных размеров, используя различные способы записи.

2. Вывести на экран значения отдельных элементов, созданных векторов и матриц.

3. Ввести значения элементов векторов и матриц, заданных в виде арифметических выражений:

- $y = [2+2/(3+4);\ \exp(5);\ \mathrm{sqrt}(10)]$
- $k = 3$

$$mm = [1 \quad k \quad k^2 \quad k^3;\ 1 \quad 1/k \quad 1/k^2 \quad 1/k^3\ ].$$

- Сформировать самостоятельно несколько подобных векторов и матриц.

4. Задать вектор-строку, используя, оператор «**:**»

- целочисленный вектор-строка $V1$ от 1 до 10 с шагом 1,
- вещественный вектор-строка $V2$ от 0 до 2 с шагом 0.1,
- записать вектор-строку из целых чисел в убывающем порядке,
- оценить результаты операции $VS = 5:1$.

5. Ввести комплексные числа и вычислить действительные и мнимые части, а также абсолютные значения, используя встроенные функции *IMAG, REAL, ABS:*

- $c1 = 3+2*I;$
- $c2 = 5+2*\mathrm{sqrt}(-1);$

- *c*З = 10.
- Сформировать самостоятельно несколько подобных чисел и произвести с ними простые арифметические действия.

6. Задать элемент матрицы *Z*(3,4)=5. Вывести матрицу Z, оценить ее размер и содержание.

2.2. Работа в режиме М-файлов

2.2.1. Операции с матрицами

Важное значение этой темы определяется необходимостью решения больших систем линейных алгебраических уравнений, которые получаются в результате применения методов дискретизации (конечные разности, конечные элементы) к уравнениям математической физики (задачи сплошных сред), а также при решении задач линейного и нелинейного программирования и т.п. Получающиеся в большинстве задач системы линейных уравнений (СЛАУ) имеют ряд особенностей. Матрицы коэффициентов уравнений могут содержать значительное количество нулевых элементов. Ненулевые элементы могут иметь специфическое диагональное расположение – ленточные матрицы. Широкое использование методов Монте-Карло связано с необходимостью генерации стохастических матриц большой размерности. Решение многих физических задач, связанных с динамикой пучков заряженных частиц, описывается с помощью различного типа матриц.

Задание 1. Формирование (генерация) матриц различного вида.

1. Сформировать матрицы случайных чисел, используя различные формы записи функции *RAND* - *rand(n), rand(m,n), rand(size(A))*. Матрица *A* должна быть заранее определена. Вычислить для этих матриц средние и среднеквадратичные значения их элементов.

2. Сформировать единичные матрицы размерностью *M*N, M*M, N*M*$_t$ используя функцию *ONES*.

3. Сформировать единичные матрицы размерностью *M*N, M*M, N*M* с единичной главной диагональю, используя функцию *EYE*.

4. Сформировать нулевые матрицы размерностью *M*N, M*M, N*M,* используя функцию *ZEROS.*

5. Транспонировать произвольные векторы и матрицы. Запись *V'* и *M'* означает транспонирование (апостроф – " ' ").

Задание 2. Формирование таблиц.

Вычислить функции $Y = \sin(X)$ и $Z = \cos(X)$ в интервале то 0 до 2π с шагом $\pi/16$. Результаты оформить в виде таблиц. Столбец аргументов и столбцы функций:

X	Y	Z
X1	Y1	Z1
X2	Y2	Z2
X3	Y3	Z3
X4	Y4	Z4

Примечание. Нужно сформировать матрицу из трех элементов – векторов *X,Y* и *Z.* Разделительные полосы и верхняя строка являются просто поясняющими элементами и не воспроизводятся на экране.

Задание 3**.** Вывод результатов.

Рассмотреть действие форматов *short, long, short e, long e, + (Plus).*

Вывести на экран числа *a* = 100, *c* = 5.73*10-8, π, матрицу размером 10*10, содержащую положительные и отрицательные числа, а также нули, используя различные форматы.

Формат задается либо с использованием команды *format* (например, >>*format short*) в командной строке, либо в меню командного окна "*file – preferences"*.

Задание 4. Работа с символьными данными.

Символьные строки в *MATLAB* вводятся в апострофах:

a= 'laboratory work'.

Запись текстовой переменной производится в форме вектора, число элементов которого равно числу введенных символов, включая пробелы. Текстовые переменные можно объединять. Например,

s=[a, 'number'].

Текстовую переменную можно использовать как оператор, команду или часть выражения. Это осуществляется с помощью встроенной функции *EVAL.*

1. Оформить заголовок работы с указанием Ф.И.О., группы, номера и названия работы.

2. Заполнить матрицу MT(10,10), используя переменную

t ='11*/(i+j-l)*'

3. Вычислить значения функций *X, Y, Z,* заданных текстовыми переменными, при изменении аргумента X в диапазоне от 0 до 1

$$t1 = \text{'sin}(x)\ ',\ \ t2 = \text{'cos}(x)\ ',\ \ z = t1+t2.$$

Результаты вывести в виде таблицы.

Задание 5. Сечения и блоки массивов.

Построить матрицы A(6,6), B(3,3), C(3,3), используя генератор случайных чисел. В матрице A произвести несколько столбцовых и строковых сечений, т.е. выделить прямоугольный фрагмент исходной матрицы. Провести столбцовое и строковое сечение массивов, используя оператор "**:**". Используя матрицы B, C в качестве блоков, сформировать три новые матрицы:

1) объединить *B* и *C* в строку
2) объединить *B* и *C* в столбец
3) объединить в матрицу

$$\begin{vmatrix} B & EYE(3) \\ C' & ONES(SIZE(C)) \end{vmatrix}$$

4) выделить из нее блок элементов со 2-й строки по 4-ю и со 2-го столбца по 5-й
5) произвести самостоятельно несколько подобных операций.

Задание 6. Редактирование матриц.

1. Сформировать матрицу MR(8,8), элементами которой являются целые числа, расположенные в порядке возрастания. Первая строка начинается с 1, вторая с 2 и т.д. Для исходной матрицы провести следующие изменения:

- удалить строку с индексом 2,
- удалить столбец с индексом 3,
- поменять местами строки с индексами 1 и 7,
- поменять местами столбцы с индексами 2 и 7,
- добавить к исходной матрице слева столбец, состоящий из 0, а справа – из 10,
- добавить к исходной матрице сверху строку, состоящую из 0, а снизу – из 10,
- вставить дополнительный столбец, состоящий из 1, между 3-м и 4-м столбцами,
- вставить дополнительную строку, состоящую из 1, между 3-й и 4-й строками,
- заменить значение элемента (5,5) на значение элемента (1,1)*20.

2. Используя генератор случайных чисел сформировать матрицу F вещественных чисел в диапазоне от 1 до 100. Преобразовать исходную матрицу, используя функции округления *FIX, FLOOR, CEIL, ROUND*. С помощью операций редактирования преобразовать исходную матрицу в матрицу F1, у которой столбцы располагаются таким образом, чтобы элементы первой строки шли в порядке возрастания.

Задание 7. Операции над матрицами.

- Сформировать матрицы *AA*(5,5), *BB*(5,5) и векторы *XX*(5), *YY*(4).
- Выполнить операции сложения, вычитания и умножения матриц *AA* и *BB*, в том числе, и унарные. Повторить операции, поменяв матрицы местами. Сравнить результаты.
- Умножить матрицу *AA* на векторы *XX* и *YY*. Умножить вектор *XX* на матрицу *AA*.
- Транспонировать матрицу *AA* и вектор *XX*.
- Выделить 4-й столбец матрицы *AA* и транспонировать его (действия осуществить одной командой). Выделить столбцовое

сечение со 2-го по 4-й столбец матрицы AA (действия осуществить одной командой).

- Задать векторы *R*(4) и *S*(4) и выполнить скалярное (вектор-строка * вектор-столбец) и внешнее (вектор-столбец * вектор-строка) умножение. Сравнить результаты.
- Умножить матрицу на скаляр.

Задание 8. Индексирование матриц. Векторное индексирование.

1. Сформировать матрицу *AM(10,10)* и вектор *P(10)*.. Сформировать вектор S(8), элементы которого целые числа от 1 до 8, расположенные в произвольном порядке.

- Выполнить операцию *T=P(S)*. Оценить результат.
- Выполнить операцию *TM=AM(S,S)*. Оценить результат.
- Произвести выборку из матрицы AM, используя векторное индексирование.

Оценить результат. Произвести несколько выборок.

2. Индексирование 0 - 1 векторами. Если в исходной матрице нужно выделить элементы строк и столбцов по какому-нибудь признаку, то используются булевы векторы, состоящие из 0 и 1.

- Задать матрицу VD(5,5), состоящую из целых чисел в диапазоне от 0 до 50, и булевский вектор *L(5)*. Исключить из матрицы те строки, элементы 3-го столбца которых меньше 25.
- Произвести еще ряд различных логических выборок.
- Оценить результат. Произвести индексирование для других логических операторов.

Задание 9. Операции над массивами.

Поэлементные операции обозначаются как знак обычной операции, но с десятичной точкой перед ним. Для произвольных матриц *A* и *B* произвести поэлементные операции умножения, сложения, вычитания, возведения в степень. Элементы матриц нужно выбирать таким образом, чтобы легко было оценить результаты.

Задание 10. Операции отношения.

Задать две произвольные матрицы и применить к ним операции отношения. (См. соответствующий раздел помощи.)

Задание 11. Функции обработки массивов.

Задать произвольные матрицы *M1, M2* и векторы *V1, V2*. Используя функции *CUMSUM, SUM, PROD, SORT* произвести вычисление сумм, произведений элементов массивов и векторов, а также их сортировку. Определить детерминант матриц *(DET),* диагональные элементы матриц *(DIAG),* а также максимальные и минимальные значения *(MAX, MIN).*

2.2.2. Организация циклов и условные операторы

Задание 12. Операторы цикла.

Используя операторы цикла *for* и *while,* сформировать массивы чисел *X(20*), *Y(20), Z(20).*

1. *X[i]*=2*(*i*-1)+*i*/2,
2. *Y[i]=Y[i*-1*]+i*,
3. *Z[i]*=exp*(i*/10*).*

Задание 13. Условные операторы.

Используя массивы из предыдущего задания и введя ограничения

XMIN=5, XMAX=13, YMIN=3, YMAX= 13, ZM1N=2, ZMAX=5,

произвести анализ значений этих массивов с помощью конструкции типа *if...else...elseif...end.*

Задание 14. Операторы выбора.

Сформировать вектор A(10), элементы которого являются целыми числами в диапазоне от 1 до 8, расположенные в произвольном порядке. Используя оператор *switch...case...otherwise...end,* сформировать вектор T, составляющими которого являются текстовые переменные, связанные с целыми числами соотношениями:

1 – красный,
2 – желтый,
3 – зеленый,
4 – синий,

5 – голубой,
6 – не определен,
7 – не определен,
8 – не определен.

2.3. Задачи вычислительной математики

Для решения задач этого раздела студенту необходимо использовать соответствующий материал из курсов высшей математики.

Наличие большой библиотеки функций, реализующей различные алгоритмы основных методов вычислительной математики встроенной в систему, значительно облегчает задачу программиста, хотя и требует ясного представления об используемых численных методах и правил обращения к этим функциям. Предложены к решению задачи: интерполирование, численное интегрирование, решение обыкновенных дифференциальных уравнений, а также задачи поисковой оптимизации. Кроме этого, предлагаются две нечисловые игровые задачи для развития алгоритмических навыков написания программ в системе MATLAB.

2.3.1. Решение традиционных задач вычислительной математики в системе MATLAB

Задача 1. Интерполяция функций. Работа с табличными функциями.

Для заданных функций создать таблицы значений в указанных интервалах и построить для них различные интерполяционные функции. Вывести информацию в виде таблиц и графиков:

1. $y=\sin(x)$, $0<x<2\pi$.
2. $y=\exp(x)$, $-1<x<1$.

Нужно использовать функции *interp*1, *spline*.

Задача 2. Численное интегрирование.

Вычислить интегралы функции одной переменной, а также функции с параметром. Использовать записи подынтегральной

функции в виде текстовой переменной, с использованием функции ***inline***, а также с использованием *M*-функций:

$$\int_0^1 x\,dx;\ \ldots\ldots \int_0^1 e^x dx;\ \ldots\ldots \int_0^{\pi} \sin(x)\,dx;\ \ldots\ldots \int_0^1 \sin(x)/x\,dx;\ \ldots\ldots \int_{-3}^{3} \sqrt{a^2+x^2}\,dx\ ;$$

$$\int_0^1 \log(x)dx;\ \int_0^1 e^x x^2 dx;\ \int_0^1 (ax+b)^2 dx;\ \int_0^1 dx/(ax+b);\ \int_0^1 dx/\sqrt{a^2+x^2}.$$

Задача 3. Решение дифференциальных уравнений (задача Коши). Решить уравнения.

