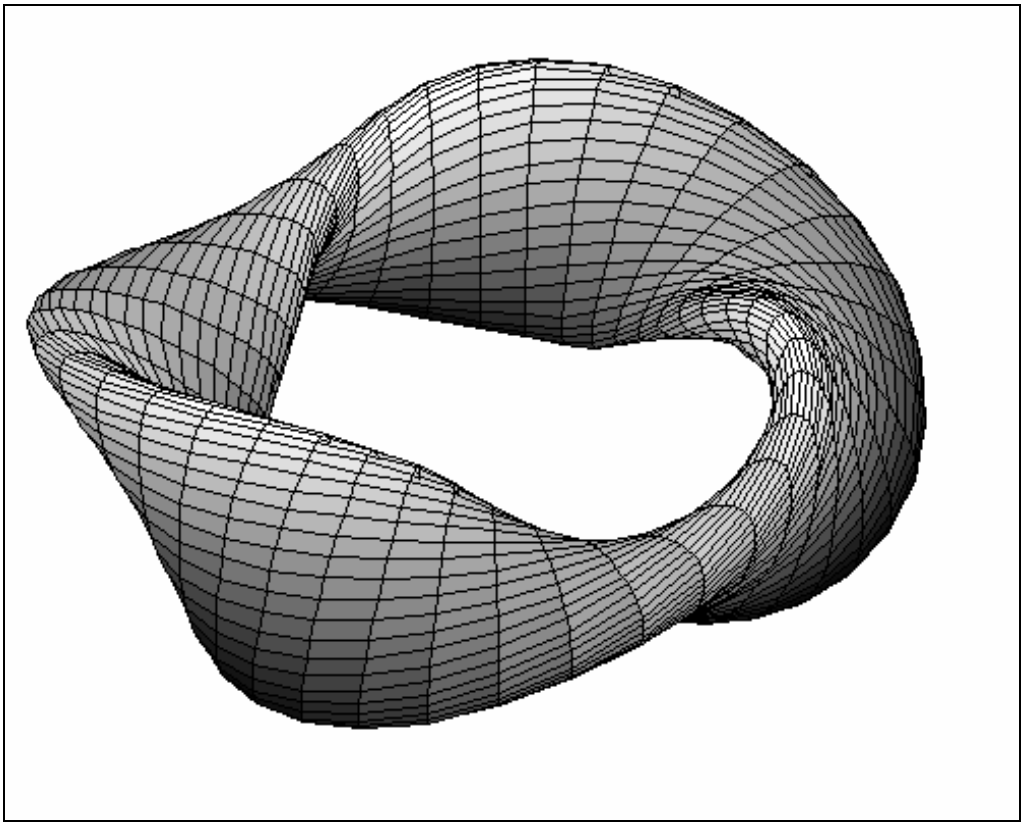
Томский Государственный Университет Систем Управления и Радиоэлектроники (ТУСУР)

Кафедра Компьютерных систем в управлении и проектировании

Черкашин М.В., Бабак Л.И.

Вычислительная математика

Методические указания для студентов специальности 230104 – «Системы автоматизированного проектирования»



ТОМСК – 2007

Черкашин М.В., Бабак Л.И.
Вычислительная математика: учеб.-методич. пособие / М.В. Черкашин, Л.И.Бабак – Томск: Томск. гос. ун-т систем упр. и радиоэлектроники, 2007. – 183 с.

В пособии представлены методические указания по выполнению лабораторных и практических работ, а также курсового проекта по дисциплине «Вычислительная математика в САПР». Кроме того, пособие включает в себя краткое описание системы для инженерных и научных расчетов MATLAB, на которой выполняются лабораторные работы.

Пособие предназначено для студентов высших технических учебных заведений, обучающихся по специальности 230104 – «Системы автоматизированного проектирования».

Содержание

1 Цель и задачи дисциплины, ее место в учебном процессе

1.1 Цель преподавания дисциплины

Цель курса «Вычислительная математика» состоит в изучении общих принципов проведения вычислительного эксперимента, методов и алгоритмов решения стандартных задач вычислительной математики, современных программных средств для автоматизации вычислений.

1.2 Задачи изучения дисциплины

В результате изучения студенты должны:

- знать: принципы проведения вычислительного эксперимента, характеристики вычислительных задач, источники погрешностей вычислений, основные методы и алгоритмы решения стандартных вычислительных задач;
- уметь: выбирать и разрабатывать численные алгоритмы решения вычислительных задач; разрабатывать программы для решения таких задач;
- иметь навыки: решения вычислительных задач с помощью современных математических пакетов.

1.3 Перечень дисциплин, усвоение которых необходимо для изучения данной дисциплины

- математика;
- дискретная математика;
- теория вероятности и математическая статистика;
- информатика;
- алгоритмические языки и программирование;
- объектно-ориентированное программирование;
- программирование под Windows.

2 Содержание дисциплины

2.1 Введение в численные методы

Предмет и история развития вычислительной математики. Этапы решения задачи на ЭВМ. Вычислительный эксперимент. Погрешности вычислительного эксперимента.

Характеристики вычислительных задач. Устойчивые и неустойчивые, корректные и некорректные задачи. Примеры некорректных задач. Требования к вычислительным методам. Устойчивость, корректность, сходимость. Пример неустойчивого алгоритма.

2.2 Погрешности округления в ЭВМ

Представление чисел в ЭВМ. Машинный нуль и машинная бесконечность. Абсолютная и относительная погрешности. Округление чисел в ЭВМ. Машинный эпсилон. Накопление ошибок округления. Классическая формула для погрешности суммы, разности, произведения и частного.

Погрешности округления при выполнении арифметических операций в ЭВМ. Погрешности суммы двух и нескольких чисел. Зависимость погрешности от порядка суммирования. Погрешности произведения двух и нескольких чисел. Алгоритм вычисления произведения чисел. Правила выполнения арифметических операций в ЭВМ. Статистические оценки погрешностей. Примеры организации вычислений.

2.3 Вычисление значений функций

Вычисление значений полинома. Схема Горнера. Вычисление элементарных функций в ЭВМ. Способы вычисления. Показательная, логарифмическая, тригонометрическая функции. Вычисление квадратного корня.

2.4 Аппроксимация функций

Понятие приближения функций. Применение аппроксимации функций в САПР. Критерии близости функций. Оптимальная аппроксимация.

Классификация задач аппроксимации.

Интерполяция функций. Задача линейной интерполяции. Линейная полиномиальная интерполяция. Интерполяционный многочлен Лагранжа. Разделенные разности и их свойства. Интерполяционная формула Ньютона.

Свойства интерполяционных моделей. Погрешность интерполяции. Многочлены Чебышева. Оптимальный выбор узлов интерполяции. Сходимость интерполяции. Теорема Фабера. Локальная интерполяция. Применение глобальной и локальной интерполяции.

Интерполяция тригонометрическими полиномами. Дискретное преобразование Фурье. Быстрое преобразование Фурье.

Интерполяция с помощью сплайнов. Понятие сплайна. Построение кубического сплайна. Основные соотношения. Применение сплайнов.

Дискретная среднеквадратичная аппроксимация. Свойство сглаживания. Получение и решение нормальных уравнений. Применение среднеквадратичной аппроксимации.

Наилучшая равномерная аппроксимация. Теорема Чебышева. Теорема Валле-Пусена. Итерационный алгоритм нахождения наилучшего равномерного приближения. Применение наилучшей аппроксимации.

Аппроксимация методом разложения в степенной ряд. Многочлен Тейлора. Погрешность приближения многочленом Тейлора. Сходимость.

Аппроксимация функций нескольких переменных. Построение поверхностей и линий уровня функции двух переменных

2.5 Численное дифференцирование и интегрирование

Прямое вычисление производных. Левая, правая и центральная разностные производные. Ошибки численного дифференцирования. Применение интерполяции.

Численное интегрирование. Формулы прямоугольников, трапеций, Симпсона. Ошибки численного интегрирования. Выбор шага интегрирования.

2.6 Моделирование случайных величин

Основные характеристики случайных величин. Получение случайных величин на ЭВМ. Генераторы случайных чисел. Метод Монте-Карло. Применение метода Монте-Карло для вычисления определенных интегралов.

2.7 Методы решения систем линейных алгебраических уравнений

Классификация и характеристики методов решения систем линейных алгебраических уравнений (СЛАУ). Прямые и итерационные методы. Методы Крамера, обратной матрицы. Метод Гаусса (схема единственного деления). Метод Гаусса с выбором главного элемента. Вычисление определителя и обратной матрицы методом Гаусса. Метод прогонки.

Погрешности решения СЛАУ. Нормы векторов и матриц. Оценка погрешностей. Число обусловленности. Оценка числа обусловленности.

Итерационные методы решения СЛАУ. Метод простых итераций (метод Якоби). Условия сходимости. Оценка числа итераций. Метод Зейделя.

2.8 Решение нелинейных уравнений с одним неизвестным

Прямые и итерационные методы решения. Число корней нелинейных уравнений. Отделение корней.

Методы уточнения корней. Метод дихотомии. Метод хорд. Метод Ньютона (касательных). Условия и скорость сходимости метода Ньютона. Модифицированный метод Ньютона и метод секущих. Глобально сходящийся метод.

Последовательный поиск корней алгебраического уравнения. Нахождение комплексных корней. Области притяжения корней.

2.9 Решение систем нелинейных уравнений

Существование, число и характер решений систем нелинейных уравнений (СНУ). Ряд Тейлора для функций многих переменных. Метод простой итерации. Метод Ньютона. Условия сходимости метода Ньютона.

Модифицированный метод Ньютона. Глобально сходящиеся модификации метода Ньютона. Метод продолжения по параметру.

2.10 Программные средства для автоматизации вычислений

Библиотеки подпрограмм для решения вычислительных задач. Универсальные системы для автоматизации математических и инженерных расчетов MathCAD, MATLAB, MAPLEV Организация систем, основные функции. Примеры решения вычислительных задач с использованием универсальных математических систем.

2.11 Учебно-методические материалы

Основная литература

1. Л.И. Турчак. Основы численных методов. – М., Наука, 1987. – 320с.
2. А.А. Самарский, А.В. Гулин. Численные методы. –М., Наука, 1989. – 432с.
3. А.А. Самарский. Введение в численные методы. –М., Наука, 1987. – 288с.
4. Ю.П. Боглаев. Вычислительная математика и программирование. –М., Высшая школа, 1990. – 544с.
5. Н.Н. Калиткин. Численные методы. –М.., Наука, 1978. – 512с.
6. Л.И. Бабак. Вычислительные методы. Курс лекций (части I и II). Томск, ТУСУР, 2002.
7. М.В. Черкашин. Вычислительные методы. Курс лекций (часть III). Томск, ТУСУР, 2002.

Дополнительная литература

8. А.Н. Тихонов, Д.П. Костомаров. Вводные лекции по прикладной математике. –М., Наука, 1984.
9. Н.С. Бахвалов, Н.П. Жидков, Г.М. Кобельков. Численные методы. –М., Наука, 1987. – 600с.
10. Е.А. Волков. Численные методы. –М., Наука, 1987. – 248с.
11. Д. Мак-Кракен, У. Дорн. Численные методы и программирование на Фортране. –М., Мир, 1977.

12. Д. Каханер, К. Моулер, С. Нэш. Численные методы и программное обеспечение. –М., Мир, 1988. –575с.
13. А.Е. Мудров. Численные методы для ПЭВМ на языках Бейсик, Фортран и Паскаль. – Томск, Packo, 1991. –324с.
14. В.В. Носач. Решение задач аппроксимации с помощью персональных компьютеров. –М., БИНОМ, 1994. –384с.
15. И.Н. Молчанов. Машинные методы решения прикладных задач. Алгебра приближенных функций. –Киев, Наукова думка, 1987.
16. В.И. Барабащук, Б.П. Креденцер, В.И. Мирошниченко. Планирование эксперимента в технике. –Киев, Техника, 1984. –200с.

Методические указания и прочие учебно-методические материалы

17. В.Г. Потемкин. Система MATLAB. Справочное пособие –М., Диалог – МИФИ, 1997. –352с.
18. В.Г. Потемкин. MATLAB 5 для студентов. –М., Диалог –МИФИ, 1998. – 314с.
19. М.В. Черкашин, Система для математических и инженерных расчетов MATLAB: учебн. пособие, Томск, 2001.
20. В.И. Ракитин, В.Е. Первушин Практическое руководство по методам вычислений с приложением программ для персональных компьютеров: учебн. пособие, -М.: Высшая шкода, 1998, - 383 с.

3 Система для математических и инженерных расчетов MATLAB

Далее приводится краткое описание известного пакета для математических и инженерных расчетов MATLAB. Цель – дать студентам навыки работы в этой системе, изучить основные приемы программирования на языке MATLAB. Кроме этого в данном руководстве содержится небольшой справочник по встроенным функциям системы.

3.1 Основные сведения о системе MATLAB

Система MATLAB (сокращение от MATrix LABoratory – МАТричная ЛАБоратория) разработана фирмой The MathWorks, Inc. (США, г. Нейтик, штат Массачусетс) и является интерактивной системой для выполнения инженерных и научных расчетов. Система MATLAB распространяется более 20 лет, она постоянно развивается: последняя версия – MATLAB 2007 (MATLAB 7.2b) выпущена весной 2007 года.

MATLAB – это высокоэффективная среда для математических, инженерных, научных вычислений, ориентированная на работу с массивами данных (матрицами), чем и обязана своему названию. Система MATLAB может выполнять операции в режиме прямых вычислений (режим интерпретатора). Это позволяет использовать ее как мощный калькулятор, в котором наряду с обычными арифметическими и алгебраическими действиями, могут использоваться и операции матричной алгебры, т.е. операции по обращению матриц, нахождению собственных чисел и векторов, решению систем линейных уравнений и многое др.

Наиболее известные области применения системы MATLAB:

– математика и вычислительные методы;
– разработка алгоритмов;
– вычислительный эксперимент, математическое и имитационное моделирование, макетирование;
– анализ данных, визуализация и обработка данных;

– научная и инженерная графика;

– разработка приложений, связанных с вычислительными алгоритмами и визуализацией данных, включая разработку пользовательского графического интерфейса (GUI).

Система MATLAB – это одновременно и операционная среда, и язык программирования. Ее можно использовать как интерпретатор, в режиме непосредственных вычислений, и как компилятор – для написания отдельных программ. Но главное достоинство среды MATLAB – это легкость ее модификации и адаптации к самым различным задачам, требующим математических вычислений, обработки данных и работы с графическими средствами. Поэтому ее с равным успехом можно применять для расчетов в математике, физике, биологии, электро- и радиотехнике, для статистических исследований и др.

3.2 Установка и состав системы MATLAB

Рассмотрим версию системы MATLAB 6.5, как наиболее удобную для использования в учебном процессе. Она обладает мощными средствами по обработке и визуализации данных, содержит большое число прикладных программ (Toolbox-ов), для реализации самых разнообразных вычислительных алгоритмов. При этом она занимает не слишком много места на жестком диске (около 600 МБайт) и работает под управлением MS Windows 98\NT\2к\XP и выше. Более поздние версии системы (MATLAB 7.x), хотя и обладают гораздо лучшими показателями, как со стороны вычислительных методов, так и по возможностям пользователя при создании собственных программ, но в тоже время требуют серьезных ресурсов от ПК и операционной системы (MS Windows XP и выше).

Установка системы на жесткий диск не представляет трудностей для пользователя, знакомого с Windows. После запуска программы **setup.exe** из состава дистрибутива (он занимает один компакт-диск), будет выведена заставка программы и диалоговое окно для ввода данных для начала установки. Далее следует выбрать логический диск и имя корневого каталога для системы

MATLAB (по умолчанию система будет установлена в каталог **c:\MATLAB**). Также перед началом установки следует указать наименования дополнительных пакетов прикладных программ (ППП), которые Вы хотите установить вместе с системой MATLAB – например, SIMULINK, Control Toolbox, Spline Tollbox и др. При необходимости ППП для MATLAB могут быть подключены к системе позже. Также следует указать, где будет размещаться расширенная система помощи – на компакт-диске или на винчестере. В первом случае – потребуется наличие CD в приводе, во втором – дополнительный объем на винчестере для копирования файлов помощи. В ходе инсталляции будет создана группа (в главном меню Windows доступном после нажатия на кнопку **ПУСК**) с ярлыками для запуска системы MATLAB и вызова системы помощи.

После установки на жесткий диск система MATLAB имеет следующую структуру: в каталоге **…\MATLAB\BIN** содержатся основные системные файлы; каталог **…\MATLAB\TOOLBOX** содержит встроенные функции системы (m-файлы) и установленные ППП; каталог **…\MATLAB\EXTERN** содержит исходные файлы и библиотеки на языках программирования C и FORTAN для создания MEX-файлов и DLL-файлов; каталог …**MATLAB\HELP** содержит контекстную систему помощи, выполненную на основе XML-технологии; в каталогах **…\MATLAB\JA** и **…\MATLAB\JAVA** находятся вспомогательные модули, выполненные на языке программирования JAVA; каталог **…\MATLAB\SYS** включает в себя графический постпроцессор GhostScript и компиляторы языков C++, JAVA, Perl. В зависимости от выбранных ППП, подключаемых к системе MATLAB, возможно наличие каталогов с файлами этих пакетов, например **…\MATLAB\SIMULINK, …\MATLAB\NOTEBOOK** и пр.

В каталоге **c:\MATLAB\TOOLBAX\LOCAL** должны быть два файла с системными установками: **matlabrc.m** – содержит основные установки для командного окна системы (MATLAB Command Window), которое является корневым объектом системы и **printopt.m** – содержит установки для текущего

принтера. Кроме этого пользователь может создать файл **startup.m** с индивидуальными установками, который будет автоматически выполняться системой при каждой загрузке MATLAB (аналог файла autoexec.bat в системе MS-DOS).

3.3 Запуск системы, основные справочные и управляющие команды

После запуска MATLAB на экране ПК появляется главное окно системы – MATLAB Command Window (см. рис. 3.1), которое является корневым (главным) объектом системы.

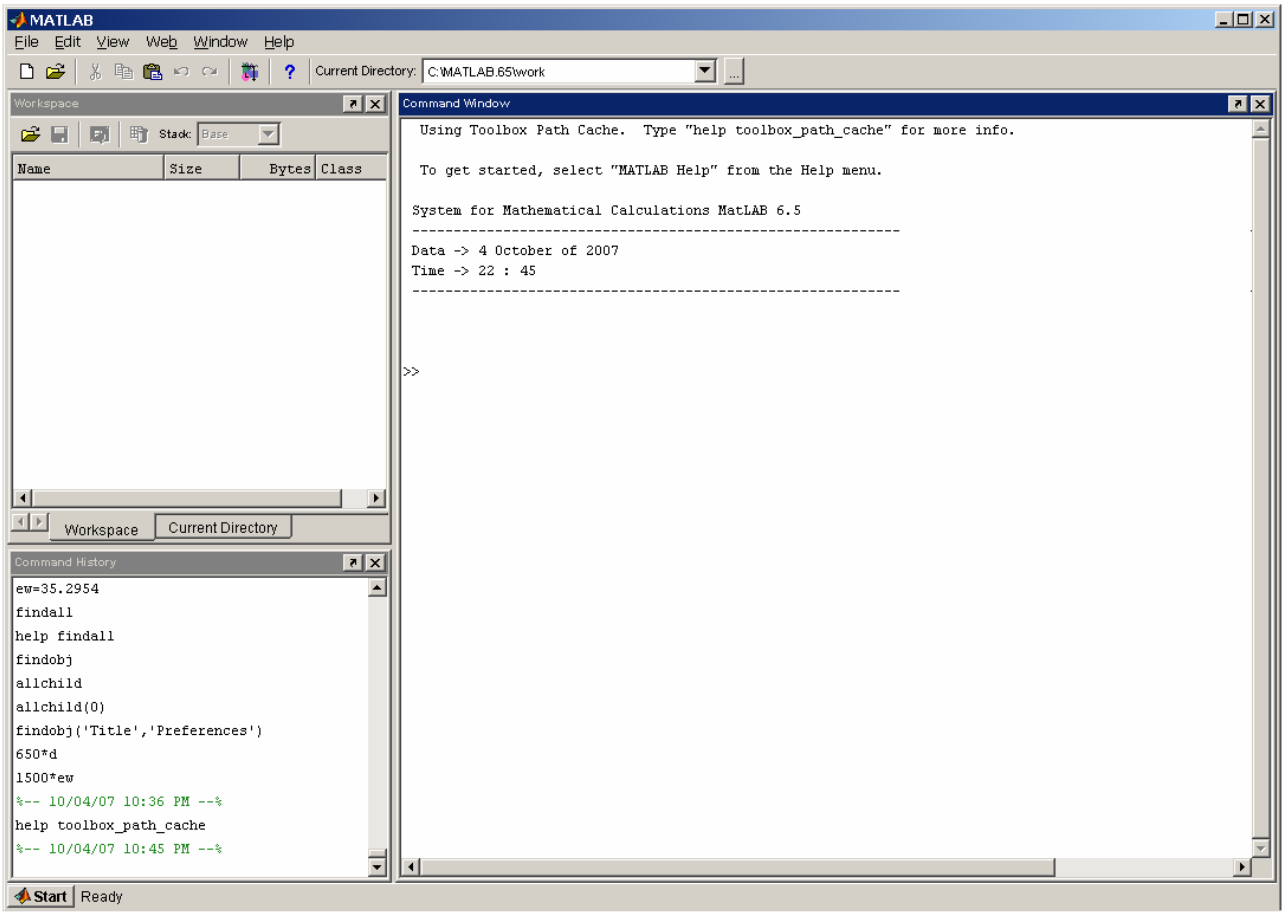


Рисунок 3.1 – Общий вид окна системы MATLAB

В командной строке главного окна можно непосредственно вводить команды, выполнять вычисления или вызывать программы (script-файлы), написанные на языке системы MATLAB.

При загрузке системы выводится подсказка вида

```
Using Toolbox Path Cache.Type "help toolbox_path_cache"
for more info.
```

`To get started, select "MATLAB Help" from the Help menu.`

которая содержит команды для первого знакомства с MATLAB. А также команды, указанные в файлах **matlabrc.m** и **startup.m.** На рис.1 видно, что дополнительно выводится дата и время запуска системы.

Доступны также другие команды: `demo` – более полная демонстрация возможностей системы и подключенных прикладных пакетов с использованием средств XML-интерфейса; `help` – вызов встроенной справочной системы MATLAB по функциям и командам; `whatsnew, info` –предназначены для вывода более полной информации о текущей версии системы MATLAB и фирме MathWorks, Inc.

Вид окна, которое появляется после вызова команды `demo`, показан на рис. 3.2. Далее можно посмотреть примеры работы с самой системой MATLAB и прикладными пакетами, входящих в ее состав (если они установлены).

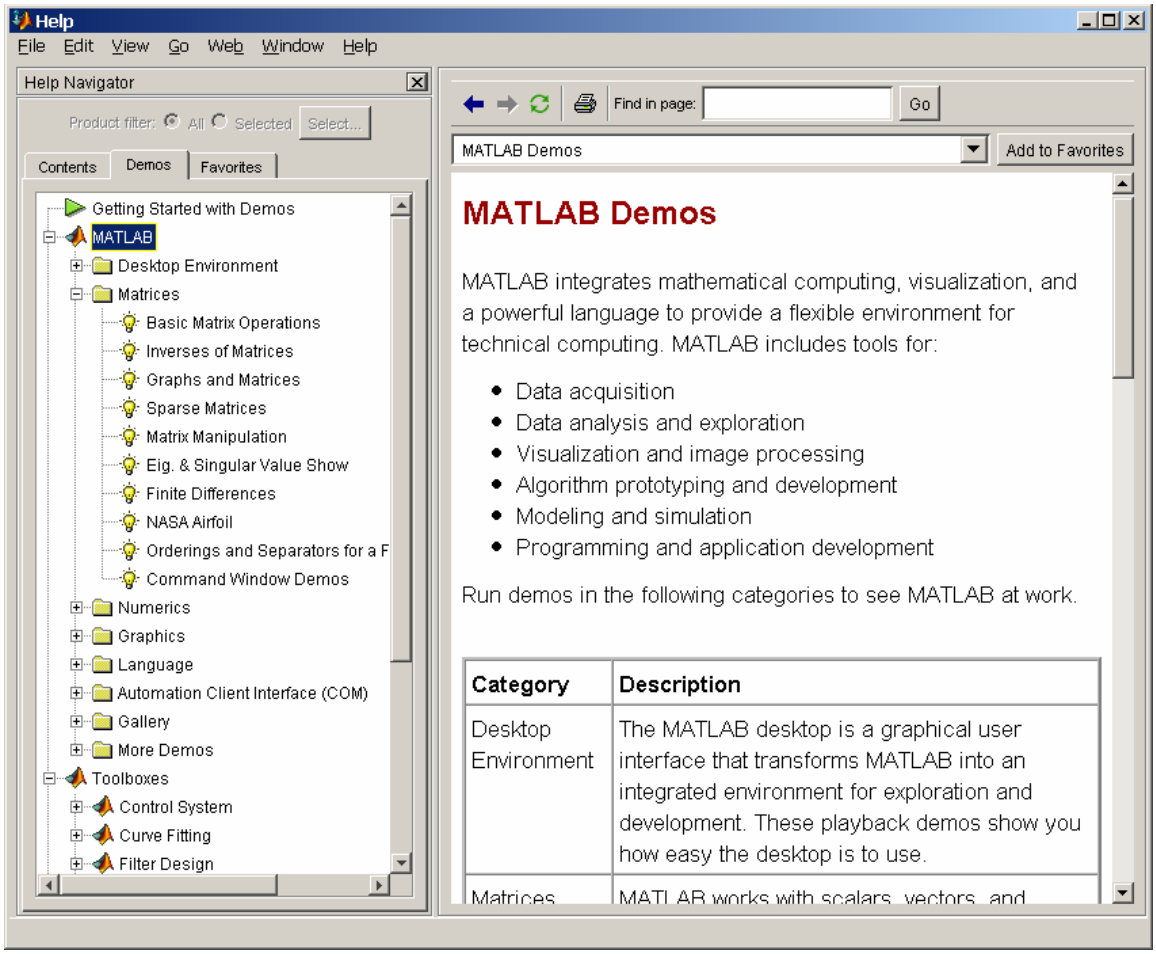


Рисунок 3.2 – Вид окна для демонстрации возможностей системы MATLAB и прикладных программ

В число демонстрационных примеров входят векторные и матричные операции, построение различных графиков (в том числе трехмерных фигур), реализация численных методов, спектральный анализ, расчет фильтров и т.д. Студентам рекомендуется перед началом работы с MATLAB внимательно просмотреть все примеры. Они прекрасно иллюстрируют возможности и разнообразие применений системы, ее высокую скорость вычислений.

Далее рассмотрим некоторые наиболее часто используемые управляющие команды и функции языка MATLAB.

`ver` **Используемая версия системы MATLAB и ППП**

Синтаксис: `ver`

`ver` < *имя ППП* >

Команда `ver` выводит на экран номера текущей версии системы MATLAB и подключенных ППП.

Команда `ver` < *имя ППП* > выводит на экран информацию о текущей версии запрошенного ППП.

См. также: `version, readme, help, whatsnew, info`

`version` **Используемая версия системы MATLAB**

Синтаксис: `version`

Команда `version` возвращает строку с указанием текущей версии системы MATLAB.

См. также: `ver, readme, help, whatsnew, info`

`help` **Справка о командах и функциях системы MATLAB**

Синтаксис: `help`

`help` < *раздел/функция* >

`help` < *спец. символ* >

Команда help без параметров выводит на экран список разделов системы MATLAB. Каждому разделу соответствует имя каталога в списке путей доступа MATLABPATH (команда path).

Команда help < *раздел/функция* > выводит перечень функций или описание самой функции. При вызове команды help следует указывать лишь имя раздела или функции.

Команда help < *спец. символ* > выводит список специальных символов системы.

См. также: lookfor, what, which, dir, pwd

path **Управление списком путей доступа системы MATLAB**

Синтаксис: path

```
p=path
path(p)
path(p1,p2)
```

Команда path выводит на экран список путей доступа в системе MATLAB. Этот список соответствует строковой переменной MATLABPATH, которая устанавливается в файлах matlabrc.m или/и startup.m.

Команда p=path возвращает строку, содержащую список путей доступа.

Команда path(p) заменяет текущий список (переменную MATLABPATH) списком p.

Команда path(p1,p2) объединяет списки p1 и p2 в один и заменяет им текущий список путей доступа.

См. также: what, dir, cd, pwd

quit **Завершение работы и выход из системы**

Синтаксис: quit

Команда quit завершает работу системы MATLAB и закрывает командное окно.

При завершении работы выводится информация о том, какое число операций с плавающей запятой было выполнено за сеанс работы. При выходе из системы все текущие переменные, находящиеся в рабочей памяти, стираются. При необходимости рабочую область памяти можно сохранить командой `save`.

`who, whos` **Вывод списка текущих переменных**

Синтаксис: `who`

`whos`

Команда `who` выводит список переменных текущей рабочей области памяти.

Команда `whos` выводит подробную информацию относительно текущих переменных, включая имя, размер и число элементов используемых массивов, длину в байтах, тип матрицы (плотная/разряженная, комплексная/действительная).

`clear` **Очистка рабочей области памяти**

Синтаксис: `clear`

`clear all`

`clear` < *список имен переменных/функций* >

`clear global`

`clear global` < *имя глобальной переменной* >

`clear functions`

Команда `clear` удаляет все переменные из рабочей области памяти.

Команда `clear all` удаляет все переменные, функции и ссылки на MEX-файлы из рабочей области памяти.

Команда `clear x1, x2` удаляет переменные или функции с именами `x1` и `x2` из текущей рабочей области памяти. Если `x1` глобальная переменная, то команда `clear x1` удаляет ее из текущей рабочей области памяти, но

оставляет ее доступной для функций, где эта переменная объявлена глобальной.

Команда `clear global` удаляет все глобальные переменные из рабочей области памяти.

Команда `clear global X` удаляет глобальную переменную X из рабочей области памяти.

Команда `clear functions` удаляет все используемые М-функции из рабочей области памяти.

`type` **Вывод на экран содержимого текстового файла**

Синтаксис: `type` < *имя файла . расширение* >

Команда `type` < *имя файла . расширение* > выводит на экран командного окна содержимое текстового файла.

Команда `type` < *имя файла* > выводит на экран содержимое М-файла (с расширением *.m).

`disp` **Вывод на экран переменных и текста**

Синтаксис: `disp` (< *переменная* > / '< *текст* >')

Команда `disp(X)` выводит на экран командного окна содержимое переменной X без указания ее имени.

Команда `disp`('< *текст* >') выводит на экран символьную строку '< *текст* >'.

После каждой команды `disp` происходит переход на новую строку.

`matlabroot` **Корневой каталог системы MATLAB**

Синтаксис: `p=matlabroot`

Команда `matlabroot` возвращает имя корневого каталога системы MATLAB.

См. также: `cd, dir, path, pwd`

`pwd` **Текущий каталог системы MATLAB**

Синтаксис: `p = pwd`

Команда `pwd` возвращает имя текущего каталога при работе в системе MATLAB.

См. также: `cd, dir, path, matlabroot`

`cd` **Просмотр и смена текущего каталога**

Синтаксис: `cd`

`cd ..`

`cd` < *имя нового каталога* >

Команда `cd` выводит на экран путь доступа к текущему каталогу.

Команда `cd ..` переход на единицу вверх по дереву каталогов.

Команда `cd` < *имя нового каталога* > изменяет текущий каталог на новый, определенный строкой < *имя нового каталога* >.

См. также: `matlabroot, dir, path, pwd, what`

`dir` **Просмотр содержимого каталога**

Синтаксис: `dir`

`dir` < *имя каталога* >

Команда `dir` выводит листинг текущего каталога.

Команда `dir` < *имя каталога* > выводит список файлов указанного каталога. Можно указывать тип файлов или путь доступа к каталогу.

См. также: `matlabroot, cd, path, pwd, what`

`what` **Вывод списка файлов с расширениями M, MAT и MEX**

Синтаксис: `what`

`what` < *имя каталога* >

Команда `what` выводит на экран списки M-, MAT- и MEX-файлов текущего каталога.

Команда `what` < *имя каталога* > выводит списки файлов указанного каталога. Путь доступа к данному каталогу должен быть описан в переменной MATLABPATH. При вызове команды `what` < *имя каталога* > можно не указывать полный путь доступа к каталогу < *имя каталога* >.

См. также: matlabroot, `cd, path, pwd, dir`

`!` **Вызов команды ОС**

Синтаксис: `!` < *команда ОС* >

Команда `!` < *команда ОС* > позволяет выполнять команду ОС из командного окна системы MATLAB.

Пример: Команда `! notepad.exe` запускает программу notepad.exe (стандартный блокнот ОС Windows 95/98) непосредственно из командного окна MATLAB.