1. $y' = -x / y$;
2. $dy / dx = y / x$;
3. $y' = x^2 + y^2$;
4. $y' = \sin(xy)$;
5. $y' = 2\sqrt{y}$;
6. $dy / dx = 1/\sqrt{1-x}$;
7. $dy / dx = y^2 - 1$;
8. $x(y^2 - 1) + y(x^2 - 1)y' = 0$;
9. $(x + y - 1) - (x - 1)y' = 0$;
10. $(x^2 y^2 + y^3) + x^5 y' = 0$;
11. $y' - x/\sqrt{1-x^2} = 0$;
12. $(1 + x^2)y' + xy = 0$;
13. $y' - 2xy/(1 - x^2) = 0$;
14. $xy' - y \sin x = 0$;
15. $(y \sin x - 1) + y' \cos x = 0$;
16. $(1 + y^2) + (xy + 1)y' = 0$;
17. $y' - y = (1 + x)y^2$;
18. $x(y^2 + 1) + (x^2 y + 2y^3)y' = 0$;
19. $y'^2 - 4y = 0$;
20. $y'^3 + 1 = 0$;
21. $y'^2 = 4x$;
22. $y'^2 - 2y' + 1 - y + x = 0$;
23. $y'^2 + xy' + y = 0$;
24. $y = \sqrt{y'^2 + 1}$;
25. $y = (2/3)y'x + (1/3)y'^2$;
26. $y'' = -1$;
27. $d^2x / dt^2 + x = 0$;
28. $y''' = yy'' + y^2 - 6x^2$;
29. $y'' = 6x$;
30. $y'' = \sin x / x$;
31. $y'' + y'^2 + 1 = 0$;
32. $y' = xy'' - y''^2$;
33. $y'' = yy' - y'^2$;
34. $y''^2 + yy'y'' - y'^3 = 0$;
35. $yy'' = y'^2 + 6xy^2$
36. $x^2 y'' + xy' + y = 0$;
37. $y'y''' - y''^2 = 0$.

Пределы интегрирования и начальные условия задать самостоятельно.

Задача 4. Решение задач оптимизации.

В зависимости от версии MATLAB нужно использовать функции *fmins, fminimax* или *fminibnd* для определения экстремумов функции. Задавая различные начальные значения, исследовать функцию на наличие нескольких экстремумов. Построить линии уровней и прямую, соединяющую стартовую точку поиска минимума и вычисленный минимум.

1. Найти минимум функций Химмельблау (рис. 2.1)

*f(x,y)=(x*x+y-11)*(x*x+y-11)+(x+y*y-7)*(x+y*y-7),*

в области определения *-6.0<x<6.0* и *-6.0<y<6.0,*

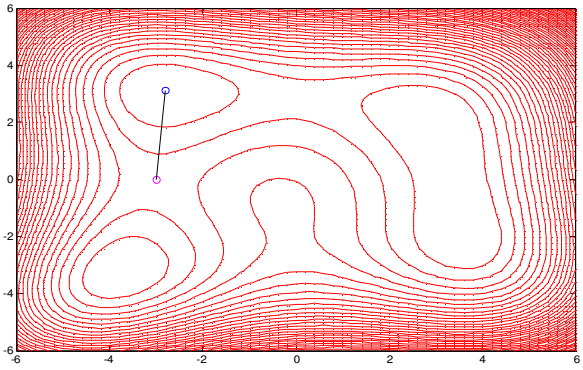


Рис. 2.1. Определение экстремума функции

2. $f(x,y) = x^2 + 2xy^3 - 5xy + y^2$ $-3 < x < 3;$ $-3 < y < 3.$
3. $f(x,y) = (1-x^2) + (2-y^2)$ $-6 < x < 6;$ $-6 < y < 6.$
4. $f(x,y) = 8x^2 + 4xy + 5y^2$ $-6 < x < 6;$ $-6 < y < 6.$
5. $f(x,y) = 2x^3 + 4xy^3 - 10xy + y^2$ $0 < x < 5;$ $0 < y < 5.$
6. $f(x,y) = 4x^2 + 3y^2 - 4xy + x$ $-6 < x < 6;$ $-6 < y < 6.$
7. $f(x,y) = 100(y-x^2) + (1-x)^2$ $-2 < x < 2;$ $-2 < y < 2.$
8. $f(x,y) = 2x^3 + 4xy^2 - 10xy + y^2$ $-3 < x < 3;$ $-3 < y < 3.$
9. $f(x,y) = (x - a^2y^2)(x - b^2y^2)$ $-3 < x < 3;$ $-3 < y < 3$ a,b – const.
10. $f(x,y) = (x^2 + (y+1)^2)(x^2 + (y-1)^2)$ $-3 < x < 3;$ $-3 < y < 3.$

2.3.2. Решение игровых и нечисловых задач

Задача 5. Кривая погони.

*Цель работы***:** Организация циклов и построение логических условий.

*Постановка задачи***:** Игра в «кошки-мышки» (рис.2.2). Мышь бежит всегда по прямой линии от места своего нахождения до норки (в нашей задаче – слева направо) с постоянной скоростью. Кошка при каждом прыжке корректирует направление своего движения по текущему положению мыши (скорость постоянна по модулю). Построить траектории движения кошки и мыши при различных соотношениях скоростей их движения и расстояниях. Графически движение объектов отобразить в пошаговом режиме. Если кошка успевает поймать мышь, то происходит изменение цвета символа, отображающего движение мыши. *(С целью адаптации студента к программированию в среде MATLAB предлагается возможный вариант решения задачи. Студент может выбрать и свое самостоятельное решение.)*

Алгоритм.

В выбранной системе координат задаются начальные данные: указывается положение кошки (*xk,yk*) и мыши (*xm,ym*), их скорости движения *vk,vm* и временной шаг движения *dt*, определяющий графический вывод символов объектов. Задаются расстояния «мышь – норка» – *s* и «кошка – траектория движения мыши» – *h*. На каждом временном шаге переопределяется направление движения кошки. Если на текущем временном шаге координаты кошки и мышки совпадут, то кошка может быть довольна.

Шаг 1: Задать начальные данные.

Шаг 2: Вычислить число временных шагов. Вычислить длину прыжка кошки. Сформировать векторы пошаговых координат для мыши (*Xm* и *Ym*) и кошки (*Xk* и *Yk*). Для кошки размерность векторов определяется по размерности векторов мыши и они изначально заполняются нулями.

Шаг 3: Цикл по временным шагам (могут быть варианты выхода из цикла): проверка координаты кошки. Если *Y*-ая координата кошки равна или больше *h*, то выход на конец цикла. Вычислить радиус-вектор «кошка-мышка» на шаге. Вычислить новые координаты кошки.

Если *X*-я координата кошки равна или больше *X*-й координаты мыши и *Y*-я координата кошки равна или больше *h*, то положить координаты кошки равными координатам мыши и выйти на конец цикла. Иначе, положить *X*–ю координату кошки равной *X*-й координате мыши, а *Y*–ю координату кошки положить равной *h*.

Конец цикла.

Шаг 4: Построить координатные оси.

Шаг 5: Цикл графического вывода «погони» по шагам.

Построить пошагово траектории движения объектов, используя оператор временной задержки. При совпадении координат кошки и мыши во времени (конец игры) изменить цвет отображения объектов (рис. 2.3)

Конец цикла.

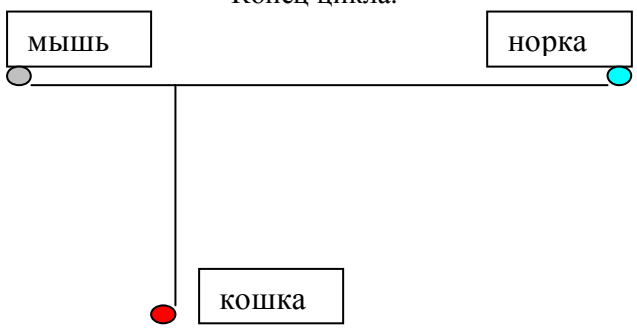


Рис. 2.2. Схема задачи «кривая погони»

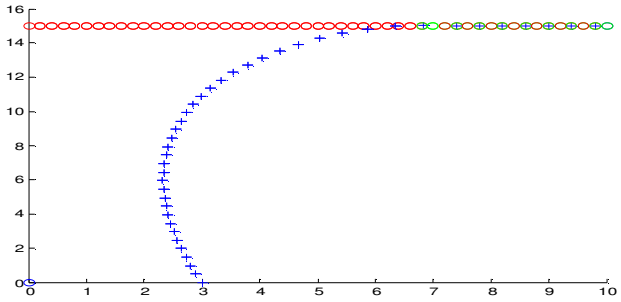


Рис. 2.3. Графическое отображение задачи «кривая погони».

Задача 6. Построение фигуры с плоскостной симметрией (эллипса).

Цель работы: Используя механизм индексации массивов, осуществить построение и трансформацию объектов с плоскостной симметрией (рис. 2.4).

Постановка задачи: В данном рассмотрении задача имеет прикладной характер – она ориентирована на изучение динамики пучка релятивистских заряженных частиц в магнитооптическом канале транспортировки. В общем случае фазовый портрет пучка занимает 6-мерный объём в пространстве $\{x,z,s,p_x,p_z,p_s\}$. В наиболее простом, но достаточно точном описании пучка, можно рассматривать три фазовых портрета пучка в координатах $\{x,p_x\}$, $\{z,p_z\}$ и $\{s,p_s\}$. Основной интерес представляет поперечная динамика пучка частиц в плоскостях $\{x,p_x\}$ и $\{z,p_z\}$. Продольный импульс частиц значительно больше поперечных импульсов, поэтому в описании пучка в фазовом пространстве можно перейти от координаты по импульсу к координате по углу расходимости пучка относительно продольной оси канала, т.е . $\{x,x'\}$, $\{z,z'\}$. В каждой их этих плоскостей фазовый портрет пучка представляет собой эллипс. В месте расположения источника частиц эллипс имеет вид «прямого эллипса», а в общем виде – наклоненного эллипса.

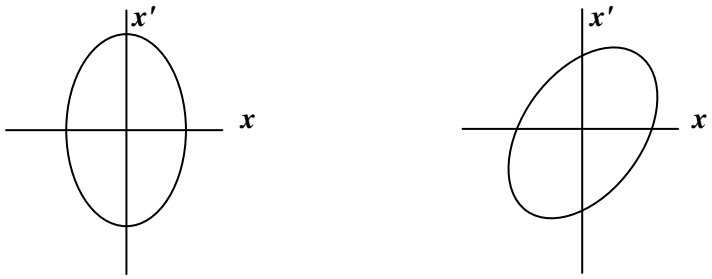


Рис. 2.4. Построение эллипса

Каноническое уравнение эллипса имеет вид:

$$(x/a)^2 + (y/b)^2 = 1,$$

где *x,y* – текущие значения координат, *a b* – полуоси эллипса. В нашем рассмотрении *x,y* – $\{x,x^{'}\}$или $\{z,z^{'}\}$.

Средствами MATLAB построить дугу эллипса в первой координатной четверти, а затем, используя эти данные и соответствующую индексацию, дополнить эллипс до полного. Для наклонного эллипса проделать подобную же операцию. Положение эллипса в пространстве определяется набором точек с координатами $\{x_i, x_i^{'}\}$ или $\{z_i, z_i^{'}\}$.

Ввести произвольную матрицу *M*(2,2), такую, что *M*(2,2) · *M*(1,1) - *M*(1,2) *M*(2,1)=1 (например, $\begin{bmatrix} 1 & l \\ 0 & 1 \end{bmatrix}$) и осуществить операцию $\begin{pmatrix} x_i \\ x_i^{'} \end{pmatrix}_{\text{вых}} = \begin{bmatrix} 1 & l \\ 0 & 1 \end{bmatrix} \begin{pmatrix} x_i \\ x_i^{'} \end{pmatrix}_{\text{вх}}$ для каждой точки, представляющей эллипс. В результате этих действий происходит трансформация исходного эллипса. При линейных преобразованиях фазового объема граница эллипса переходит в новую границу.

2.4. Задачи электрофизики

Четвертый класс задач посвящен расчету характерных параметров элементов электрофизических устройств и установок. Первые две задачи связаны с исследованием устойчивости и колебательными процессами, протекающими в ускорителях. Многие периодические процессы можно моделировать простейшим гармоническим осциллятором (задачи 1, 2). Задачи 3, 12 посвящаются расчету распределения электростатических полей при различном расположении определенных зарядов. Подобные поля используются в некоторых классах установок как для ускорения пучков заряженных частиц, так и для их фокусировки.

Конфигурация высокочастотных полей в волноводах (на примере прямоугольного волновода) решается в задаче 4. Фокусирующие возможности статических магнитных и электрических полей в традиционных для ЭФУ элементах (квадрупольная линза, соленоид, диполь и т.п.), а также актуальные вопросы каналов транспортировки рассматриваются в задачах 5, 6, 7, 9. Движение частиц в скрещенных электромагнитных полях изучается задачах 8, 10 и 11.

Задача 1. Гармонический осциллятор.

Цель работы: Исследование работы гармонического осциллятора, решение системы дифференциальных уравнений [1].

Постановка задачи: Гармонический осциллятор (маятник) в условиях отсутствия сопротивления среды или трения, а также при отсутствии внешней вынуждающей силы, совершает незатухающие свободные колебания, которые можно описать уравнением

$$\frac{d^2x}{dt^2}+\omega^2 x=0$$

для заданной частоты ω и начальных условий x_0 и x_0'.

В классических разделах механики частота колебаний $\omega=\sqrt{\frac{g}{l}}$ (математический маятник, l – длина маятника, g – ускорение свободного падения), $\omega=\sqrt{\frac{k}{m}}$ (пружинный маятник, k – жесткость пружины, m – масса маятника).

Для построения принципиальной зависимости *x(t)* выбрать значения частот ω = 1, 2, 3, 4. При рассмотрении биений рассмотреть соотношение частот ≈ 1/10, амплитуду высшей гармоники выбрать значительно меньшей амплитуды основной гармоники.

Рассмотреть начальные условия x_o и x_o' как с нулевыми, так и с ненулевыми значениями.

Задание: Построить зависимости координаты и скорости от времени для заданной частоты ω и начальных условий x_0 и x_0'. Сложить несколько гармонических колебаний с различными (кратными) частотами и различными начальными условиями. Определить

условия возникновения биений. Определить амплитуду колебаний. Решение этого дифференциального уравнения записывается через тригонометрические функции $\sin(\omega t)$ и $\cos(\omega t)$, где t – время колебания осциллятора, а ωt – фаза колебания.

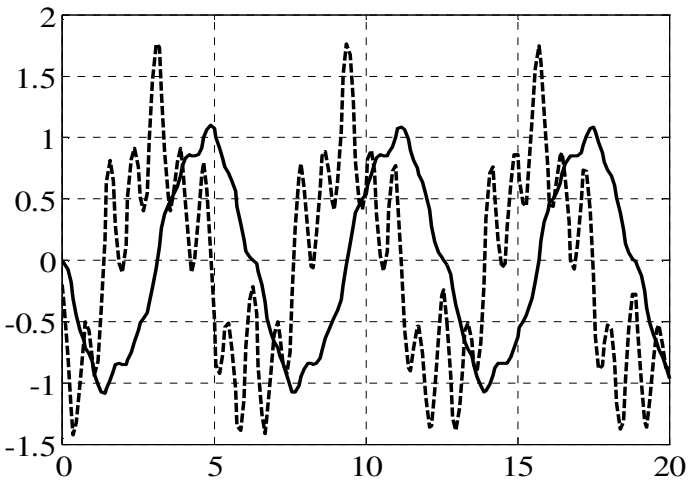


Рис. 2.5. Гармонические колебания. Биения

Выбрать частоту колебаний основной гармоники $\omega_0 = 1$, а частоты высших гармоник в диапазоне целых чисел [1 ÷ 10], время колебаний $t = 20$, начальные условия x_0 и $x_0^{'}$ можно задавать в диапазоне [-1 ÷ 1]. Дополнительно можно ввести параметр – амплитудный множитель для каждой частоты. На рисунке рис. 2.5 приведен примерный вид колебаний с биениями.

Задача 2. Устойчивость движения заряженных частиц в циклическом ускорителе.

Цель работы: На основе решения уравнения гармонического осциллятора найти условие устойчивого движения заряженных частиц в циклических ускорителях со слабой фокусировкой [2].

Постановка задачи: В циклических ускорителях со слабой фокусировкой устойчивое движение частиц по кольцевой орбите ра-

диусом R (рис. 2.6) обеспечивается выбором соответствующей конфигурации магнитного поля в поворотных магнитах ускорителя. Поле в дипольных магнитах является неоднородным и спадает по радиусу к краю магнита. Этот спад поля характеризуется коэффициентом n– показателем спада магнитного поля.

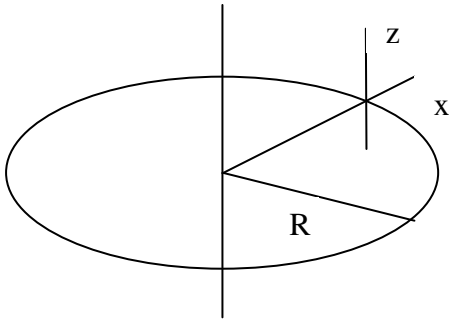


Рис. 2.6. Равновесная траектория заряженной частицы

Равновесной траекторией движения заряженных частиц в ускорителе можно (в самом простом приближении) считать движение по идеальной круговой орбите, лежащей в медианной плоскости магнитов. Радиус круговой орбиты может достигать нескольких километров, в то время как радиус вакуумной камеры, в которой ускоряются частицы, измеряется несколькими сантиметрами. Одной из задач проектирования циклических ускорителей является выбор параметров магнитной системы, при которых частица не «высаживаются» на стенки вакуумной камеры. В действительности, заряженные частицы в процессе движения по кольцу ускорителя совершают колебания относительно этой равновесной орбиты в вертикальном (z) и радиальном (x) направлениях. Эти колебания описываются в первом приближении уравнениями гармонического осциллятора.

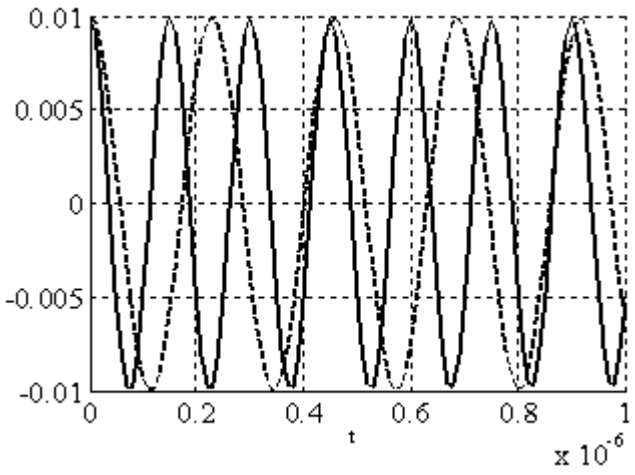


Рис. 2.7. Траектория заряженной частицы в вертикальном и радиальном направлении

Движение (колебания относительно равновесной орбиты) заряженных частиц в вертикальном (z) и радиальном (x) направлениях описывается в первом приближении уравнениями гармонического осциллятора

$$\ddot{z} + \omega^2 n z = 0$$

$$\ddot{x} + \omega^2 (1-n) x = 0,$$

где ω – круговая частота обращения (для решения данной задачи ≈ 50 ÷ 100 МГц). Время движения частицы (предел интегрирования) – 0,5 ÷ 1,0 мкс.

Задание: Найти и объяснить условия устойчивого движения заряженных частиц в обеих плоскостях (выбор n), построить траектории движения. Начальные условия выбрать исходя из максимальной амплитуды колебаний – 1 см.

Построить траектории частиц в вертикальном и радиальном направлениях (рис. 2.7).

Задача 3. Расчет конфигурации поля для статической системы зарядов.

Цель работы: Исследовать конфигурацию электростатических полей произвольной системы зарядов.

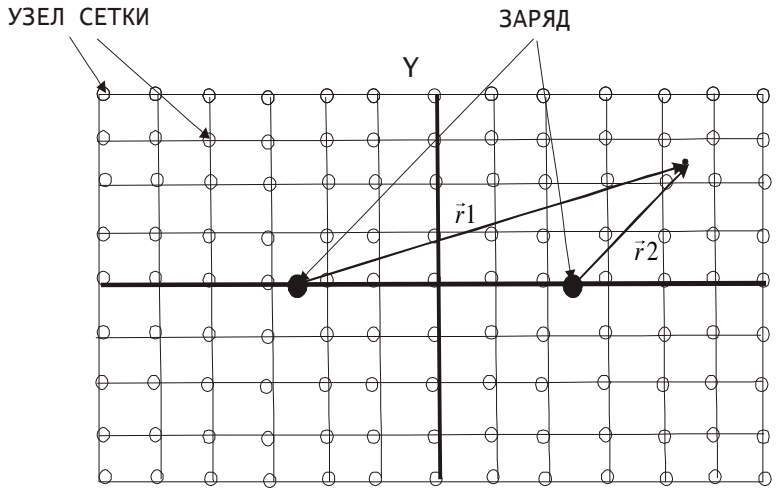


Рис. 2.8. Сетка для построения распределения потенциала поля

Постановка задачи: Используя выражения для потенциала и напряженности электростатического поля точечного заряда в произвольной точке пространства [3,4].

Задание: Задача разделяется на два варианта – система двух зарядов и система произвольного числа зарядов. Для системы двух зарядов – получить аналитическое и численное решение, сравнить результаты. Для произвольной системы зарядов получить численное решение.

1. Система двух зарядов (рис. 2.8). Построить силовые линии и эквипотенциальные линии для системы двух зарядов, расположенных на оси *X* симметрично относительно начала координат на расстояниях *« +a»* и *« -a»* соответственно. Рассмотреть два случая – когда заряды одного знака и когда заряды имеют разные знаки (рис. 2.9.).

Методические указания**.**

- Запишите выражения для потенциала и напряженности электростатического поля точечного заряда в произвольной точке пространства.
- При определении потенциала и напряженности электростатического поля используйте принцип суперпозиции полей.
- При интегрировании дифференциального уравнения силовых линий (аналитическое решение задачи) используйте замену переменных $v=(x-a)/y$ и $u=(x+a)/y$.
- Для получения хорошо различимого графического вида линий нужно согласовывать (подбирать) диапазоны изменения «*X*» и «*Y*» с шагами сетки по обеим координатам, а также с месторасположением зарядов «*a*».
- Варианты, когда заряды одного знака и когда заряды имеют разные знаки, отразить в разных частях экрана.

2. Произвольная система зарядов (рис. 2.10).

Алгоритм:

- Шаг 1: В заданной области определения задается матрица координат системы зарядов, формируется координатная сетка.
- Шаг 2: В цикле по числу зарядов формируется 3-мерный массив радиусов-векторов от каждого заряда к каждому узлу сетки.
- Шаг 3: В цикле по числу зарядов формируется 3-мерный массив значений потенциалов в каждом узле сетки от каждого заряда.
- Шаг 4: Исходя из принципа суперпозиции полей, определяется суммарный потенциал в узлах координатной сетки (*Us*).
- Шаг 5:Напряженность поля $\vec{E} = -\mathrm{grad}(Us)$.

Используются функции:

meshgrid – задается сетка, на которой строятся линии уровня (U = const*)*;

subplot – одновременная выдача графиков в разных частях экрана;

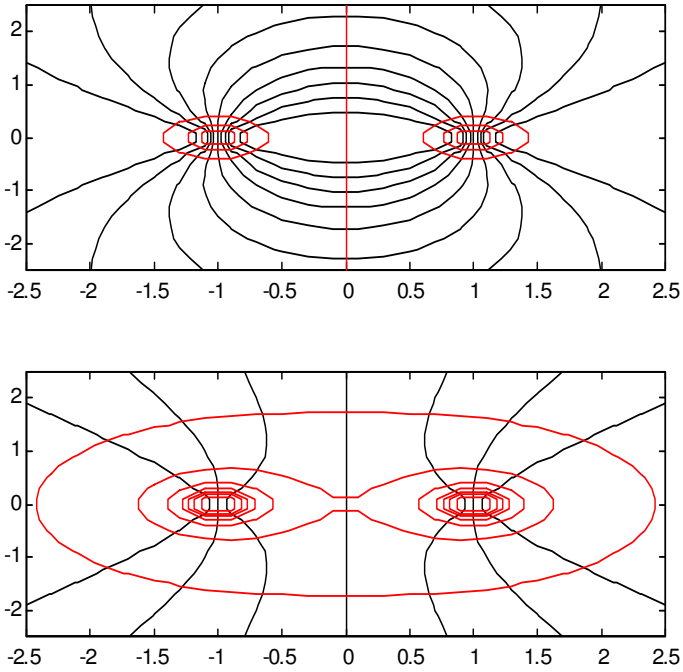


Рис. 2.9. Эквипотенциали и силовые линии системы двух зарядов

hold on, hold off – включение/выключение режима сохранения графика;
дополнительно можно использовать функции графического режима, позволяющие дополнить график текстовой информацией и цветовыми эффектами;
gradient(*Us*) – вычисление градиента;
quiver(*X,Y,Px,Py*) – построение силовых линий поля;
contour(*X,Y,Us,60,'r'*) – построение линий потенциала.