`clc` **Очистка командного окна**

Команда `clc` очищает командное окно системы и устанавливает курсор в верхний левый угол.

См. также: `home`

`echo` **Переключение режима вывода на экран содержимого m-файлов**

Синтаксис: `echo`

```
echo on/off
echo on all/off all
```

Команда `echo` управляет режимом вывода на экран (переключает) содержимого файла-сценария при выполнении.

Команда `echo on` включает режим вывода на экран текста script-файла.

Команда `echo on all` включает режим вывода на экран текста script-файла и всех М-функций, вызываемых в этом файле.

Команда `echo off` выключает режим вывода на экран текста script-файла.

Команда `echo off all` включает режим вывода на экран текста m-файла и m-функций.

3.4 Работа в режиме прямых вычислений

Система MATLAB создана таким образом, что любые (подчас весьма сложные вычисления) можно выполнять в режиме прямых вычислений, т.е. без программы. Это превращает MATLAB в необычайно мощный калькулятор, который способен производить не только обычные для калькуляторов вычисления (например, выполнять арифметические операции и вычислять некоторые элементарные функции), но и операции с векторами и матрицами, комплексными числами, с рядами и многочленами. Можно почти мгновенно задать и вывести графики различных функций – от простой синусоиды до сложной трехмерной фигуры.

Работа с системой в режиме прямых вычислений носит диалоговый характер. Пользователь набирает на клавиатуре вычисляемое выражение, редактирует его (если нужно) и завершает ввод нажатием клавиши ENTER. При этом действует простейший строчный редактор. Перечислим его команды:

Комбинация клавиш	Назначение
→	перемещение курсора вправо на один символ
←	перемещение курсора влево на один символ
Ctrl →	перемещение курсора вправо на одно слово
Ctrl←	перемещение курсора влево на одно слово
Home	перемещение курсора в начало строки
End	перемещение курсора в конец строки
↑ и ↓	перелистывание строк вверх или вниз
Del	стирание символа, на котором установлен курсор
Back_Space	стирание символа слева от курсора
Ins	включение/выключение режима вставки

Возможность перемещения курсора по словам в командной строке полезна при редактировании сложных выражений. Для вызова предыдущей команды следует использовать клавиши ↑ или ↓, которые позволяют просматривать историю вводимых команд.

В некоторых случаях вводимое математическое выражение может оказаться настолько длинным, что для него не хватит одной строки - 80 символов. В этом случае часть выражения можно перенести на новую строку с помощью знака многоточия … (три или более точек).

Например: `s=1-1/2*1/3-1/4* 1/5 - 1/6 * 1/7 ...`
`- 1/8 * 1/9 - 1/10 * 1/11 - 1/12;`

Иногда в ходе вывода данных вычислений появляется сокращение NaN (от слов Not a Number – не число). Оно обозначает операцию неопределенности, например вида 0/0 или Inf/Inf, где Inf – системная переменная со значением машинной бесконечности. Могут появляться и различные сообщения об ошибках (на английском языке). Например, при делении на 0 конечного числа появляется сообщение `Warning: Devide by Zero`. (Предупреждение: Деление на ноль). Весь диапазон представления чисел в системе лежит от 10^{-308} до 10^{308}.

Система MATLAB содержит несколько системных переменных: pi – число $\pi \approx 3{,}1415926\ldots$; inf – значение машинной бесконечности; ans -– переменная, хранящая результат последней операции и обычно вызывающая его отображение на экране дисплея. Эти переменные можно использовать в математических выражениях.

Начнем с простейших вычислений. Для вычисления математических выражений достаточно набрать их по общепринятым правилам (как на Бейсике или Паскале) и завершить ввод нажатием клавиши ENTER. Значение вычисленного выражения будет присвоено переменной ans и выведено на экран дисплея.

Пример:

```
» 2*3-1
ans =
     5

» 2*sin(1)
ans =
    1.6829

» 10*(1-exp(-2))
ans =
    8.6466
```

В системе MATLAB можно задавать переменные. Для этого используется операция присваивания, вводимая знаком равенства (=) в следующем виде:
<Имя_переменной> = <Выражение> [;]

Знак ; (точка с запятой) в конце выражения необязателен, однако играет важную роль. Он определяет открытые и закрытые команды системы. Если знак ; в конце выражения не стоит, то выражение является открытым и система MATLAB после вычисления (после нажатия на клавишу ENTER) присваивает результат вычислений переменной ans и выводит его на экран командного окна. В противном случае, когда мы поставим в конце выражения ; , вывода результата на экран не будет. При написании программ (script-файлов и функций) на языке MATLAB все выражения рекомендуется закрывать ; , чтобы предотвратить лишний вывод информации в командном окне.

Имя переменной может содержать сколько угодно символов, но запоминаются и идентифицируются только 19 первых символов. Имя переменной не должно совпадать с именами функций и процедур системы. Переменные могут быть обычными и индексированными, т.е. элементами векторов или матриц (см. далее). Могут использоваться и символьные переменные (строки), причем символьные значения заключаются в апострофы.

Например: `txt = 'Demo';`

Символьная переменная при этом рассматривается как вектор, состоящий из отдельных символов.

В арифметических выражениях можно использовать следующие знаки арифметических операции: + (сложение); - (вычитание); * (умножение); / (деление слева направо); \ (деление справа налево); ^ (возведение в степень) и др. Полный список операторов можно получить, используя справочную систему MATLAB командой `help ops`.

В выражениях также можно использовать и доступные системе функции (алгебраические, тригонометрические, обратные тригонометрические, гиперболические, обратные гиперболические и др.). Перечень некоторых математических функций представлен в приложении.

Система MATLAB работает как с действительными, так и с комплексными числами вида z = Re(z) + i*Im(z),
где i (или j) - мнимая единица, т.е. квадратный корень из –1, Re(z) - действительная часть комплексного числа z, а Im(z) - его мнимая часть. Тогда, если Re(z) = 2, а Im(z) = 3.5, то число z можно задать в виде *z* = 2 + 3.5*i или z = 2 + 3.5*j.

Функция `real(z)` возвращает действительную часть комплексного числа Re(z), а функция `imag(z)` – мнимую Im(z). Для получения модуля комплексного числа используется функция `abs(z)`, а для вычисления аргумента – `angle (z)`.

MATLAB – система, специально предназначенная для проведения сложных вычислений с векторами, матрицами и многочленами. При этом она по умолчанию предполагает, что каждая заданная переменная - это вектор или матрица. Все определяется конкретным значением переменной. Например, если задано X = 1, то это значит, что X есть вектор с единственным элементом, имеющим значение 1. Если надо задать вектор из трех элементов, то их значения надо перечислить в квадратных скобках, разделяя пробелами.

Например, ввод
```
» V = [1 2 3]
V =
```

```
1     2     3
```

задает вектор V, имеющий три элемента со значениями 1, 2 и 3. После ввода вектора система выводит его на экран дисплея.

Задание матрицы требует указания различных строк. Для различения строк также используется знак ; (точка с запятой).

Например:

```
» A=[ 1 2 3; 4 5 6; 7 8 9]
A =
     1     2     3
     4     5     6
     7     8     9
```

Мы задаем квадратную матрицу A, которую потом можно вывести на экран, набрав в командной строке ее имя A и нажав клавишу ввода ENTER. Матрицы можно вводить и по-другому (при вводе строки нажимается клавиша ENTER):

```
» A = [ 1 2 3
        4 5 6
        7 8 9 ];
```

Такой способ может оказаться предпочтительным при вводе больших матриц.

Возможен ввод элементов матриц и векторов в виде арифметических выражений, содержащих любые доступные системе функции.

Пример:

```
» V = [2*2/(3*4) exp(5) sqrt(10)];
      » V
      V =
          0.3333  148.4132    3.1623
```

Для указания отдельного элемента вектора или матрицы используются выражения вида `V(i)` или `M(i,j)`. Например, легко определить значение элемента матрицы A, стоящего на месте (2, 2):

```
» A(2,2)
```

```
ans =
     5
```

он равен 5. Если нужно присвоить элементу A(i, j) новое значение x, то нужно использовать выражение A(i,j)=x

Например, если элементу A(2, 2) надо присвоить значение 10, следует записать

```
» A(2,2)=10
```

Выражение вида `A(i)` с одним индексом дает доступ к элементам матрицы, развернутым в один столбец. Такая матрица образуется из исходной, если подряд выписать ее столбцы. Следующий пример поясняет такой доступ к элементам матрицы A:

```
» A=[1 2 3; 4 5 6; 7 8 9]
A =
     1     2     3
     4     5     6
     7     8     9
» A(2)
ans =
     4
» A(8)
ans =
     6
» A(5)=100
A =
     1     2     3
     4   100     6
     7     8     9
```

Возможно задание векторов и матриц с комплексными элементами.

Пример:

```
» CM=[ [1 2; 3 4] + i*[5 6; 7 8] ]
CM =
1.0000 + 5.0000i   2.0000 + 6.0000i
3.0000 + 7.0000i   4.0000 + 8.0000i
```

или по-другому

```
» CM=[ 1+5i 2+6i; 3+7i 4+8i]
CM =
   1.0000 + 5.0000i   2.0000 + 6.0000i
   3.0000 + 7.0000i   4.0000 + 8.0000i
```

Наряду с операциями над отдельными элементами матриц векторов система позволяет производить операции умножения, деления и возведения в степень сразу над всеми элементами, т.е. над массивами. Для этого перед операцией ставится точка. Например, оператор * означает умножение для векторов или матриц (по правилам матричной алгебры), а оператор .* - почленное умножение всех элементов массива. Таким образом, если M - матрица, то M.*2 даст матрицу, все элементы которой умножены на число 2. Необходимо также помнить, что при математических операциях с матрицами и векторами, их размеры должны соответствовать друг другу.

Пример:

```
» A = [1 2; 4 5]
A =
     1     2
     4     5
» B=[ 1 0; 0 1]
B =
     1     0
     0     1
» A*B
ans =
     1     2
     4     5
» A.*B
ans =
     1     0
     0     5
```

Объединение малых матриц в большую. Описанный способ задания матриц позволяет выполнить операцию объединения малых матриц в одну большую. Например, создадим вначале матрицу размером 3x3:

```
» A=magic(3)
```

```
A =
     8     1     6
     3     5     7
     4     9     2
```

Теперь можно построить матрицу B, содержащую четыре матрицы:

```
» B=[A A+16; A+32 A.*3]
B =
     8     1     6    24    17    22
     3     5     7    19    21    23
     4     9     2    20    25    18
    40    33    38    24     3    18
    35    37    39     9    15    21
    36    41    34    12    27     6
```

Полученная матрица имеет уже размер 6x6.

Применение оператора : (двоеточие). Очень часто необходимо произвести формирование упорядоченных числовых последовательностей. Такие последовательности нужны например для создания векторов или значений абсциссы при построении графиков. Для этого в MATLAB используется оператор : (двоеточие):

Ситаксис: <Начальное_значение>:[<Шаг>]:<Конечное_значение>

Данная конструкция порождает возрастающую последовательность чисел, которая начинается с <Начального_значения>, идет с заданным <Шагом> и завершается <Конечным_значением>. Если <Шаг> не задан, по умолчанию он принимается равным 1. Примеры применения оператора : даны ниже.

Пример:

```
» 1:5
ans =
     1     2     3     4     5

» i=0:2:10
i =
     0     2     4     6     8    10
» j=10:-2:1

j =
    10     8     6     4     2
» V=0:pi/2:2*pi
```

```
V =
         0   1.5708   3.1416   4.7124   6.2832
» 5:-1:0
ans =
     5     4     3     2     1     0
```

Выражения с оператором : могут использоваться в качестве аргументов функций для получения множества их значений. Например, вычисление функции sin(x) в диапазоне от 0 до 3 с шагом 0,5:

```
» sin(0:0.5:3)
ans =
  0   0.4794   0.8415   0.9975   0.9093   0.5985   0.1411
  1
```

Оператор : часто используется для вывода только части матрицы на экран или применения в математических операциях.

Пример:

```
A=[1 2 3 4;5 6 7 8; 9 10 11 12 ; 13 14 15 16]
A =
     1     2     3     4
     5     6     7     8
     9    10    11    12
    13    14    15    16
```

вывод подматрицы, состоящей из элементов 1, 2 и 3 строки и 2, 3 и 4 столбцов матрицы A.

```
» A(1:3, 2:4)
ans =
     2     3     4
     6     7     8
    10    11    12
```

вывод только 1 и 3 строк матрицы A (вектор номеров строк указан с шагом 2)

```
» A(1:2:4, :) % -
ans =
     1     2     3     4
     9    10    11    12
```

Для формирования матриц и выполнения ряда матричных операций возникает необходимость удаления отдельных столбцов или строк матрицы. Для этого используется пустые квадратные скобки [] (так называемая "пустая матрица"). Проделаем это над матрицей A:

```
» A=[1 2 3; 4 5 6; 7 8 9]
A =
     1     2     3
     4     5     6
     7     8     9
```

Удалим второй столбец, используя оператор : (двоеточие):

```
» A(:,2)=[]
A =
     1     3
     4     6
     7     9
```

А теперь удалим вторую строку:

```
» A(2,:)=[]
A =
     1     3
     7     9
```

Помимо встроенных в систему операторов и функций могут использоваться и внешние программы, т.е. написанные пользователем файлы с расширением *.m. Можно как воспользоваться встроенными m-функциями, уже включенными в поставку системы, так и создать свои собственные. В итоге MATLAB оказывается расширяемой системой, легко адаптируемой под решение конкретных математических задач, интересующих пользователя.

В системе MATLAB внешние программы (написанные пользователем) используются точно так же, как и встроенные функции и операторы. Никаких особых указаний по их применению делать не надо. Разумеется, скорость вычислений по внешним программам несколько ниже, чем по встроенным функциям или операторам. При этом вычисления происходят следующим

образом: вначале система быстро определяет, есть ли слово среди служебных слов системы. Если оно есть, то нужные вычисления выполняются сразу; если нет, то система ищет m-файл с соответствующим именем на диске. Если файла нет, то выдается сообщение об ошибке. Если файл найден, он загружается в память (компилируется) и исполняется. Список доступных m-файлов в текущем каталоге выводится с помощью команды `what`.

3.5 Рабочая область

Рабочая область системы MATLAB – это область памяти, в которой размещены переменные системы. Содержимое этой области можно просмотреть из командной строки с помощью команд `who` и `whos`. Команда `who` выводит только имена переменных, а команда `whos` – еще и дополнительную информацию о размерах массивов и типе переменной.

Загрузка и сохранение рабочей области. Команды `save` и `load` позволяют в любой момент времени сохранить содержимое рабочей области или загрузить новые данные в процессе сеанса работы. С помощью этих команд можно также осуществлять экспорт и импорт ASCII-файлов.

Сохранение переменных рабочей области. Команда `save` позволяет сохранить содержимое рабочей области в двоичном MAT-файле (он имеет расширение *.mat), который можно в дальнейшем вызвать командой `load`.

Спецификация формата файла. Для того чтобы управлять форматами файлов, следует в команде `save` в дополнение к имени файла и списку переменных использовать следующие флаги:

Флаг	Пояснение
-mat	Двоичный MAT-файл (по умолчанию)
-ascii	ASCII-формат (8 цифр)
-ascii –double	ASCII-формат (16 цифр)
-ascii –double -tabs	Формат с разделителями и метками табуляции
-append	Добавить данные к существующему MAT-файлу

Синтаксис: `save` < *имя_файла* . [*расширение*] > *список_переменных – флаг*

`save` – сохранение всех переменных (содержимого рабочей области) в файл matlab.mat в текущем каталоге;

`save aaa` – сохранение содержимого рабочей области в файл aaa.mat в формате системы MATLAB;

`save my.txt a -ASCII` - сохранение переменной a в файл my.txt в виде ASCII-кодов (текстовый формат);

`save my a1`, a2, a3 `-ASCII` - сохранение переменных a1, a2, a3 в файл my.mat в двоичном формате.

Когда содержимое рабочей области сохраняется в ASCII-формате, то рекомендуется единовременно сохранять только одну переменную. Если сохраняется более одной переменной, то система MATLAB создаст файл ASCII-файл, который нельзя будет в дальнейшем загрузить в MATLAB, используя команду `load`. По умолчанию, если пользователь не указывает расширение в имени файла и не устанавливает флаг, система присваивает файлу расширение < *имя_файла* >.mat.

Загрузка рабочей области. Команда `load` позволяет загрузить MAT-файл, который был ранее сохранен с помощью команды `save`. При загрузке MAT-файла новые значения одноименных переменных будут записаны взамен старых. Если MAT-файл имеет расширение, отличающееся от *.mat, то необходимо использовать флаг -mat; в противном случае MATLAB будет считать форматом файла ASCII-формат.

Загрузка файлов данных в ASCII-формате. Команда `load` позволяет также выполнять импорт файлов данных в ASCII-формате; она преобразует содержимое файла в переменную с именем файла только без расширения. Например, применение команды `load tides.dat` создает в рабочей области системы MATLAB переменную с именем `tides`. Если исходный файл в ASCII-формате имеет m строк с n значениями в каждой строке, то результатом будет массив чисел размера mxn.

Синтаксис: `load` < *имя_файла* . [*расширение*] > *-флаг*

Пример:

`load` – загрузка файла matlab.mat из текущего каталога;

`load aaa` – загрузка файла aaa.mat в формате системы MATLAB;

`load aaa.txt -mat` – загрузка файла aaa.txt в формате системы MATLAB;

`load my.txt` – загрузка файла my.txt в текстовом формате (при этом содержимое файла будет помещено в переменную (матрицу) с именем my)

Замечание: Для сохранения или загрузки последовательности файлов, имена которых имеют общий корень и дополнительный целочисленный суффикс, необходимо использовать структуру цикла.

Например, следующая конструкция позволяет сохранить квадраты чисел от 1 до 10 в файлах с именами `data1, …, data10`:

```
file = 'data';
for i = 1:10,
    j = i.^2;
    save([file int2str(i)],'j');
end
```

Команды `load` и `save` допускают использование группового символа * в качестве замены ряда символов в шаблоне имени переменной.

Например, команда `save rundate x*` сохраняет все переменные, имена которых начинаются с символа `x` в файле с именем `rundata.mat`. Точно также команда `load testdata ex1*95` загружает все переменные, имена которых начинаются с символов `'ex1'` и заканчиваются символами `'95'`, независимо от того, какие символы размещены между ними.

Удаление переменных из рабочей области. В памяти компьютера введенные переменные занимают определенное место. Использовании матриц и векторов больших размеров при сложных вычислениях может привести к нехватке памяти. Поэтому возникает необходимость удаления из памяти (рабочего пространства) MATLAB ненужных переменных. Для очистки

рабочего пространства используется функция `clear` в разных формах (см. ранее гл.3).

Стертая из рабочего пространства переменная становится недоступной для использования. Если необходимо лишь очистить переменную от значений, не удаляя ее из рабочего пространства, следует использовать «пустую матрицу», т.е. использовать выражения вида `A = []`. В этом случае имя переменной сохраняется и в дальнейшем ее можно использовать для вычислений.

Необходимо отметить, что переменные, объявленные внутри m-функций «не видны» за ее пределами, т.е. ее нельзя использовать вне функции. Если необходимо, чтобы переменные были общими для несколько m-функций (или m-сценариев) необходимо объявлять их как глоабльные с помощью команды `global`:

Синтаксис: global <список переменных>

Объявление глобальных переменных должно быть вначале всех m-функций (m-сценариев), где эти переменные будут использоваться. При этом глобальные переменные будут храниться в специальной области памяти системы MATLAB. Их нельзя удалить с помощью простой команды `clear`. Для удаления необходимо использовать команду `clear global`.

3.6 Программирование в среде MATLAB

Файлы, которые содержат команды языка MATLAB, называются m-файлами. Для создания m-файла используется любой текстовый редактор; вызову m-файла предшествует присваивание значений входным аргументам; результатом является значение выходной переменной. Таким образом, вся процедура включает две операции:

1) Создать m-файл, используя текстовый редактор:

```
function c = myfile(a, b)
   c = sqrt((a.^2)+(b.^2))
```

2) Вызвать m-файл из командной строки или из другого m-файла:

```
» a=7.5;
» b=3.342;
» c=myfile(a,b)
c =
    8.2109
```

Типы m-файлов. Существует два типа m-файлов: m-сценарии (m-script) и m-функции (m-function) со следующими характеристиками

m-сценарий	m-функция
Не использует входных и выходных аргументов	Использует входные и выходные аргументы
Оперирует с данными из рабочей области	По умолчанию, внутренние переменные являются локальными по отношению к функции
Предназначен для автоматизации последовательности шагов, которые нужно выполнять много раз	Предназначена для расширения возможностей языка MATLAB (библиотеки функций, пакеты прикладных программ)

Структура m-файла. m-файл, оформленный в виде функции состоит из следующих компонентов

Строка определения функции Первая строка комментария (help-строка)	`function f = fact(n)` `% FACT Вычисление факториала.`
Комментарии	`% fact(n) возвращает n! – факториал` `% числа n`
Тело функции	`f = prod(1:n);`

Структура этой простейшей функции содержит компоненты, которые являются общими для любых функций системы MATLAB:

– **Строка определения функции** задаёт имя, количество и порядок следования входных и выходных аргументов.

- **Первая строка комментария** (help-строка) определяет назначение функции или сценария. Она выводится на экран с помощью команд `lookfor` или `help <имя_файла>`.
- **Комментарий** выводится на экран вместе с первой строкой при использовании команды `help <имя_файла>`.
- **Тело функции** - это программный код, который реализует вычисления и присваивает значения выходным аргументам.

3.7 Создание m-файлов. m-сценарии. m-функции

m-файлы являются обычными текстовыми файлами, которые создаются с помощью текстового редактора. Система MATLAB включает в себя специальный встроенный редактор/отладчик с подствекой синтаксиса, хотя можно использовать и любой другой текстовый редактор, работающий с ASCII-кодами (например, «Блокнот» или текстовый редактор файлового менеджера FAR, Total Commander, и др.).

m-сценарии (m-script)

Сценарии являются самым простым типом m-файла – у них нет входных и выходных аргументов. Они используются для автоматизации многократно выполняемых вычислений. Сценарии оперируют данными из рабочей области и могут генерировать новые данные для последующей обработки в этом же файле. Данные, которые используются в сценарии, сохраняются в рабочей области после завершения сценария и могут быть использованы для дальнейших вычислений.

Пример: Следующие операторы вычисляют радиус-вектор `rho` для различных тригонометрических функций от угла `theta` и строят последовательность графиков в полярных координатах.

Строка комментария (help-строка)	`% m-file petals – сценарий` `% построения лепесткового графика`

Вычисления	`theta = -pi:0.01:pi;`
	`rho(1, :) = 2*sin(5*theta).^2;`
	`rho(2, :) = cos(10*theta).^3;`
	`rho(3, :) = sin(theta).^2;`

Команды графического вывода

```
rho(4, :) = 5*cos(3.5*theta).^3;
for i = 1:4
      polar (theta, rho(i, :))
     pause
end
```

Создайте m-файл `petals.m`, введя указанные выше операторы. Этот файл является сценарием. Ввод команды `petals.m` в командной строке системы MATLAB вызывает выполнение операторов этого сценария.

После того, как сценарий отобразит первый график, нажмите любую клавишу, чтобы перейти к следующему графику. В сценарии отсутствуют входные и выходные аргументы; программа `petals.m` сама создаёт переменные, которые сохраняются в рабочей области системы MATLAB. Когда выполнение завершено, переменные (`i`, `theta` и `rho`) остаются в рабочей области. Для того чтобы увидеть этот список, следует воспользоваться командой `whos`.

m-функции (m-function)

m-функции являются m-файлами, которые допускают наличие входных и выходных аргументов. Они работают с переменными в пределах собственной рабочей области, отличной от рабочей области системы MATLAB.

Пример: Функция `average` - это достаточно простой m-файл, который вычисляет среднее значение элементов вектора:

```
function y = average(x)
% AVERAGE Среднее значение элементов вектора.
% AVERAGE(X), где X - вектор. Вычисляет среднее значение
%                        элементов вектора.
% Если входной аргумент не является вектором,
% генерируется ошибка.
    [m,n] = size(x);
    if (~((m == 1) | (n == 1)) | (m == 1 & n == 1))
       error('Входной массив должен быть вектором')
    end
    y =sum(x)/length(x);  % -> Собственно вычисление
```

Попробуйте ввести эти команды в m-файл, именуемый `average.m`. Функция `average` допускает единственный входной и единственный выходной аргументы. Для того чтобы вызвать функцию `average`, надо ввести следующие операторы:

```
» z = 1:99;
» average(z)
» ans = 50
```

Структура m-функции. Любая m-функция состоит из:

- строки определения функции;
- первой строки комментария;
- собственно комментария;
- тела функции;
- строчных комментариев.

Строка определения функции. Строка определения функции сообщает системе MATLAB, что файл является m-функцией, а также определяет список входных аргументов.

*Пример***:** Строка определения функции `average` имеет вид:

```
function y = average(x)
```

Здесь:

function - ключевое слово, определяющее m-функцию;

y - выходной аргумент;

average - имя функции;

x - входной аргумент.

Каждая функция в системе MATLAB содержит строку определения функции, подобную приведенной.

Если функция имеет более одного выходного аргумента, список выходных аргументов помещается в квадратные скобки. Входные аргументы, если они присутствуют, помещаются в круглые скобки. Для отделения аргументов во входном и выходном списках применяются запятые.

Пример:

```
function [x, y, z] = sphere(theta, phi, rho)
```

Имена входных переменных могут, но не обязаны совпадать с именами, указанными в строке определения функции.

Первая строка комментария. Для функции `average` первая строка комментария выглядит так:

```
% AVERAGE Среднее значение элементов вектора
```

Это – первая строка текста, которая появляется, когда пользователь набирает команду `help <имя_функции>`. Кроме того, первая строка комментария выводится на экран по команде поиска `lookfor`. Поскольку эта строка содержит важную информацию об m-файле, она должна быть тщательно составлена.

Тело функции. Тело функции содержит код языка MATLAB, который выполняет вычисления и присваивает значения выходным аргументам. Операторы в теле функции могут состоять из вызовов функций, программных конструкций для управления потоком команд, интерактивного ввода/вывода, вычислений, присваиваний, комментариев и пустых строк.

Пример:

Тело функции `average` включает ряд простейших операторов программирования:

Оператор вызова функции size
Начало оператора if
Сообщение об ошибке
Конец оператора if
Вычисление и присваивание

```
[m, n] = size(x);
      if (~((m == 1) | (n==1)) | (m ==1 &
                    n==1))
   Error('Input должно быть вектором')
end
y = sum(x)/length(x);
```

Как уже говорилось ранее, комментарии отмечаются знаком (%). Строка комментария может быть размещена в любом месте m-файла, в том числе и в конце строки.

Пример:

```
% -- Найти сумму всех элементов вектора x
    y = sum(x)     % -- Использована функция sum
```

Кроме строк комментариев в текст М-файла можно включать пустые строки. Однако надо помнить, что пустая строка может служить указателем окончания подсказки.

Имена m-функций. В системе MATLAB на имена m-функций налагаются те же ограничения, что и на имена переменных - их длина не должна превышать 31 символа. Более точно, имя может быть и длиннее, но система MATLAB принимает во внимание только первые 31 символ. Имена m-функций должны начинаться с буквы; остальные символы могут быть любой комбинацией букв, цифр и подчеркиваний.

Имя файла, содержащего m-функцию, составляется из имени функции и расширения *.m : (например, `average.m`).

Если имя файла и имя функции в строке определения функции разные, то используется имя файла, а внутреннее имя игнорируется. Хотя имя функции, определенное в строке определения функции, может и не совпадать с именем файла, настоятельно рекомендуется использовать одинаковые имена.

Двойственность функций и команд. Команды системы MATLAB – это операторы вида:

```
load
help
```

Многие команды могут быть модифицированы добавлением операндов:

```
load August17.dat
help magic
type rank
```

Альтернативный метод задания модификаторов - определить их в качестве строковых аргументов функции:

```
load('August17.dat')
help('magic')
type('rank')
```

В этом заключается двойственность понятий команды и функции в системе MATLAB. Любая команда вида

```
command argument
```

может быть записана в форме функции

```
command('argument').
```

Преимущество функционального описания проявляется, когда строка аргументов формируется по частям. Следующий пример показывает, как может быть обработана последовательность файлов `August1.dat, August2.dat`, и т.д. Здесь используется функция `int2str`, которая переводит целое число в строку символов, что помогает сформировать последовательность имён файлов.

```
for d = 1:31
     s = ['August', int2str(d), '.dat']
     load(s) % - Загрузить файл с именем August'd'.dat
     %  --- Операторы обработки файла ...
end
```

m-функцию можно вызвать из командной строки системы MATLAB или из других m-файлов, обязательно указав все необходимые атрибуты – входные аргументы в круглых скобках, выходные аргументы в квадратных скобках.

Назначение имени. Когда появляется новое имя, система MATLAB проверяет:

- Не является ли новое имя именем переменной.
- Не является ли это имя именем подфункции, то есть функции, которая размещена в этом же m-файле и является вызываемой.
- Не является ли оно именем частной функции, размещаемой в каталоге `private`. Этот каталог доступен только m-файлам, размещенным на один уровень выше.
- Не является ли оно именем функции в пути доступа системы MATLAB. В этом случае система использует тот m-файл, который встречается первым в пути доступа.

В случае дублирования имен система MATLAB использует первое имя в соответствии с вышеприведенной четырехуровневой иерархией. Следует отметить, что в системе MATLAB допускается переопределять функцию по правилам объектно-ориентированного программирования.

Вызов функции. При вызове m-функции, система MATLAB транслирует функцию в псевдокод и загружает в память. Это позволяет избежать повторного синтаксического анализа. Псевдокод остаётся в памяти до тех пор, пока не будет использована команда `clear` или завершен сеанс работы.

3.8 Типы переменных

Локальные и глобальные переменные. Использование переменных в m-файле ничем не отличается от использования переменных в командной строке, а именно:

- переменные не требуют объявления; прежде чем переменной присвоить значение, необходимо убедиться, что всем переменным в правой части значения присвоены;
- любая операция присваивания создает переменную, если это необходимо, или изменяет значение существующей переменной;
- имена переменных начинаются с буквы, за которой следует любое количество букв, цифр и подчеркиваний; система MATLAB различает символы верхнего и нижнего регистров;
- имя переменной не должно превышать 31 символа. Более точно, имя может быть и длиннее, но система MATLAB принимает во внимание только первые 31 символ.

Обычно каждая m-функция, задаваемая в виде m-файла, имеет собственные локальные переменные, которые отличны от переменных других функций и переменных рабочей области. Однако, если несколько функций и рабочая область объявляют некоторую переменную глобальной, то все они используют единственную копию этой переменной. Любое присваивание этой переменной распространяется на все функции, где она объявлена глобальной.

Пример:

Допустим, требуется исследовать влияние коэффициентов a и b для модели «хищник-жертва», описываемой уравнениями Лотке-Вольтерра:

$$\begin{cases} \dot{y}_1 = y_1 - \alpha\, y_1 y_2 \\ \dot{y}_2 = -y_2 + \beta\, y_1 y_2 \end{cases}$$

Создадим m-файл `lotka.m`:

```
    function yp = lotka(t, y)
% LOTKA уравнения Лотке-Вольтерра для модели хищник-жертва
global ALPHA BETA
yp = [y(1)-ALPHA*y(1)*y(2); -y(2)+BETA*y(1)*y(2)];
```

Затем через командную строку введем операторы:

```
» global ALPHA BETA
» ALPHA = 0.01;
» BETA = 0.02;
» [t,y] = ode23('lotka2',[0 10],[1; 1]);
» lot(t,y)
```

Команда `global` объявляет переменные ALPHA и BETA глобальными и следовательно, доступными в функции `lotka.m`. Таким образом, они могут быть изменены из командной строки, а новые решения будут получены без редактирования m-файла `lotka.m`.

Для работы с глобальными переменными необходимо:

1. объявить переменную как глобальную в каждой m-функции, которая необходима эта переменная. Для того чтобы переменная рабочей области была глобальной, необходимо объявить ее как глобальную из командной строки;
2. в каждой функции использовать команду `global` перед первым появлением переменной; рекомендуется указывать команду `global` в начале m-файла.

Имена глобальных переменных обычно более длинные и более содержательные, чем имена локальных переменных, и часто используют заглавные буквы. Это необязательно, но рекомендуется, чтобы обеспечить

удобочитаемость кода языка MATLAB и уменьшить вероятность случайного переопределения глобальной переменной.

Специальные переменные. Некоторые m-функции возвращают специальные переменные, которые играют важную роль при работе в среде системы MATLAB:

`ans`	Последний результат; если выходная переменная не указана, то MATLAB использует переменную ans.
`eps`	Точность вычислений с плавающей точкой; определяется длиной мантиссы и для PC eps = 2.220446049250313e-016
`realmax`	Максимальное число с плавающей точкой, представимое в компьютере; для PC realmax = 1.797693134862316e+308.
`realmin`	Минимальное число с плавающей точкой, представимое в компьютере; для PC realmin = 2.225073858507202e-308.
`pi`	Специальная переменная для числа p: pi=3.141592653589793e+000.
`i, j`	Специальные переменные для обозначения мнимой единицы
`inf`	Специальная переменная для обозначения символа бесконечности
`NaN`	Специальная переменная для обозначения неопределенного значения - результата операций типа: 0/0, inf/inf.
`computer`	Специальная переменная для обозначения типа используемого компьютера; для PC - PCWIN.
`flops`	Специальная переменная для обозначения количества операций с плавающей точкой.
`version`	Специальная переменная для хранения номера используемой версии системы MATLAB.

Соответствующие М-функции, генерирующие эти специальные переменные, находятся в каталоге elmat и поддержаны online-подсказкой.

3.9 Операторы системы MATLAB. Объединение операторов в арифметические выражения

Операторы системы MATLAB делятся на три категории:

- Арифметические операторы позволяют конструировать арифметические выражения и выполнять числовые вычисления.
- Операторы отношения позволяют сравнивать числовые операнды.
- Логические операторы позволяют строить логические выражения.

Логические операторы имеют самый низкий приоритет относительно операторов отношения и арифметических операторов.

Арифметические операторы. При работе с массивом чисел установлены следующие уровни приоритета среди арифметических операций :

уровень 1: поэлементное транспонирование (.'), поэлементное возведение в степень (.^), эрмитово-сопряженное транспонирование матрицы (') , возведение матрицы в степень (^);

уровень 2: унарное сложение (+), унарное вычитание (-);

уровень 3: умножение массивов (.*), правое деление (./), левое деление массивов (.\), умножение матриц (*), решение систем линейных уравнений - операция (/), операция (\);

уровень 4: сложение (+), вычитание (-);

уровень 5: оператор формирования массивов (:).

Внутри каждого уровня операторы имеют равный приоритет и вычисляются в порядке следования слева направо. Заданный по умолчанию порядок следования может быть изменен с помощью круглых скобок

Арифметические операторы допускают использование индексных выражений:

Пример:

```
b = sqrt(A(2))+2*B(1);
b = 7
```

Арифметические операторы системы MATLAB работают, как правило, с массивами одинаковых размеров. Для векторов и прямоугольных массивов оба операнда должны быть одинакового размера, за исключением единственного случая, когда один из них - скаляр. Если один из операндов скалярный, а другой нет, в системе MATLAB принято, что скаляр расширяется до размеров второго операнда и заданная операция применяется к каждому элементу. Такая операция называется расширением скаляра.

Операторы отношения. В системе MATLAB определено 6 следующих операторов отношения:

< Меньше

<= Меньше или равно

> Больше

> = Больше или равно

= = Равно тождественно

~ = Не равно

Операторы отношения выполняют поэлементное сравнение двух массивов равных размерностей. Для векторов и прямоугольных массивов, оба операнда должны быть одинакового размера, за исключением случая, когда один из них скаляр. В этом случае MATLAB сравнивает скаляр с каждым элементом другого операнда. Позиции, где это соотношение истинно, получают значение 1, где ложно – 0.

Операторы отношения, как правило, применяется для изменения последовательности выполнения операторов программы. Поэтому они чаще всего используются в теле операторов `if`, `for`, `while`, `switch`. Операторы отношения всегда выполняются поэлементно

Пример:

Выполним сравнение двух массивов, используя условие A<B:

```
» A = [2 7 6; 9 0 -1; 3 0.5 6];
»  B = [8 0.2 0; -3 2 5; 4 -1 7];
» A<B
ans =
     1     0     0
     0     1     1
     1     0     1
```

Полученная матрица указывает позиции, где элемент A меньше соответствующего элемента B.

При вычислении арифметических выражений операторы отношения имеют более низкий приоритет, чем арифметические, но более высокий, чем логические операторы.

Операторы отношения могут применяться к многомерным массивам, для которых одна из размерностей равна нулю, при условии, что оба массива - одинакового размера или один из них - скаляр. Однако выражения типа `A == [ ]` применимы только к массивам размера 0x0 или 1x1, а в других случаях вызывают ошибку.

Поэтому наиболее универсальный способ проверить, является ли массив пустым – это применить функцию `isempty(A)`.

Логические операторы. В состав логических операторов системы MATLAB входят следующие операторы:

&	И
|	ИЛИ
~	НЕТ

Логические операторы реализуют поэлементное сравнение массивов одинаковых размерностей. Для векторов и прямоугольных массивов оба операнда должны быть одинакового размера, за исключением случая, когда один из них скаляр. В последнем случае MATLAB сравнивает скаляр с каждым элементом другого операнда. Позиции, где это соотношение истинно, получают значение 1, где ложно - 0.

Каждому логическому оператору соответствует некоторый набор условий, которые определяю результат логического выражения:

Логическое выражение с оператором AND (&) является истинным, если оба операнда - истинны. Если элементами логического выражения являются числа, то выражение истинно, если оба операнда отличны от нуля.

Пример:

Пусть заданы два числовых вектора:

```
» u = [1 0 2 3 0 5];
```

```
» v = [5 6 1 0 0 7];
```

и логическое выражение с оператором AND (&):

```
» u & v
ans =
     1     0     1     0     0     1
```

Логическое выражение с оператором OR (|) является истинным, если один из операндов или оба операнда логически истинны. Выражение ложно, только если оба операнда логически ложны. Если элементами логического выражения являются числа, то выражение ложно, если оба операнда равны нулю.

Пример:

Используем векторы u и v, определенные выше, и выполним логическое выражение с оператором OR (|):

```
» u|v
ans =
     1     1     1     1     0     1
```

Логическое выражение с оператором NOT (~) строит отрицание. Результат логически ложен, если операнд истинен, и истинен, если операнд ложен. Если элементами логического выражения являются числа, то любой операнд, отличный от нуля, становится нулем, и любой нулевой операнд становится единицей.

Пример:

```
» ~u
ans =     0     1     0     0     1     0
```

Логические функции. В дополнение к логическим операторам в состав системы MATLAB включено ряд логических функций:

Функция `xor(a,b)` реализует операцию ИСКЛЮЧИТЕЛЬНОЕ ИЛИ. Выражение, содержащее ИСКЛЮЧИТЕЛЬНОЕ ИЛИ истинно, если один из операндов имеет значение TRUE, а другой FALSE. Для числовых выражений,

функция возвращает 1, если один из операндов отличен от нуля, а другой - нуль.

Пример:

Рассмотрим два числовых операнда a и b:

```
» a=1;
» b=1;
» xor(a,b)
ans =
     0
```

Функция `all` возвращает 1, если все элементы вектора истинны или отличны от нуля

Пример:

Пусть задан вектор u и требуется проверить его на условие «все ли элементы меньше 3». Если это условие выполняется, то выдается сообщение «Все элементы меньше 3»:

```
» u= [ 1 2 3 4 0];
» if all(u<3),
 disp(' Все элементы меньше 3');
  end
```

В данном случае никакого сообщения не появится, но если в качестве вектора u взять

```
» u= [ 0 1 2 1 0];
```

то результатом будет сообщение:

```
 Все элементы меньше 3
```

В случае массивов функция `all` проверяет столбцы, то есть является ориентированной по столбцам.

Функция `any` возвращает 1, если хотя бы один из элементов аргумента отличен от нуля; иначе, возвращается 0. В случае обработки массивов функция `any` является столбцовоориентированной.

Функции `isnan` и `isinf` возвращают 1 для NaN и Inf, соответственно. Функция `isfinite` истинна только для величин, которые не имеют значения inf или NaN.

Пример:

Рассмотрим следующие два числовых массива A и B

```
A = [0 1 5;2 NaN -inf];
B = [0 0 15; 2 5 inf];
```

Образуем массив C и применим перечисленные выше функции

```
C = A./B
C =
```

NaN	Inf	0.3333
1.0000	NaN	NaN

Isfinite(C)	isnan(C)	Isinf(C)
ans =	ans =	ans =
0 0 1	1 0 0	0 1 0
1 0 0	0 1 1	0 0 0

Функция `find` определяет индексы элементов массива, которые удовлетворяют заданному логическому условию. Как правило, она используется для создания шаблонов для сравнения и создания массивов индексов. В наиболее употребительной форме функция `txt=find(x<условие>)` возвращает вектор индексов тех элементов, которые удовлетворяет заданному условию.

Пример:

```
» A=magic(4)
A =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
» txt=find(A>8);
» A(txt)=100*ones(1,length(txt))
```

```
A =

   100     2     3   100
     5   100   100     8
   100     7     6   100
     4   100   100     1
```

Функция вида `[i,j]=find(x)` позволяет получить индексы ненулевых элементов прямоугольного массива. Функция вида `[i,j,s]=find(x)` возвращает кроме того и их значения в виде вектора s.

Объединение операторов в арифметические выражения

Теперь вы имеете возможность строить выражения, которые используют любую комбинацию арифметических, логических операторов и операторов отношения.

Управление последовательностью исполнения операторов. Существуют четыре основных оператора управления последовательностью исполнения инструкций:

оператор условия `if`, в сочетании с оператором `else` и `elseif` выполняет группу инструкций в соответствии с некоторыми логическими условиями;

- оператор переключения `switch`, в сочетании с операторами `case` и `otherwise` выполняет различные группы инструкций в зависимости от значения некоторого логического условия;
- оператор условия `while` выполняет группу инструкций неопределенное число раз, в соответствии с некоторым логическим условием завершения;
- оператор цикла `for` выполняет группу инструкций фиксированное число раз.

Все операторы управления включают оператор `end`, чтобы указать конец блока, в котором действует этот оператор управления.

`if...else...elseif...end` **Оператор условия**

Синтаксис:

```
if  <логическое_выражение>        if <логическое_выражение>      if<  логическое_выражение>
инструкции                        инструкции                     инструкции
end                                 else                          elseif <лог-кое_выраж-е>
                                  инструкции                     инструкции
                                    end                           else
                                                                  инструкции
                                                                  end
```

Описание:

Оператор условия `if .... end` вычисляет некоторое логическое выражение и выполняет соответствующую группу инструкций в зависимости от значения этого выражения. Если логическое выражение истинно, то MATLAB выполнит все инструкции между `if` и `end`, а затем продолжит выполнение программы в строке после `end`. Если условие ложно, то MATLAB пропускает все утверждения между `if` и `end` и продолжит выполнение в строке после `end`.

Пример:

```
if rem(a,2)==0,
     disp('a - четное число ')
     b = a/2;
end
```

Если логическое условие включает переменную, не являющуюся скаляром, то утверждение будет истинным, если все элементы отличны от нуля.

Пример:

Пусть задана матрица X; запишем следующий оператор условия:

```
if X
     инструкции
end
```

Этот оператор равносилен следующему:

```
if all(X(:))
```

```
    инструкции
end
```

Операторы `if...else...end` и `if...elseif...end` создают дополнительные ветвления внутри тела оператора `if`:

1. Оператор `else` не содержит логического условия. Инструкции, связанные с ним, выполняются, если предшествующий оператор `if` (и возможно `elseif`) ложны. Оператор `elseif` содержит логическое условие, которое вычисляется, если предшествующий оператор `if` (и возможно `elseif`) ложны. Инструкции, связанные с оператором `elseif` выполняются, если соответствующее логическое условие истинно.
2. Оператор `elseif` может многократно использоваться внутри оператора условия `if`.

Пример:

Рассмотрим фрагмент программы:

```
if n<0,   % Если n<0, вывести сообщение об ошибке
disp('Введенное число должно быть положительным');
elseif rem(n,2)==0, % Если n≥0 и четное, разделить на два
a=n/2;
else  % Если n≥0 и нечетное, увеличить на 1 и разделить
a=(n+1)/2;
end
```

Если в операторе if условное выражение является пустым массивом, то такое условие ложно. То есть оператор условия вида

```
if A,
    S1
else
    S0
end
```

выполнит инструкции S0 только тогда, когда A - пустой массив.

`switch...case...otherwise...end` **Оператор переключения**

Синтаксис:

`switch` *<выражение> % выражение - это обязательно скаляр или строка*

`case` *<значение1>*

инструкции % выполняются, если < выражение> =< значение1>

`case` *<значение2>*

инструкции % выполняются, если <выражение> = < значение2>

`otherwise`

инструкции % выполняются, если <выражение> не совпало ни с одним из значений

`end`

Оператор переключения `switch` работает только в системе MATLAB 5.x и выше. Оператор `switch...case_1...case_k...otherwise...end` выполняет ветвления, в зависимости от значений некоторой переменной или выражения.

Оператор переключения включает:

- Заголовок `switch`, за которым следует вычисляемое выражение (скаляр или строка).
- Произвольное количество групп `case`; Заголовок группы состоят из слова `case`, за которым следует возможное значение выражения, расположенное на одной строке. Последующие строки содержат инструкции, которые выполняются для данного значения выражения. Выполнение продолжается до тех пор, пока не встретится следующий оператор `case` или оператор `otherwise`. На этом выполнение блока `switch` завершается
- Группа `otherwise`. Заголовок включает только слово `otherwise`, начиная со следующей строки размещаются инструкции, которые выполняются, если значение выражения оказалось не обработанным ни одной из групп `case`. Выполнение завершается оператором `end`.

– Оператор `end` является последним в блоке переключателя.

Оператор `switch` работает, сравнивая значение вычисленного выражения со значениями групп `case`. Для числовых выражений оператор `case` выполняется, если <значение>= =<выражение>. Для строковых выражений, оператор `case` истинен, если `strcmp`(значение, выражение) истинно.

Пример:

Рассмотрим оператор `switch` со следующими условиями: он проверяет переменную `input_num`; если `input_num` равно -1, 0 или 1, то операторы `case` выводят на экран соответствующее сообщения. Если значения выражения `input_num` не равно ни одному из этих значений, то выполнение переходит к оператору `otherwise`.

```
switch input_num
    case –1, disp('минус один');
    case 0, disp('ноль');
    case 1, disp('один');
    otherwise, disp('другое значение');
end
```

`while...end` **Оператор цикла с неопределенным числом операций**

Синтаксис:

`while` <*выражение*>

инструкции

`end`

Описание:

Оператор цикла с неопределенным числом операций `while ... end` многократно выполняет инструкцию или группу инструкций, пока управляющее выражение истинно.

Если выражение использует массив, то все его элементы должны быть истинны для продолжения выполнения. Чтобы привести матрицу к скалярному значению, следует использовать функции `any` и `all`.

Пример:

Этот цикл с неопределенным числом операций находит первое целое число n, для которого n! - записывается числом, содержащим 100 знаков:

```
n=1;
while prod(1:n)<1e100,
n=n+1;
end
```

Досрочный выход из `while`-цикла может быть реализован с помощью оператора `break`. Если в операторе `while`, управляющее условие является пустым массивом, то такое условие ложно, то есть оператор вида

```
while A,
S1,
end
```

никогда не выполнит инструкции S1, если A - пустой массив.

`for...end` **Оператор цикла с определенным числом операций**

Синтаксис:

`for` *<перем-я_цикла>=<нач-ное_знач-е>:<приращение>:<конеч-ное_знач-е>*

инструкции

`end`

Описание:

Оператор цикла `for .... end` выполняет инструкцию или группу инструкций предопределенное число раз. По умолчанию приращение равно 1. Можно задавать любое приращение, в том числе отрицательное. Для положительных индексов выполнение завершается, когда значение индекса

превышает <конечное_значение>; для отрицательных приращений выполнение завершается, когда индекс становится меньше чем <конечное_значение>.

Пример:

Этот цикл выполняется пять раз:

```
for i=2:6,
x(i)=2*x(i-1);
end
```

Допустимы вложенные циклы типа:

```
for i=1:m,
   for j=1:n,
A(i,j)=1/(i + j - 1);
   end
end
```

В качестве переменной цикла `for` могут использоваться массивы. Например, рассмотрим массив A размера mxn. Оператор цикла

```
for i = A
    инструкции
end
```

определяет переменную цикла i как вектор A(:, k). Для первого шага цикла k равно 1; для второго k равно 2, и так далее, пока k не достигнет значения n. То есть цикл выполняется столько раз, сколько столбцов в матрице A. Для каждого шага i - это вектор, содержащий один из столбцов массива A.

Для досрочного выхода из цикла можно использовать оператор прерывания `break`. Он часто применяется совместно с условными выражениями. Когда этот оператор встречается в тексте программы, цикл досрочно прекращается и управление передается последующим (стоящим после слова `end`) операторам программы.

Следует помнить, что тело цикла выполняется столько раз, сколько раз меняется переменная цикла. Поэтому следует избегать применения циклов там,

где MATLAB допускает альтернативные операции в векторной или матричной форме. К примеру, рассмотрим фрагмент программы

```
i=0;
for t=0:0.001:1,
    i=i+1;
    y(i)=sin(t);
end
```

его можно заменить на следующий:

```
t=0:0.001:1;
y=sin(y);
```

который не только проще, но и выполняется намного быстрее (примерно в 20-30 раз).

Частные каталоги. Они введены в систему MATLAB, начиная с версии 5.0. Частные каталоги представляют собой подкаталог с именем private родительского каталога. М-файлы частного каталога доступны только М-файлам родительского каталога. Поскольку файлы частного каталога не видимы вне родительского каталога, они могут иметь имена совпадающие, с именами файлов других каталогов системы MATLAB. Это удобно в тех случаях, когда пользователь создает собственные версии некоторой функции, сохраняя оригинал в другом каталоге. Поскольку MATLAB просматривает частный каталог раньше каталогов стандартных функций системы MATLAB, он в первую очередь находит функцию из частного каталога.

3.10 Ввод информации

В процессе выполнения m-файла пользователь может:

- вывести на экран запрос и ввести соответствующую информацию с клавиатуры;
- сделать паузу до нажатия клавиши;
- использовать графический интерфейс пользователя (GUI).

Формирование запроса для ввода с клавиатуры. Функция `input` выводит на экран запрос и ждет ответа пользователя. Ёе синтаксис выглядит следующим образом:

```
n = input('запрос')
txt = input('запрос','s')
```

Функция `input` возвращает введённое с клавиатуры значение. Если пользователь вводит арифметическое выражение, функция вычисляет его и возвращает соответствующее значение. Функция полезна для реализации диалоговых прикладных программ.

Функция `input` может также возвращать не числовое, а строковое выражение, вводимое пользователем. Для ввода символьного выражения необходимо добавить строку 's’ к списку параметров функции:

Пример:

```
name = input('Введите адрес ->','s');
```

Задание паузы. В некоторых случаях целесообразно устанавливать паузу между отдельными шагами алгоритма, например, при выводе графиков.

Команда `pause` <без параметров> останавливает выполнение до тех, пока не будет нажата какая-нибудь клавиша. Чтобы реализовать паузу в n секунд, необходимо применить оператор `pause(n)`.

Выход в оболочку DOS. Для обращения к программам, написанным на языках C или Fortran, можно использовать команду перехода в среду DOS, которая обозначается символ (`!`). Это позволяет выполнять автономную внешнюю программу по аналогии с выполнением M-функции. Такая M-функция с вызовом внешней автономной программы равносильна M-файлу, который реализует следующие условия:

- Сохраняет переменные на диске.
- Выполняет внешнюю программу, которая читает файлы данных, обрабатывает их, и записывает результаты на диск.
- Загружает обрабатываемый файл в рабочую область.

3.11 Повышение эффективности обработки m-файлов

Этот раздел описывает методы повышения быстродействия при выполнении программы и управление памятью:

- векторизация циклов;
- предварительное размещение векторов.

MATLAB - это язык, специально разработанный для обработки массивов и выполнения матричных операций. Всюду, где это возможно, пользователь должен учитывать это обстоятельство.

Векторизация циклов. Под векторизацией понимается преобразование циклов `for` и `while` к эквивалентным векторным или матричным выражениям. При векторизации алгоритма ускоряется выполнение m-файла.

Пример:

Вот один из способов вычислить 1001 значение функции синуса на интервале [0 10], используя оператор цикла:

```
i=0;
for t = 0:.01:10,
    i=i+1;
    y(i)=sin(t);
end
```

Эквивалентная векторизованная форма имеет вид

```
t = 0:.01:10;
y = sin(t);
```

В этом случае вычисления выполняются намного быстрее, и такой подход в системе MATLAB является предпочтительным. Время выполнения этих М-файлов можно оценить, используя команды `tic` и `toc`.

Предварительное выделение памяти. В системе MATLAB есть возможность для существенного сокращения времени выполнения программы за счёт предварительного размещения массивов для выходных данных. Предварительное распределение избавляет от необходимости изменять массив при увеличении его размеров. Используйте соответствующие функции

предварительного выделения памяти, как это показано в таблице для различных типов массивов

Тип массива	Функция	Примеры

Массив чисел	zeros	`y = zeros(1, 100)`
		`for i = 1:100,`
		`  y(i) = det(X^i);`
		`end`

Массив записей	struct	`data=struct([1 3],'x',[1 3],'y',[5 6])`
		`data(3).x = [9 0 2];`
		`data(3).y = [5 6 7];`

Массив ячеек	cell	`B = cell(2, 3)`
		`B{1, 3} = 1:3;`
		`B{2, 2} = 'string'`

Предварительное выделение памяти позволяет избежать фрагментации памяти при работе с большими матрицами. В ходе сеанса работы системы MATLAB, память может стать фрагментированной из-за работы механизмов динамического распределения и освобождения памяти. Это может привести к появлению большого количества фрагментов свободной памяти, и непрерывного пространства памяти может оказаться недостаточно для хранения какого-либо большого массива.

Предварительное выделение памяти позволяет также определить непрерывную область, достаточную для проведения всех вычислений

Переменные и память. Память выделяется для переменной всякий раз, когда происходит присваивание этой переменной какого-то значения (т.е. во время обращения к этой переменной).

Для экономии памяти надо:

- избегать использовать одни и те же переменные в качестве входных и выходных аргументов функции, поскольку они будут передаваться ссылкой;
- после использования переменной целесообразно либо присвоить ей пустой массив, либо удалить с помощью команды `clear` <имя_переменной>;
- стремиться использовать переменные повторно.

Глобальные переменные. При объявлении глобальной переменной в таблицу переменных просто помещается флаг. При этом не требуется дополнительной памяти.

Например, последовательность операторов

```
a=5;
global a;
```

определяет переменную `a` как глобальную и формируется дополнительная копия этой переменной.

Функция `clear a` удаляет переменную `a` из рабочей области системы MATLAB, но сохраняет её в области глобальных переменных.

Функция `clear global a` удаляет переменную `a` из области глобальных переменных.

3.12 Работа с графическими средствами системы MATLAB

Одно из достоинств системы MATLAB – обилие средств графики, начиная от команд построения простых графиков функций одной переменной в декартовой системе координат и кончая комбинированными и демонстрационными графиками с элементами анимации. Особое внимание в системе уделено трехмерной графике с функциональной окраской отображаемых фигур.

Пользователь может на одном графике строить несколько кривых, выделяя их различными цветами или различными метками. Графики могут иметь линейную, логарифмическую или полулогарифмическую шкалу. По умолчанию устанавливается режим автомасштабирования графиков, так что они имеют максимально большие размеры. Отметим основные операторы графики в системе MATLAB:

- `plot` – построение графика в линейном масштабе;
- `loglog` – построение графика в логарифмическом масштабе;
- `semilogx` – построение графика в полулогарифмическом масштабе – логарифмический масштаб по оси *x*;

- `semilogy` – построение графика в полулогарифмическом масштабе – логарифмический масштаб по оси *y*;
- `polar` - построение графика в полярной системе координат;
- `mesh` - построение графика трехмерной поверхности в виде сетки;
- `surf` – построение графика функции двух переменный в виде гладкой ("залитой")поверхности ;
- `contour` – построение графика с контурными линиями уровнями равных высот для функции двух переменных;
- `bar` или `hist` – построение графика столбцовой гистограммы.

Для окончательного оформления графиков необходимо предусмотреть нанесение надписей по осям координат, титульной надписи и надписей с произвольной ориентацией относительно графиков. Для этой цели служат следующие операторы:

- `text` – вывод надписи в заданное место графика;
- `title` – задание титульной надписи для графика;
- `xlabel` - задание надписи по оси *x***;**
- `ylabel` – задание надписи по оси y;
- `grid on/off` – включение/выключение масштабной сетки.

Ниже рассмотрим применение графических операторов на ряде конкретных примеров.

2D графика

Функции одной переменной $y(x)$ находят широкое применение в практике математических и инженерных расчетов, а также при компьютерном математическом моделировании. Для отображения таких функций часто используют прямоугольную декартовую систему координат.

Команда plot служит для построения графиков функции одной переменной в декартовой системе координат. Эта команда имеет ряд параметров, рассматриваемых ниже

plot **построение графика функции одной переменной**

Синтаксис: `plot(X,Y,S)`

Строит график функции $y(x)$, координаты точек (x, y) которой берутся из векторов одинакового размера X и Y. Если X или Y – матрица, то строится семейство графиков по данным, содержащимся в матрице. Тип линии графика можно задавать с помощью строковой константы S.

Значениями константы S могут быть следующие символы:

Константа S	Цвет линии
y	желтый
m	фиолетовый
b	синий
r	красный
g	зеленый
w	белый
k	черный
c	голубой

Константа S	Тип точки
.	точка
o	окружность
x	крест
+	плюс
*	звездочка

Константа S	Тип линии
-	сплошная
:	пунктирная
-.	штрих-пунктирная
--	Штриховая

Таким образом, с помощью строковой константы S можно менять цвет линии, представлять узловые точки различными метками и менять тип линии графика. Если в команде plot отсутствует указание на цвет линий и точек, то он выбирается автоматически из таблицы цветов. Если линий больше шести, то выбор цветов повторяется.

Пример: Построение графика функции $y=\sin(x)$

```
»% график функции sin(x)
» x=0:0.01:2*pi; % формирование вектора x
» plot(x,sin(x),'-w');
```

Команда `plot(Y)` строит график *y*(*i*), где значения *y* берутся из вектора Y, а *i* представляет собой индекс соответствующего элемента. Если Y содержит комплексные числа, то выполняется команда `plot(real(Y),imag(Y))`. Во всех других случаях мнимая часть данных игнорируется.

Пример: построение фигуры Лиссажу для кратности частот 1/3.

```
% построение фигуры Лиссажу
x=0:0.02*pi:2*pi;
k1=1;
k2=3;
y=sin(k1*x)+i*cos(k2*x);
plot(y);
axis([-1.2 1.2 -1.2 1.2]); % задание масштаба осей координат
```

Результат выполнения этой программки показан на рис. 3.3.

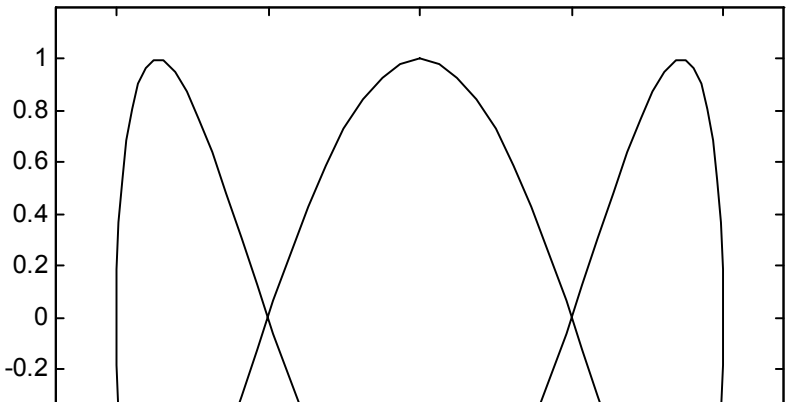


Рисунок 3.3 – Фигура Лиссажу для соотношения частот 1/3

Система MATLAB позволяет на одном графике строить несколько кривых одновременно:

`plot(X1,Y1,S1,X2,Y2,S2,X3,Y3,S3,…)` – эта команда строит на одном графике ряд линий, представленных данными вида (X_i,Y_i,S_i), где X_i,Y_i – векторы значений функций, S_i – строковые константы, определяющие вид графика.

Пример: рассмотрим пример построения графика нескольких функций с различным стилем представления каждой

```
% графики двух функций со спецификацией линий каждого графика
a=-2*pi; b=2*pi; % интервал по оси x построения графиков
N=100; % число точек линий
N1=10; % число меток
delx=(b-a)/(N-1); % расчет шага для расчета x
x=a:delx:b; % формирование вектора x
delx1=(b-a)/(N1-1);
x1=a:delx1:b; % формирование вектора x для меток
y1=sin(x); y1m=sin(x1); % расчет первой функции
y2=sin(x)./x; y2m=sin(x1)./x1; % расчет второй функции
plot(x,y1,'-.w',x1,y1m,'ow',x,y2,'-b');
```

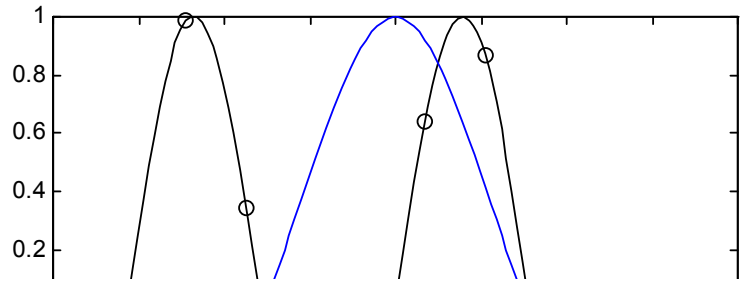


Рисунок 3.4 – Пример построения нескольких графиков в одном графическом окне

Оформление и комбинирование графиков

После того, как график уже построен, MATLAB позволяет выполнить его дополнительное форматирование или оформление в нужном виде. Соответствующие операторы рассмотрены ниже.

Установка титульной надписи и подписей на осях. Для установки над графиком титульной надписи используется команда `title`.

Синтаксис: `title('string')` – установка над графиком надписи, заданной строковой переменной `string`.

Для установки подписей возле осей *x*, *y* или *z* используются команды:

```
xlabel('string')
ylabel('string')
zlabel('string')
```

подпись также определятся строковой переменной `string`.

Ввод текста в графическом окне. Часто возникает необходимость добавления текста в определенное место графика, например для обозначения той или иной кривой графика. Для этого используются команды `text` и `gtext`:

`text` **ввод текста в поле графика**

Ситнаксис: `text(X,Y,'string')`

`text(X,Y,Z,'string')`

добавляет на график (двухмерный или трехмерный) текст, заданный строкой 'string', так что начало текста расположено в точке с координатами (X,Y,Z).

`gtext` **ввод текста в поле графика с помощью мыши**

Ситнаксис: `gtext('string')`

добавляет на график (двухмерный или трехмерный) текст, заданный строкой 'string'. Место вывода текста определяется с помощью мыши.

Управление свойствами осей координат. Обычно графики в системе MATLAB строятся в режиме автоматического масштабирования. Используя команду axis можно управлять формой осей координат и масштабом.

`axis` **установка масштаба и типа осей координат**

Ситнаксис:

```
axis([xmin xmax ymin ymax])
axis([xmin xmax ymin ymax zmin zmax])
axis('auto')
axis(axis)
V=axis
axis('square')
axis('equal')
axis('image')
axis('normal')
axis('off') или axis off
axis('on') или axis on
```

Описание:

- `axis([Xmin Xmax Ymin Ymax])` – ручная установка масштаба по осям x и y для текущего двухмерного графика;
- `axis([Xmin Xmax Ymin Ymax Zmin, Zmax])` – ручная установка масштаба по осям x и y для текущего трехмерного графика;
- `axis auto` - включение режима автоматического масштабирования (стоит по умолчанию);
- `axis('square')` – устанавливает текущие оси координат в виде квадрата (или куба в трехмерном случае);
- `axis('equal')` – включает масштаб с одинаковым расстоянием между метками по осям *x*, *y* и *z*;

- `axis('image')` – команда устанавливает масштаб, который обеспечивает квадратные размеры пикселей;
- команда `axis('normal')` восстанавливает полноразмерный масштаб, отменяя масштабы, установленные командами `axis('square')` и `axis('equal')` – данный режим стоит по умолчанию;
- `axis(axis)` – фиксирует текущие значения масштабов для последующих графиков, как если бы был включен режим hold;
- функция `V=axis` возвращает вектор-строку масштабов по осям для активного графика. Если график двумерный, то V имеет 4 компонента; если трехмерный - 6 компонентов;
- `axis('off')` – выключает отображение осей, их обозначения и маркеры на экране;
- `axis('on')` – включает отображение осей, их обозначения и маркеры на экране (стоит по умолчанию).

`grid` **нанесение масштабной сетки**

Синтаксис:

```
grid on
grid off
grid
```

Описание:

Команда `grid on` наносит координатную сетку на текущие оси.

Команда `grid off` удаляет координатную сетку.

Команда `grid` выполняет роль переключателя с одной функции на другую.