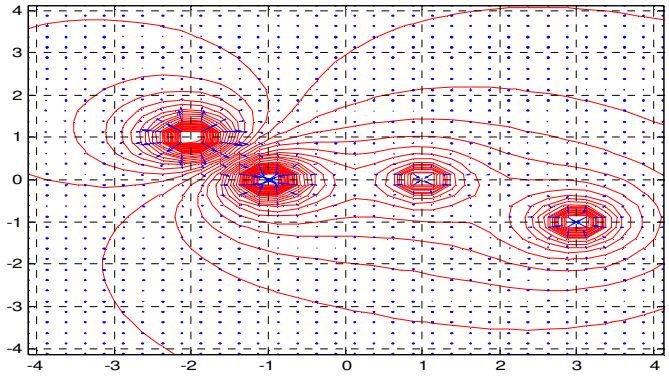


Рис. 2.10. Эквипотенциали и силовые линии произвольной системы зарядов

Задача 4. Расчет конфигурации ВЧ-поля в прямоугольном волноводе.

Цель работы: Исследовать конфигурацию электрических и магнитных полей в прямоугольном волноводе [5].

Постановка работы: Прямоугольный волновод представляет собой трубу прямоугольного поперечного сечения (рис. 2.11), в котором распространяется электромагнитная волна. В зависимости от типа волны изменяется конфигурация силовых линий поля. Необходимо построить силовые линии электрического и магнитного полей в прямоугольном волноводе. Рассматриваются два типа полей H_{mn} и E_{mn}. Ширина поперечного сечения волновода – «*а*», высота – «*в*». Соотношение сторон в сечении волновода – *а/в* ≈ 2/1. Силовые линии электрического и магнитного поля для обоих типов волн H_{mn} и E_{mn} строятся в поперечной плоскости *XY*. Кроме того, в плоскости *XZ* строится силовая линия магнитного поля для волн типа H_{mn}, а для волн типа E_{mn} строится силовая линия электрического поля. Уравнения силовых линий имеют вид:

$$dx / E_x = dy / E_y;\ \ dx / H_x = dy / H_y; dx / H_x = dy / H_z;$$
$$dx / E_x = dy / E_y;\ \ dx / E_x = dy / E_z; dx / H_x = dy / H_y.$$

Задание: Рассмотреть типы волн в прямоугольном волноводе

$$E_{11}, E_{12}, E_{21}, E_{22}, H_{10}, H_{11}, H_{20}, H_{21} \quad [5].$$

Решение: Для заданных выражений составляющих электрических и магнитных полей задача решается аналитически. Полученные уравнения силовых линий изображаются графически средствами MATLAB.

Используются функции:
meshgrid – задается сетка, на которой строятся линии уровня,
contour – прорисовывает линии уровня,
subplot – одновременная выдача графиков в разных частях экрана,
hold on, hold off – включение/выключение режима сохранения графика.

Дополнительно можно использовать функции графического режима, позволяющие дополнить график текстовой информацией и цветовыми эффектами.

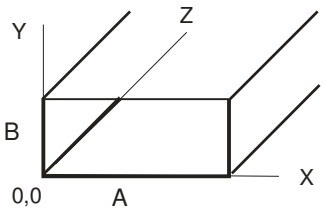


Рис. 2.11. Прямоугольный волновод

Для решения задачи необходимо создать соответствующие сетки полей. Решения дифференциальных уравнений силовых линий представляют собой произведения или отношения функций cos(.) и sin(.).

Исходные соотношения.

1. Составляющие полей для волн типа H_{mn}:

$$E_x = -j\frac{n\pi}{b}\cos(\frac{m\pi}{a}x)\sin(\frac{n\pi}{b}y);$$

$$E_y = j\frac{m\pi}{a}\sin(\frac{m\pi}{a}x)\cos(\frac{n\pi}{b}y);$$
$$E_z = 0;$$

$$H_x = -j\frac{m\pi}{a}\sin(\frac{m\pi}{a}x)\cos(\frac{n\pi}{b}y);$$
$$H_y = -j\frac{n\pi}{b}\cos(\frac{m\pi}{a}x)\sin(\frac{n\pi}{b}y);$$
$$H_z = \pi^2(\frac{m^2}{a^2}+\frac{n^2}{b^2})\cos(\frac{m\pi}{a}x)\cos(\frac{n\pi}{b}y)$$

2. Составляющие полей для волн типа E_{mn}:

$$E_x = j\frac{m\pi}{a}\cos(\frac{m\pi}{a}x)\sin(\frac{n\pi}{b}y);$$
$$E_y = j\frac{n\pi}{b}\sin(\frac{m\pi}{a}x)\cos(\frac{n\pi}{b}y);$$
$$E_z = \pi^2(\frac{m^2}{a^2}+\frac{n^2}{b^2})\sin(\frac{m\pi}{a}x)\sin(\frac{n\pi}{b}y);$$

$$H_x = -j\frac{n\pi}{b}\sin(\frac{m\pi}{a}x)\cos(\frac{n\pi}{b}y);$$
$$H_y = j\frac{m\pi}{a}\cos(\frac{m\pi}{a}x)\sin(\frac{n\pi}{b}y)$$
$$H_z = 0.$$

Во всех формулах знак «=» записан условно, так как опущены постоянные коэффициенты, не влияющие на картину распределения силовых линий. Параметры m и n – волновые моды – принимают целочисленное значение от 0 и выше. Они определяют число вариаций поля по горизонтальной и вертикальной оси соответственно. Низшая мода волны $H_{mn} \rightarrow H_{10.}$ Низшая мода волны $E_{mn} \rightarrow$

E_{11}. Параметр j определяет фазовый сдвиг между компонентами полей. Значения a и b можно принять a = 10 см и b = 5 см.

Примечание. При построении силовых линий (рис. 2.12) необходимо учитывать, что производится масштабирование значений поля в узлах сетки. Отношения типа sin(π/2)/sin(0) дают бесконечно большие значения, что приводит к «прижиманию» линий на графике к боковым границам волновода. Исключить этот эффект можно преобразовав сетки таким образом, чтобы точки (где аргумент тригонометрических функций принимает значения 0 или π) оказались «выколотыми» из исходной сетки.

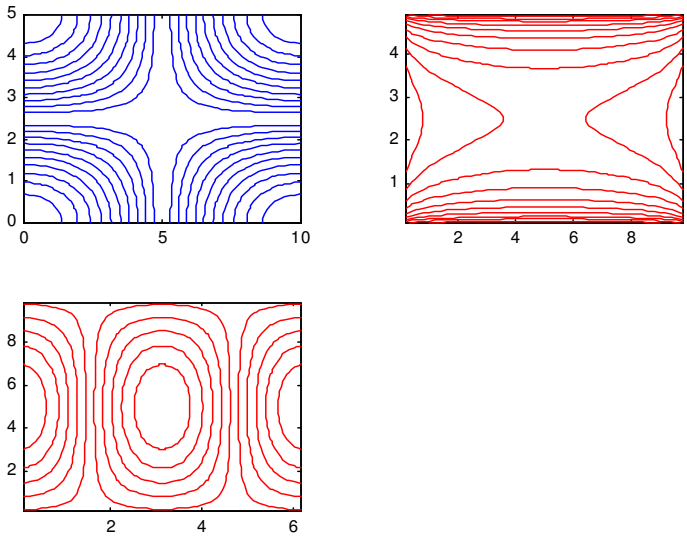


Рис. 2.12. Распределение силовых линий электрического и магнитного полей в волноводе

Задача 5. Взаимодействие заряженных частиц с магнитным полем.

Цель работы: Исследовать фокусирующие свойства квадрупольной магнитной линзы [6].

Постановка задачи: Квадрупольная линза – магнитное устройство, имеющее четыре магнитных полюса, поперечное сечение кото-

рого приведено на рис. 2.13. Длина линзы – *l*. Заряженные частицы двигаются в пространстве между полюсами. В первом приближении поле в магнитном квадруполе определяется градиентом поля [6]

$$g = dB_z / dx \mid_{x=z=0} = dB_x / dz \mid_{x=z=0}$$

и, соответственно, $B_x = g * z, \quad B_z = g * x$,

где x и z горизонтальная и вертикальная оси симметрии квадруполя. Потенциальная функция *Fi* записывается относительно осей повернутых на 45° (x_1,y_1) и имеет вид $B = -g * x_1 * y_1$. Она определяет конфигурацию силовых линий магнитного поля. Компоненты поля B_x, B_z определяются из условия $\vec{B} = -\mathrm{grad}(Fi)$

Для построения силовых линий и вектора градиента поля используются функции *meshgrid(x,z); contour(X,Z,Fi,'r'); gradient(Fi); hold on; quiver(X,Z,Bx,Bz); hold off* (рис. 2.14).

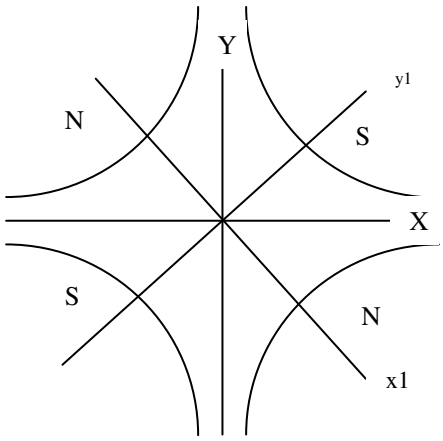


Рис. 2.13. Схема расположения магнитных полюсов квадруполя

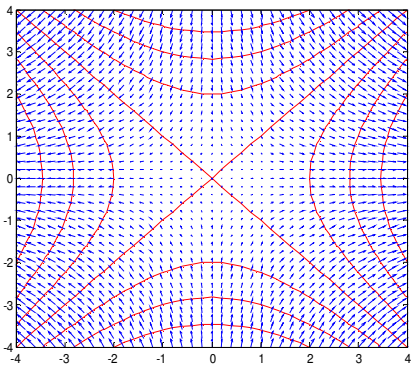


Рис. 2.14. Распределение эквипотенциалей и силовых линий магнитного поля

Взаимодействие заряженной частицы с магнитным полем определяется силой Лоренца

$$\vec{F} = q*[\vec{V}*\vec{B}],$$

где $\vec{V}$ - скорость движения частицы, q – заряд частицы.

Уравнение движения частицы имеет вид $d\vec{p}/dt = \vec{F}$, где p – импульс заряженной частицы. Исходя из конфигурации магнитного поля квадруполя, оказывается, что в одной из своих плоскостей симметрии он оказывает на заряженные частицы фокусирующее действие, а в другой – дефокусирующее (рис. 2.15). В результате преобразований уравнения движения заряженной частицы в поле квадруполя принимают вид.

$$\ddot{x} + kx = 0 \qquad \text{и} \qquad \ddot{z} - kz = 0$$

Для фокусирующей и дефокусирующей плоскостей. Решение записывается в виде матриц:

$$\begin{vmatrix} z \\ z' \end{vmatrix} = \begin{vmatrix} ch(\theta) & l/\theta * sh(\theta) \\ \theta/l * sh(\theta) & ch(\theta) \end{vmatrix} \begin{vmatrix} z_0 \\ z_0' \end{vmatrix},$$

$$\begin{vmatrix} x \\ x' \end{vmatrix} = \begin{vmatrix} \cos(\theta) & l/\theta * \sin(\theta) \\ -\theta/l * \sin(\theta) & \cos(\theta) \end{vmatrix} \begin{vmatrix} x_0 \\ x_0' \end{vmatrix},$$

где $\theta = \sqrt{3 * g / p} * l$ – пролетный угол, $k=3*g/p$, g – градиент магнитного поля *[кгс/см]*, p - импульс заряженных частиц *[Гэв/с]*, l- длина линзы *[м]*.