`hold` **управление режимом сохранения текущего графического окна**

Синтаксис:

```
hold on
```

```
hold off
hold
```

Описание:

Команда `hold on` включает режим сохранения текущего графика и свойств объекта axes, так что последующие команды приведут к добавлению новых графиков в графическом окне.

Команда `hold off` выключает режим сохранения графика.

Команда `hold` реализует переключение от одного режима к другому.

Пояснение:

Команды `hold` воздействуют на значения свойства NextPlot объектов figure и axes:

- `hold on` присваивает свойству NextPlot для текущих объектов figure и axes значение add;
- `hold off` присваивает свойству NextPlot для текущих объектов figure и axes значение replace.

`subplot` **разбиение графического окна**

Синтаксис:

```
subplot(m, n, p)
subplot(h)
subplot(mnp)
```

Описание:

Данная команда выполняется перед обращением к функциям построения графиков для одновременной выдачи нескольких графиков в различных частях графического окна.

Команды `subplot(mnp)` или `subplot(m, n, p)`, где `mnp` - 3 цифры, производит разбивку графического окна на несколько подокон, создавая при этом новые объекты axes; значение m указывает, на сколько частей разбивается окно по горизонтали, n - по вертикали, а p - номер подокна,

куда будет выводиться очередной график. Эти же команды могут использоваться для перехода от одного подокна к другому.

Команды `clf, subplot(111), subplot(1,1,1)` выполняют одну и ту же функцию - удаляют все подокна и возвращают графическое окно в штатное состояние.

Пример: В верхней части экрана строится функция y1 = sin(x), в нижней - y2 = log(abs(y)).

```
x = -1:.1:1;
y1 = sin(x);
subplot(2, 1, 1), plot(x, y1)
y2 = log(abs(y1));
subplot(2, 1, 2), plot(x, y2)
```

`colormap` **Указание палитры цветов**

Синтаксис:

```
colormap(C)
colormap('default')
C = colormap
caxis(caxis)
```

Описание:

Палитра цветов C - это матрица размера mx3 действительных чисел из диапазона [0.0 1.0]. Строка k палитры сформирована из трех чисел, которые указывают интенсивность красного, зеленого и синего цветов, то есть C(k,:)=[r(k) g(k) b(k)].

Команда `colormap(C)` устанавливает палитру согласно матрице C. Если значение элемента матрицы выходит за пределы интервала [0 1], выдается сообщение об ошибке

```
Colormap must have values in [0,1].
```

Значения элементов палитры должны быть в интервале [0 1].

Команды `colormap('default')` или `colormap(hsv)` устанавливают штатную палитру, которая соответствует модели hue-saturation-value (оттенок-насыщенность-значение). Последовательность цветов этой палитры соответствует цветам радуги.

red	Красный	Каждый
-	-	охотник
yellow	Желтый	желает
green	Зеленый	знать,
cyan	Голубой	где
blue	Синий	сидит
magenta	Фиолетовый	фазан

В рамках системы MATLAB реализованы следующие палитры:

bone	Grey-scale with tinge of blue color map	Серая палитра с оттенком синего
cool	Shades of cyan and magenta color map	Палитра с оттенками голубого и фиолетового
copper	Linear copper-tone color map	Линейная палитра в оттенках меди
flag	Alternating red, white, blue, and black color map	Палитра с чередованием красного, синего и черного
gray	Linear grey-scale color map	Линейная палитра в оттенках серого
hot	Black-red-yellow-white color map	Палитра с чередованием черного, красного, желтого и белого
hsv	Hue-saturation-value color map	Палитра радуги
jet	Variant of hsv	Разновидность hsv-палитры
pink	Pastel shades of pink color map	Розовая палитра с оттенками пастели
prism	Prism (red-orange-yellow-green -blue-violet) color map	Палитра с чередованием красного, оранжевого, желтого, зеленого, синего и фиолетового
white	All white color map	Белая палитра

Специальные графики

Система MATLAB также позволяет выполнять построение графиков функции одной переменной в специфической форме: в полярной системе координат, в логарифмическом масштабе, построение гистограмм и столбцовых диаграмм, векторные диаграммы и пр. Рассмотрим несколько примеров.

`loglog` **график функции в логарифмическом масштабе**

Синтаксис:

```
loglog(x, y)
loglog(x, y, s)
loglog(x1, y1, s1, x2, y2, s2, ...)
```

Описание:

Команды `loglog(...)` равносильны функциям `plot`, за исключением того, что они используют по обеим осям логарифмический масштаб вместо линейного.

Пример: Построим график y = exp(x) в логарифмическом масштабе:

```
x = logspace(-1, 2);
loglog(x, exp(x))
grid on
```

`semilogx semilogy` **график функции в полулогарифмическом масштабе**

Синтаксис:

```
semilogx(x, y)                    semilogy(x, y)
semilogx(x, y, s)                 semilogy(x, y, s)
semilogx(x1,y1,s1,x2,y2,s2,...)   semilogy(x1,y1,s1,x2,y2,s2,...)
```

Описание:

Команды `semilogx(...)` используют логарифмический масштаб по оси x и линейный масштаб по оси y.

Команды `semilogy(...)` используют логарифмический масштаб по оси y и линейный масштаб по оси x.

Пример: Построим график y = exp(x) в полулогарифмическом масштабе по оси y:

```
x = 0:0.1:100;
semilogy(x, exp(x))
grid on
```

`polar` **график функции в полярной системе координат**

Синтаксис:

polar(phi, rho)

polar(phi, rho, s)

Описание:

Команды `polar(...)` реализуют построение графиков в полярных координатах, задаваемых углом phi и радиусом pho.

Пример: Построим график функции rho=sin(2*phi)*cos(2*phi) в полярных координатах

```
phi = 0:0.01:2*pi;
polar(phi,sin(2*phi).*cos(2*phi));
```

Результат представлен на рис. 3.5.

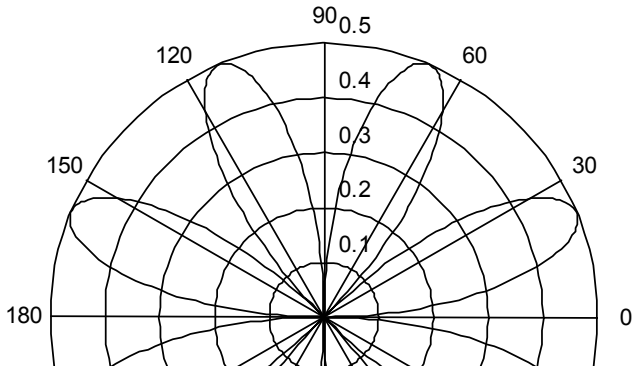


Рисунок 3.5 – График функции sin(x)*cos(x) в полярной системе координат

`bar` **график функции в виде столбцовой диаграммы**

Синтаксис:

```
bar(y)bar(x, y)
[xb,yb]=bar(...)
bar(y,'<тип_линии>')
bar(x,y,'<тип_линии>')
```

Описание:

Команда `bar(y)` выводит график элементов одномерного массива y в виде столбцовой диаграммы.

Команда `bar(x, y)` выводит график элементов массива y в виде столбцов в позициях, определяемых массивом x, элементы которого должны быть упорядочены в порядке возрастания.

Если X и Y - двумерные массивы одинаковых размеров, то каждая диаграмма определяется соответствующей парой столбцов и они надстраиваются одна над другой.

Команды `bar(y,'<тип_линии>')`, `bar(x,y,'<тип_линии>')` позволяют задать тип линий, используемых для построения столбцовых диаграмм, по аналогии с командой `plot`.

Функция `[xb, yb] = bar(...)` не выводит графика, а формирует такие массивы xb и yb, которые позволяют построить столбцовую диаграмму с помощью команды `plot(xb, yb)`.

Пример: Построить график функции $y = \mathrm{e}^{-x^2}$ в виде столбцовой диаграммы.

```
x=-2.9:0.2:2.9;
bar(x, exp(-x.*x));
```

`hist` **Построение гистограммы**

Синтаксис:

```
hist(y)
hist(y, n)
hist(y, x)
[y, x] = hist(y, ...)
```

Описание:

Команды `hist(...)` подсчитывают и отображают на графике количество элементов массива y, значения которых попадают в заданный интервал; для этого весь диапазон значений y делится на n интервалов (по умолчанию 10) и подсчитывается количество элементов в каждом интервале.

Команда `hist(y)` выводит гистограмму для 10 интервалов.

Команда `hist(y, n)` выводит гистограмму для n интервалов.

Команда `hist(y,x)` выводит гистограмму с учетом диапазона изменения переменной *x*. Здесь x должен быть вектором.

Функции `[y,x] = hist(y, ...)` формируют такие массивы y и x, что `bar(x, y)` является гистограммой.

Пример: Построить гистограмму для 10 000 случайных чисел, распределенных по нормальному и равномерному законам.

```
y1= randn(10000,1);
y2=rand(10000,1);
subplot(211);
hist(y1, 20);
subplot(212);
hist(y2,20)
```

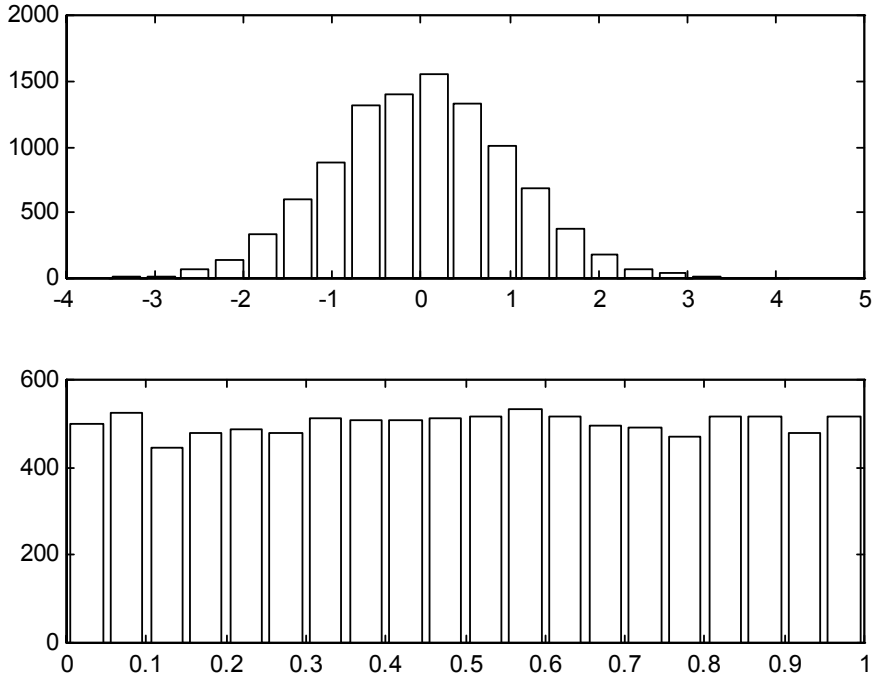


Рисунок 3.6 – Гистограммы нормального и равномерного распределений случайной величины

3D графика

В системе MATLAB предусмотрено несколько команд и функций для построения трехмерных графиков. Значения элементов числового массива рассматриваются как *z*-координаты точек над плоскостью, определяемой координатами *x* и *y*. Возможно несколько способов соединения этих точек. Первый из них - это соединение точек в сечении (функция `plot3`), второй - построение сетчатых поверхностей (функции `mesh` и `surf`). Поверхность, построенная с помощью функции `mesh`, - это сетчатая поверхность, ячейки которой имеют цвет фона, а их границы могут иметь цвет, который определяется свойством EdgeColor графического объекта surface. Поверхность, построенная с помощью функции `surf`, - это сетчатая поверхность, у которой может быть задан цвет не только границы, но и ячейки; последнее управляется свойством FaceColor графического объекта surface. Заинтересованному читателю рекомендуем обратиться к документации по системе MATLAB.

Создание массивов данных для трехмерной графики. Трехмерные поверхности обычно описываются функцией двух переменных $z=f(x, y)$. Специфика построения трехмерных графиков требует не просто задания ряда значений x и y, то есть векторов X и Y. Она требует определения для координат x и y специальных двухмерных массивов – матриц. Для создания таких массивов служит функция `meshgrid`. Она используется в основном совместно с функциями построения графиков трехмерных поверхностей и фигур.

`meshgrid` **Формирование двумерных массивов X и Y для построения 3-D графиков**

Синтаксис:

```
[X,Y]=meshgrid(x, y)
[X,Y]=meshgrid(x)
```

Описание:

Функция `[X,Y]=meshgrid(x,y)` задает сетку на плоскости x-y в виде двумерных массивов X, Y, которые определяются одномерными массивами x и y. Строки массива X являются копиями вектора x, а столбцы - копиями вектора y. Формирование таких массивов упрощает вычисление функций двух переменных, позволяя применять операции над массивами.

Функция `[X,Y]=meshgrid(x)` представляет собой упрощенную форму записи для функции `[X,Y] = meshgrid(x, x)`.

Пример:

```
» [X,Y]=meshgrid(1:4,-3:3)
X =
     1     2     3     4
     1     2     3     4
     1     2     3     4
     1     2     3     4
     1     2     3     4
     1     2     3     4
     1     2     3     4
```

```
Y =

    -3    -3    -3    -3
    -2    -2    -2    -2
    -1    -1    -1    -1
     0     0     0     0
     1     1     1     1
     2     2     2     2
     3     3     3     3
```

Используя полученные массивы X и Y, далее осуществляют расчет функции $z=f(x, y)$ для выполнения построения графика. График трехмерной функции по полученной матрице z можно строить разными операторами.

`Plot3` **Построение линий и точек в трехмерном пространстве**

Синтаксис:

```
plot3(x, y, z)
plot3(x, y, z, s)
plot3(x1, y1, z1, s1, x2, y2, z2, s2, ...)
```

Описание:

Команды `plot3(...)` являются трехмерными аналогами функции `plot(...)`.

Команда `plot3(x,y,z)`, где x, y, z - одномерные массивы одинакового размера, строит точки с координатами x(i), y(i), z(i) и соединяет их прямыми линиями.

Команда `plot3(x,y,z,S)` позволяет выделить график функции $z=f(x, y)$, указав способ отображения линии, способ отображения точек, цвет линий и точек с помощью строковой переменной S.

Команда `plot3(x1,y1,z1,s1,x2,y2,z2,s2, ...)` позволяет объединить на одном графике несколько функций z1(x1, y1), z2(x2, y2), ..., определив для каждой из них свой способ отображения.

Обращение к команде `plot3` вида `plot3(x,y,z,s1,x,y,z,s2)` позволяет для графика $z=f(x, y)$ определить дополнительные свойства, для

указания которых применения одной строковой переменной s1 недостаточно, например при задании разных цветов для линии и для точек на ней.

Пример: Построим график функции $z = xe^{(-x^2-y^2)}$ в трехмерном пространстве (см. рис. 3.7).

```
[X,Y]=meshgrid([-2:0.1:2]);    % построение сетки значений для x и y
Z=X.*exp(-X.^2-Y.^2); % расчет функции
plot3(X, Y, Z)
```

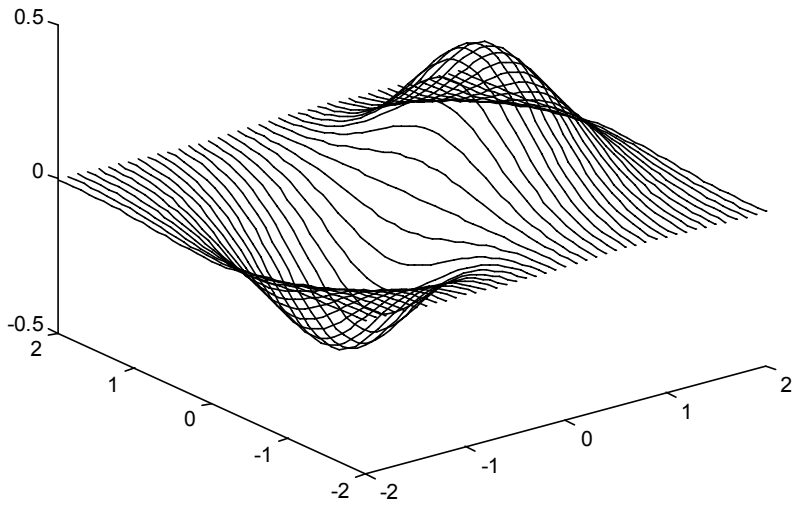


Рисунок 3.7 – График функции $z = xe^{(-x^2-y^2)}$ построенный с помощью оператора plot3

`mesh, meshc`

Трехмерная сетчатая поверхность

Синтаксис:

`mesh(X,Y,Z,C)`	`meshc(X,Y,Z,C)`
`mesh(x,y,Z,C )`	`meshc(x,y,Z,C )`
`mesh(Z,C)`	`meshc(Z,C)`
`mesh(X,Y,Z)`	`meshc(X,Y,Z)`
`mesh(x,y,Z)`	`meshc (x,y,Z)`
`mesh(Z)`	`meshc(Z)`

Описание:

Команда `mesh(X,Y,Z,C)` выводит на экран сетчатую поверхность для значений массива Z, определенных на множестве значений массивов X и Y. Цвета узлов поверхности задаются массивом C. Цвета ребер определяются свойством EdgeColor объекта surface. Можно задать одинаковый цвет для всех ребер, определив его в виде вектора [r g b] интенсивности трех цветов - красного, зеленого, синего. Если определить спецификацию none, то ребра не будут прорисовываться. Если определить спецификацию flat, то цвет ребер ячейки определяется цветом того узла, который был первым при обходе этой ячейки. Поскольку одни и те же ребра обходятся несколько раз, то цвета будут замещаться. Если определить спецификацию interp, то будет реализована линейная интерполяция цвета между вершинами ребра.

Применение функции shading после обращения к функции `mesh` изменяет спецификации свойств EdgeColor и FaceColor согласно следующей таблице.

Свойство	Применяемая функция		
	mesh	shading flat	shading interp
EdgeColor	flat	flat	interp
FaceColor	Цвет фона	Цвет фона	Цвет фона

Команда `mesh(x,y,Z,C)` выполняет ту же функцию, но вместо двумерных массивов X, Y использует одномерные вектора, такие что если `length(x)=n`, а `length(y)=m`, то `[m,n]=size(Z)`. В этом случае узлы сетчатой поверхности определяются тройками {x(j), y(i), Z(i, j)}, где вектор x определяет столбцы массива Z, а y - строки.

Команда `mesh(Z,C)` использует сетку, которая определяется одномерными массивами x = 1:n и y=1:m.

Команды `mesh(X, Y, Z)`, `mesh(x,y,Z)`, `mesh(Z)` используют в качестве массива цвета C=Z, то есть цвет в этом случае пропорционален высоте поверхности.

Группа команд `meshc(...)` в дополнение к трехмерным поверхностям строит проекцию линий постоянного уровня.

Пример: Построим трехмерную поверхность функции $z = \sin(\sqrt{x^2 + y^2}) / (\sqrt{x^2 + y^2})$.

```
[X,Y ] = meshgrid([-8 :0.5:8]);
R=sqrt(X.^2+Y.^2)+eps;
Z=sin(R)./R;
mesh(X,Y,Z)
```

Результат выполнения данного фрагмента программы представлен на рис. 3.8.

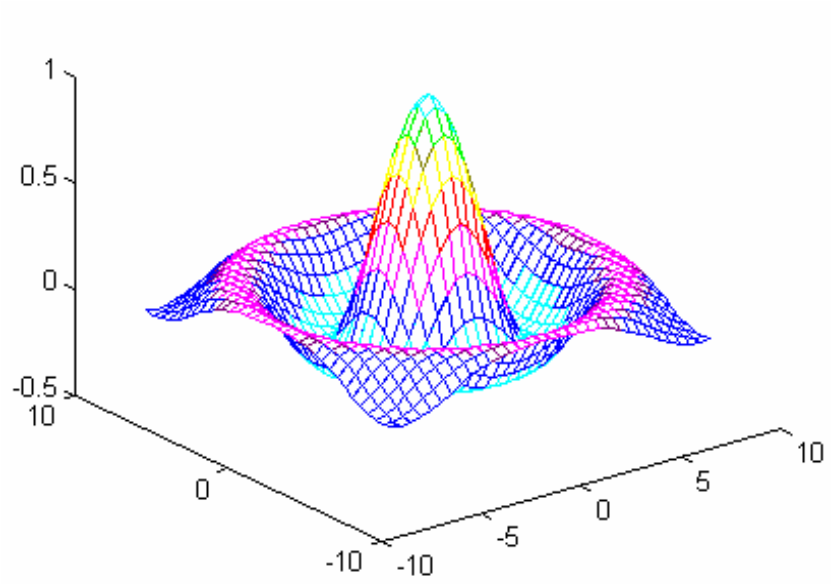


Рисунок 8 – График трехмерной поверхности, выполненный с помощью функции `mesh`

`colorbar` **Шкала палитры для трехмерного графика**

Синтаксис:

```
colorbar('vert')
colorbar('horiz')
colorbar(h)
colorbar
```

Описание:

Команда `colorbar('vert')` добавляет к текущему графику вертикальную шкалу палитры.

Команда `colorbar('horiz')` добавляет к текущему графику горизонтальную шкалу палитры.

Команда `colorbar(h)` добавляет к графику с дескриптором h шкалу палитры. Размещение шкалы реализуется автоматически в зависимости от соотношения ширины и высоты графика.

Команда `colorbar` без аргументов размещает на текущем графике новую вертикальную шкалу палитры или обновляет прежнюю.

`contour` **Изображение линий уровня для трехмерной поверхности**

Синтаксис:

`contour(Z)`	`contour(x, y, Z)`
`contour(Z, n)`	`contour(x, y, Z, n)`
`contour(Z, v)`	`contour(x, y, Z, v)`
`contour(...,'тип_линии')`	
`C = contour(...)`	
`[C, h] = contour(...)`	

Описание:

Команда `contour(Z)` рисует двумерные линии уровня для массива данных Z, определяющих поверхность в трехмерном пространстве без учета диапазона изменения координат x и y.

Команда `contour(x,y,Z)`, где x и y - векторы, рисует линии уровня для массива данных Z с учетом диапазона изменения координат x и y.

Команды `contour(Z,n)`, contour(x, y, Z, n) рисует n линий уровня для массива данных Z; по умолчанию, n равно 10.

Команды `contour(Z,V)`, `contour(x,y,Z,V)` рисуют линии уровня для заданных значений, которые указаны в векторе V.

Команда `contour(...,'<тип_линии>')` рисует линии уровня, тип и цвет которых определяются параметром <тип_линии> команды `plot`.

Функция `C=contour(...)` возвращает массив C описания линий уровней по аналогии с функцией `contourc` для последующего использования командой `clabel`.

Функция `[C, h]=contour(...)` возвращает массив C и вектор-столбец дескрипторов h графических объектов `line` для каждой линии уровня.

Пример: Построить линии уровня функции $z = xe^{(-x^2-y^2)}$.

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

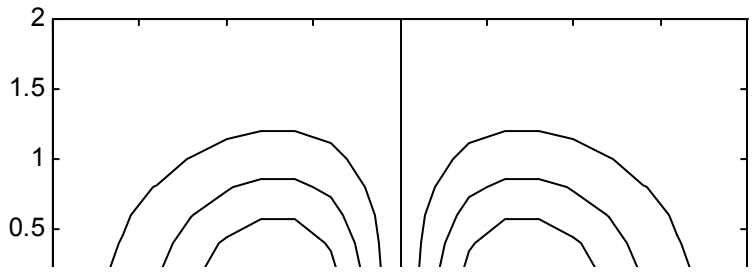


Рисунок 3.9 – Линии уровня функции $z = xe^{(-x^2-y^2)}$

Диагностические сообщения: Если меньшая размерность массива Z оказывается меньше двух, то выдается сообщение

```
Matrix must be 2-by-2 or larger.
```

Матрица должна иметь размер 2x2 или больше.

Ограничения: При работе с командой и функцией `contour` предполагается, что элементы массивов x и y монотонно возрастают.

3.13 Заключение

Данное руководство не является полным описанием системы MATLAB. Цель, которую преследовал автор – познакомить студентов с системой, показать основные приемы работы и программирования в ней. Для более глубокого изучения MATLAB следует обращаться к документации и дополнительной литературе. Следует отметить, что в последнее время MATLAB завоевывает все большую популярность среди пакетов для математических и инженерных расчетов в России. Это подтверждает появление большого числа публикация и книг о MATLAB и различных прикладных пакетах, входящих в его состав.

3.14 Список дополнительных источников по системе MATLAB

1. www.mathworks.com - сайт фирмы-разработчика системы MATLAB. Содержит последние новости о системе и прикладных пакетах, имеет архив свободно распространяемых программ на языке системы MATLAB.
2. www.matlab.ru - сайт пользователей системы MATLAB в нашей стране, имеет ссылки на другие ресурсы, посвященный MATLAB.
3. www.exponenta.ru - очень полезный сайт для начинающих и уже продвинутых пользователей MATLAB. Здесь можно найти книгу и справочник по операторам языка MATLAB Потемкина В.Г. Сайт содержит полезную информацию и по другим математическим пакетам Maple, Mathematica, MathCAD и пр. Также есть ссылки на другие полезные ресурсы.
4. Дьяконов В.П. Справочник по применению системы PC MathLAB. – М.: Наука, ФизМфаЛит, 1993.
5. Потемкин В.Г. MATLAB - справочное пособие – М.: ДИАЛОГ-МИФИ, 1997 – подробный справочник по основным функциям системы MATLAB.
6. Потемкин В.Г. MATLAB 5 для студентов – М.: ДИАЛОГ-МИФИ, 1998.
7. Потемкин В.Г. Система инженерных и научных расчетов MATLAB 5.x в 2-х т. – М.: ДИАЛОГ-МИФИ, 1999.
8. Дьяконов В.П. MATLAB: учебный курс. – СПб: ПИТЕР, 2001. – полезная книга для начинающих пользователей. Курс изучения системы разбит на 20 уроков по разным темам.
9. Потемкин В.Г. Инструментальные средства MATLAB 5.x – М.: ДИАЛОГ-МИФИ, 2000. – книга посвящена вопросам создания интерфейсов в среде MATLAB на основе технологии GUI и созданию самостоятельно выполняемых приложений. Рекомендуется для тех, кто уже знаком с основами программирования в среде пакета MATLAB.

4 Лабораторные работы

4.1 Введение

Целью данного курса лабораторных работ является закрепление на практике теоретических знаний полученных в ходе изучение дисциплины «Вычислительная математика».

Предполагается, что студенты уже освоили базовые методы вычислительной математики и имеют опыт программирования на алгоритмических языках. Весь курс работ поставлен в среде системы для математических и инженерных расчетов MATLAB. Использование данной системы обусловлено тем, что она позволяет отвлечься от чистого программирования и сосредоточить внимание на использовании методов вычислительной математики для решения конкретных задач. В своем составе MATLAB содержит большое число самых разнообразных подпрограмм для решения широкого круга задач различного назначения: задачи линейной алгебры, решение систем уравнений (линейных и нелинейных), приближение функций и аппроксимация данных, численное дифференцирование и интегрирование, задачи одномерной и многомерной оптимизации и многое другое.

Курс лабораторных работ начинается с изучения самой системы и освоения языка программирования MATLAB. Вторая лабораторная работа посвящена изучению линейной интерполяции функций с помощью полиномов. В третьей лабораторной работе студенты должны применить метод наименьших квадратов для выравнивания экспериментальных результатов.

В ходе выполнения каждой работы оформляется отчет, который должен содержать теоретические сведения, разработанный алгоритм, текст программы, результаты тестирования (вычислительного эксперимента) и выводы по полученным результатам.

4.2 Лабораторная работа № 1. Знакомство с пакетом для математических и инженерных расчетов MATLAB

Вводные замечания

Назначение данной работы – знакомство с системой для математических и инженерных расчетов MATLAB. Студенты должны научиться работать с системой в режиме прямых вычислений и освоить принципы программирования на внутреннем языке MATLAB.

Своим названием (MATrix LABoratory) система MATLAB обязана ориентации на матричные и векторные операции. Система содержит также средства работы с графикой, методы численного анализа и оптимизации и многое другое.

Система MATLAB может работать как в режиме непосредственных вычислений (диалоговый режим), так и в режиме операционной среды, т.е. программу на языке MATLAB можно запускать на выполнение из командного окна среды. Первое знакомство с системой можно получить, запустив в командном окне MATLAB файл demo.m, который показывает основные возможности системы. Для этого, получив приглашение к вводу (символ »), необходимо набрать имя m-файла `demo` и нажать Enter.

Файлы, содержащие команды на языке MATLAB имеют расширение *.m. Они бывают двух типов: script-файлы и функции. Script-файл представляет собой самостоятельную программу, написанную на языке системы MATLAB. Функции (function) представляют собой встроенные в систему или написанные пользователем подпрограммы, которые вызываются обращением к ним по имени с указанием входных аргументов и возвращают вектор, матрицу, скалярное или символьное значение. Информацию по каждой из внутренних функций системы MATLAB можно получить с помощью команды `help` <имя функции>. Команда `help` без параметра выдает информацию о всех функциях системы.

Задание на лабораторную работу № 1

I часть – Изучение команд и операторов MATLAB

Ознакомиться с основными командами системы MATLAB:

Справочные команды - DEMO, WHATSNEW,VERSION, HELP, HELPWIN,

Управление командным окном – CLC, CLF, CLEAR, CD, DIR HOME, PAUSE, DISP, PATH, QUIT, FLOATS.

Арифметические и логические операторы – +, -, *, /, \, ^, ’, <, >, <=, >=, = =, ~ =, &, |, ~, ANS, PI, I, J.

Операторы ввода, вывода данных и операторы циклов – INPUT, PAUSE, ERROR, FPRINTF, SPRINTF, FOR…END, WHILE…END, IF…ELSE…END, BREAK, MENU.

Основные команды для работы с графикой – FIGURE, CLOSE, HOLD, SUBPLOT, PLOT, LOGLOG, POLAR, SEMILOGX,SEMILOGY, MESHGRID, PLOT3, MESH,SURF, CONTOUR.

Оформление графиков – GRID, TEXT, TITLE, XLABEL, YLABEL, GTEXT, CLABEL, AXIS.

Замечания:

а) В системе MATLAB реализовано два типа арифметических операций:

- операции над матрицами и векторами в соответствии с правилами линейной алгебры (матричные операции);
- поэлементные операции над массивами.

Чтобы различать эти операции, перед знаком поэлементной операции ставится точка (для операторов *, /, \, ^).

Пример:

1) X=A./B – поэлементное деление матрицы A на матрицу B. Размеры матриц Аи B должны быть одинаковыми. Результатом является массив с элементами $x_{ij} = \frac{a_{ij}}{b_{ij}}$.

2) X=A/B – деление матрицы A на матрицу B по правилам линейной алгебры (это эквивалентно $X=A\cdot B^{-1}$, где B^{-1} – обратная матрица).

б) Операции отношения и логические операторы выполняют поэлементное сравнение двух массивов. Они возвращают в качестве результата массив того же размера, элементы которого равны единице, если результат сравнения или выполнения логической операции равен истине, и нулю – в противном случае.

в) Логические операторы &, |, ~ соответствуют операторам булевой алгебры AND, OR, NOT.

II часть – Работа с системой MATLAB в режиме непосредственных вычислений

Запустить файл `demo.m` и ознакомится с основными возможностями системы MATLAB. Проделать в режиме непосредственных вычислений следующие операции:

1) Работа с матрицами и векторами.

- Ввод векторов, матриц, строковых переменных (из командной строки и с помощью команды INPUT).
- Вывод произвольного элемента матрицы (вектора) или нескольких произвольных строк или столбцов. Формирование матрицы меньшего размера из исходной матрицы.
- Операции матричного сложения, умножения, деления (умножения на обратную матрицу), возведения в степень, вычисление определителя матрицы, нахождение обратной матрицы, транспонирование матрицы.
- Операции поэлементного умножения, деления, возведения в степень векторов и матриц.
- Решение систем линейных уравнений вида AX=B с помощью стандартного решателя системы (использование оператора \).

2) Работа с функциями.

Для функции $f(x)$ (см. таблицу вариантов заданий в приложении А) в указанных пределах выполнить следующие действия:

- Создать файл-функцию (m-файл типа function).
- Построить график $y = f(x)$. Добавить подписи к осям и заголовок графика.
- Вычислить интеграл (с помощью функции QUAD) на указанном промежутке.
- Построить 3D-график произведения заданной функции, зависящей от переменной x, на функцию $\sin(y)/y$ (следует пользоваться функциями MESHGRID и MESH).

III часть - Программирование в среде пакета MATLAB

1) Составить программу на языке MATLAB (script-файл), выполняющую матричные операции и работу с заданной функцией (см. п.п. 1 и 2 раздела II). Программа должна выполнять следующие действия:

- Запрос ввода пользователем двух матриц;
- Выбор операции над матрицами с помощью меню;
- Выполнение выбранной операции и вывод результата;
- Построение графика функции $y=f(x)$;
- Расчет интеграла для функции $y=f(x)$.

2) Написать программу построения фигуры Лиссажу. Отношение частот $k1/k2$ указано в таблице вариантов заданий (см. табл. 4.1). Фигура Лиссажу определяется комплексной функцией вида $y(x)=\cos(k1*x)+j*\sin(k2*x)$, где $j=\sqrt{-1}$ – мнимая единица. При этом независимая переменная x изменяется в пределах от 0 до 2π.

3) Написать программу для построения трехмерной поверхности функции $z=f(x, y)$ следующего вида $z = f(x).*\sin(y)./y$, где $f(x)$ берется из таблицы вариантов заданий (см. приложение А). Задать цветовую шкалу для значений z, используя команды COLORMAP и COLORBAR.

4) Написать программу для проверки генератора случайных чисел. На одном графическом экране построить гистограммы нормального и равномерного законов распределений (использовать операторы RAND, RANDN, SUBPLOT и HIST).

Отчет по данной работе представляет собой программу (один или несколько m-файлов) на языке MATLAB. Листинг программы в письменном виде или саму программу в виде m-файла следует прислать для проверки в ТМЦ ДО ТУСУР. Программа должна содержать все необходимые комментарии, объясняющие работу отдельных фрагментов и основных операторов.

Таблица 4.1 – Варианты заданий на лабораторную работу № 1

№ варианта	Функция $f(x)$	Интервал аргумента x для $f(x)$	Отношение частот $k1/k2$
1	$e^{\sin(x)}$	[-3, 3]	3/5
2	$\sqrt{\frac{3x^2+1}{x-1}}$	[2, 5]	2/5
3	sin(x)cos(x)	[-2, 4]	3/4
4	$e^{\cos(x)}$	[-1, 2]	2/3
5	e^{-x^2}	[0, 5]	3/5
6	$\text{arctg}(\frac{x}{100})$	[0, 100]	5/6
7	$e^{x\cos(x)}$	[-2, 3]	2/3
8	$\log(x^2+\sin(x))$	[1, 3]	3/7
9	$\frac{\sqrt{1+x^2}}{\sin(x)}$	[0, 2]	5/8
10	$\text{tg}(\sin(x)-\cos(x))$	[1, 7]	3/8

4.3 Лабораторная работа № 2. Решение систем линейных алгебраических уравнений методом Гаусса

Целью данной лабораторной работы является изучения одного из основных методов решения систем линейных алгебраических уравнений – метода Гаусса. А также закрепление навыков программирования в среде системы для инженерных расчетов MATLAB.

Вводные замечания

Система алгебраических линейных уравнений (СЛАУ) имеет вид:

$$\left.\begin{array}{l} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \vdots \quad\quad \vdots \quad\quad\quad \vdots \quad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \end{array}\right\} \qquad (4.1)$$

Также СЛАУ можно записать в матричной форме

$$\mathbf{A}\boldsymbol{x}=\boldsymbol{b},$$

где $\boldsymbol{x}=[x_1,x_2,\ldots,x_n]^{\mathrm{T}}$ – вектор-столбец неизвестных; $\boldsymbol{b}=[b_1,b_2,\ldots,b_n]^{\mathrm{T}}$ – вектор-столбец коэффициентов правой части системы (вектор свободных членов); **A** – матрица nxn коэффициентов левой части СЛАУ. Значок $^{\mathrm{T}}$ означает транспонирование матрицы (вектора), т.е. замену строки соответствующим столбцам.

В общем случае система (4,1) будет иметь единственное решение, если выполняются следующие условия:

1) число неизвестных x_i $(i=\overline{1,n})$ равно числу уравнений, т.е. матрица коэффициентов **A** должна быть квадратной;

2) определитель матрицы коэффициентов **A** не равен нулю, т.е. det**A**≠0.

Если det**A**=0, то в системе (4,1) есть линейно зависимые уравнения и СЛАУ в общем случае не имеет единственного решения. Система уравнений, у которой определитель матрицы коэффициентов равен нулю (det**A**=0), называется *вырожденной*.

Для решения СЛАУ используются два класса методов: прямые и итерационные. *Прямые методы* являются универсальными и используются для решения СЛАУ сравнительно невысокого порядка ($n \le 100…200$). Прямые методы – это методы, которые теоретически позволяют получить точное решение системы уравнений за конечное число действий. Они используют аналитические выражения, которые дают точный результат, если не делать округления чисел в процессе расчетов. К прямым методам относятся, в частности, метод обратной матрицы, метод Крамера и др.

На практике при выполнении вычислений на ЭВМ неизбежны ошибки округления. Поэтому решение СЛАУ на ЭВМ с использованием прямых методов осуществляется с определенной погрешностью, которая возрастает с увеличением порядка системы уравнений.

В данной лабораторной работе студентам предлагается реализовать наиболее распространенный метод решения СЛАУ – метод Гаусса. Этот метод при реализации в виде программы требует гораздо меньшего числа вычислений, чем методы обратной матрицы, Крамера и пр. Метод Гаусса применим только к СЛАУ с невырожденной матрицей коэффициентов ($\det\mathbf{A} \neq 0$).

Основные положения метода Гаусса

Метод Гаусса основан на приведении матрицы к треугольному виду. Это достигается последовательным исключением неизвестных из уравнений системы. Сначала при использовании первого уравнения в качестве ведущего исключается x_1 из всех последующих уравнений СЛАУ. Затем с помощью второго уравнения исключается x_2 из третьего и всех последующих уравнений и т.д. Этот процесс называется *прямым ходом*. Он продолжается до тех пор, пока в левой части последнего (n-го) уравнения не останется лишь один член с неизвестным x_n, т.е. пока матрица коэффициентов **A** системы не будет приведена к треугольному виду.

На произвольном *k*-м шаге прямого хода формулы для пересчета коэффициентов имеют следующий вид:

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - \frac{a_{ik}^{(k-1)} \cdot a_{kj}^{(k-1)}}{a_{kk}^{(k-1)}};\ b_i^{(k)} = b_i^{(k-1)} - b_k^{(k-1)} \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}};\ i,j = \overline{k,n};\ \ k = \overline{1,n-1}.$$

Это основные формулы для прямого хода метода Гаусса.

Обратный ход метода Гаусса состоит в последовательном вычислении искомых неизвестных: решая последнее уравнение приведенной системы, находим неизвестное x_n. Далее, используя это значение, из предыдущего (n–1)-го уравнения вычисляем x_{n-1}, и т.д. Последним найдем x_1 из первого уравнения.

Общая формула расчета переменных x_k ($k = \overline{n,1}$) имеет следующий вид

$$x_n = \frac{b_n}{a_{nn}};$$

$$x_k = \left[b_k - \sum_{j=k+1}^{n} a_{kj} x_j \right] \Big/ a_{kk},\ k = n-1, \ldots, 1.$$

В процессе исключения неизвестных приходится выполнять деление неизвестных на коэффициенты a_{11}, a_{22}, … и т.д. Поэтому они должны быть отличными от нуля, в противном случае необходимо соответственным образом переставить уравнения СЛАУ. Перестановка уравнений должна быть предусмотрена в вычислительном алгоритме при его реализации на ЭВМ.

Недостаток стандартного метода Гаусса (метода единственного деления). При малой величине главного элемента $a_{kk}^{(k-1)}$ каждое уравнение системы во время прямого хода необходимо умножать на большое число $1/a_{kk}^{(k-1)}$, что приводит к росту ошибки округления при расчетах. Для уменьшения этой ошибки в качестве главного элемента следует выбирать наибольший элемент в строке, столбце или во всей матрице **A**. Это делается с помощью соответствующей перестановки столбцов и/или строк матрицы коэффициентов **A** и вектора свободных членов ***b***.

Стандартные функции системы MATLAB для работы с СЛАУ

Вся система MATLAB ориентирована на работу с матрицами, поэтому все функции линейной алгебры полностью реализованы в виде стандартных подпрограмм или встроенных процедур. Рассмотрим наиболее часто употребляемые.

norm – вычисление нормы векторов и матриц

Синтаксис: `n=norm(A,p)`

Функция `n=norm(A,p)` вычисляет p-норму матрицы **A**. В качестве параметра p можно использовать значения 1, 2 или inf. Матрица A имеет размер *n*x*m*.

`n=norm(A,1)` – вычисляет 1-норму матрицы A: $\|\mathbf{A}\|_1 = \max\limits_{j=1,m} \{\sum\limits_{i=1}^{n} |a_{ij}|\}$ - максимальная сумма элементов столбцов;

`n=norm(A,inf)` – вычисляет max-норму (∞-норма) матрицы **A**: $\|\mathbf{A}\|_\infty = \max\limits_{i=1,n} \{\sum\limits_{j=1}^{m} |a_{ij}|\}$ - максимальная сумма элементов строк;

`n=norm(A,2)` – вычисляет 2-норму (евклидову норму) матрицы **A**: $\|\mathbf{A}\|_2 = \sqrt{\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{m} |a_{ij}^2|}$.

cond – вычисление числа обусловленности матрицы

Синтаксис: `k=cond(A)`

Функция `k=cond(A)` возвращает число обусловленности матрицы A. Число обусловленности k – есть мера относительной погрешности при операциях обращения матрицы. Оно характеризует близость матрицы к вырождению. Матрицы с большим по величине числом обусловленности (близкие к вырожденной) дают большие ошибки при решении СЛАУ или обращении матрицы.

rcond – оценка числа обусловленности матрицы

Синтаксис: `k=rcond(A)`

Функция `k=rcond(A)` возвращает величину, обратную значению числа обусловленности матрицы A относительно 1-нормы. Если матрица A хорошо обусловлена, то значение k близко к единице, если матрица A плохо обусловлена, то значение k близко к нулю.

det – определитель матрицы

Синтаксис: `d=det(A)`

Функция `d=det(A)` вычисляет определитель квадратной матрицы.

inv – вычисление обратной матрицы

Синтаксис: `Y=inv(A)`

Функция `Y=inv(A)` вычисляет матрицу, обратную квадратной матрице A. В случае, когда матрица A плохо масштабирована или близка к вырожденной выдается соответствующее предупреждение.

На практике вычисление явной обратной матрицы требуется не часто. Если необходимо искать решение СЛАУ, то в среде MATLAB рекомендуется использовать решатели систем, т.е. операторы вида x=A\b или x=A/b, а не операцию x=inv(A)*b. Расчеты с помощью решателя систем выполняются быстрее и с большей точностью.

**/ ** – решатели систем линейных уравнений

Синтаксис: `X=B\A`

`X=B/A`

Функция `X=B\A` находит решение системы уравнений вида AX=B, где A – прямоугольная матрица размера m x n и B – матрица размера n x k.

Функция `X=B/A` находит решение системы уравнений вида XA=B, где A – прямоугольная матрица размера n x m и B – матрица размера m x k.

lu – треугольное LU разложение матрицы

Синтаксис: `[L,U,P]=lu(A)`

Функция `[L,U,P]=lu(A)` находит разложение для квадратной матрицы A в виде трех составляющих – нижней треугольной матрицы L, верхней треугольной матрицы U и матрицы перестановок P, так что PA=LU.

Алгоритм LU-разложения основан на методе исключения Гаусса. LU-разложение широко используется при вычислении определителей, нахождении обратных матриц и при решении СЛАУ.

Для получения информации по другим функциям системы MATLAB для работы с линейной алгеброй см. дополнительную литературу или встроенный HELP среды.

Задание на лабораторную работу № 2

В ходе данной лабораторной работы студенты должны выполнить следующие пункты:

1) реализовать метод исключения Гаусса без выбора главного элемента в виде процедуры на языке системы MATLAB (блок-схема алгоритма приведена ниже);
2) протестировать реализованную процедуру решения СЛАУ на следующих тестовых задачах:

а) СЛАУ с хорошо обусловленной матрицей коэффициентов размером 50x50. Для формирования матрицы коэффициентов A и вектора свободных

членов b можно использовать генераторы случайных чисел `rand` и `randn` (например, для матрицы A=rand(N) и вектора b=randn(N,1)).

б) СЛАУ с плохо обусловленной матрицей коэффициентов размером 50x50. Для получения плохо обусловленной матрицы можно использовать функции системы MATLAB `A=hilb(N)` или `A=invhilb(N)`, которые генерируют плохо обусловленную матрицу Гильберта. Матрицы с большим числом обусловленности $\text{cond}(A) \to \infty$ (близкие к вырожденной) при использовании функции A=hilb(N) или A=invhilb(N) получаются при $N \geq 15$.

Для каждой сформированной матрицы коэффициентов A вычислить определитель `det(A)`, число обусловленности `cond(A)` и оценку числа обусловленности `rcond(A)`.

3) Сравнить полученное решение (вектор ***x***) с результатами расчетов с помощью стандартного решателя системы MATLAB `x=A\b`.
4) оформить отчет по лабораторной работе, который должен включать следующие *обязательные* разделы:

   а) описание алгоритма метода исключения Гаусса (включая блок-схему);

   б) текст программы решения СЛАУ методом Гаусса с необходимым числом комментариев;

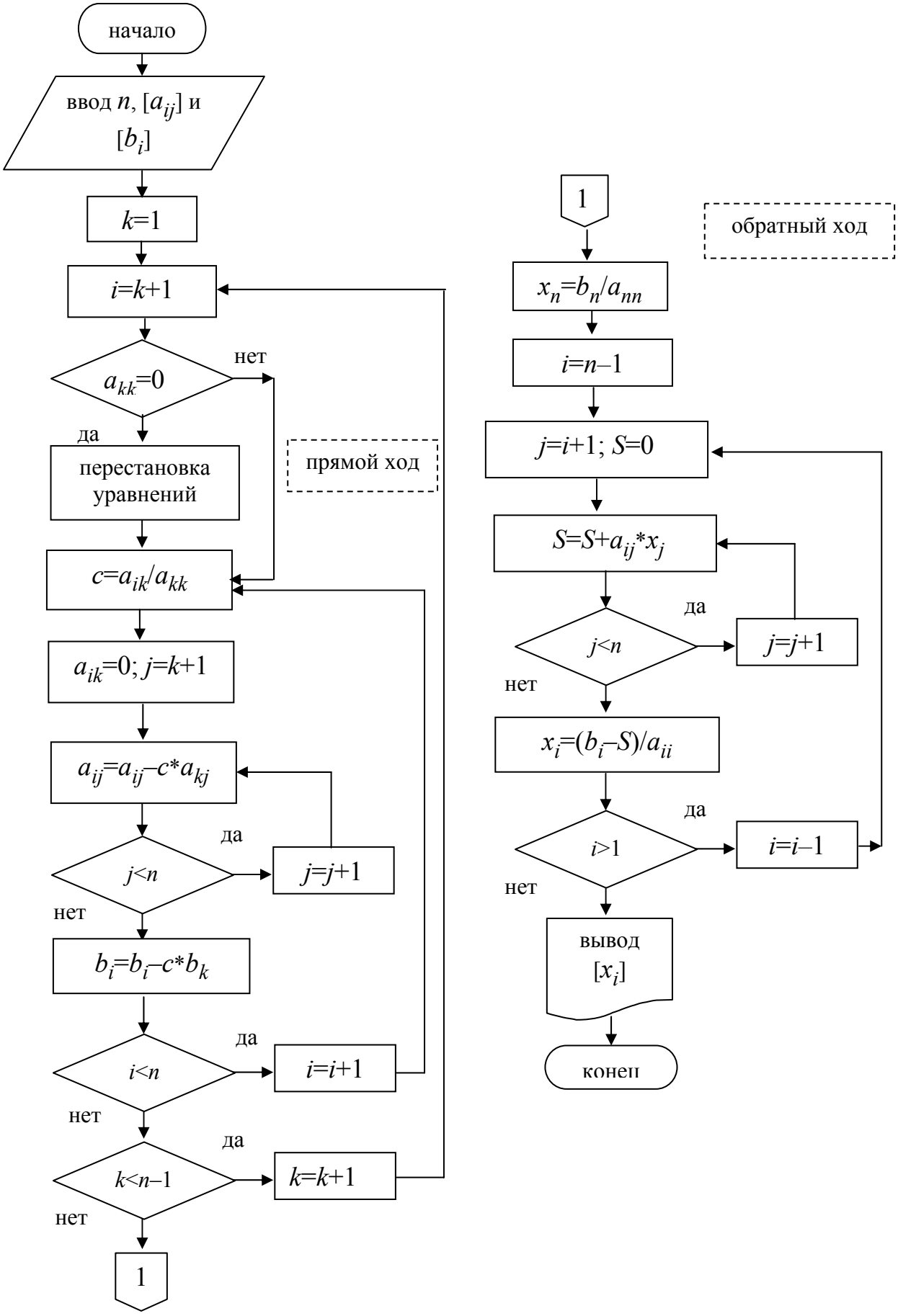
   в) результаты тестирования процедуры;

      для каждой тестовой задачи необходимо привести вектор решений ***x***, полученный своей программой и стандартным решателем MATLAB, значение определителя det(A) и величину числа обусловленности cond(A) матрицы коэффициентов и оценку числа обусловленности rcond(A).

   г) выводы о проделанной работе.

Отчет по данной лабораторной работе выполняется с помощью текстового редактора MSWord или письменно. Программа должна содержать все необходимые комментарии, объясняющие работу отдельных фрагментов и основных операторов.

Блок-схема алгоритма исключения Гаусса (без выбора главного элемента) для решения СЛАУ



4.4 Лабораторная работа № 3. Линейная полиномиальная интерполяция функций

Целью данной лабораторной работы является исследование поведения интерполирующего полинома и точности интерполяции в зависимости от числа узлов и вида приближаемой функции.

Теоретические сведения

Интерполяция – это нахождение функции, которая в некоторых заранее указанных точках совпадает с заданной функцией.

Пусть на отрезке $[a, b]$ задана $n+1$ точка $x_0, x_1, \ldots, x_n$ (*узлы интерполяции*). Пусть в этих точках известны значения *интерполируемой* функции $f(x)$:

$$y_i = f(x_i),\ i = \overline{0,n}.$$

Требуется построить функцию $\varphi(x)$, принадлежащую известному классу и принимающую в узлах интерполяции те же значения, что и $f(x)$, т.е.

$$\varphi(x_i) = y_i,\ i = \overline{0,n}. \qquad (4.2)$$

Геометрически это означает, что нужно найти кривую $y=\varphi(x)$ определённого вида, проходящую через заданную систему точек (x_i, y_i), $i = \overline{0,n}$ (см. рис. 4.1). Функция $\varphi(x)$ называется *интерполирующей* функцией.

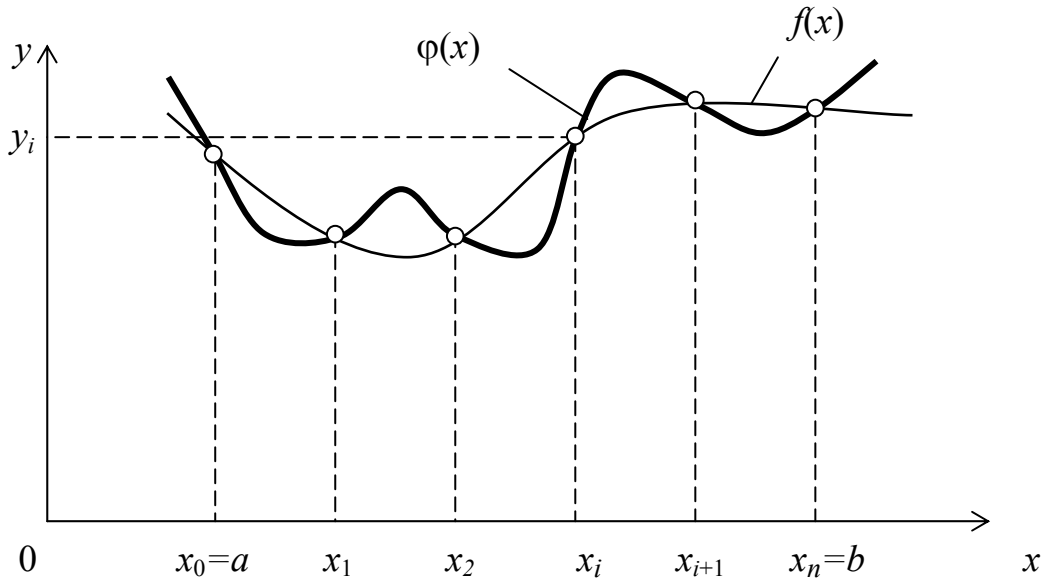


Рисунок 4.1 – Интерполяция исходной функции $f(x)$ интерполирующей функцией $\varphi(x)$

Рассмотрим задачу *линейной интерполяции*. При этом интерполирующая функция имеет следующий вид:

$$\varphi(x) = \varphi(\vec{C}, x) = C_0\varphi_0(x) + C_1\varphi_1(x) + \ldots + C_m\varphi_m(x) = \sum_{k=0}^{m} C_k\varphi_k(x), \quad (4.3)$$

где $\varphi_0(x)$, $\varphi_1(x)$, … , $\varphi_m(x)$ – базисные функции, вид которых задан; C_k – искомые коэффициенты.

Используя условие (4.2) и выражение (4.3), получаем систему уравнений

$$\left.\begin{aligned} &\varphi(\vec{C}, x_0) = C_0\varphi_0(x_0) + C_1\varphi_1(x_0) + \ldots + C_m\varphi_m(x_0) = y_0 \\ &\varphi(\vec{C}, x_1) = C_0\varphi_0(x_1) + C_1\varphi_1(x_1) + \ldots + C_m\varphi_m(x_1) = y_1 \\ &\ldots \\ &\varphi(\vec{C}, x_n) = C_0\varphi_0(x_n) + C_1\varphi_1(x_n) + \ldots + C_m\varphi_m(x_n) = y_n \end{aligned}\right\} \quad (4.4)$$

Единственное решение системы (4.4) существует при двух условиях:

1) число точек (x_i, y_i), $i = \overline{0,n}$ равно числу коэффициентов C_k, $k = \overline{0,m}$, т.е. $m = n$.
2) система уравнений (4.4) должна быть невырожденной, т.е. определитель системы $\Delta \neq 0$.

Таким образом, если выполняются вышеуказанные условия, то через точки (x_i, y_i) проходит *единственная* функция вида $\varphi(\vec{C}, x) = \sum_{k=0}^{m} C_k\varphi_k(x)$.

На практике в качестве интерполирующей функции $\varphi(x)$ часто используются алгебраические полиномы различного вида, так как полиномы легко вычислять, дифференцировать и интегрировать. При этом интерполяция носит название *полиномиальной*.

В случае *линейной полиномиальной интерполяции* базисные функции имеют следующий вид: $\varphi_0(x) = x^0 = 1$, $\varphi_1(x) = x^1 = x$, $\varphi_2(x) = x^2$, … , $\varphi_m(x) = x^m$. Интерполирующая функция при этом имеет вид полинома степени m: $\varphi(x) = P_m(x) = C_0 + C_1 x + C_2 x^2 + \ldots + C_m x^m$. При $m = n$ система (4.4) примет вид

$$\left.\begin{array}{l} C_0 + C_1 x_0 + C_2 x_0^2 + ... + C_n x_0^n = y_0 \\ C_0 + C_1 x_1 + C_2 x_1^2 + ... + C_n x_1^n = y_1 \\ ... \qquad\qquad ... \qquad\qquad ... \\ C_0 + C_1 x_n + C_2 x_n^2 + ... + C_n x_n^n = y_n \end{array}\right\} \qquad (4.5)$$

В матричной форме систему (4.5) можно переписать как **A*C=B**, где

$$\mathbf{A} = \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix}$$ – матрица Ван дер Монда; $\mathbf{B} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$; $\mathbf{C} = \begin{bmatrix} C_0 \\ C_1 \\ \vdots \\ C_n \end{bmatrix}$.

Решением системы (3.4) будет вектор коэффициентов полинома **C**. Так как определитель матрицы Ван дер Монда всегда отличен от нуля (при $x_i \neq x_j$), то решение системы (4.5) – единственное. Для решения системы (4.5) необходимо найти обратную матрицу **A**. В этом случае решением (4.5) будет $\mathbf{C}=\mathbf{A}^{-1}*\mathbf{B}$.

Вывод: Таким образом, через заданные на интервале $[a, b]$ точки (x_i, y_i), $i = \overline{0,n}$ всегда можно провести *единственный интерполяционный полином* степени n $P_n(x)=C_0 + C_1 x + \ + C_2 x^2 + \ldots + C_n x^n$, коэффициенты которого находятся в результате решения системы (4.5).

Выражение (4.2) определяет поведение функции $\varphi(x)$ только в узлах интерполяции (x_i, y_i), $i = \overline{0,n}$. Между узлами $\varphi(x)$ *может вести себя произвольным образом*, сколь угодно далеко, в принципе, отклоняясь от зависимости $f(x)$. Определить погрешность приближения можно, используя выражение для абсолютной ошибки $\varepsilon = | f(x) - \varphi(x) |$.

Ошибка полиномиальной интерполяции. Лучший способ проверить качество интерполяции – вычислить значения интерполирующей функции в большом числе точек и построить график. Однако в некоторых ситуациях качество интерполянта можно проанализировать. Предположим, что величина y_i представляет собой точные значения известной функции $f(x)$ в точках x_i. Пусть $P_n(x)$ – единственный полином n-й степени, интерполирующий функцию по этим точкам (x_i, y_i), $i = \overline{0,n}$. Предположим, что во всех точках $x \in [a, b]$

функция $f(x)$ имеет $(n+1)$ непрерывную производную. Тогда можно показать, что абсолютная ошибка интерполяции $\varepsilon(x) = |f(x) - P_n(x)|$ определяется выражением

$$\varepsilon(x) \le \frac{|(x-x_0)(x-x_1)...(x-x_n)|}{(n+1)!} M_{n+1} = \frac{|\omega_h(x)|}{(n+1)!} M_{n+1}, \qquad (4.6)$$

где $M_{n+1} = \max\limits_{x\in[a,b]} |f^{(n+1)}(x)|$ - максимальное значение $(n+1)$-й производной функции $f(x)$ на интервале $[a, b]$; $\omega_h(x) = \prod\limits_{i=0}^{n}(x - x_i)$.

Теперь посмотрим, что получится, если интерполировать известную функцию $f(x)$ все в большем и большем числе точек на фиксированном интервале. Выражение для погрешности (4.6) состоит из трех разных частей: факториал и произведение разностей $\omega_h(x)$ с увеличением n уменьшают ошибку, но порядок производной при этом растет. Для многих функций величина M_{n+1} увеличивается быстрее, чем $(n+1)!$. В результате полиномиальные интерполянты далеко не всегда сходятся к обычной непрерывной функции. Практический эффект выражается в том, что *интерполирующий полином высокой степени может вести себя очень плохо* в точках, отличных от узлов интерполяции (x_i, y_i), $i = \overline{0,n}$. Поэтому на практике обычно используют интерполирующие полиномы степени не выше 5-6.

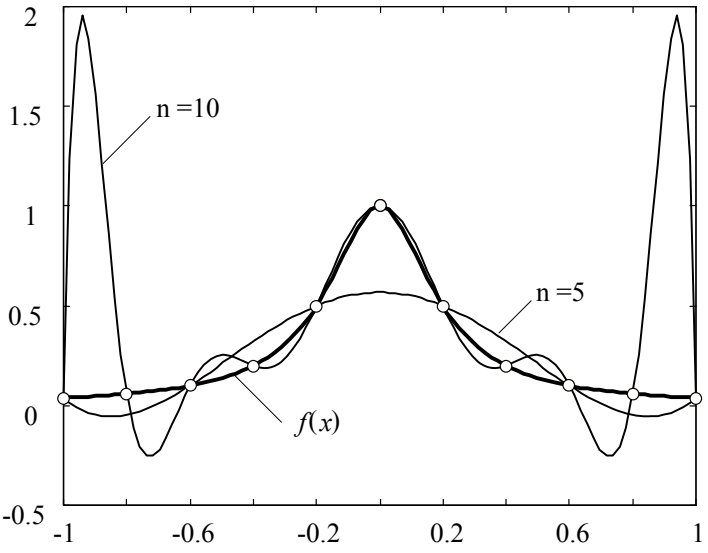


Рисунок 4.2 – Интерполяция функции Рунге полиномом степени n

Примером может служить функция Рунге вида $f(x)=1/(1+25x^2)$, график которой представлен на рис. 4.2. С увеличением порядка интерполирующего полинома $P_n(x)$ при равномерном распределении узлов интерполяции на

интервале [–1, 1] происходит ухудшение качества приближения на краях интервала. Это объясняется тем, что производные функции Рунге, которые фигурируют в выражении для погрешности интерполяции (3.5), быстро растут с увеличением числа n.

Формула (4.6) показывает, что точность приближения зависит не только от числа узлов интерполяции (т.е. порядка интерполирующего полинома), но и от *их расположения на интервале* $[a, b]$. В простейшем случае выбирается равномерное расположение точек x_i, $i = \overline{0,n}$ на интервале $[a, b]$ с шагом $\Delta x = (b–a)/n$. Однако, как показывает практика, равномерное расположение не является оптимальным с точки зрения лучшего приближения $\varphi(x)$ к зависимости $f(x)$. Более оптимальным для полиномиальной интерполяции является расположение узлов на интервале $[a, b]$ по формуле

$$x_i = \frac{a+b}{2} + \frac{b-a}{2}\cos\frac{(2i+1)\pi}{2n+2},\ i = \overline{0,n}. \qquad (4.7)$$

Выражение (4.7) определяет так называемое *оптимальное распределение* узлов интерполяции на интервале $[a, b]$.

Содержание лабораторной работы № 3

При выполнении лабораторной работы используется программа-макет `polinom.m`, написанная в среде пакета для математических и инженерных расчетов MATLAB. Программа предназначена для решения задачи линейной полиномиальной интерполяции и позволяет вывести графики исходной $f(x)$, приближающей $\varphi(x)=P_n(x)$ функций и ошибки интерполяции $\varepsilon(x)$, коэффициенты полинома $P_n(x)$, значение максимальной ошибки $\varepsilon_{\max} = \max\limits_{x\in[a,b]}(\varepsilon(x))$ на интервале аппроксимации $[a, b]$.

Студентам предлагается выполнить следующие задания в ходе данной лабораторной работы:

1) Изучить предлагаемую программу-макет polinom.m (текст программы см. в Приложении А). Исходные интерполируемые функции (см. табл. 4.3 вариантов заданий) записать в виде m-файлов со структурой function.

2) Исследовать точность приближения с помощью полинома $P_n(x)$ следующих трех функций (см. таблицу вариантов в приложении А):

а) функция 1 из табл. 4.3 заданий на лабораторную работу;

б) полином $P_m(x)$ из табл. 4.3 (выполнить расчет для случаев $n<m$, $n=m$, $n>m$, где n - порядок интерполирующего полинома);

в) функция 2 из табл. 4.3 заданий.

Исследования выполнить при различном порядке n интерполирующего полинома $P_n(x)$ и способе распределения узлов на интервале. Например, n = 3, 4, 5, 7, 10 – не менее 5-6 опытов для каждой из функций.

3) По результатам испытаний для каждой исходной функции заполнить таблицу или построить график зависимости максимальной ошибки ε_{max} от порядка интерполирующего полинома n (числа узлов). Кроме этого, в случае 2б, для полинома $P_m(x)$ заполнить таблицу (см. табл. 4.2) и сравнить коэффициенты исходного $P_m(x)$ и интерполирующего $P_n(x)$ полиномов для случаев $n<m$, $n=m$, $n>m$).

Таблица 4.2 – Исследование интерполяции функции $P_m(x)$

Порядок полинома $P_n(x)$	Равномерное распределение узлов		Оптимальное распределение узлов	
	коэффициенты полинома $P_n(x)$	ε_{max}	коэффициенты полинома $P_n(x)$	ε_{max}
$n<m$	$C_0 =$		$C_0 =$	
	$C_1 =$		$C_1 =$	
	…		…	
	$C_n =$		$C_n =$	
$n=m$	$C_0 =$		$C_0 =$	
	$C_1 =$		$C_1 =$	
	…		…	
	$C_n =$		$C_n =$	
$n>m$	$C_0 =$		$C_0 =$	
	$C_1 =$		$C_1 =$	
	…		…	
	$C_n =$		$C_n =$	

4) Оформить отчет по лабораторной работе. Отчет должен содержать следующие обязательные разделы:

– Теоретическая часть (понятие полиномиальной интерполяции, формирование системы уравнений, описать используемый в программе

метод решения полученной системы, ошибка приближения, способ распределения узлов интерполяции на интервале);

- Вариант задания – номер варианта, вид приближаемых функций и интервалы интерполяции (из таблицы вариантов заданий в приложении А);
- Графики исходной функции $f(x)$, интерполирующего полинома $P_n(x)$ и ошибки $\varepsilon(x)$ на интервале интерполяции – для всех заданных приближаемых функций $f(x)$ для одного из значений n;
- Зависимость максимальной ошибки интерполяции от порядка интерполирующего полинома (количества узлов) в виде графика или таблицы для всех заданных приближаемых функций $f(x)$;
- Выводы по полученным результатам. В выводах объяснить зависимость ошибки интерполяции от степени интерполирующего полинома (использовать формулу для ошибки), от вида исходной функции и способа распределения узлов на интервале интерполяции.