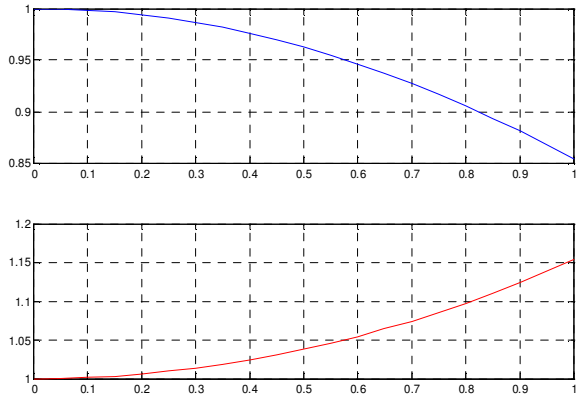


Рис. 2.15. Траектории заряженных частиц в фокусирующей и дефокусирующей плоскостях

Задание: Построить эквипотенциали и силовые линии магнитного поля квадруполя и траектории заряженной частицы в обеих плоскостях движения.

Начальные значения $x_{0,}$ z_0 задаются в миллиметрах, а расходимость пучка заряженных частиц x_0', z_0' - в миллирадианах. (Для примера, $x_{0,}= z_0=1$ мм, $x_0' = z_0' =1$ мрад.)
импульс частиц $\boldsymbol{p}$ = 10 – 200 ГэВ/с,
длина квадруполя $\boldsymbol{l}$ = 0.25 – 2.0 м,
градиент магнитного поля $\boldsymbol{g}$ = 0.2 – 1.0 кгс/см.

Задача 6. Расчет динамики заряженных частиц в магнитооптическом канале транспортировки.

Цель работы: Исследовать фокусирующие свойства канала транспортировки пучков заряженных частиц высоких энергий [6].

Постановка задачи: Простейший фокусирующий объектив – дублет квадрупольных линз (см. предшествующую задачу). На схеме (рис. 2.16) $Q1$ и $Q2$ – квадрупольные линзы.

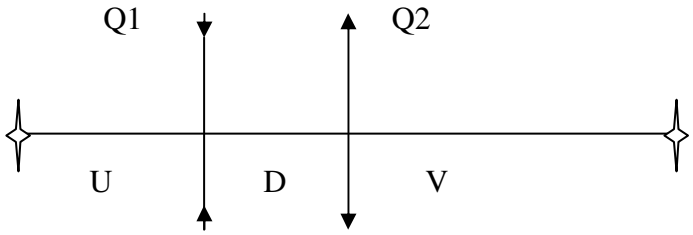


Рис. 2.16. Схема канала транспортировки заряженных частиц

Расстояние от источника до первой (дефокусирущей) линзы U, расстояние между линзами D, расстояние от второй (фокусирущей) линзы до изображения V. В матричном виде движение заряженной частицы в свободном пространстве описывается как:

$$\begin{vmatrix} x \\ x' \end{vmatrix} = \begin{vmatrix} 1 & L \\ 0 & 1 \end{vmatrix} \begin{vmatrix} x_0 \\ x_0' \end{vmatrix}.$$

Нужно задать исходный фазовый портрет пучка (его граница - «прямой эллипс») и трансформировать его в конец канала, т.е. воздействовать на каждую точку границы эллипса матрицей передачи данного участка канала. Подобрать геометрические параметры канала и пролетные углы квадруполей таким образом, чтобы в плоскости изображения фазовый эллипс пучка принял (по возможности) «прямое» положение.

Фокусное расстояние любого объектива можно определить следующим образом. Запишем матрицу M (матрицу передачи участка U-$Q1$-D-$Q2$) в виде произведения матриц отдельных участков канала. Нужно подобрать значение расстояния U таким, чтобы элемент

матрицы M_{22} равнялся нулю. Это значение U и будет фокусным расстоянием объектива F.

Задание: Построить траекторию частицы и фазовые портреты пучка на выходе каждого элемента канала и в плоскости изображения.

Начальные значения x_0 задаются в миллиметрах, а расходимость пучка заряженных частиц x_0', – в миллирадианах. (Для примера, $x_0 = z_0 = 1$ мм, $x_0' = z_0' = 1$ мрад.)

импульс частиц p = 10 – 200 ГэВ/с,

длина квадруполя l = 0.25 – 2.0 м,

градиент магнитного поля g = 0.2 – 1.0 кгс/см,

расстояние между линзами – D = 0.5 – 5.0 м.

Расстояние от источника до первой (дефокусирущей) линзы U должно быть больше фокусного расстояния дублета F. В качестве начального значения можно принять $U = 2F$. Расстояние от второй (фокусирущей) линзы до изображения (V) подбирается таким образом, что размеры пучка становятся минимальными («прямой» фазовый эллипс).

Задача 7. Фокусировка нерелятивистских электронов одиночной диафрагмой.

Цель работы: Изучение фокусирующих свойств отдельной диафрагмы [7].

Постановка задачи: Отдельная диафрагма (рис. 2.17) является простейшим устройством, позволяющим изменять траекторию движения заряженных частиц (электронная линза). Электроны, проходя сквозь круглое отверстие в плоской пластине, которая разделяет области с разной напряженностью электрического поля, будут отклоняться относительно продольной оси системы.

Этот эффект используется для фокусировки пучка электронов на мишень. Общее действие линзы может быть как фокусирующим, так и дефокусирующим, в зависимости от параметров системы. Распределение потенциала электрического поля $\phi(r, z)$ облас-

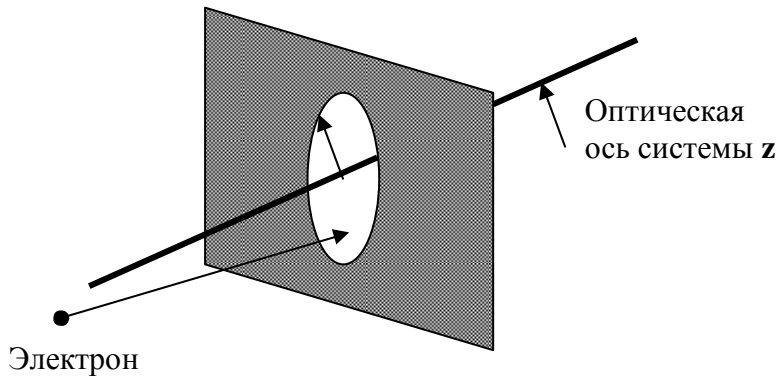


Рис. 2.17. Схема фокусировки электронов одиночной диафрагмой

ти диафрагмы описывается полуэмпирической зависимостью [7]:

$$\phi(r,z) = \phi_0 - (E_1 + E_2)/2 * z + (E_1 - E_2)/\pi * |z| * (Arctg(\mu) + 1/\mu),$$
$$\mu = [(z^2 + r^2 - R^2)/2/R^2 + 1/2 * \{4 * z^2/R^2 + (z^2/R^2 + r^2/R^2 - 1)^2\}^{1/2}]^{1/2},$$

где E_1, E_2 – напряженности электрического поля слева и справа от диафрагмы, R – радиус отверстия, r, z – радиальная и продольная координаты, ϕ_0 – потенциал диафрагмы.

На рис. 2.18. приведен вариант распределения электростатического поля по обе стороны диафрагмы, сплошные линии – эквипотенциали, стрелками отображена напряженность поля.

Уравнение движения электрона в электростатическом поле имеет вид:

$$d\vec{p}/dt = e * \vec{E}.$$

Задание: Построить эквипотенциали и силовые линии отдельной диафрагмы. Варьируя параметры системы, оценить изменения конфигурации электрического поля. Записать уравнение движения электрона. Рассчитать траектории движения электронов по информации о значениях напряженности поля в узлах сетки, используя значения градиента потенциала электрического поля в области

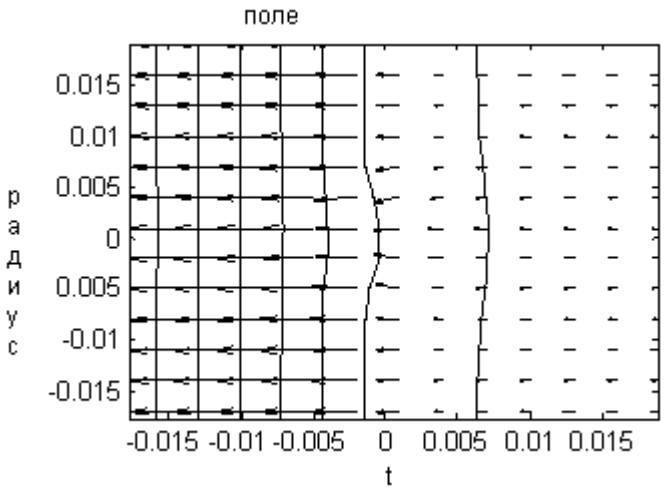


Рис. 2.18. Распределение эквипотенциалей и силовых линий электрического поля

отверстия в диафрагме. Оценить величины вертикальных и горизонтальных составляющих градиента и объяснить механизм фокусировки. Найти условия фокусировки и дефокусировки электронов отдельной диафрагмой.

В качестве начальных данных можно принять следующие значения:

Fi0 = 0 – потенциал диафрагмы (В)
E1 = 5000 – напряженность поля слева от диафрагмы (В/м)
E2 = 0 – напряженность поля справа от диафрагмы (В/м)
Rd = 0.005 – радиус отверстия в диафрагме (м)
e = -1.6e-19 – заряд электрона
m = 9.1e-31 масса электрона
c = 3e8 – скорость света
bet = 0.001– относительная продольная скорость электрона
*vz = bet*c* – продольная скорость электрона
vr = 0 – радиальная скорость электрона
tk = 4.0e-7 – предел интегрирования по времени
zr0 = [-0.005 vz 0.005 vr] – вектор начальных условий
hz = 0.003 – шаг сетки в метрах
hr = hz – шаг сетки в метрах
Z = -0.05:hz:0.05 – продольная координата в метрах

$R = Z$ – радиальная координата в метрах.

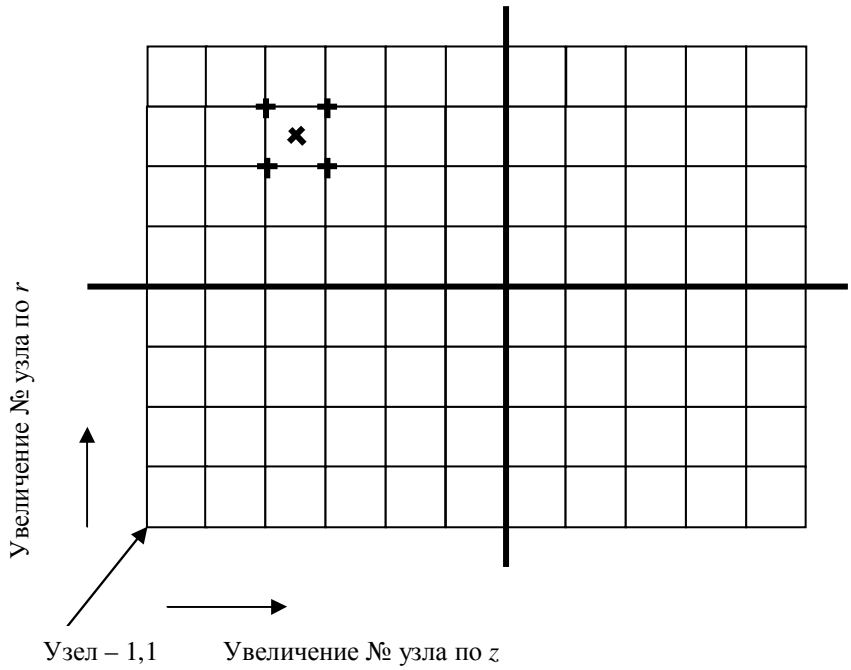


Рис. 2.19. Сетка для интерполяции значений электрического поля

Методические указания.

Задача аксиально-симметричная. Рассматриваем плоскость $\{z,r\}$.

Выделить диапазоны изменения продольной и радиальной координат.

В выделенной области задать узлы сетки, в которых вычисляется напряженность поля. Шаг сетки лучше выбрать таким образом, чтобы не было узлов, приходящихся на нулевые координаты. Это обеспечит прорисовку эквипотенциалей без разрыва линий.

В записи функции правых частей системы дифференциальных уравнений, описывающих движение электрона, использовать двумерную интерполяцию по узлам сетки.

Узлы, помеченные прямым крестом, – узлы сетки с номерами *nz* и *(nz-1)* по оси *z* и номерами *nr* и *(nr-1)* по оси *r*, между которыми

проводится двумерная интерполяция значений электрического поля для вычисления поля в точке, помеченной косым крестом (рис. 2.19).

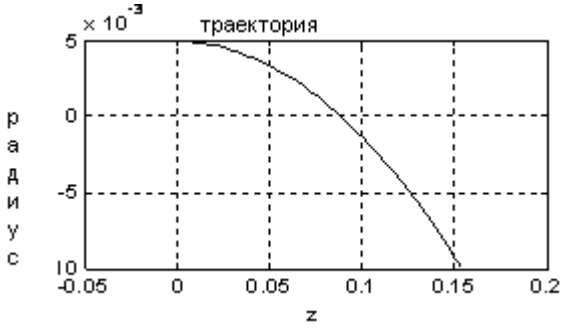


Рис. 2.20. Траектория электрона при прохождении диафрагмы

Задача 8. Движение заряженной частицы в скрещенных электрических и магнитных полях (рис. 2.21).

Цель работы: Исследовать характер движения заряженной частицы в различных сочетаниях действия электрического и магнитного полей [8].

Постановка задачи: Движение заряженной частицы в скрещенных полях описывается уравнением:

$$d\vec{P} / dt = q * (\vec{E} + [\vec{V} \times \vec{B}]).$$

Рассмотрим типичную задачу – движение заряженной частицы в скрещенных магнитном ($\vec{B}$) и электрическом ($\vec{E}$) полях. Для нерелятивистской частицы отношение ее заряда к массе практически постоянно и можно записать уравнение движения как

$$d\vec{r} / dt = q / m * (\vec{E} + [\vec{V} \times \vec{B}]).$$

Скорость частицы $\vec{V}$ можно записать как $\beta\vec{c}$, где c – скорость света. Для нерелятивистской частицы относительная скорость имеет значения: $\left|\vec{\beta}\right| < 0.1 \ (\beta = \frac{v}{c})$.

В медианной плоскости движение частицы описываемся системой уравнений.

$$\ddot{z} = 0; \quad \ddot{x} = \frac{q}{m}(\dot{y}B_z + E_x); \quad \ddot{y} = -\frac{q}{m}(\dot{x}B_z - E_y).$$

Положим $B_z = B_0 - ky$, т.е. поле линейно спадает вдоль оси *y* (*k* – коэффициент, определяющий скорость спадания поля).

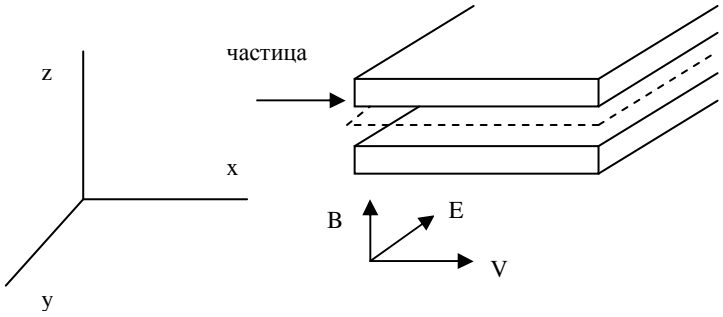


Рис. 2.21. Схема ориентации электрического и магнитного полей

Частные случаи (рис. 2.22):

- движение частицы в постоянном поперечном электрическом поле,
- движение частицы в медианной плоскости дипольного магнита с постоянным и линейно спадающим к краю магнитным полем,
- движение частицы в медианной плоскости дипольного магнита с постоянным или линейно спадающим к краю магнитным

полем и перпендикулярным к нему постоянным электрическим полем.

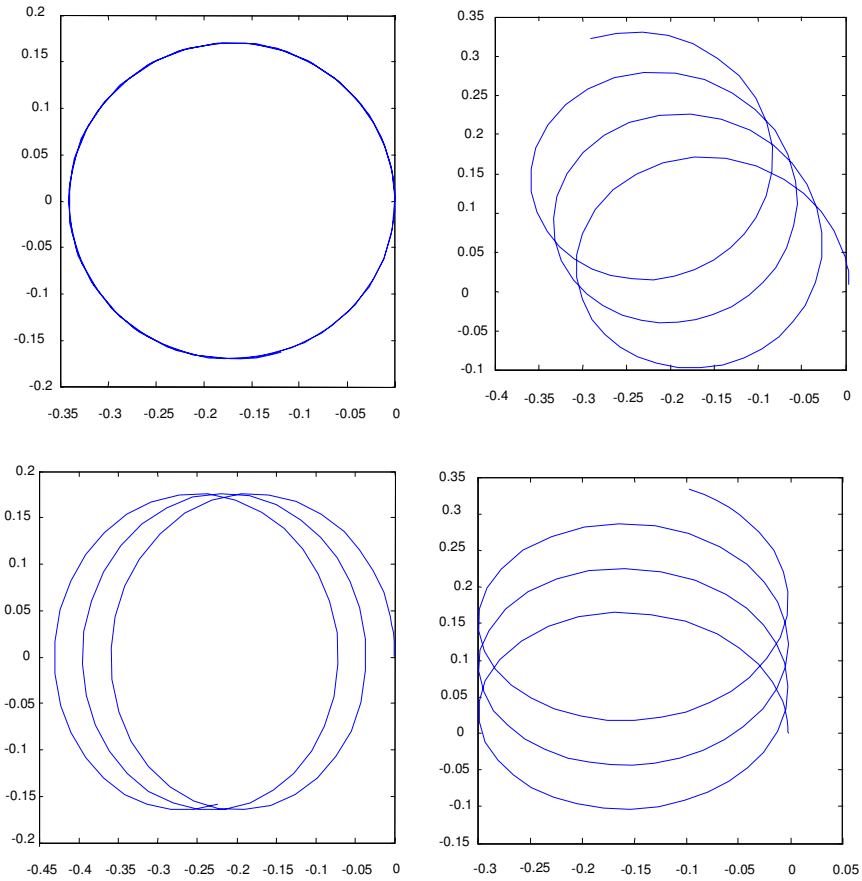


Рис. 2.22. Траектории движения электронов

Задание: Рассмотреть траектории частицы в зависимости от различных сочетаний направлений векторов $\vec{V}$, $\vec{E}$, $\vec{B}$ для нерелятивистских электронов:

магнитное поле B = 0.001 – 0.1 Тл,

электрическое поле E = 1000 В,
коэффициент спада магнитного поля k = 0.001 – 0.01.

Задача 9. Фокусировка нерелятивистских электронов соленоидом (рис. 2.23).

Цель работы: Исследовать фокусирующие свойства соленоида.

Постановка задачи: Движение заряженной (нерелятивистской) частицы происходит в продольном магнитном поле. Фокусировка частиц происходит за счет нелинейных полей на краях соленоида (краевая фокусировка) [9].

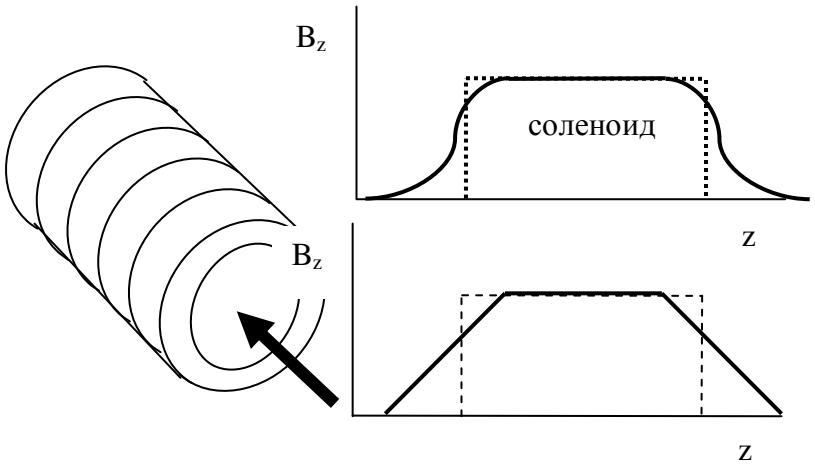


Рис. 2.23. Распределение поля в соленоиде

Реальное распределение магнитного поля по продольной оси соленоида можно разделить на три области. Центральная область с однородным постоянным магнитным полем, направленным вдоль оси соленоида. Две краевые области, в которых магнитное поле спадает до нуля. Эти области достаточно хорошо можно аппроксимировать линейной зависимостью. Длина этих участков порядка апертуры соленоида.

Задача аксиально-симметричная. Заряженная частица имеет продольную скорость много большую, чем поперечная ее составляющая. Траектория частицы – пространственная спираль, проекция

которой на поперечную плоскость представляет собой окружность. Уравнение движения частицы в общем виде записывается как

$$dP/dt = q*([\vec{V}\times\vec{B}]).$$

В соленоиде магнитное поле $\vec{B}$ в центральной области имеет только продольную составляющую, а в краевых областях – продольную и радиальную составляющие.

Для центральной области решение уравнения движения имеет вид:

$$\begin{pmatrix} x_2 \\ x_2' \\ y_2 \\ y_2' \end{pmatrix} = \begin{pmatrix} 1 & l*\sin\theta/\theta & 0 & l(1-\cos\theta)/\theta \\ 0 & \cos\theta & 0 & \sin\theta \\ 0 & -l(1-\cos\theta)/\theta & 1 & l*\sin\theta/\theta \\ 0 & -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} x_1 \\ x_1' \\ y_1 \\ y_1' \end{pmatrix}.$$

Для краевых областей – входной и выходной – матрицы имеют вид:

$$\begin{pmatrix} x_1 \\ x_1' \\ y_1 \\ y_1' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & \theta/(2*l) & 0 \\ 0 & 0 & 1 & 0 \\ -\theta/(2*l) & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_0' \\ y_0 \\ y_0' \end{pmatrix};$$

$$\begin{pmatrix} x_3 \\ x_3' \\ y_3 \\ y_3' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -\theta/(2*l) & 0 \\ 0 & 0 & 1 & 0 \\ \theta/(2*l) & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_2 \\ x_2' \\ y_2 \\ y_2' \end{pmatrix},$$

т.е. они рассматриваются как преломляющие плоскости на входе и выходе соленоида, где θ – угол поворота по круговой орбите при прохождении участка соленоида длиной l.

Движение по окружности происходит с циклотронной частотой $\omega = \frac{e}{m}B$. Угол, на который повернется частица по азимуту, опре-

деляется длиной соленоида и продольной составляющей скорости частицы V_z :

$$\theta = \frac{e}{m_0 c}\frac{Bl}{\beta_n \gamma}; \qquad r = \frac{m_0 c \beta_n \gamma}{eB},$$

где r – радиус круговой орбиты электрона, e, m_0 - заряд и – масса покоя электрона, скорость света, β,γ – относительная скорость и относительная масса электрона, β_n – нормальная составляющая скорости электрона, $\vec{B}$ – магнитная индукция поля соленоида.

Фокусировка соленоидами применяется для электронных пучков.

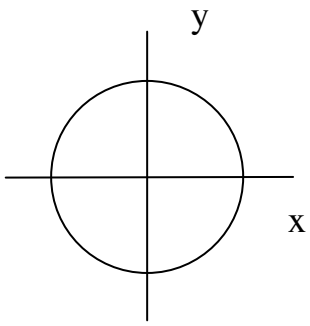


Рис. 2.24. Поперечное сечение пучка

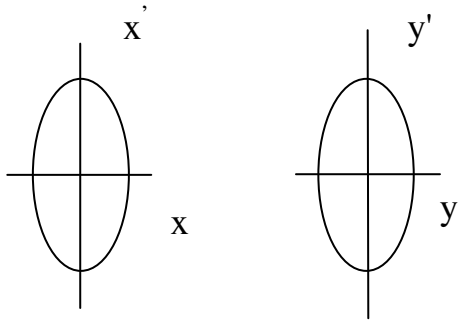


Рис. 2.25. Фазовый портрет пучка в плоскостях ***x*** и ***y***

y

Задание: Примерные параметры системы:

$\beta = 0.05 - 0.1$, $\gamma \approx 1$, $B = 0.01 - 0.03$ Тл, $l = 0.1 - 0.5$ м .

Диаметр сечения пучка -2 мм, максимальная угловая расходимость частиц в пучке 10 мрад.

Для заданных параметров соленоида и заряженной частицы:

- вычислить радиус круговой орбиты;
- определить длину соленоида, при которой электрон совершает полный оборот (2π) в плоскости перпендикулярной продольной оси движения;
- вывести матрицы передачи каждого участка, а также суммарную матрицу всего соленоида и произведение матриц крайних областей соленоида. Прокомментировать численные результаты;

- построить преобразованный фазовый объем пучка (эллипс) после прохождения соленоида.

Задача 10. Расчет динамики заряженных частиц в дипольном магните (рис. 2.26).

Цель работы: Рассмотреть фокусирующие свойства дипольного магнита [6].