Таблица 4.3 – Варианты заданий на лабораторную работу № 3

№ варианта	Функция 1	Интервал	Полином $P_m(x)$	Интервал	Функция 2	Интервал
1	$\sin(x)$	$[0, \frac{\pi}{2}]$	$3x^5+12x-7$	[0, 2]	$\lvert x-5 \rvert$	[0, 10]
2	$\cos(x)$	$[0, \frac{2\pi}{3}]$	$4x^7-5x^5+8x^3+12$	[–2, 3]	$\lvert x+3 \rvert$	[–8, 5]
3	$\cos(x)+\sin(x)$	$[-\frac{\pi}{3}, \frac{\pi}{3}]$	$7x^7-9x^4-7x^2+12$	[–4, 1]	$\lvert 2-x \rvert$	[–2, 4]
4	$\cos(x)-\sin(x)$	$[-\frac{\pi}{2}, \frac{\pi}{2}]$	$12x^6-3x^5-7x^3+11$	[–1, 2]	$\lvert 2x-2 \rvert$	[–5, 8]
5	$\sin(x)-\cos(x)$	$[-\frac{\pi}{4}, \frac{\pi}{4}]$	$3x^9-12x^3-18$	[–3, 2]	$\lvert 2x+1 \rvert$	[–5, 5]
6	$\sin(x)\cos(x)$	$[0, \frac{\pi}{3}]$	$5x^4-4x^2+9$	[–4, 7]	$\lvert 4-2x \rvert$	[0, 8]
7	$1+\sin(x)$	$[0, \pi]$	$-2x^4+7x^2-8x$	[–1, 3]	$\lvert x \rvert+5$	[–9, 9]
8	$1+\cos(x)$	$[0, \frac{\pi}{2}]$	$-8x^7+12x^4-8x^2+19$	[–3, 6]	$\lvert 3x \rvert - 4$	[–7, 8]
9	$1-\sin(x)$	$[0, \frac{\pi}{3}]$	$9x^4-7x^2-21$	[–4, 5]	$\lvert 5-7x \rvert$	[–4, 2]
10	$1-\cos(x)$	$[-\frac{\pi}{2}, \frac{\pi}{2}]$	$-2x^8+11x^5-4x^2-8$	[–2, 3]	$\lvert 2x+3 \rvert$	[–6, 4]

4.5 Лабораторная работа № 4. Применение метода наименьших квадратов для сглаживания и выравнивания экспериментальных данных

Выполнение проектных операций и процедур в системах автоматизированного проектирования (САПР) основано на оперировании математическими моделями проектируемых объектов или технологических процессов.

Для решения задач, связанных с получением и хранением моделей, в состав современных САПР входят специальные информационные подсистемы. Важным разделом математического обеспечения таких подсистем является метод наименьших квадратов, который используется, в частности, для обработки результатов экспериментального определения параметров объектов и процессов при получении их математических моделей.

Цель работы: изучение методики сглаживания и выравнивания экспериментальных данных и получения математических моделей объектов и процессов по методу наименьших квадратов.

Теоретические сведения

Пусть данные некоторого эксперимента представлены в виде таблицы значений переменных x и y:

Таблица 4.4 – Исходные данные для аппроксимации

x_i	x_1	x_2	…	x_n
y_i	y_1	y_1	…	y_n

где x – независимая, а y – зависимая переменные.

Можно поставить задачу об отыскании аналитической зависимости между x и y, т.е. некоторой формулы $y = \varphi(x)$, явным образом выражающей y как функцию от x. Естественно требовать, чтобы график искомой функции $y = \varphi(x)$ изменялся плавно и не слишком уклонялся от экспериментальных точек (x_i, y_i). Поиск такой функциональной зависимости называют *«сглаживанием» или «выравниванием» экспериментальных данных* (см. рис. 4.3):

В частности, рассматриваемая задача встречается при построении математических моделей объектов и процессов, характеристики которых получены путем *экспериментальных измерений*. В этом случае требуется найти математическую модель $y = \varphi(x)$, которая достаточно хорошо будет приближать экспериментальные характеристики (x_i, y_i) реального объекта или процесса. Функцию $y = \varphi(x)$ иногда называют *эмпирической* зависимостью (моделью).

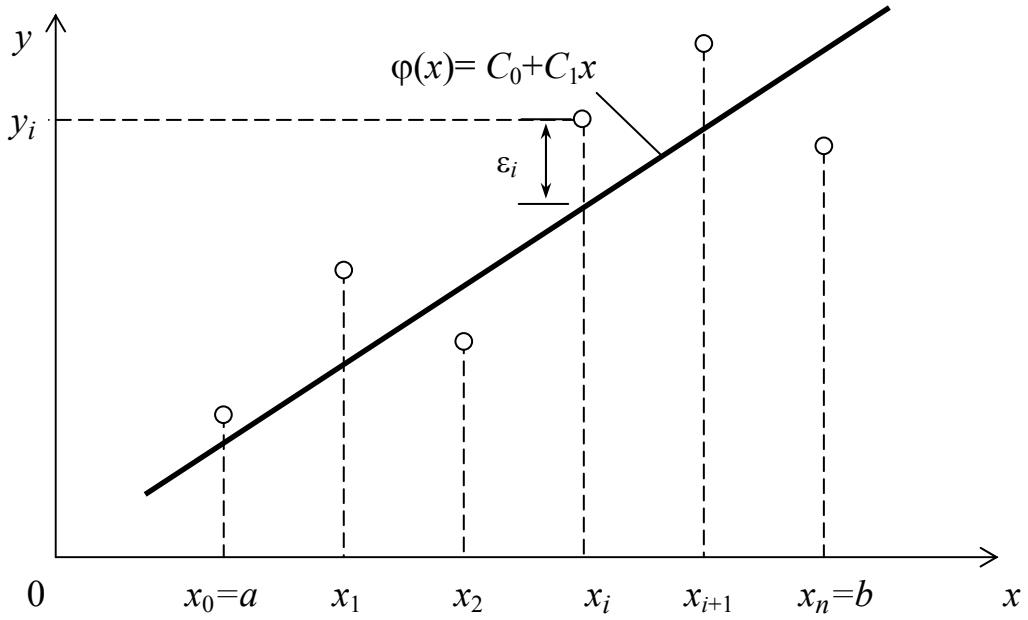


Рисунок 4.3 – Аппроксимация табличных данных (x_i, y_i) функцией $\varphi(x)$

Если для получения модели $y = \varphi(x)$ использовать глобальную интерполяцию функции в узлах x_i, $i = \overline{0,n}$ с помощью полинома, то при большом числе узлов степень интерполяционного полинома будет высока, это приведет к усложнению вычислений и «колебаниям» $y = \varphi(x)$ между узлами. Между тем, на практике даже при большом количестве точек обычно нет необходимости в использовании полиномов высокой степени.

Кроме того, так как значения аппроксимируемой функции $f(x)$ в точках x_i (т.е. значения y_i в табл. 4.4) находятся в результате измерений, они будут содержать некоторые ошибки. Если применить интерполяцию измеренных величин $y_i = f(x_i)$, то полученная математическая модель $y = \varphi(x)$ будет тщательно повторять все ошибки измерений.

Задачу сглаживания экспериментальных данных можно решать, используя *метод наименьших квадратов* (МНК). Согласно МНК указывается вид аппроксимирующей функции $y = \varphi(x)$ (математической модели):

$$y = \varphi(x, C_0, C_1, C_2, \dots, C_m) = \varphi(\vec{C}, x), \qquad (4.8)$$

где $C_0, C_1, C_2, \dots, C_m$ – неизвестные числовые параметры (коэффициенты модели).

Например, часто используется функция $\varphi(\vec{C}, x)$ в виде *обобщенного полинома* (линейной комбинации заданных *базисных функций* $y = \varphi_j(x)$, $j = \overline{0,m}$):

$$\varphi(\vec{C},x) = C_0\varphi_0(x) + C_1\varphi_1(x) + \dots + C_m\varphi_m(x) = \sum_{j=0}^{m} C_j\varphi_j(x). \qquad (4.9)$$

Если выбрать систему базисных функций $\varphi_j(x) = x^j$, $j = \overline{0,m}$, то функция $\varphi(\vec{C},x)$ будет представлять собой обычный алгебраический полином:

$$\varphi(\vec{C},x) = P_m(x) = C_0 + C_1x + C_2x^2 + \dots + C_mx^m.$$

Аналогично, функция (4.9) может описывать тригонометрические, экспоненциальные полиномы и т.д.

При использовании аппроксимирующей функции в виде (4.9) задача нахождения коэффициентов C_j, $j = \overline{0,m}$ называется *линейной задачей дискретной среднеквадратической аппроксимации* или *линейной задачей МНК*.

Наилучшими значениями параметров $C_0, C_1, C_2, \dots, C_m$ считают те, для которых сумма квадратов уклонений функции $\varphi(\vec{C},x)$ в точках x_i от экспериментальных значений y_i ($i = \overline{0,n}$) является минимальной, т.е. функция

$$\varepsilon(C_0, C_1, \dots, C_m) = \varepsilon(\vec{C}) = \sum_{i=0}^{n} (\varphi(\vec{C}, x_i) - y_i)^2 \qquad (4.10)$$

достигает минимума. Таким образом, в *методе наименьших квадратов минимизируется сумма квадратов разностей* между значениями функции $y = \varphi(\vec{C},x)$ и измеренными значениями y_i в точках x_i:

$$\varepsilon = \sum_{i=0}^{n} \varepsilon_i^2 = \sum_{i=0}^{n} [\varphi(\vec{C}, x_i) - y_i]^2 = \min, \qquad (4.11)$$

где ε_i – разность в i-ой точке значений функции $y = \varphi(\vec{C}, x)$ и y_i. Величина ε называется *среднеквадратической ошибкой*. Выбор функции ошибки в виде (4.11) удобен с практической точки зрения, так как делает функцию $y = \varphi(\vec{C}, x)$, построенную по экспериментальным данным, нечувствительной к случайным ошибкам измерений.

Получим уравнения для вычисления коэффициентов C_j, $j = \overline{0, m}$ по методу наименьших квадратов. Минимизируемая ошибка является функцией неизвестных коэффициентов C_j :

$$\varepsilon = f(C_0, C_1, \ldots, C_m).$$

Как известно, минимум функции нескольких переменных достигается в точке, где равны нулю все частные производные этой функции:

$$\frac{\partial \varepsilon}{\partial C_0} = \frac{\partial \varepsilon}{\partial C_1} = \ldots = \frac{\partial \varepsilon}{\partial C_m} = 0. \qquad (4.12)$$

Дифференцируя функцию ошибки (4.10) по переменным $C_0, C_1, \ldots, C_m$, согласно (4.12), получим систему уравнений следующего вида

$$\left.\begin{aligned}
&\frac{\partial \varepsilon}{\partial C_0} = 2\sum_{i=0}^{n} [\varphi(\vec{C}, x_i) - y_i] \frac{\partial \varphi(\vec{C}, x_i)}{\partial C_0} = 0 \\
&\ldots \\
&\frac{\partial \varepsilon}{\partial C_j} = 2\sum_{i=0}^{n} [\varphi(\vec{C}, x_i) - y_i] \frac{\partial \varphi(\vec{C}, x_i)}{\partial C_j} = 0 \\
&\ldots \\
&\frac{\partial \varepsilon}{\partial C_m} = 2\sum_{i=0}^{n} [\varphi(\vec{C}, x_i) - y_i] \frac{\partial \varphi(\vec{C}, x_i)}{\partial C_m} = 0
\end{aligned}\right\} \qquad (4.13)$$

Заметим, что в общем случае система (4.13) является нелинейной.

Далее будем полагать, что функция $\Phi(\vec{C}, x_i)$ является *линейной* относительно параметров $C_0, C_1, \ldots, C_m$, т.е. имеет вид (4.9).

Из (4.9) следует, что $\frac{\partial \varphi(\vec{C},x)}{\partial C_j} = \varphi_j(x)$. С учетом этого система (4.13) примет следующий вид (общий множитель 2 в уравнениях опустим):

$$\left.\begin{aligned}
&\frac{\partial \varepsilon}{\partial C_0} = \sum_{i=0}^{n}[C_0\varphi_0(x_i) + C_1\varphi_1(x_i) + ... + C_m\varphi_m(x_i) - y_i]\,\varphi_0(x_i) = 0\\
&\frac{\partial \varepsilon}{\partial C_1} = \sum_{i=0}^{n}[C_0\varphi_0(x_i) + C_1\varphi_1(x_i) + ... + C_m\varphi_m(x_i) - y_i]\,\varphi_1(x_i) = 0\\
&...\\
&\frac{\partial \varepsilon}{\partial C_j} = \sum_{i=0}^{n}[C_0\varphi_0(x_i) + C_1\varphi_1(x_i) + ... + C_m\varphi_m(x_i) - y_i]\,\varphi_j(x_i) = 0\\
&...\\
&\frac{\partial \varepsilon}{\partial C_m} = \sum_{i=0}^{n}[C_0\varphi_0(x_i) + C_1\varphi_1(x_i) + ... + C_m\varphi_m(x_i) - y_i]\,\varphi_m(x_i) = 0
\end{aligned}\right\} \quad (4.14)$$

Полученная система уравнений является системой (m+1) линейных алгебраических уравнений с (m+1) неизвестным $C_0, C_1, \ldots, C_m$ и называется *нормальной системой*. Ее решение определяет искомые коэффициенты функции $y=\varphi(\vec{C},x_i)$.

При решении систем линейных уравнений на ЭВМ обычно используют стандартные подпрограммы. В этом случае систему уравнений необходимо представить в стандартной матричной форме:

$$\mathbf{A}*\mathbf{C}=\mathbf{B}, \quad (4.15)$$

где **A** – квадратная матрица размером (m+1)×(m+1), составленная из коэффициентов системы линейных уравнений; **C** – вектор-столбец, содержащий (m+1) неизвестных переменных $C_0, C_1, \ldots, C_m$; **B** – вектор-столбец свободных членов системы линейных уравнений.

Представим (4.14) в матричном виде (4.15). Для этого сгруппируем слагаемые в уравнениях (4.14) таким образом, чтобы каждое уравнение с номером j ($j = \overline{0,m}$) приняло следующий вид

$$C_0\sum_{i=0}^{n}\varphi_0(x_i)\,\varphi_j(x_i)+C_1\sum_{i=0}^{n}\varphi_1(x_i)\,\varphi_j(x_i)+\ldots+C_m\sum_{i=0}^{n}\varphi_m(x_i)\,\varphi_j(x_i)=\sum_{i=0}^{n}y_i(x_i)\,\varphi_j(x_i).$$

В результате получим систему уравнений, представляющую собой развернутую запись системы (4.15)

$$\left.\begin{array}{l} a_{00}C_0+a_{01}C_1+\ldots+a_{0m}C_m=b_0 \\ a_{10}C_0+a_{11}C_1+\ldots+a_{1m}C_m=b_1 \\ \ldots \qquad\qquad\qquad\qquad \ldots \\ a_{j0}C_0+a_{j1}C_1+\ldots+a_{jm}C_m=b_j \\ \ldots \qquad\qquad\qquad\qquad \ldots \\ a_{m0}C_0+a_{m1}C_1+\ldots+a_{mm}C_m=b_m \end{array}\right\} \qquad (4.16)$$

Здесь $a_{jk}=\sum_{i=0}^{n}\varphi_k(x_i)\,\varphi_j(x_i)$ - элементы матрицы **A**; первый индекс (j) коэффициента a_{jk} равен номеру уравнения, а второй (k) – номеру соответствующей неизвестной переменной; $b_j=\sum_{i=0}^{n}y_i\varphi_j(x_i)$ – элементы вектора–столбца свободных членов **B**; $j,k=\overline{0,m}$.

Система линейных уравнений (4.16) при невысоком порядке может быть вручную решена по правилу Крамера или с помощью метода подстановки. Решение же систем линейных уравнений на ЭВМ чаще всего проводится с использованием метода исключения неизвестных (методы Гаусса или *LU*-разложения).

Выравнивание экспериментальных данных на основе МНК

Изучим подробнее случай, когда аппроксимирующая зависимость (4.8) имеет два неизвестных коэффициента: $y=\varphi(x, C_0, C_1)$. Используя выражение (4.13) и опуская несложные выкладки, получим систему двух уравнений с двумя неизвестными C_0, C_1:

$$\left.\begin{array}{l} \dfrac{\partial\varepsilon}{\partial C_0}=\sum_{i=0}^{n}[\varphi(C_0,C_1,x_i)-y_i]\dfrac{\partial\varphi(C_0,C_1,x_i)}{\partial C_0}=0 \\ \dfrac{\partial\varepsilon}{\partial C_1}=\sum_{i=0}^{n}[\varphi(C_0,C_1,x_i)-y_i]\dfrac{\partial\varphi(C_0,C_1,,x_i)}{\partial C_1}=0 \end{array}\right\} \qquad (4.17)$$

Рассмотрим, в частности аппроксимацию экспериментальных данных с помощью линейной функции (см. рис. 4.3)

$$y = \varphi(x, C_0, C_1) = C_0 + C_1 x. \qquad (4.18)$$

В этом случае, очевидно, $\frac{\partial \varphi}{\partial C_0} = 1$, $\frac{\partial \varphi}{\partial C_1} = x$ и система (4.17) примет вид:

$$\left.\begin{aligned} \sum_{i=0}^{n}[(C_0 + C_1 \cdot x_i) - y_i] = 0 \\ \sum_{i=0}^{n}[(C_0 + C_1 \cdot x_i) - y_i] \cdot x_i = 0 \end{aligned}\right\} \rightarrow \left\{\begin{aligned} C_0(n+1) + C_1 \sum_{i=0}^{n} x_i = \sum_{i=0}^{n} y_i \\ C_0 \sum_{i=0}^{n} x_i + C_1 \sum_{i=0}^{n} x_i^2 = \sum_{i=0}^{n} x_i y_i \end{aligned}\right\} \qquad (4.19)$$

Систему (4.19) можно записать в матричном виде (4.15), где

$$\mathbf{A} = \begin{bmatrix} n+1 & \sum_{i=0}^{n} x_i \\ \sum_{i=0}^{n} x_i & \sum_{i=0}^{n} x_i^2 \end{bmatrix}; \ \mathbf{B} = \begin{bmatrix} \sum_{i=0}^{n} y_i \\ \sum_{i=0}^{n} x_i \ y_i \end{bmatrix}; \ \mathbf{C} = \begin{bmatrix} C_0 \\ C_1 \end{bmatrix}.$$

Среднеквадратичная аппроксимация с помощью линейной функции (прямой линии) $\varphi(x) = C_0 + C_1 x$ целесообразна, очевидно, тогда, когда экспериментальные данные (x_i, y_i) приближенно описывают линейную зависимость y от x (об этом можно судить непосредственно по расположению точек (x_i, y_i) на координатной плоскости).

Однако подобная аппроксимация может быть использована и для более сложных зависимостей, если применить замену переменных. В этом случае выбирают новые переменные

$$X = \psi_1(x, y); \qquad Y = \psi_2(x, y) \qquad (4.20)$$

так, чтобы преобразованные экспериментальные данные

$$X_i = \psi_1(x_i, y_i); \qquad Y_i = \psi_2(x_i, y_i) \qquad (4.21)$$

в новой системе координат X, Y давали точки (X_i, Y_i), менее отклоняющиеся от линейной зависимости. Для аппроксимирующей прямой в новой системе координат

$$Y = \varphi'(X, C_0', C_1')$$

коэффициенты C_0' и C_1' можно определить из уравнений (4.19), где вместо (x_i, y_i) подставляют соответствующие значения (X_i, Y_i). Нахождение подходящих зависимостей (4.21) называют *выравниванием экспериментальных данных*.

После нахождения функции $Y = \varphi'(X, C_0', C_1')$ находят функцию $y = \varphi(x, C_0, C_1)$, выполнив обратную замену переменных. Функция $y = \varphi(x)$ определена неявно уравнением

$$\psi_2(x, y) = C_0' + C_1'\psi_1(x, y).$$

Явное выражение для $\varphi(x)$ получают их этого уравнения, решая его относительно y (заметим, что это возможно не всегда).

Пример. Экспериментальная зависимость y от x представлена таблицей:

x_i	1	2	3	4	5
y_i	7,1	27,8	62,1	110	161

Установить вид эмпирической формулы, описывающей эту зависимость, используя аппроксимирующую функцию $y = \varphi(x, C_0, C_1)$ с двумя параметрами C_0 и C_1. Определить наилучшие значения коэффициентов C_0 и C_1.

Решение. Легко убедиться в том, что экспериментальные точки (x_i, y_i) не располагаются вблизи прямой. Произведем замену переменных $X = \ln x$, $Y = \ln y$ и составим таблицу экспериментальных данных в новых переменных (X_i, Y_i)

$X_i = \ln x_i$	0,000	0,693	1,099	1,386	1,609
$Y_i = \ln y_i$	1,960	3,325	4,129	4,700	5,081

Теперь точки (X_i, Y_i) лежат приблизительно на прямой (см. рис. 4.4). Значения коэффициентов C_0' и C_1' эмпирической зависимости $Y = C_0' + C_1'X$ (в новых переменных) находятся из системы уравнений (4.19)

$$\left.\begin{aligned} C_0'(n+1) + C_1' \sum_{i=0}^{n} x_i &= \sum_{i=0}^{n} y_i \\ C_0' \sum_{i=0}^{n} x_i + C_1' \sum_{i=0}^{n} x_i^2 &= \sum_{i=0}^{n} x_i y_i \end{aligned}\right\} \rightarrow \left.\begin{aligned} 5 \cdot C_0' + 4{,}787 \cdot C_1' &= 19{,}196 \\ 4{,}787 \cdot C_0' + 6{,}200 \cdot C_1' &= 21{,}535 \end{aligned}\right\}$$

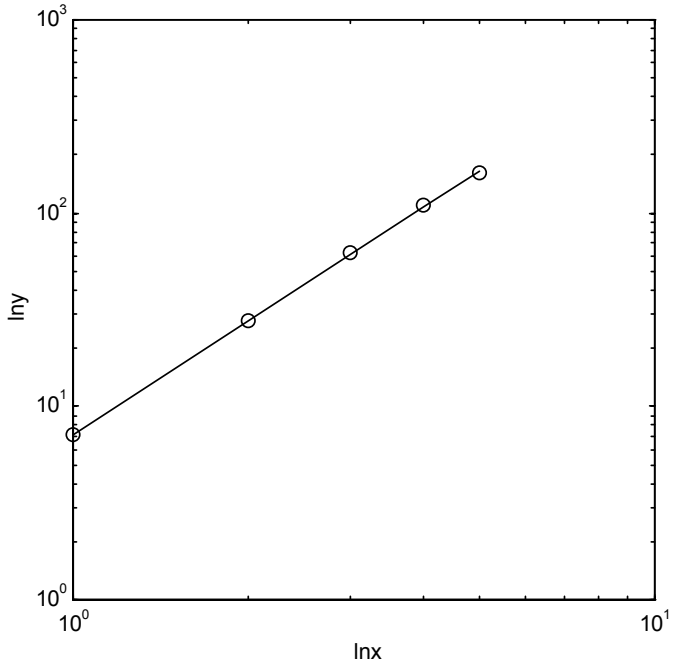


Рисунок 4.4 – Исходные точки и график аппроксимирующей функции в логарифмическом масштабе

Решив эту систему, получим $C_0' = 1{,}97,\ C_1' = 1{,}95.$

Найдем теперь связь между исходными переменными x и y в виде функции $y = \varphi(x, C_0, C_1)$, выполнив обратную замену переменных. Неявное уравнение, выражающее связь между x и y, имеет вид

$$\ln y = C_0' + C_1' \ln x = 1{,}97 + 1{,}95 \ln x.$$

Отсюда легко получить явную зависимость между x и y в виде степенной функции, выполнив антилогарифмирование:

$$y = e^{C_0' + C_1' \ln x} = e^{C_0'} e^{C_1' \ln x} = e^{C_0'} e^{\ln x^{C_1'}} = e^{C_0'} x^{C_1'}.$$

Обозначим $e^{C_0'} = C_0,\ C_1' = C_1$. В результате получим эмпирическую формулу в виде

$$y = \varphi(x, C_0, C_1) = C_0 x^{C_1} = 7{,}16\, x^{1{,}95}, \qquad (4.22)$$

где $C_0 = e^{C_0'} = e^{1,97} = 7,16$, $C_1 = C_1' = 1,95$.

Сравнение экспериментальных данных с результатами вычислений по формуле (4.22) в соответствующих точках представлены в виде таблицы

x_i	1	2	3	4	5
y_i	7,1	27,8	62,1	110	161
$y = 7,16\,x^{1,95}$	7,16	27,703	61,081	107,04	165,39

Как видно, эмпирическая формула (4.15) хорошо приближает эти данные.

Заметим, что параметры C_0' и C_1' в зависимости (4.15) в принципе можно найти из решения нелинейных уравнений (4.10). Однако применение способа выравнивания существенно упрощает вычисления этих параметров. В данном случае $C_0 = e^{C_0'}$, $C_1 = C_1'$.

Часто используемые аппроксимирующие зависимости $y = \varphi(x, C_0, C_1)$ с двумя параметрами C_0 и C_1 приведены в табл. 4.5. Здесь же указано, преобразование переменных для приведения зависимости к линейному виду $Y = C_0' + C_1' X$, а также связь коэффициентов C_0 и C_1 с коэффициентами C_0' и C_1'.

Таблица 4.5 – Вид эмпирических формул для приведения данных к линейной зависимости

№	Преобразование переменных	Эмпирическая формула $\varphi(x, C_0, C_1)$
1	$X = x,\ Y = xy$	$y = C_0 + \dfrac{C_1}{x}$, $C_0 = C_1'$, $C_1 = C_0'$
2	$X = x,\ Y = \dfrac{1}{y}$	$y = \dfrac{1}{C_0 + C_1 x}$, $C_1 = C_1'$, $C_0 = C_0'$
3	$X = x,\ Y = \dfrac{x}{y}$	$y = \dfrac{x}{C_0 + C_1 x}$, $C_1 = C_1'$, $C_0 = C_0'$
4	$X = x,\ Y = \ln y$	$y = C_0 C_1^{\,x}$, $C_0 = e^{C_0'}$, $C_1 = e^{C_1'}$
5	$X = \ln x,\ Y = y$	$y = C_1 \ln x + C_0$, $C_0 = C_1'$, $C_1 = C_0'$
6	$X = \ln x,\ Y = \ln y$	$y = C_0 x^{C_1}$, $C_0 = C_0'$, $C_1 = C_1'$

Естественно, наряду с шестью предложенными формулами преобразования к переменным X, Y следует проверить возможность применения линейной зависимости непосредственно к исходным данным (x_i, y_i). Условием выбора наилучшей эмпирической формулы является наименьшее уклонение исходных или преобразованных экспериментальных данных от прямой. Уклонение данных от прямой в каждом варианте выравнивания будем определять величиной

$$d = \left(\frac{\sum_{i=0}^{n} (Y_i - C_1' X_i - C_0')^2}{\sum_{i=0}^{n} Y_i^2} \right)^{\frac{1}{2}}. \qquad (4.23)$$

Для наилучшей эмпирической формулы величина d является наименьшей.

Задание на лабораторную работу № 4

Таблица вариантов заданий к лабораторной работе представлена ниже (табл. 4.6). Студентам необходимо выполнить следующие действия:

1) нанести на координатную сетку x, y экспериментальные точки (x_i, y_i);

2) выбрать одну из шести предложенных формул преобразования к переменным X, Y так, чтобы преобразованные экспериментальные данные (X_i, Y_i) наименее уклонялись от прямой линии;

3) методом наименьших квадратов найти значения коэффициентов C_0' и C_1' в уравнении прямой $Y = C_0' + C_1' X$;

4) найти явный вид эмпирической формулы $y = \varphi(x, C_0, C_1)$ и построить график эмпирической функции на плоскости x, y, оценить максимальную ошибку приближения;

5) оформить отчет по лабораторной работе.

Указания: в ходе выполнения лабораторной работы необходимо написать программу на языке системы MATLAB, реализующую вычисление коэффициентов C_0 и C_1 функции $y = \varphi(x, C_0, C_1)$ на основе МНК. Выбор формулы преобразования можно сделать как автоматически (по величине

уклонения d (4.23) от прямой), так и вручную – т.е. пользователь сам выбирает тип преобразования на основе оценки расположения точек (x_i, y_i) на плоскости x, y или точек (X_i, Y_i) на плоскости X, Y.

Таблица 4.6 – Варианты заданий на лабораторную работу № 4

№ варианта	x	1	2	3	4	5
1	y	1,1	1,4	1.6	1,7	1,9
2	y	1,05	1,55	1,7	1,75	1,8
3	y	0,4	0,55	0,13	0,09	0,07
4	y	7,5	6,2	5,5	3,5	3
5	y	8,2	5,9	4,9	4	3,2
6	y	7,2	5,9	4,9	4	3,2
7	y	7,1	6,1	4,9	4	3,1
8	y	0,55	0,7	0,77	0,82	0,85
9	y	1,1	1,55	1,9	2,3	2,6
10	y	1,1	1,55	1,9	2,25	2,5
11	y	5,1	4,4	3,2	2,7	2,55
12	y	5,1	3,4	3,2	2,7	2,55
13	y	1,9	5,5	10	15	21
14	y	3	3,5	3,67	3,75	3,8
15	y	0,25	0,09	0,07	0,05	0,04
16	y	0,25	0,111	0,071	0,053	0,042
17	y	0,20	0,28	0,33	0,36	0,38
18	y	4,8	5,76	6,912	8,294	9,95
19	y	1	3,08	4,3	5,16	5,83
20	y	0,33	0,5	0,6	0,67	0,71
21	y	1,5	1,75	1,83	1,87	1,9
22	y	1	0,2	0,11	0,077	0,059
23	y	1	0,4	0,33	0,31	0,29
24	y	2,25	3,37	5,06	7,59	11,4
25	y	2	2,69	3,1	3,39	3,61

Содержание отчета

Отчет по данной лабораторной работе должен содержать следующие обязательные разделы:

1) цель работы;

2) вариант задания и постановку задачи;

3) математические выкладки при получении системы уравнений, результирующие формулы для вычисления коэффициентов C_0 и C_1;

4) текст программы вычисления коэффициентов C_0, C_1 по заданным экспериментальным точкам (x_i, y_i) с достаточным для понимания количеством комментариев;

5) результаты расчетов. Выходными данными для программы будут

- величина уклонения d, для всех способов выравнивания;
- значения коэффициентов C_0, C_1 для эмпирической функции $y = \varphi(x, C_0, C_1)$;
- график функции $y = \varphi(x, C_0, C_1)$ и точек (x_i, y_i) на плоскости x, y;
- график функции $Y = C_0' + C_1' X$ и точек (X_i, Y_i) на плоскости X, Y;

6) выводы о проделанной работе.

Отчет по лабораторной работе выполняется письменно или в электронном виде с применением текстового редактора MS Word.

5 Курсовое проектирование

Целью курсового проектирования является: углубленное изучение методов и алгоритмов решения различных вычислительных задач; разработка программ для решения вычислительных задач.

5.1 Темы курсовых проектов

I Линейная алгебра

1) Решение СЛАУ методом Гаусса с частичным выбором главного элемента (в строке или столбце)

2) Решение СЛАУ методом простых итераций

3) Решение СЛАУ итерационным методом Гаусса-Зейделя

4) Обращение матрицы на основе метода Гаусса с частичным выбором главного элемента

Основные требования к программе:

- программа должна быть оформлена в виде самостоятельной процедуры (подпрограммы);
- входные данные задаются с клавиатуры и из внешнего текстового файла;
- результат решения выводится на экран и во внешний текстовый файл;
- для тестирования программы написать программу-оболочку, с помощью которой можно быстро изменять входные данные и просматривать результаты;
- проверить работоспособность программы на плохо обусловленной (близкой к вырождению) задаче. Для этого сформировать входную матрицу коэффициентов с одним или двумя маленькими диагональными элементами (порядка 10^{-6}-10^{-8});
- для сравнения результатов можно использовать различные системы математических расчетов, например MATLAB или MathCAD и пр.

II Приближение функций и аппроксимация данных

5) Аппроксимация периодических функций рядом Фурье.

6) Аппроксимация экспериментальных данных на основе метода наименьших квадратов – в качестве аппроксимирующей функции использовать полином степени m, т.е. $y = \Phi(\vec{C}, x) = C_0 + C_1 x + C_2 x^2 + \ldots + C_m x^m$.

7) Построение графика функции $y=f(x)$ с использованием кубических сплайнов.

Основные требования к программе:

- для задач 6 и 7 исходные данные считываются из внешнего текстового файла;
- необходимо построить графики исходных данных и результата приближения в одной системе координат;
- результаты вычислений (коэффициенты приближающих функций для задач 5 и 6) выводить на экран и во внешний текстовый файл

III Численное интегрирование функций

8) Программа расчета определенных интегралов методом прямоугольников (правых или левых и центральных)

9) Программа расчета определенных интегралов методом трапеций

10) Программа расчета определенных интегралов методом Симпсона (парабол)

Основные требования к программе:

- процедура интегрирования должна быть оформлена в виде самостоятельной подпрограммы;
- шаг интегрирования должен выбираться автоматически по заданной точности решения;
- выполнить тестирование подпрограммы на нескольких функциях;
- для тестирования подпрограммы написать программу-оболочку, с помощью которой можно осуществлять выбор тестирующей функции и просматривать результаты;

– необходимо исследовать влияние шага интегрирования на точность нахождения интеграла (построить график зависимости ошибки интегрирования от величины шага).

IV Нахождение корней (нулей) нелинейных уравнений

11) Нахождение нулей функции $y = f(x)$ методом дихотомии

12) Нахождение нулей функции $y = f(x)$ методом хорд

13) Нахождение нулей функции $y = f(x)$ методом Ньютона – для нахождения значений производной на каждом шаге можно использовать аппроксимацию конечными разностями

14) Нахождение нулей функции $y = f(x)$ модифицированным методом Ньютона

15) Нахождение нулей функции $y = f(x)$ методом секущей

Основные требования к программе:

– процедура нахождения нулей должна быть оформлена в виде самостоятельной подпрограммы;

– перед уточнением значения нуля указанным методом, необходимо выполнить предварительное отделение нулей на указанном интервале изменения функции;

– для тестирования подпрограммы написать программу-оболочку, с помощью которой можно осуществлять ввод исходных данных, выбор тестовых функций и просмотр результатов;

– для тестирования можно использовать несколько функций: тригонометрические функции, алгебраический полином и др.

V Решение системы нелинейных уравнений

16) Решение системы нелинейных уравнений (СНУ) методом Ньютона – для расчета элементов матрицы Якоби на каждом шаге использовать аппроксимацию частных производных конечно-разностными соотношениями

17) Решение системы нелинейных уравнений методом простых итераций

Основные требования к программе:

- процедура решения СНУ должна быть оформлена в виде самостоятельной подпрограммы;
- для тестирования подпрограммы написать программу-оболочку, с помощью которой можно осуществлять ввод исходных данных, выбор тестовых функций и просмотр результатов;
- начальное приближение задается самим пользователем или автоматически выбирается программой.

VI Дополнительные задачи повышенной трудности

18) Нахождение нулей алгебраического полинома – необходимо найти все корни (в том числе и комплексные).

19) Построение графика функции двух переменных $z=f(x, y)$ в виде поверхности – для сглаживания данных использовать триангуляцию или двухмерные сплайны

20) Построение выпуклой оболочки множества точек на плоскости – т.е. нахождение я границы множества точек.

Дополнительные указания:

Выбор среды реализации программы остается за разработчиком. Готовая программа должна быть самостоятельным приложением (т.е. в виде самостоятельно исполняемого exe файла). Приветствуется реализация в среде ОС Windows с использованием сред визуального программирования Delphi, Visual C++, C Builder, и пр.

Предложенные в качестве тем для курсовых проектов задачи не являются окончательными. Студент может выбрать в качестве темы курсового проекта (по согласованию с преподавателем) свою задачу, если она связана с реализацией определенного численного алгоритма или метода.

5.2 Общие требования к содержанию пояснительной записки курсовых проектов, связанных с разработкой программного обеспечения

Пояснительная записка (ПЗ) к курсовому проекту (работе), должна включать в себя следующие разделы:

1. **Титульный лист** – пример оформления титульного листа представлен в приложении Б.

2. Аннотация (Реферат) – содержит краткое описание (реферат) выполненной работы (2-3 предложения). Перечисляются ключевые слова, указывается количество страниц и приложений. Реферат размещают на отдельной странице. Заголовком служит слово «Реферат», написанное прописными буквами по центру страницы.

3. Задание на проектирование – формулируется по выбранному варианту. Пример оформления задания на проектирования представлен в приложении В.

4. Содержание ПЗ – нумерованный по страницам список разделов ПЗ. Нумерация страниц ПЗ – сквозная: титульный лист имеет первый номер. На листе с содержанием обычно рисуется рамка для текстовых документов (см. приложение В).

Замечание: по последним требованиям СТП ТУСУР рамки может и не быть, если курсовой проект не связан с разработкой какого-то конкретного устройства или блока.

5. Введение – содержит общую информацию по проекту: назначение программного продукта, необходимость его разработки и пр.

6. Постановка задачи – включает в себя анализ технического задания, представленного в разделе «задание на проектирование». Здесь формулируются требования к разрабатываемому программному продукту и среде разработки (т.е. осуществляем выбор среды реализации).

7. Обзор литературы – данный раздел содержит состояние проблемной области на момент начала работы. Включает в себя наличие аналогов, обзор

существующих алгоритмов, их достоинства и недостатки, обзор и краткие характеристики возможных средств реализации разработки программ и т.д.

8. Основной раздел ПЗ – содержит описание хода работы над программным продуктом и результаты тестирования готовой программы. Может состоять из нескольких подразделов. Желательно отразить следующие моменты:

1) Описание алгоритмов программы и используемых математических моделей (расчетные формулы, блок-схемы используемых алгоритмов, общее описание алгоритмов);

2) Описание реализации программы (структурная схема программы, структура и типы данных, основные процедуры, их назначение и взаимосвязь, входные и выходные данные программы и отдельных процедур, смысл основных (глобальных) переменных программы, построение файловой системы, используемые средства среды программирования и т.д.)

3) Описание программы для пользователя (описание меню, порядок работы с программой, входные и выходные данные, экранные формы и пр.) – данный раздел может быть оформлен в виде инструкции для пользователя.

4) Тестирование программы (описание тестовых задач и результатов тестирования).

9. Заключение – в данном разделе формулируем основной итог работы: сопоставление желаемых и полученных результатов, встретившиеся проблемы, целесообразность и направление дальнейшего совершенствования программного продукта и т.д.

10. Список литературы – список источников, используемых при работе над проектом. Может содержать не только литературные источники, но и ссылки на различные ресурсы в сети INTERNET. Пример оформления списка литературы представлен в Приложении Д.

11. Приложение – обычно в приложение выносят следующее:
блок-схемы алгоритмов, структурную схему и листинг программы, результаты тестирования, экранные формы разработанной программы и т. д.

В текст ПЗ следует включать не весь листинг программы целиком, а только текст процедур, выполняющих основные действия и расчеты.

5.3 Указания к оформлению ПЗ:

ПЗ пишется в редакторе MS Word шрифтом Times New Roman, размером 12-14 пунктов, на листе формата А4. Нумерация страниц должна быть сквозной, первой страницей является титульный лист (номер страницы на титульном листе не ставится). Номер страницы проставляется вверху справа. Заголовки разделов пишутся прописными буквами посередине текста. Заголовки подразделов пишутся с абзаца строчными буквами, кроме первой прописной. В заголовке не допускаются переносы слов. Точку в конце заголовка не ставят. Если заголовок состоит из двух предложений, то их разделяют точкой.

Необходимо обратить внимание на следующее:

1) наличие нумерации и подписей к рисункам, нумерация и заголовки таблиц;

2) наличие заголовков приложений и их сквозная нумерация;

3) выполнение блок-схем алгоритмов и структурных схем программы в соответствии с требованиями ГОСТа (СТП ТУСУР);

4) обязательное детальное комментирование текста программы;

5) оформление списка литературы в соответствии с требованиями ГОСТа;

6) наличие в тексте ссылок на используемую литературу (в тексте ПЗ в квадратных скобках пишется номер источника).

Оценка за курсовой проект выставляется с учетом качества выполнения программного продукта и пояснительной записки.

6 Практическая и самостоятельная работа

6.1 Общие указания

В ходе самостоятельных и практических занятий студентам необходимо выполнить 5 практических работ. При решении каждой задачи необходимо полностью приводить ход решения. Ответ на задачу пишется отдельной строкой под решением. Везде, где возможно, следует выполнять проверку полученного ответа.

Ответы даются письменно или в электронном варианте с использованием тестового редактора MS Word.

6.2 Практическая работа № 1. Представление чисел и погрешности при вычислениях на ЭВМ

Практическая работа № 1 выполняется после изучения разделов 2.1–2.2 и содержит 3 задачи.

Вариант – 1

1. Представить число 8912,342 в виде десятичной дроби (разложить по степеням числа 10).
2. Определить количество верных цифр в узком и широком смысле для приближенного числа a, если известна его предельная относительная погрешность $\overline{\delta}(a)$: $a = 1{,}8921;\ \overline{\delta}(a) = 0{,}1\%$.
3. Вычислить выражение и найти предельные абсолютную и относительную погрешности. В ответе сохранить все верные цифры и одну сомнительную. Все исходные числа даны с верными знаками в узком смысле.

$$Y = \frac{3{,}07 \cdot 326}{36{,}4 \cdot 323}.$$

Вариант – 2

1. Определить абсолютную погрешность $\Delta(a)$ приближенного числа a по его относительной погрешности $\delta(a)$: $a = 13267;\ \delta(a) = 0{,}1\%$.
2. Округлить сомнительные цифры числа a, оставив в нем верные знаки в узком смысле: $a = 47{,}453 \pm 0{,}024$.

3. Вычислить выражение и найти предельные абсолютную и относительную погрешности. В ответе сохранить все верные цифры и одну сомнительную.

$$X = \frac{a \cdot b}{\sqrt[3]{c}};$$

$$a = 3{,}85 \pm 0{,}001;\ b = 2{,}0435 \pm 0{,}0004;\ c = 962{,}6 \pm 0{,}01.$$

Вариант – 3

1. Определить количество верных цифр в узком и широком смысле для приближенного числа a, если известна его предельная абсолютная погрешность $\overline{\delta}(a)$: $a = 0{,}3941;\ \overline{\delta}(a) = 0{,}25 \cdot 10^{-2}$.
2. Округлить сомнительные цифры числа a, оставив в нем верные знаки в узком смысле, если $a = 3{,}2873$

   и предельная относительная погрешность $\overline{\delta}(a) = 0{,}1\%$.
3. Определить относительную погрешность вычисления объема цилиндра V, если погрешность измерения радиуса основания R равна 1% и погрешность измерения высоты H равна 1%.

Вариант – 4

1. Найти предельные абсолютную и относительную погрешности приближенного числа a, если оно имеет только верные знаки в узком смысле: $a = 0{,}7538$.
2. Определить, какое из измерений выполнено точнее – 80 км с ошибкой 20 м или 8 см с ошибкой 2 мм (сравнить относительные погрешности измерений).
3. При измерении радиуса R круга с точностью до 0,5 см получилось число 12 см. Найти предельные абсолютную и относительную погрешности при вычислении площади круга.

Вариант – 5

1. Округлить по правилу симметричного округления число 2,1514 последовательно до тысячных, сотых и десятых. Найти предельные абсолютную и относительную погрешности каждого результата.
2. С каким числом верных знаков в узком смысле следует взять числа arctg(6) и ln(30), чтобы относительная погрешность была не более 0,1% ?

3. Каждое ребро куба, измеренное с точностью до 0,02 см, оказалось равным 8 см. Найти предельные абсолютную и относительную погрешности при вычислении объема куба.

Вариант – 6

1. Определить, какое из приближенных равенств точнее (более точным является то равенство, предельная относительная погрешность которого меньше): $\frac{6}{25} \approx \frac{1}{4}$ или $\frac{1}{3} \approx 0{,}333$.
2. С каким числом верных знаков в узком смысле следует взять числа arctg(9) и ln(45), чтобы относительная погрешность была не более 0,1% ?
3. Найти предельные абсолютную и относительную ошибки при вычислении напряжения U по закону Ома ($U = I \cdot R$), если ток I равен $2 \pm 0{,}1$ и сопротивление R равно 10 Ом с точностью 10%.

Вариант – 7

1. Определить абсолютную погрешность $\Delta(a)$ приближенного числа a по его относительной погрешности $\delta(a)$: $a = 15278;\ \delta(a) = 0{,}2\%$.
2. Определить, какое из измерений выполнено точнее – 100 км с ошибкой 30 м или 9 см с ошибкой 4 мм (сравнить относительные погрешности измерений).
3. Реактивное сопротивление емкости (в Омах) задается формулой

$$X_C = 1 \big/ (2 \cdot \pi \cdot f \cdot C),$$

где f – частота в герцах,

C – емкость в фарадах.

Указать границы возможных значений X_C для $f = 400 \pm 1$ Гц и $C = 10^{-7}\,\Phi \pm 10\%$.

Вариант – 8

1. Определить количество верных цифр в узком и широком смысле для приближенного числа a, если известна его предельная абсолютная погрешность $\overline{\delta}(a)$: $a = 0{,}2937;\ \overline{\delta}(a) = 0{,}15 \cdot 10^{-2}$.

2. Определить, какое из приближенных равенств точнее (более точным является то равенство, предельная относительная погрешность которого меньше): $\frac{7}{15} \approx 0{,}467$ или $\sqrt{30} \approx 5{,}48$.
3. С какой точностью следует определить радиус основания R и высоту H цилиндрической банки, чтобы её вместимость можно было вычислить с точностью 1% ?

Вариант – 9

1. Представить число 6834,148 в виде десятичной дроби (разложить по степеням числа 10).
2. Определить количество верных цифр в узком и широком смысле для приближенного числа a, если известна его предельная относительная погрешность $\overline{\delta}(a)$: $a = 2{,}1756;\ \overline{\delta}(a) = 0{,}2\%$.
3. Найти допустимые абсолютные погрешности аргументов x_1 и x_2, которые позволяют вычислить значения функции f с 4 верными знаками:

$$f = \ln(x_1 + x_2{}^2);$$
$$x_1 = 3{,}2835, \quad x_2 = 0{,}93221.$$

Вариант – 10

1. Округлить сомнительные цифры числа a, оставив в нем верные знаки в узком смысле: $a = 23{,}927 \pm 0{,}016$.
2. Найти предельные абсолютную и относительную погрешности приближенного числа a, если оно имеет только верные знаки в узком смысле: $a = 0{,}2937$.
3. С каким числом верных знаков в узком смысле следует взять значения аргумента x, чтобы получить значение функции f с точностью 0,1%:

$$f = x^3 \cdot \sin x, \quad x = \sqrt{2}.$$

6.3 Практическая работа № 2. Решение СЛАУ

Практическая работа № 2 выполняется после изучения раздела 2.7 и содержит 3 задачи.

Вариант – 1

1. Для матрицы **A** вычислить обратную матрицу (использовать алгебраические дополнения элементов матрицы). Убедиться, что $\mathbf{A}\cdot\mathbf{A}^{-1}=\mathbf{E}$.

$$\mathbf{A}=\begin{bmatrix}1 & -3 & 2\\ 3 & -4 & 0\\ 2 & -5 & 3\end{bmatrix}.$$

2. Решить СЛАУ методом простой итерации, сделать 3 шага итерационного процесса (при необходимости для выполнения условий сходимости сначала преобразовать систему).

$$\left.\begin{aligned} x_1+3\cdot x_2-x_3 &= 5,\\ 2\cdot x_1-x_2+5\cdot x_3 &= 4,\\ 3\cdot x_1+x_2+x_3 &= -1. \end{aligned}\right\}$$

3. Решить СЛАУ методом Гаусса с выбором главного элемента в столбце:

$$\left.\begin{aligned} 2\cdot x_1-x_2+3\cdot x_3 &= -4,\\ x_1+3\cdot x_2-x_3 &= 2,\\ 5\cdot x_1+2\cdot x_2+x_3 &= 5. \end{aligned}\right\}$$

Вариант – 2

1. Для матрицы **A** вычислить нормы $\|\mathbf{A}\|_1$, $\|\mathbf{A}\|_2$, $\|\mathbf{A}\|_\infty$.

$$\mathbf{A}=\begin{bmatrix}-0{,}3 & 1{,}2 & -0{,}2\\ -0{,}1 & -0{,}2 & 1{,}6\\ -1{,}5 & -0{,}3 & 0{,}1\end{bmatrix}.$$

2. Решить СЛАУ методом обратной матрицы. Найти число обусловленности системы уравнений. Оценить возможное отклонение $\dfrac{\|\Delta x\|}{\|x\|}$ от полученного

решения, если допустить относительную ошибку в левой части $\frac{\|\Delta b\|}{\|b\|} = 0{,}01$ (использовать любую из норм).

$$\left.\begin{aligned} x_1 \qquad\quad - x_3 &= 2, \\ x_1 + x_2 + \quad x_3 &= 6, \\ x_2 + 3 \cdot x_3 &= 5. \end{aligned}\right\}$$

3. Решить СЛАУ методом Зейделя, сделать 3 шага итерационного процесса (при необходимости для выполнения условий сходимости сначала преобразовать систему).

$$\left.\begin{aligned} x_1 + 3 \cdot x_2 - x_3 &= 5, \\ 2 \cdot x_1 - x_2 + 5 \cdot x_3 &= 4, \\ 3 \cdot x_1 + x_2 + x_3 &= -1. \end{aligned}\right\}$$

Вариант – 3

1. Найти произведение **X**·**Y**, если:

а) $\mathbf{X} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$; $\mathbf{Y} = \begin{bmatrix} 4 & 5 \end{bmatrix}$; б) $\mathbf{X} = \begin{bmatrix} 10 & 17 & 8 \end{bmatrix}$; $\mathbf{Y} = \begin{bmatrix} 12 \\ 7 \\ 5 \end{bmatrix}$.

2. Решить систему уравнений: а) обычным методом Гаусса (схема единственного деления); б) методом Гаусса с выбором главного элемента. Все вычисления производить с точностью до 4-х значащих цифр. Вычислить число обусловленности системы уравнений (использовать любую из норм), сделать выводы.

$$\left.\begin{aligned} x + 592 \cdot y &= 437 \\ 592 \cdot x + 4308 y &= 2251 \end{aligned}\right\}$$

3. Показать, что для СЛАУ процесс решения по методу Зейделя сходится. Определить необходимое число итераций для нахождения корней системы с точностью до 10^{-4}.

$$\left.\begin{aligned} x_1 + 3 \cdot x_2 - x_3 &= 5, \\ 2 \cdot x_1 - x_2 + 5 \cdot x_3 &= 4, \\ 3 \cdot x_1 + x_2 + x_3 &= -1. \end{aligned}\right\}$$

Вариант – 4

1. Найти произведение **A**·**X**, если:

$$\mathbf{A}=\begin{bmatrix}1 & 2 & -4\\ 3 & 0 & -2\\ 2 & 2 & -3\end{bmatrix};\ \mathbf{X}=\begin{bmatrix}-2\\ -3\\ -4\end{bmatrix}.$$

2. Вычислить определитель методом Гаусса:

$$\Delta=\begin{bmatrix}4 & -3 & 2\\ 2 & 5 & -3\\ 5 & 6 & -2\end{bmatrix}.$$

3. Решить СЛАУ методом простой итерации, сделать 3 шага итерационного процесса (при необходимости для выполнения условий сходимости сначала преобразовать систему).

$$\left.\begin{aligned} 2\cdot x_1 + x_2 - 2\cdot x_3 &= 3,\\ 3\cdot x_1 + 5\cdot x_2 - 2\cdot x_3 &= 4,\\ -x_1 + 3\cdot x_2 + 4\cdot x_3 &= 3. \end{aligned}\right\}$$

Вариант – 5

1. Решить СЛАУ по методу обратной матрицы:

$$\left.\begin{aligned} 2\cdot x_1 + x_2 &= 5\\ 3\cdot x_1 - 2\cdot x_2 &= 4 \end{aligned}\right\}$$

2. Вычислить обратную матрицу методом Гаусса:

$$\mathbf{A}=\begin{bmatrix}2 & -1 & -1\\ 3 & 4 & -2\\ 3 & -2 & 4\end{bmatrix}.$$

3. Решить СЛАУ методом Зейделя, сделать 3 шага итерационного процесса (при необходимости для выполнения условий сходимости сначала преобразовать систему).

$$\left.\begin{aligned} 2\cdot x_1 + 4\cdot x_2 - 3\cdot x_3 &= 2,\\ -4\cdot x_1 + 2\cdot x_2 + 3\cdot x_3 &= 5,\\ 3\cdot x_1 + 2\cdot x_2 - 3\cdot x_3 &= -3. \end{aligned}\right\}$$

Вариант – 6

1. Для матрицы **A** вычислить обратную матрицу (использовать алгебраические дополнения элементов матрицы). Убедиться, что $\mathbf{A} \cdot \mathbf{A}^{-1} = \mathbf{E}$.

$$\mathbf{A} = \begin{bmatrix} 2 & 4 & -3 \\ -4 & 2 & 3 \\ 3 & 2 & -3 \end{bmatrix}.$$

2. Решить систему уравнений методом прогонки:

$$\left.\begin{aligned} 2 \cdot x_1 - x_2 &= -4, \\ x_1 + 3 \cdot x_2 - x_3 &= 2, \\ 2 \cdot x_2 + x_3 &= 5. \end{aligned}\right\}$$

3. Показать, что для СЛАУ процесс решения по методу Зейделя сходится. Определить необходимое число итераций для нахождения корней системы с точностью до 10^{-4}.

$$\left.\begin{aligned} 2 \cdot x_1 + 3 \cdot x_2 - x_3 &= 3, \\ 4 \cdot x_1 + 2 \cdot x_2 - 2 \cdot x_3 &= 4, \\ -4 \cdot x_1 + 5 \cdot x_2 + x_3 &= -3. \end{aligned}\right\}$$

Вариант – 7

1. Для матрицы **A** вычислить нормы $\mathbf{A}\|_1$, $\|\mathbf{A}\|_2$, $\|\mathbf{A}\|_\infty$.

$$\mathbf{A} = \begin{bmatrix} 0{,}2 & 0{,}6 & -0{,}3 \\ 0{,}3 & -0{,}2 & -0{,}4 \\ -1{,}2 & -0{,}2 & 0{,}4 \end{bmatrix}.$$

2. Решить систему уравнений: а) обычным методом Гаусса (схема единственного деления); б) методом Гаусса с выбором главного элемента. Все вычисления производить с точностью до 4-х значащих цифр. Вычислить число обусловленности системы уравнений (использовать любую из норм), сделать выводы.

$$\left.\begin{aligned} x + 347 \cdot y &= 129 \\ 285 \cdot x + 2430 y &= 1067 \end{aligned}\right\}$$

3. Решить СЛАУ методом простой итерации, сделать 3 шага итерационного процесса (при необходимости для выполнения условий сходимости сначала преобразовать систему).

$$\left.\begin{aligned} x_1 + 3\cdot x_2 - x_3 &= 5,\\ 2\cdot x_1 - x_2 + 5\cdot x_3 &= 4,\\ 3\cdot x_1 + x_2 + x_3 &= -1. \end{aligned}\right\}$$

Вариант – 8

1. Найти произведение **X**·**Y**, если:

а) $\mathbf{X} = \begin{bmatrix} -2 \\ 4 \\ 7 \end{bmatrix}$; $\mathbf{Y} = \begin{bmatrix} 3 & -2 & 8 \end{bmatrix}$; $\mathbf{X} = \begin{bmatrix} 12 & 5 & 14 \end{bmatrix}$; $\mathbf{Y} = \begin{bmatrix} 13 \\ 9 \\ 7 \end{bmatrix}$.

2. Решить СЛАУ методом обратной матрицы. Найти число обусловленности системы уравнений. Оценить возможное отклонение $\frac{\|\Delta x\|}{\|x\|}$ от полученного решения, если допустить относительную ошибку в левой части $\frac{\|\Delta b\|}{\|b\|} = 0{,}01$ (использовать любую из норм).

$$\left.\begin{aligned} 2\cdot x_1 - 2\cdot x_3 &= 1,\\ 3\cdot x_1 + x_2 + x_3 &= 4,\\ 2\cdot x_1 - x_2 + 2\cdot x_3 &= 2. \end{aligned}\right\}$$

3. Решить СЛАУ методом Зейделя, сделать 3 шага итерационного процесса (при необходимости для выполнения условий сходимости сначала преобразовать систему).

$$\left.\begin{aligned} 2\cdot x_1 + x_2 - 2\cdot x_3 &= 3,\\ 3\cdot x_1 + 5\cdot x_2 - 2\cdot x_3 &= 2,\\ -x_1 + 3\cdot x_2 + 4\cdot x_3 &= -3. \end{aligned}\right\}$$

Вариант – 9

1. Найти произведение **A**·**X**, если: $\mathbf{A} = \begin{bmatrix} 2 & 1 & -2 \\ 3 & 5 & -2 \\ -1 & 3 & 4 \end{bmatrix}$; $\mathbf{X} = \begin{bmatrix} 4 \\ -1 \\ 3 \end{bmatrix}$.

2. Решить СЛАУ методом Гаусса с выбором главного элемента в столбце:

$$\left.\begin{aligned} x_1 - 2\cdot x_2 - 3\cdot x_3 &= -2,\\ 5\cdot x_1 - 4\cdot x_2 + 2\cdot x_3 &= 1,\\ 3\cdot x_1 - x_2 + 2\cdot x_3 &= 3. \end{aligned}\right\}$$

3. Показать, что для СЛАУ процесс решения по методу Зейделя сходится. Определить необходимое число итераций для нахождения корней системы с точностью до 10^{-4}.

$$\left.\begin{aligned} 2\cdot x_1 + 3\cdot x_2 - x_3 &= 2,\\ 4\cdot x_1 + 2\cdot x_2 - 2\cdot x_3 &= -3,\\ -4\cdot x_1 + 5\cdot x_2 + x_3 &= -1. \end{aligned}\right\}$$

Вариант – 10

1. Решить СЛАУ методом обратной матрицы:

$$\left.\begin{aligned} 3\cdot x_1 + 2\cdot x_2 &= 7\\ 4\cdot x_1 - 2\cdot x_2 &= 3 \end{aligned}\right\}$$

2. Вычислить определитель методом Гаусса:

$$\Delta = \begin{bmatrix} 2 & 4 & -3 \\ -4 & 2 & 3 \\ 3 & 2 & -3 \end{bmatrix}.$$

3. Решить СЛАУ методом простой итерации, сделать 3 шага итерационного процесса (при необходимости для выполнения условий сходимости сначала преобразовать систему).

$$\left.\begin{aligned} 2\cdot x_1 + x_2 - 2\cdot x_3 &= 3,\\ 3\cdot x_1 + 5\cdot x_2 - 2\cdot x_3 &= 1,\\ -x_1 + 3\cdot x_2 + 4\cdot x_3 &= 4. \end{aligned}\right\}$$

6.4 Практическая работа № 3. Решение нелинейных уравнений

Практическая работа № 3 выполняется после изучения разделов 2.8–2.9 и содержит 2 задачи.

Вариант – 1

1. Вычислить методом дихотомии (деления отрезка пополам) корень уравнения $x^3 - x^2 + 3 = 0$ на интервале [–2; –1]. Сделать три итерации.
2. Решить систему уравнений $\begin{cases} f_1(x_1, x_2) = x_1^3 - x_2 = 0 \\ f_2(x_1, x_2) = x_1^2 + x_2^2 - 1 = 0 \end{cases}$ методом Ньютона-Рафсона. Начальная точка $x^{(0)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. Сделать две итерации указанным методом. Оценить ошибку решения.

Вариант – 2

1. Вычислить методом хорд корень уравнения $x^3 - x^2 + 3 = 0$ на интервале [–2; –1]. Сделать три итерации.
2. Решить систему уравнений $\begin{cases} f_1(x_1, x_2) = x_1 - x_2^2 = 0 \\ f_2(x_1, x_2) = x_1^2 + x_2^2 - 1 = 0 \end{cases}$ модифицированным методом Ньютона. Начальная точка $x^{(0)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. Сделать две итерации указанным методом. Оценить ошибку решения.

Вариант – 3

1. Вычислить методом Ньютона корень уравнения $x^3 - x^2 + 3 = 0$ на интервале [–2; –1]. Сделать три итерации.
2. Решить систему уравнений $\begin{cases} f_1(x_1, x_2) = x_1 - x_2^2 = 0 \\ f_2(x_1, x_2) = x_1^2 + x_2^2 - 1 = 0 \end{cases}$ разностным методом Ньютона. Начальная точка $x^{(0)} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ шаг $h = 0{,}1$. Сделать две итерации указанным методом. Оценить ошибку решения.

Вариант – 4

1. Вычислить модифицированным методом Ньютона корень уравнения $x^3 - x^2 + 3 = 0$ на интервале [–2; –1]. Сделать три итерации.
2. Решить систему уравнений $\begin{cases} f_1(x_1, x_2) = x_1 - x_2^2 = 0 \\ f_2(x_1, x_2) = x_1^2 + x_2^2 - 1 = 0 \end{cases}$ методом секущих.

   Начальная точка $x^{(1)} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}; x^{(0)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix};$. Сделать две итерации указанным методом. Оценить ошибку решения.

Вариант – 5

1. Вычислить методом секущих корень уравнения $x^3 - x^2 + 3 = 0$ на интервале. [–2; –1]. Сделать три итерации, на первой итерации выбрать две точки на расстоянии $h = 0{,}1$.
2. Решить систему уравнений $\begin{cases} f_1(x_1, x_2) = x_1 - x_2^2 = 0 \\ f_2(x_1, x_2) = x_1^2 + x_2^2 - 1 = 0 \end{cases}$ методом Ньютона-Рафсона. Начальная точка $x^{(0)} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$. Сделать две итерации указанным методом. Оценить ошибку решения.

Вариант – 6

1. Уточнить корень уравнения $x^2 - 6 = 0$ методом Ньютона и модифицированным методом Ньютона. Сделать три итерации, начальная точка $x_0 = 3$.
2. Решить систему уравнений $\begin{cases} f_1(x_1, x_2) = x_1^2 - x_2 = 0 \\ f_2(x_1, x_2) = x_1^2 + x_2^2 - 1 = 0 \end{cases}$ разностным методом Ньютона. Величина шага $h = 0{,}1$. Начальная точка $x^{(0)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. Сделать две итерации указанным методом. Оценить ошибку решения.

Вариант – 7

1. Вычислить методом дихотомии (деления отрезка пополам) корень уравнения $2 \cdot x^3 - 3 \cdot x^2 + 2 = 0$ на интервале $[-5; -2]$. Сделать три итерации.
2. Решить систему уравнений $\begin{cases} f_1(x_1, x_2) = x_1 - x_2^2 = 0 \\ f_2(x_1, x_2) = x_1^2 + x_2^2 - 1 = 0 \end{cases}$ методом Ньютона-Рафсона. Начальная точка $x^{(0)} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$. Сделать две итерации указанным методом. Оценить ошибку решения.

Вариант – 8

1. Вычислить методом хорд корень уравнения $2 \cdot x^3 - x^2 + 1 = 0$ на интервале $[-3; -1]$. Сделать три итерации.
2. Решить систему уравнений $\begin{cases} f_1(x_1, x_2) = x_1 - x_2^2 = 0 \\ f_2(x_1, x_2) = x_1^2 + x_2^2 - 1 = 0 \end{cases}$ методом простых итераций. Начальная точка $x^{(0)} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$. Сделать две итерации указанным методом. Оценить ошибку решения.

Вариант – 9

1. Вычислить методом Ньютона корень уравнения $x^3 - 2 \cdot x^2 + 2 = 0$ на интервале $[-3; -2]$. Сделать три итерации.
2. Решить систему уравнений $\begin{cases} f_1(x_1, x_2) = x_1^3 - x_2 = 0 \\ f_2(x_1, x_2) = x_1^2 + x_2^2 - 1 = 0 \end{cases}$ модифицированным методом Ньютона. Начальная точка $x^{(0)} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$. Сделать две итерации указанным методом. Оценить ошибку решения.

Вариант – 10

1. Вычислить модифицированным методом Ньютона корень уравнения $2 \cdot x^3 - 3 \cdot x^2 + 1 = 0$ на интервале $[-3; -2]$. Сделать три итерации.
2. Решить систему уравнений $\begin{cases} f_1(x_1, x_2) = x_1^2 + x_2^2 - 1 = 0 \\ f_2(x_1, x_2) = x_1^2 + 2x_1 + x_2^2 = 0 \end{cases}$ методом простых итераций. Начальная точка $x^{(0)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. Сделать две итерации указанным методом. Оценить ошибку решения.

6.5 Практическая работа № 4. Интерполяция и аппроксимация данных

Практическая работа № 3 выполняется после изучения разделов 2.8–2.9 и содержит 2 задачи.