Постановка задачи: Секторный магнит – дипольный магнит с формой полюсных наконечников в виде сектора круга. В системе секторного магнита траектория осевой заряженной частицы строго перпендикулярна к входной и выходной плоскости магнита, т.е. в криволинейной системе координат

$$x_{вх} = 0, \quad x'_{вх} = 0,$$
$$x_{вых} = 0, \quad x'_{вых} = 0.$$

Для идеализированного магнита уравнения движения заряженной частицы в горизонтальной и вертикальной плоскостях записываются следующим образом:

$$x'' - (k - 1/\rho)x = -1/\rho \frac{\Delta p}{p}; \quad z'' + kz = 0,$$

где ρ – радиус кривизны траектории частицы в горизонтальной плоскости;

$1/\rho[м^{-1}] = 0.03\dfrac{B[кгс]}{p[Гэв/с]}$, B – магнитная индукция, p – импульс частицы.

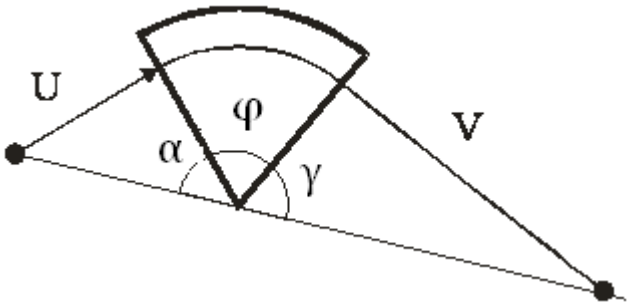


Рис. 2.26. Фокусировка секторным магнитом

В данном приближении решения задачи положим, что поле в магните постоянно и однородно ($k = 0$) и в пучке отсутствует разброс частиц по импульсу ($\frac{\Delta p}{p} = 0$). Решение уравнений движения в матричном виде запишется как

$$\begin{bmatrix} x \\ x' \end{bmatrix} = \begin{bmatrix} \cos\varphi & \rho\sin\varphi \\ -\frac{1}{\rho}\sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} x_0 \\ x_0' \end{bmatrix}; \qquad \begin{bmatrix} z \\ z' \end{bmatrix} = \begin{bmatrix} 1 & \rho\varphi \\ 0 & 1 \end{bmatrix} \begin{bmatrix} z_0 \\ z_0' \end{bmatrix},$$

где φ - пролетный угол магнита.

На рисунке U – расстояние от источника частиц до магнита, V – расстояние от магнита до изображения источника матрицы передачи этих участков, где x, z, x', z' – линейные и угловые параметры пучка:

$$\begin{bmatrix} 1 & U \\ 0 & 1 \end{bmatrix}; \quad \begin{bmatrix} 1 & V \\ 0 & 1 \end{bmatrix}.$$

Задание: Рассмотреть два варианта – прохождение через магнитную систему электронов и протонов.

Электроны:

импульс 0.001 – 0.1 ГэВ/с;

магнитная индукция 0.1 – 1.0 кгс.

Протоны:

импульс 1.0 – 100 ГэВ/с;

магнитная индукция 1.0 – 10.0 кгс.

Пролетный угол 30 – 90 градусов (задается в радианах).

Для выбранного варианта магнита рассчитать радиус кривизны траектории и матрицу передачи в горизонтальной плоскости. Оценить величину элемента этой матрицы M_{21} ($F=1/M_{21}$ приблизительно равно фокусному расстоянию системы). Задать $U \approx 2F$.

Вычислить общую матрицу системы U-M-V. Подобрать параметр V таким образом, чтобы элемент M_{12} общей матрицы системы стремился к нулю. В этом случае на расстоянии V от выхода магнита получается изображение источника. Построить фазовые портреты пучка (фазовые эллипсы) в этой точке продольной оси системы.

Начальные размеры пучка $x_{0,=}$ $z_{0=}$*1 мм,* $x_0' = z_0' = 1$ *мрад.*

Задача 11. Расчет динамики заряженных частиц в магнитном зеркале (рис. 2.27, 2.28).

Цель работы: Рассмотреть «зеркальные» свойства дипольного магнита.

Постановка задачи: Магнитное зеркало – это обычный дипольный магнит установленный таким образом, что заряженная частица входит в пространство между полюсами магнита под углом φ к боковой грани магнита. Под действием магнитного поля траектория частицы поворачивает относительно входного направления и выходит через ту же грань магнита под тем же углом φ. Эта симметрия траектории частицы сохраняется и при учете краевого спадающего поля магнита.

Движение частицы в медианной плоскости дипольного магнита с постоянным или линейно спадающим краевым магнитным полем описывается уравнением:

$$dP / dt = q * ([\vec{V} \times \vec{B}]) .$$

Скорость частицы $\vec{V}$ можно записать как $\beta\vec{c}$, где c – скорость свет; $\vec{B}$ – магнитная индукция, q – заряд частицы.

В медианной плоскости движение частицы описываемся системой уравнений:

$$\ddot{z} = 0;\ \ddot{x} = \frac{q}{m}\dot{y}B_z;$$

$$\ddot{y} = -\frac{q}{m}\dot{x}B_z.$$

Положим $B_z = B_0 - ky$, т.е. магнитное поле линейно спадает вдоль оси y (k – коэффициент, определяющий скорость спадания поля).

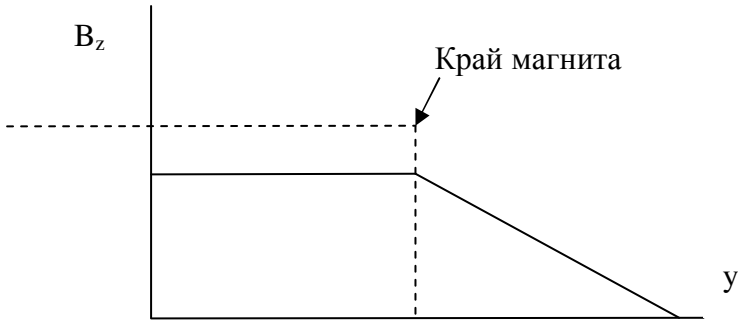


Рис. 2.27. Распределение краевого магнитного поля

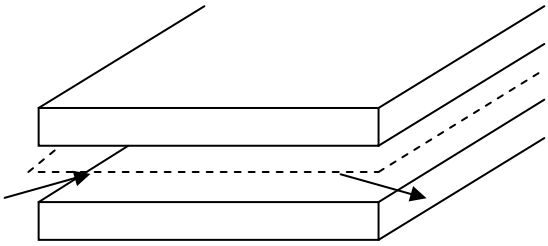


Рис. 2.28. Схема магнитного зеркала

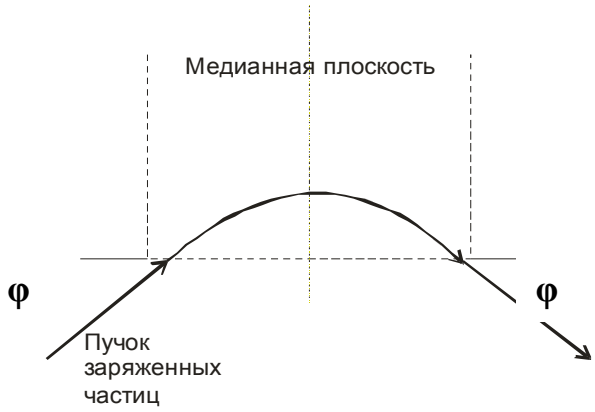


Рис. 2.29. Траектория заряженных частиц в магнитном зеркале

Задание: Рассмотреть прохождение через магнитную систему электронов, построить траектории частиц (рис. 2.29) для различных параметров магнита. Вычислить радиус круговой орбиты электронов и их угол поворота по круговой орбите.

Электроны:

импульс 0.001 – 0.1 ГэВ/с;

магнитная индукция 0.1 – 1.0 кгс.

$$r = \frac{m_0 c \beta\gamma}{qB}; \quad \theta = \frac{q}{m_0 c}\frac{Bl}{\beta\gamma} \ ,$$

где θ – угол поворота по круговой орбите , r – радиус круговой орбиты частицы, q, m_0 – заряд и масса покоя частицы, с – скорость света, β,γ – относительная скорость и относительная масса частицы, B – магнитная индукция поля.

Задача 12. Расчет конфигурации внешнего электростатического поля с внесенным в него проводящим или диэлектрическим шаром (рис. 2.30).

Цель работы: Изучить изменения однородного электрического поля при внесении в него проводящего или диэлектрического шара [10]. Исследовать изменение конфигурации электрического поля при различных сочетаниях диэлектрических свойств среды и шара.

Постановка задачи: Во внешнее однородное электрическое поле вносится шар. В одном случае шар проводящий, во втором случае – диэлектрический. Происходит изменение конфигурации первоначального распределения поля. Для диэлектрического шара распределение поля зависит от соотношения значений диэлектрической проницаемости внешней среды и шара.

Задание: Построить эквипотенциали электростатического поля в системе «*внешнее однородное поле – шар*».

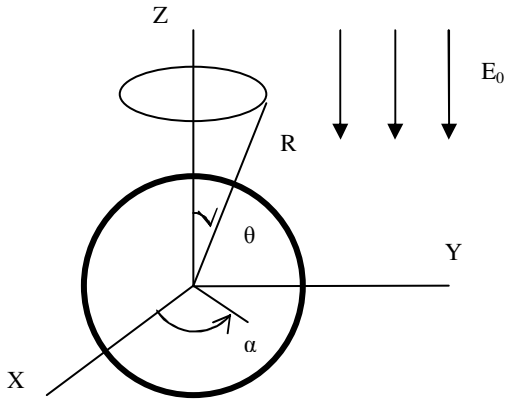


Рис. 2.30. Шар во внешнем поле

Отобразить картину поля в системе – «равномерное внешнее поле с напряженностью E_0 и шар (металлический или диэлектрический)». Шар может быть как заряженным, так и незаряженным. Радиус шара – *a,* внешнее поле E_0 направлено вдоль оси ***z***. Поле данной системы описывается уравнением Лапласа (в сферической системе координат):

$$\frac{1}{R^2}\frac{\partial(R^2 * \partial\varphi/\partial R)}{\partial R} + \frac{1}{R^2 \sin\theta}\frac{\partial(\sin\theta * \partial\varphi/\partial\theta)}{\partial\theta} = 0,$$

где φ – потенциал поля.

Для проводящего шара, если он не заряжен, все точки плоскости *xoy*, проходящей через центр шара, имеют один и тот же потенциал – φ_0. Для заряженного шара (заряд – Q) потенциал на его поверхности равен:

$$\frac{Q}{4\pi\xi_0\xi_a a}.$$

Для всех точек пространства вне шара потенциал определяется как

$$\varphi = \frac{Q}{4\pi\xi_0\xi_a R} + \varphi_0 + E_0(R - \frac{a^3}{R^2})\cos\theta.$$

Так как потенциал зависит только от R и θ, напряженность электрического поля имеет только две составляющие:

$$E_R = \frac{Q}{4\pi\xi_0\xi_a R^2} - E_0(1 + \frac{2a^3}{R^3})\cos\theta,$$

$$E_\vartheta = E_0(1 - \frac{a^3}{R^3})\sin\theta,$$

где ξ_0, ξ_a – диэлектрическая постоянная и относительная диэлектрическая проницаемость среды.

Для диэлектрического незаряженного шара потенциал внутренней области φ_i и потенциал внешней области φ_e определяются как

$$\varphi_i = \varphi_0 + E_0 \frac{3\xi_e}{2\xi_e + \xi_i} R\cos\theta,$$

$$\varphi_e = \varphi_0 + E_0(R + \frac{a^3}{R^2}\frac{\xi_e - \xi_i}{2\xi_e + \xi_i})\cos\theta,$$

где ξ_e, ξ_i – абсолютные диэлектрические проницаемости среды и шара.

Принципиальный алгоритм решения задачи.

1. Задача решается в полярных координатах.
2. Рассматриваются две области – внутри и вне шара.
3. Для каждой области в отдельности:

а) создается двумерная сетка в полярных координатах в диапазонах [0÷a] по радиусу и [0÷2π] по азимуту для внутренней области шара и диапазонах [a÷3a] по радиусу и [0÷2π] по азимуту для внешней

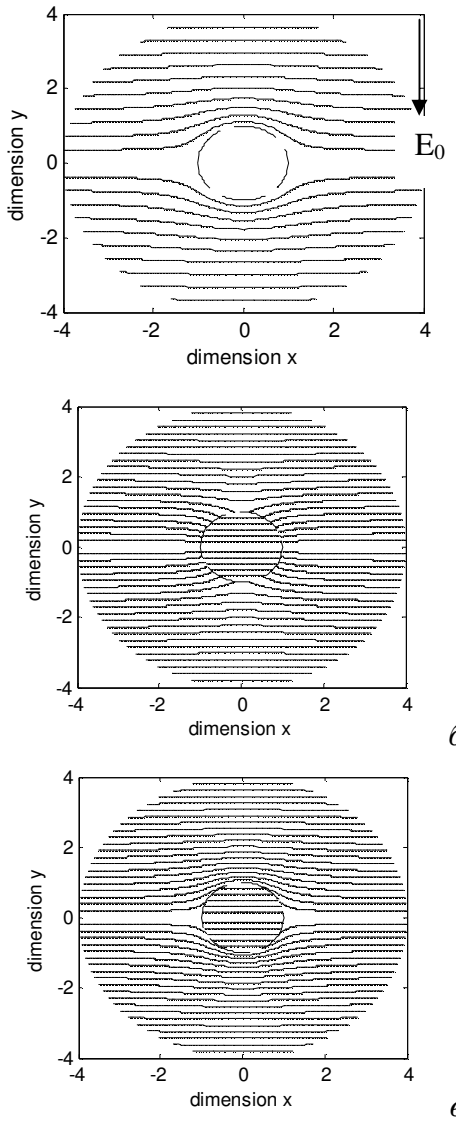


Рис. 2.31. Эквипотенциали электрического поля:
(*а* – проводящий шар, *б, в* – диэлектрический шар).

области. Шаг сетки подбирается таким образом, чтобы линий уровня выглядели достаточно гладкими. (Функция *meshgrid*);
b) сетки из полярных координат преобразуется в сетки декартовых координат. Графически это выражается в том, что область построения графиков оторажается в виде круга;
c) вычисляется массивы значений потенциала поля в узлах сетки для обеих областей и строятся эквипотенциали. (Функция *pol2cart*).
4. Эквипотенциали обеих областей совмещаются на одном графике.
5. На графике также прорисовывается граница шара (окружность).

В случае проводящего шара внутренняя область не рассматривается так как электрическое поле внутри шара отсутствует.

Эквипотенциали являются непрерывными линиями, следовательно, нужно обеспечит их «сшивку» на границе раздела областей. Технически это можно осуществить подбором количества линий уровня во внешней и внутренней областях. В случае диэлектрического шара нужно рассмотреть варианты, когда диэлектрическая постоянная среды больше диэлектрической постоянной шара и наоборот. Важно отметить качественное различие этих двух вариантов (рис. 2.31). Если значение диэлектрической постоянной внешней области больше значения диэлектрической постоянной внутренней области, то эквипотенциали в приграничной области (с внешней стороны) имеют тенденцию «загибаться» к поверхности шара. В противном случае наблюдается обратная тенденция. На рис. 2.31 отражены оба варианта сочетаний диэлектрических постоянных, а также картина поля для проводящего шара.

Следует также отметить, что нас в данной постановке задачи интересует только конфигурация полей, а не их численные значения, поэтому величины заряда на проводящем шаре, напряженности внешнего постоянного электрического поля, а также внешнего потенциала не имеют принципиального значения.

Контрольные вопросы к главе 2

1. Напишите уравнение колебаний гармонического осциллятора.
2. Как выражается потенциал и напряженность электрического поля системы статических зарядов?

3. Напишите уравнение движения заряженной частицы в электрическом и магнитном полях.

4. Как отображаются параметры пучка заряженных частиц в фазовом пространстве?

5. Магнитная квадрупольная линза. Как осуществляется фокусировка и дефокусировка заряженных частиц? Матрицы передачи квадрупольной линзы.

6. Дипольный магнит. Как осуществляется поворот заряженных частиц? Матрицы передачи дипольного магнита.

7. Соленоид. Как осуществляется фокусировка заряженных частиц? Матрицы передачи соленоида.

8. В чем заключается матричный способ описания движения пучка заряженных частиц в канале транспортировки?

9. Как, используя матричный аппарат описания движения пучка заряженных частиц, можно выбрать необходимые параметры канала транспортировки?

10. Объясните характер движения заряженной частицы в постоянном во времени магнитном и электрическом поле.

11. Одиночная диафрагма. Объясните конфигурацию электрического поля и характер движения электрона.

12. Прямоугольный волновод. Объясните конфигурацию электрического и магнитного поля в волноводе для различных мод.

13. Как влияет проводящий или диэлектрический шар на изменение конфигурации внешнего электрического поля?

ПРИЛОЖЕНИЕ

Символы, операторы и функции системы MATLAB

В записи любого выражения используются специальные символы и операторы. Приведем основные значения ряда наиболее употребительных символов и операторов.

[ ] – квадратные скобки используются для записи векторов и матриц,

( ) – круглые скобки используются для традиционного построения вложенных арифметических конструкций. В круглые скобки заключены аргументы функций. При записи элементов массива его индексы также заключаются в круглые скобки,

= – оператор присвоения переменной некого значения,

== – оператор отношения (равенства),

‘ – верхний кавычка означает операцию транспонирования матриц и векторов,

. – десятичная точка в записи вещественных чисел,

*** , ^, /, -, +** – операции умножения, взведения в степень, деления, сложения и вычитания,

.* , .^, ./, or **.\.** - операции умножения, взведения в степень, правого и левого деления,

, – запятая разделяет индексы элементов матрицы и аргументы функции,

; – точка с запятой внутри квадратных скобок означает конец строки матрицы. Этим символом можно отделять один оператор программы от другого, что позволяет записать в одной строке программы несколько операторов. Кроме того, запись «**;**» после любого выполняемого оператора программы означает «подавление» вывода на экран результатов выполнения этого оператора (только результатов, а не самого действия),

: – двоеточие используется в записи векторов,

% – процент. Запись в строке после символа % является комментарием.

Операторы отношений и логические операторы

- == – тождественно
- ~= – не тождественно
- < – меньше
- > – больше
- <= – меньше или равно
- >= – больше или равно
- & – логическое «И»
- | – логическое «ИЛИ»
- ~ – логическое «НЕТ»

Команды общего назначения

help – помощь
clear – очистка командного окна, рабочей области, буфера команд
path – путь доступа

Операторы организации программ

(Организация циклов и уловные операторы)

if – условный оператор
else – оператор в структуре if
elseif – оператор в структуре if
for – цикл
while – цикл с предусловием
break – оператор прерывания в структуре цикла
switch – оператор в структуре цикла
case – оператор переключений
otherwise – оператор в структуре выбора
return – возвращение в вызывающую программу
end – конец цикла, условного оператора и т.п.

Операторы вычисления

feval – вычисление функции по её имени

Массивы и матрицы специального вида

zeros–- формирование массива нулей
ones – формирование массива единиц
eve – формирование единичной матрицы
rand – формирование массива случайных чисел (равномерное распределение)

Анализ и обработка данных

sum – возвращает сумму элементов каждого столбца массива;
cutsum–- возвращает промежуточные результаты суммирования элементов каждого столбца массива;
sort – упорядочивает элементы каждого столбца массива по возрастанию;
max – для одномерного массива возвращает наибольший элемент; для двумерного массива возвращает вектор-строку, состоящую из наибольших элементов каждого столбца;
min – для одномерного массива возвращает наименьший элемент; для двумерного массива возвращает вектор-строку, состоящую из наименьших элементов каждого столбца.

Тригонометрические функции

sin – синус
cos – косинус
tan – тангенс
cot – котангенс
asin – арксинус
acos – арккосинус
atan – арктангенс
sinh – гиперболический синус

cosh – гиперболический косинус
tanh – гиперболический тангенс
coth – гиперболический котангенс
asinh – гиперболический арксинус
acosh – гиперболический арккосинус
atanh – гиперболический арктангенс

Экспонента и логарифмы

exp – экспонента
log – натуральный логарифм
log10 – логарифм по основанию 10
log2 – логарифм по основанию 2
pow2 – возведение в квадрат
sqrt – квадратный корень

Базовые операции над матрицами

size – размер массива
length – длина вектора
ndims – количество размерностей массива
numel – число элементов в массиве
disp – вывод значений переменных и текста на экран

Преобразование матриц

cat – объединение массивов
reshape – преобразование размеров многомерного массива
diag – формирование или извлечение диагоналей матрицы
rot90 – поворот матрицы на 90 градусов против часовой стрелки

Операции над строками

strcat - горизонтальное объединение строк
strvcat – вертикальное объединение строк

Некоторые значения (по умолчанию)

pi – 3.1415926535897....,
i, j – мнимая единица,
inf – бесконечность,
NaN – нечисловое значение.

Функции округления

fix – усечение дробной части числа,
floor – округление до меньшего целого,
ceil – округление до большего целого,
round – округление до ближайшего целого,
mod – остаток от деления с учетом знака,
sign – знак числа.

Решение систем линейных алгебраических уравнений

cholinc – решение СЛАУ методом Холецкого.

Численное интегрирование и дифференцирование

quad, quad8 – численное интегрирование.

Интерполяция табличных данных

interp1 – одномерная интерполяция,
spline – интерполяция сплайнами.

Вычисление минимумов функций

fminibnd – вычисление минимума функции одной переменной,
fminsearch (fmins) – вычисление минимума функции нескольких переменных.

Вычисление конечных разностей

diff – вычисление конечных разностей,
gradient – вычисление градиента функции.

Решение систем обыкновенных дифференциальных уравнений

ode23, ode45 – решение ОДУ методом Рунге-Кутта.

Вычисление градиента функции

gradient – приближенное вычисление градиента функции на двумерной сетке

Интерпретация текстовых строк

eval – текст, содержащийся в символьной переменной, воспринимается как команда, оператор, или часть выражения

Вывод графической информации

В состав системы MATLAB входит мощная графическая подсистема, которая поддерживает средства двумерной и трехмерной графики, а также средства презентационной графики. Элементарные графические функции системы MATLAB позволяют представить результаты расчетов в виде графиков (семейства графиков), а также отобразить картину распределения электромагнитных полей на плоскости. Графики сопровождаются соответствующими подписями и оцифровкой координат.

Двумерная графика

hold on, hold off – функция управления режимом графического окна.

Задание осей координат, надписи и пояснения

subplot – разбиение графического окна на несколько окон,
polar – график в полярных координатах *t,*
plot – график в линейных координатах *t,*
comet – движение отображающей точки по траектории,
title – заголовок графика,
xlabelX – обозначение оси *x,*
ylabelY – обозначение оси *y,*
legend – пояснение к графику,
grid – координатная сетка,
meshgrig – формирование двумерных массивов *X* и *Y* координатной сетки,
contour – изображение линий уровня для трехмерной поверхности,
quiver – график поля направлений.

СПИСОК ЛИТЕРАТУРЫ

1. Савельев И.В. Курс общей физики, книга 1. Механика. М., 2003.
2. Ливингуд Дж. Принципы работы циклических ускорителей. М.: ИЛ, 1963.
3. Савельев И.В. Курс общей физики, книга 2. Электричество и магнетизм. М., 2003.
4. Смайт В. Электростатика и электродинамика. М., ИЛ, 1954.
5. Фельштейн А.П., Явич Л.Р., Смирнов В.П. Справочник по элементам волноводной технике. М.: Госэнергиздат, 1963.
6. Штеффен К. Оптика пучков высоких энергий. М.: Мир, 1969.
7. Хокс П., Каспер Э. Основы электронной оптики. М., Мир: 1993.
8. Ландау Л.Д., Лившиц Е.М. Теория поля. М.: Наука, 1967.
9. Бенфорд А. Транспортировка пучков заряженных частиц. М.: Атомиздат, 1969.
10. Бессонов Л.А. Теоретические основы электротехники. Электромагнитное поле. М.: Высшая школа, 1978.
11. Потемкин В.Г. Систем инженерных и научных расчетов MATLAB., т.т.1, 2. М.: Диалог МИФИ, 1999.

Аверьянов Герман Петрович

Будкин Валерий Андреевич

Дмитриева Валентина Викторовна

АВТОМАТИЗАЦИЯ ПРОЕКТИРОВАНИЯ ЧСАСТЬ 1. РЕШЕНИЕ ЗАДАЧ ЭЛЕКТРОФИЗИКИ В СИСТЕМЕ MATLAB

Компьютерный практикум

Редактор Шумакова Н.В.

Оригинал-макет изготовлен

Подписано в печать. Формат 60x84 1/16
Уч.-изд.л. 7,0. Печ.л. 7,0. Тираж 200 экз.
Изд. № 062-1. Заказ №

Московский инженерно-физический институт
(государственный университет).
115409, Москва, Каширское ш., 31

Типография издательства «Тровант».
г. Троицк Московской обл.

112