Вариант – 1

1. Функция $y = x^3$ интерполируется на интервале [2; 4] полиномом

$$P_2(x) = c_0 + c_1 \cdot x + c_2 \cdot x^2$$

1) Найти оптимальные (чебышевские) узлы интерполяции.

2) Оценить ошибку интерполяции при таких узлах.

3) Выполнить интерполяцию и сравнить оценку с действительной ошибкой в точке $x = 3{,}5$.

2. Экспериментально получена зависимость дальности приема радиосигнала D от интенсивности осадков I:

I, мм/час	10	30	50
D, км	100	60	30

С использованием МНК найти приближенную зависимость $D(I)$ виде квадратного трехчлена.

Вариант – 2

1. Методом интерполяции найти первую производную функции $y = f(x)$ в точке $x = 3$. Функция задана числовыми значениями. Указание: вначале найти интерполирующий полином.

x	2	4	6
y	1	3	8

2. Экспериментально получена зависимость дальности приема радиосигнала D от интенсивности осадков I:

I, мм/час	10	30	50
D, км	100	60	30

С использованием МНК найти приближенную зависимость $D(I)$ в виде квадратного трехчлена.

Вариант – 3

1. Функция $y = \sqrt{x}$ интерполируется полиномом $P(x)$ в точках

$$x_0 = 100; x_1 = 121;\ x_2 = 144.$$

Оцените, с какой точностью с помощью полинома можно вычислить $\sqrt{117}$? Выполнить интерполяцию и сравнить оценку с действительной ошибкой.

2. Для функции $y = f(x)$, заданной таблицей, с помощью МНК найдите коэффициенты аппроксимирующей её нелинейной зависимости $y = a \cdot x^b$.

Указание: вначале выполните преобразование аппроксимирующей функции к линейному виду.

x	1	2	3
y	4	6	12

Вариант – 4

1. Функция $y = x^3$ интерполируется полиномом $P(x)$ в точках $x = 1; 2; 3$.

   1) Найти коэффициенты полинома.

   2) Оценить ошибку интерполяции в точке $x = 2{,}5$.

   3) Сравнить оценку с действительной ошибкой.

2. Экспериментально получена зависимость дальности приема D сигнала от мощности передатчика $P_{\text{п}}$. С использованием МНК найти приближенную зависимость $D(P_{\text{п}})$ в виде квадратного трехчлена.

$P_{\text{п}}$, кВт	2	3	4	5
D, км	120	190	230	260

Вариант – 5

1. Дана таблично заданная функция $y = f(x)$. Пользуясь формулой Лагранжа, найти значение функции в точке $x = 2{,}75$.

x	2	2,5	3	3,5
y	5	7	4	1

2. Экспериментально получена зависимость дальности приема D сигнала от мощности передатчика $P_{\text{п}}$. С использованием МНК найти приближенную зависимость $D(P_{\text{п}})$ в виде квадратного трехчлена.

$P_{\text{п}}$, кВт	1	2	3	4
D, км	100	150	170	200

Вариант – 6

1. Для таблично заданной функции $y = \mathrm{e}^x$ построить интерполяционный полином Ньютона $P_2(x)$ второго порядка и с его помощью вычислить $e^{3{,}62}$. Оценить ошибку вычисленного значения и сравнить оценку с действительной ошибкой.

x	3,60	3,65	3,70	3,75
y	36,598	38,475	40,447	42,521

2. Для функции $y = f(x)$, заданной таблицей, с помощью МНК найдите коэффициенты аппроксимирующей её нелинейной зависимости $y = a \cdot x^b$.

Указание: вначале выполните преобразование аппроксимирующей функции к линейному виду.

x	1	3	7
y	6	10	18

Вариант – 7

1. Для функции $y = f(x)$, заданной числовыми значениями, найти коэффициенты интерполирующего полинома $P(x)$ непосредственно из условий интерполяции.

x	2	3	4
y	5	8	17

2. Экспериментально получена зависимость дальности приема D сигнала от мощности передатчика $P_{\text{п}}$. С использованием МНК найти приближенную зависимость $D(P_{\text{п}})$ в виде квадратного трехчлена.

$P_{\text{п}}$, кВт	1	2	3	4
D, км	90	120	160	200

Вариант – 8

1. Функция $y = x^3$ интерполируется на интервале [3; 5] полиномом

$$P_2(x) = c_0 + c_1 \cdot x + c_2 \cdot x^2.$$

Найти оптимальные (чебышевские) узлы интерполяции. Оценить ошибку интерполяции при таких узлах. Выполнить интерполяцию и сравнить оценку с действительной ошибкой в точке $x = 3{,}5$.

2. Экспериментально получена зависимость дальности приема радиосигнала D от интенсивности осадков I:

I, мм/час	20	40	60
D, км	150	90	40

С использованием МНК найти приближенную зависимость $D(I)$ в виде квадратного трехчлена.

Вариант – 9

1. Методом интерполяции найти первую производную функции $y = f(x)$ в точке $x = 5$. Функция задана числовыми значениями. Указание: вначале найти интерполирующий полином.

x	3	6	8
y	5	4	3

2. Для функции $y = f(x)$, заданной таблицей, с помощью МНК найдите коэффициенты аппроксимирующей её нелинейной зависимости $y = a \cdot x^b$.

x	2	3	4
y	7	9	14

Указание: вначале выполните преобразование аппроксимирующей функции к линейному виду.

Вариант – 10

1. Функция $y = \sqrt[3]{x}$ интерполируется полиномом $P(x)$ в точках

$$x_0 = 50; x_1 = 110; \; x_2 = 180.$$

Оцените, с какой точностью с помощью полинома можно вычислить $\sqrt[3]{147}$? Выполнить интерполяцию и сравнить оценку с действительной ошибкой.

2. Экспериментально получена зависимость дальности приема радиосигнала D от интенсивности осадков I:

3.

I, мм/час	15	25	55
D, км	90	50	20

С использованием МНК найти приближенную зависимость $D(I)$ в виде квадратного трехчлена.

6.6 Практическая работа № 5. Численное вычисление производных и интегралов

Практическая работа № 5 выполняется после изучения раздела 2.5 и содержит 2 задачи.

Вариант – 1

1. Функция $y = f(x)$ задана таблицей. В точке $x = 0{,}1$ найти значения $f'(x)$ и $f''(x)$ по формулам левой, правой и центральной разностных производных.

x	0,00	0,10	0,15
y	0,0000	0,20134	0,30452

2. Вычислить интеграл $\int_0^1 x^2 dx$ по формулам прямоугольников, трапеций, Симпсона (взять 5 значений подынтегральной функции, т.е. $n = 4$). Рассчитать оценки погрешности вычислений интеграла, сравнить с действительными погрешностями.

Вариант – 2

1. Дан полином

$$P(x) = 3 \cdot x^2 - 2 \cdot x - 1.$$

Вычислить при x =2 (шаг $h = 0{,}1$):

1) $P'(x)$ по формулам левой, правой и центральной разностных производных.

2) $P'(x)$ с помощью интерполяции по 5 узлам.

3) $P''(x)$ по формуле центральной разностной производной.

Для всех случаев рассчитать оценки погрешности вычисления производных, сравнить с действительными погрешностями.

2. Определить, с какой точностью можно вычислить $\int_0^1 \sin(e^x)dx$, используя 9 значений подынтегральной функции (n = 8):

   1) по формуле прямоугольников;

   2) по формуле трапеций;

   3) по формуле Симпсона.

Вариант – 3

1. Функция $y = f(x)$ задана таблицей. В точке x=0,15 найти значения $f'(x)$ и $f''(x)$ по формулам левой, правой и центральной разностных производных.

x	0,00	0,15	0,20
y	0,0000	0,40389	0,23091

2. Сколько узлов следует взять для вычисления интеграла $\int_0^1 \frac{dx}{1+x}$ с точностью 10^{-4} при использовании формулы Симпсона ?

Вариант – 4

1. Функция $y = f(x)$ задана таблицей. В точке $x = 0{,}12$ найти значения $f'(x)$ и $f''(x)$ по формулам левой, правой и центральной разностных производных.

x	0,00	0,12	0,24
y	0,0000	0,34045	0,76507

2. Определить, с какой точностью можно вычислить $\int_0^1 \cos(\mathrm{e}^x)dx$, используя 9 значений подынтегральной функции (n = 8):

   1) по формуле прямоугольников;

   2) по формуле трапеций;

   3) по формуле Симпсона.

Вариант – 5

1. Дан полином $P(x) = 4 \cdot x^2 - 2 \cdot x - 5$.

   Вычислить при $x = 3$ (шаг $h = 0{,}1$):

   1) $P'(x)$ по формулам левой, правой и центральной разностных производных.

   2) $P'(x)$ с помощью интерполяции по 5 узлам.

   3) $P''(x)$ по формуле центральной разностной производной.

   Для всех случаев рассчитать оценки погрешности вычисления производных, сравнить с действительными погрешностями.

2. Определить, с какой точностью можно вычислить $\int_0^1 \mathrm{tg}(\mathrm{e}^x)dx$, используя 9 значений подынтегральной функции ($n = 8$):

   1) по формуле прямоугольников;

   2) по формуле трапеций;

   3) по формуле Симпсона

Вариант – 6

1. Функция $y = f(x)$ задана таблицей. В точке $x = 0{,}2$ найти значения $f'(x)$ и $f''(x)$ по формулам левой, правой и центральной разностных производных.

x	0,00	0,20	0,35
y	0,0000	0,43251	0,47819

2. Сколько узлов следует взять для вычисления интеграла $\int_0^1 \frac{dx}{1-x}$ с точностью 10^{-4} при использовании формулы Симпсона ?

Вариант – 7

3. Функция $y = f(x)$ задана таблицей. В точке $x = 0{,}15$ найти значения $f'(x)$ и $f''(x)$ по формулам левой, правой и центральной разностных производных.

x	0,00	0,15	0,25
y	0,0000	0,12765	0,25034

4. Вычислить интеграл $\int_0^1 2 \cdot x^3 dx$ по формулам прямоугольников, трапеций, Симпсона (взять 5 значений подынтегральной функции, т.е. $n = 4$). Рассчитать оценки погрешности вычислений интеграла, сравнить с действительными погрешностями.

Вариант – 8

1. Дан полином $P(x) = 2 \cdot x^2 - 4 \cdot x - 3$. Вычислить при x =2 (шаг h = 0,1):

   1) $P'(x)$ по формулам левой, правой и центральной разностных производных.

   2) $P'(x)$ с помощью интерполяции по 5 узлам.

   3) $P''(x)$ по формуле центральной разностной производной.

   Для всех случаев рассчитать оценки погрешности вычисления производных, сравнить с действительными погрешностями.

2. Определить, с какой точностью можно вычислить $\int_0^1 \mathrm{ctg}(\mathrm{e}^{2 \cdot x}) dx$, используя 9 значений подынтегральной функции (n = 8):

   1) по формуле прямоугольников;

   2) по формуле трапеций;

   3) по формуле Симпсона.

Вариант – 9

1. Функция $y = f(x)$ задана таблицей. В точке $x = 0{,}2$ найти значения $f'(x)$ и $f''(x)$ по формулам левой, правой и центральной разностных производных.

x	0,00	0,20	0,30
y	0,0000	0,10283	0,20654

2. Сколько узлов следует взять для вычисления интеграла $\int_0^1 \frac{dx}{2 + 3 \cdot x}$ с точностью 10^{-4} при использовании формулы Симпсона ?

Вариант – 10

1. Дан полином $P(x) = 3 \cdot x^2 + 5 \cdot x - 2$. Вычислить при $x = 2$ (шаг $h = 0{,}1$):

1) $P'(x)$ по формулам левой, правой и центральной разностных производных.

2) $P'(x)$ с помощью интерполяции по 5 узлам.

3) $P''(x)$ по формуле центральной разностной производной.

Для всех случаев рассчитать оценки погрешности вычисления производных, сравнить с действительными погрешностями.

2. Сколько узлов следует взять для вычисления интеграла $\int_0^1 \frac{dx}{x-1}$ с точностью 10^{-4} при использовании формулы Симпсона ?

7 Примеры решения практических работ

Рассмотрим решение задач, аналогичных представленным в разделе 6.

Задача 1

Округлить по правилу симметричного округления число a^*=3,1415926 последовательно до тысячных, сотых и десятых. Найти предельные абсолютную и относительную погрешности каждого результата.

Решение:

При симметричном округлении за предельную абсолютную погрешность приближенного числа a принимается половина единицы последнего сохраняемого разряда числа. Таким образом, при симметричном округлении числа a^*=3,1415926 получим следующий результат:

a_1=3,1 – предельная абсолютная погрешность равна $\overline{\Delta}(a_1)$=0,05;

a_2=3,14 – предельная абсолютная погрешность равна $\overline{\Delta}(a_2)$=0,005;

a_3=3,142– предельная абсолютная погрешность равна $\overline{\Delta}(a_1)$=0,0005;

Предельную относительную погрешность числа можно определить по формуле:

$$\overline{\delta}(a)=\frac{\overline{\Delta}(a)}{|a|}.$$

Для нашей задачи получим:

$$\overline{\delta}(a_1)=\frac{\overline{\Delta}(a_1)}{|a_1|}=\frac{0{,}05}{3{,}1}=0{,}016129\approx 1{,}61\%;$$

$$\overline{\delta}(a_2)=\frac{\overline{\Delta}(a_2)}{|a_2|}=\frac{0{,}005}{3{,}14}=0{,}001592\approx 0{,}16\%;$$

$$\overline{\delta}(a_3)=\frac{\overline{\Delta}(a_3)}{|a_3|}=\frac{0{,}0005}{3{,}142}=0{,}000159\approx 0{,}016\%$$

Задача 2

С каким числом верных знаков в узком смысле следует взять число arcsin(1), чтобы относительная погрешность была не более 0,1% ?

Решение:

Решение вначале вычислим:

$$a^*=\arcsin(1)=\pi/2\approx 1{,}57079\ \ldots$$

По значению предельной относительной погрешности

$$\bar{\delta}(a^*)=0{,}1\ \%=0{,}001$$

найдем предельную относительную погрешность числа a^*:

$$\bar{\Delta}(a^*) = a^* \cdot \bar{\delta}(a^*) = 1{,}5709\ldots \cdot 0{,}001 \approx 0{,}0158$$

Представим число a^* в виде

$a^* = 1{,}57079\ldots = 1\cdot 10^0 + 5\cdot 10^{-1} + 7\cdot 10^{-2} + 0\cdot 10^{-3} + 7\cdot 10^{-4} + 9\cdot 10^{-5} + \ldots$ и составим таблицу

Цифра	1	5	7	0	7	9	…
Единица разряда	1	0,1	0,01	0,001	0,0001	0,00001	…
Половина единицы разряда	0,5	0,05	0,005	0,0005	0,00005	0,000005	…

Сравнивая для каждой цифры половину единицы разряда с абсолютной погрешностью числа $\bar{\Delta}(a^*) = 0{,}00158$, получим, что a^* следует записать в виде a=1,57.

Проверка: $\delta(a) = \dfrac{|a^* - a|}{|a|} = \dfrac{|1{,}57079... - 1{,}57|}{1{,}57} \approx 0{,}0005 = 0{,}05\ \% < 0{,}1\ \%.$

Ответ: arcsin(1)=1,57 – число верных знаков (в узком смысле) равно 3.

Задача 3

Найти допустимые абсолютные погрешности аргументов x_1 и x_2, которые позволяют вычислить функцию f с точностью $\bar{\delta}(f) = 0{,}2\ \%$:

$$f = \sqrt{x_1^2 + x_2^2};$$
$$x_1 = 2{,}4563, \quad x_2 = 0{,}8473.$$

Решение:

Значение функции при x_1=2,4563 и x_2=0,8473 равно $f(x_1, x_2) \approx 2{,}59833$. Предельную абсолютную погрешность $\bar{\Delta}(f)$ можно оценить следующим образом:

$$\overline{\Delta}(f) = |\ \overline{\delta}(f) \cdot f(x_1, x_2)\ | = 0{,}002 \cdot 2{,}59833 \approx 0{,}0051967 = 0{,}006.$$

Используя принцип равных влияний для предельных абсолютных погрешностей аргументов заданной функции, получим:

$$\overline{\Delta}(x_1) = \frac{1}{2}\frac{\overline{\Delta}(f)}{\left|\frac{\partial f}{\partial x_1}\right|} = \frac{1}{2}\frac{\overline{\Delta}(f)}{\sqrt{\frac{x_1}{x_1^2 + x_2^2}}} = \frac{1}{2}\frac{0{,}006}{\sqrt{\frac{2{,}4563}{2{,}4563^2 + 0{,}8473^2}}} \approx 0{,}00497;$$

$$\overline{\Delta}(x_2) = \frac{1}{2}\frac{\overline{\Delta}(f)}{\left|\frac{\partial f}{\partial x_2}\right|} = \frac{1}{2}\frac{\overline{\Delta}(f)}{\sqrt{\frac{x_2}{x_1^2 + x_2^2}}} = \frac{1}{2}\frac{0{,}006}{\sqrt{\frac{0{,}8473}{2{,}4563^2 + 0{,}8473^2}}} \approx 0{,}00847.$$

Таким образом, величины аргументов x_1=2,4563 и x_2=0,8473 следует брать с предельными абсолютными погрешностями $\overline{\Delta}(x_1) = 0{,}005$ и $\overline{\Delta}(x_2) = 0{,}0085$.

Ответ: $\overline{\Delta}(x_1) = 0{,}005$; $\overline{\Delta}(x_2) = 0{,}0085$.

Задача 4

Для матрицы $\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ вычислить число обусловленности cond(**A**).

Решение: число обусловленности можно найти по формуле

$$\text{cond}(\mathbf{A}) = \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\|,$$

где $\|\ \mathbf{A}\ \|$ – норма матрицы **A**; $\|\mathbf{A}^{-1}\|$ – норма обратной матрицы $\mathbf{A}^{-1}$.

Найдем обратную матрицу, используя метод алгебраических дополнений.

Обратную к $\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}$ матрицу $\mathbf{A}^{-1}$ можно получить по формуле

$$\mathbf{A}^{-1} = \frac{1}{\Delta}\begin{bmatrix} A_{11} & A_{21} & \dots & A_{n1} \\ A_{12} & A_{22} & \dots & A_{2n} \\ \vdots & & & \vdots \\ A_{1n} & A_{2n} & \dots & A_{nn} \end{bmatrix},$$

где Δ – определитель матрицы **A**;

A_{ij} – алгебраическое дополнение элемента a_{ij}.

$$A_{ij} = (-1)^{(i+j)} \cdot M_{ij},$$

где M_{ij} – минор элемента a_{ij}, т.е. определитель $(n-1)$-го порядка, получаемый из определителя Δ вычеркиванием i-й строки и j-го столбца.

Для нашей задачи $\Delta = 1 \cdot 4 - 2 \cdot 3 = -2$.

$A_{11}=(-1)^{1+1} \cdot 4 = 4$; $A_{12}=(-1)^{1+2} \cdot 3 = -3$; $A_{21}=(-1)^{2+1} \cdot 2 = -2$; $A_{22}=(-1)^{2+2} \cdot 1 = 1$.

Отсюда получим обратную матрицу $\mathbf{A}^{-1} = \frac{1}{-2}\begin{bmatrix} 4 & -2 \\ -3 & 1 \end{bmatrix} = \begin{bmatrix} -2 & 1 \\ 1{,}5 & -0{,}5 \end{bmatrix}$.

Используя разные нормы матриц (∞-норму и 1-норму), найдем число обусловленности матрицы **A**.

а) ∞-норма: $\|\mathbf{A}\|_\infty = \max\limits_{i=1,n} \{\sum\limits_{j=1}^{n} |a_{ij}|\}$ – максимальная сумма элементов строк.

Для нашей задачи:

$$\|\mathbf{A}\|_\infty = \max\{\,|1|+|2|,\,|3|+|4|\,\} = \{3, 7\} = 7;$$

$$\|\mathbf{A}^{-1}\|_\infty = \max\{\,|-2|+|1|,\,|1{,}5|+|-0{,}5|\,\} = \max\{3, 2\} = 3.$$

б) 1-норма: $\|\mathbf{A}\|_1 = \max\limits_{j=1,n} \{\sum\limits_{i=1}^{n} |a_{ij}|\}$ - максимальная сумма элементов столбцов.

Для нашей задачи:

$$\|\mathbf{A}\|_1 = \max\{\,|1|+|3|,\,|2|+|4|\,\} \;\; \max\{4, 6\} = 6;$$

$$\|\mathbf{A}^{-1}\|_1 = \max\{\,|-2|+|1{,}5|,\,|1|+|-0{,}5|\,\} = \max\{3{,}5,\, 1{,}5\} = 3{,}5.$$

По полученным нормам рассчитаем число обусловленности матрицы

$$\mathrm{cond}(\mathbf{A}) = \|\mathbf{A}\|_\infty \cdot \|\mathbf{A}^{-1}\|_\infty = 7 \cdot 3 = 21;$$

$$\mathrm{cond}(\mathbf{A}) = \|\mathbf{A}\|_1 \cdot \|\mathbf{A}^{-1}\|_1 = 6 \cdot 3{,}5 = 21.$$

Ответ: число обусловленности равно $\mathrm{cond}(\mathbf{A})=21$.

Задача 5

Решить систему методом Гаусса с выбором главного элемента в строке:

$$\begin{cases} 2x_1 + x_2 + x_3 = 0 \\ 5x_1 + x_2 - x_3 = -5 \\ -2x_1 + x_2 - 2x_3 = 1 \end{cases}$$

Решение:

1 Прямой ход

Первый шаг:

а) выполним выбор главного элемента в 1-м столбце, получим новую систему

$$\begin{cases} 5x_1 + x_2 - x_3 = -5 \\ 2x_1 + x_2 + x_3 = 0 \\ -2x_1 + x_2 - 2x_3 = 1 \end{cases}$$

перестановкой 1-й и 2-й строк в исходной СЛАУ.

б) для исключения переменной x_1 из второго уравнения умножим первое уравнение на коэффициент $-\left(\frac{a_{21}}{a_{11}}\right) = -\frac{2}{5}$, и сложим его со вторым уравнением:

$$2\,x_1 - \frac{2\cdot 5}{5}x_1 + x_2 - \frac{2\cdot 1}{5}x_2 + x_3 - \frac{2\cdot(-1)}{5}x_3 = 0 - \frac{2\cdot(-5)}{5}.$$

Теперь упростим:

$$0\,x_1 + \left(1 - \frac{2}{5}\right)x_2 + \left(1 + \frac{2}{5}\right)x_3 = 0 + 2;$$

$$0\,x_1 + 0{,}6\,x_2 + 1{,}4\,x_3 = 2.$$

в) для исключения x_1 из третьего уравнения умножим первое уравнение на коэффициент $-\left(\frac{a_{31}}{a_{11}}\right) = \frac{2}{5}$ и сложим его с 3-м уравнением:

$$-2\,x_1 + \frac{2\cdot 5}{5}x_1 + x_2 + \frac{2\cdot 1}{5}x_2 - 2\,x_3 + \frac{2\cdot(-1)}{5}x_3 = 1 + \frac{2\cdot(-5)}{5};$$

$$0\,x_1 + \left(1 + \frac{2}{5}\right)x_2 + \left(-2 - \frac{2}{5}\right)x_3 = 1 - 2;$$

$$0\,x_1+1{,}4\,x_2-2{,}4x_3=-1.$$

В результате этих преобразований после первого шага получим:

$$\begin{cases} 5x_1 + \quad x_2 - \quad x_3 = -5 \\ 0x_1 + 0{,}6x_2 + 1{,}4x_3 = 2 \\ 0x_1 + 1{,}4\,x_2 - 2{,}4x_3 = -1 \end{cases}.$$

Видно, что коэффициенты при x_1 во втором и третьем уравнении равны нулю.

Второй шаг:

а) выполним выбор главного элемента во втором столбце (первую строку не трогаем), получим новую систему

$$\begin{cases} 5x_1 + \quad x_2 - \quad x_3 = -5 \\ 0x_1 + 1{,}4\,x_2 - 2{,}4x_3 = -1 \\ 0x_1 + 0{,}6x_2 + 1{,}4x_3 = 2 \end{cases}$$

перестановкой 2-й и 3-й строк системы.

в) теперь с помощью второго уравнения можно исключить переменную x_2 из третьего уравнения системы. Умножим второе уравнение на коэффициент $-\left(\frac{a_{32}^{(1)}}{a_{22}^{(1)}}\right)=-\frac{0{,}6}{1{,}4}$ и сложим его с 3-м уравнением:

$$0{,}6\,x_2+\frac{(-0{,}6)\cdot 1{,}4}{1{,}4}x_2+1{,}4\,x_3+\frac{(-0{,}6)\cdot(-2{,}4)}{1{,}4}x_3=2+\frac{0{,}6\cdot(-1)}{1{,}4};$$

$$0\,x_2+2{,}4286\,x_3=2{,}4286.$$

В результате получаем следующую систему:

$$\begin{cases} 5x_1 + \quad x_2 - \quad\quad x_3 = -5 \\ 0x_1 + 1{,}4\,x_2 - \quad 2{,}4x_3 = -1 \\ 0x_1 + \quad 0x_2 + 2{,}4286x_3 = 2{,}4286 \end{cases}.$$

2 Обратный ход

Вторая часть метода Гаусса, называемая обратным ходом, заключается в последовательной подстановке найденных переменных x_3 и x_2 во второе и первое уравнения СЛАУ и их решения.

Первый шаг:

а) находим значение переменной x_3 из последнего уравнения

$$x_3=2{,}4286/2{,}4286=1;$$

б) подставляем полученное значение x_3 во второе уравнение системы:

$$0\,x_1+1{,}4\,x_2-2{,}4\cdot 1=-1,$$

решая которое, получим значение $x_2=(-1+2{,}4)/1{,}4 = 1$.

Второй шаг:

Подставляем найденные значения x_3 и x_2 в первое уравнение системы:

$$5\,x_1+1\cdot 1-1\cdot 1=-5;$$

$$x_1=(-5)/5=-1;$$

$$x_1=-1.$$

В результате получаем решение заданной системы линейных уравнений $\boldsymbol{x}=[-1,\ 1,\ 1]^{\mathrm{T}}$.

Проверка: $\begin{cases} 2\cdot(-1)+1+1=0 \\ 5\cdot(-1)+1-1=-5 \\ -2\cdot(-1)+1-2\cdot 1=1 \end{cases}$.

Ответ: вектор неизвестных равен $\boldsymbol{x}=[-1,\ 1,\ 1]^{\mathrm{T}}$.

Задача 6

Решить систему итерационным методом Гаусса-Зейделя, сделать 3 шага итерационного процесса. (При необходимости для выполнения условий сходимости сначала преобразовать систему).

$$\begin{cases} 3x_1+x_2+x_3=6 \\ 2x_1-x_2+4x_3=-1 \\ 0x_1-2x_2+x_3=-3 \end{cases}$$

Решение:

Преобразуем исходную систему уравнений к виду, удобному для итерационного процесса. Для этого выразим неизвестные x_1, x_2 и x_3 из первого, второго и третьего уравнений СЛАУ. Получим:

$$\begin{cases} x_1 = -\frac{1}{3}x_2 - \frac{1}{3}x_3 + 2 \\ x_2 = 2x_1 + 4x_3 + 1 \\ x_3 = 0x_1 + 2\,x_2 - 3 \end{cases}$$

В матричном виде приведенная система имеет вид $\boldsymbol{x} = \alpha\boldsymbol{x} + \beta$,

где $\alpha = \begin{bmatrix} 0 & -\frac{1}{3} & -\frac{1}{3} \\ 2 & 0 & 4 \\ 0 & 2 & 0 \end{bmatrix}$; $\beta = \begin{bmatrix} 2 \\ 1 \\ -3 \end{bmatrix}$.

Проверим условие сходимости итераций. Исходная система не отвечает условиям диагонального преобладания

$$|a_{ii}| > \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}|,\ i = \overline{1,n};\ \ |a_{jj}| > \sum_{\substack{i=1 \\ i\neq j}}^{n} |a_{ij}|,\ j = \overline{1,n}. \qquad (*)$$

Следовательно, достаточное условие сходимости не выполняется.

Исследуем также сходимость метода, используя нормы матрицы α для приведенной системы:

а) ∞-норма: $\|\alpha\|_\infty = \max\limits_{i=\overline{1,3}} \{\sum\limits_{j=1}^{3} |\alpha_{ij}|\}$=max{(0+$\frac{1}{3}$+$\frac{1}{3}$); (2+0+4); (0+2+0)}=

$$= \max\{\frac{2}{3};\ 6;\ 2\} = 6.$$

б) 1-норма: $\|\alpha\|_1 = \max\limits_{j=\overline{1,3}} \{\sum\limits_{i=1}^{3} |\alpha_{ij}|\}$=max{(0+2+0); ($\frac{1}{3}$+0+3); ($\frac{1}{3}$+4+0)}=

$$= \max\{2;\ 3\tfrac{1}{3};\ 4\tfrac{1}{3}\} = 4{,}33.$$

Так как обе нормы больше 1, то достаточное условие сходимости итерационного процесса $\|\alpha\| < 1$ не выполняется.

Преобразуем исходную систему таким образом, чтобы получить матрицу **A** с преобладающими диагональными элементами. Для этого переставим местами 2 и 3 строки:

$$\begin{cases} 3x_1 + x_2 + x_3 = 6 \\ 0x_1 - 2x_2 + x_3 = -3 \\ 2x_1 - x_2 + 4x_3 = -1 \end{cases}$$

Проверим условия сходимости (*):

для строк матрицы **A**: | 3 | > | 1 | + | 1 |; | –2 | > | 0 | + | 1 |; | 4 | > | 2 | + | –1 |;

для столбцов матрицы **A**: | 3 | > | 0 | + | 2 |; | –2 | = | 1 | + | –1 |; | 4 | > | 1 | + | 1 |.

Как видно, теперь условия диагонального преобладания выполняются.

Приведем полученную СЛАУ к итерационному виду. Получим следующую систему

$$\begin{cases} x_1 = -\frac{1}{3}x_2 - \frac{1}{3}x_3 + 2 \\ x_2 = 0x_1 + \frac{1}{2}x_3 + \frac{3}{2} \\ x_3 = -\frac{2}{4}x_1 + \frac{1}{4}x_2 - \frac{1}{4} \end{cases}$$

или в матричном виде приведенная система имеет вид $\boldsymbol{x} = \alpha\boldsymbol{x} + \beta$,

где $\alpha = \begin{bmatrix} 0 & -\frac{1}{3} & -\frac{1}{3} \\ 0 & 0 & \frac{1}{2} \\ -\frac{1}{2} & \frac{1}{4} & 0 \end{bmatrix}$; $\beta = \begin{bmatrix} 2 \\ \frac{3}{2} \\ -\frac{1}{4} \end{bmatrix}$.

Покажем, что и для приведенной системы условия сходимости выполняются. Вычислим нормы матрицы α:

а) ∞-норма: $\|\alpha\|_\infty = \max\limits_{i=1,3}\{\sum\limits_{j=1}^{3}|\alpha_{ij}|\} = \max\{(0+\frac{1}{3}+\frac{1}{3});\ (0+0+\frac{1}{2});\ (\frac{1}{2}+\frac{1}{4}+0)\} =$

$$= \max\{\frac{2}{3};\ \frac{1}{2};\ \frac{3}{4}\} = 0{,}75.$$

б) 1-норма: $\|\alpha\|_1 = \max\limits_{j=1,3}\{\sum\limits_{i=1}^{3}|\alpha_{ij}|\} = \max\{(0+0+\frac{1}{2});\ (\frac{1}{3}+0+\frac{1}{4});\ (\frac{1}{3}+\frac{1}{2}+0)\} =$

$$= \max\{\frac{1}{2};\ \frac{7}{12};\ \frac{10}{12}\} = 0{,}83.$$

Как и следовало ожидать, в данном случае условие сходимости выполняется и по ∞-норме и по 1-норме.

Выполним 3 шага итерационного процесса. В качестве начального приближения возьмем вектор свободных членов преобразованной системы: $\beta = \begin{bmatrix} 2 \\ 1,5 \\ -0,25 \end{bmatrix}$. На каждом шаге оценим ошибку решения по формуле

$$\varepsilon_k \le \frac{\| \beta \|}{1 - \| \beta \|} \cdot \| x^{(k)} - x^{(k-1)} \|.$$

1-е приближение:

$$\begin{cases} x_1^{(1)} = 0 \cdot x_1^{(0)} - 0,33 \cdot x_2^{(0)} - 0,33 \cdot x_3^{(0)} + 2 = 0 \cdot 2 - 0,33 \cdot 1,5 - 0,33 \cdot (-0,25) + 2 = 1,5875 \\ x_2^{(1)} = 0 \cdot x_1^{(1)} + 0 \cdot x_2^{(0)} + 0,5 \cdot x_3^{(0)} + 1,5 = 0 \cdot 1,5875 + 0 \cdot 1,5 + 0,5 \cdot (-0,25) + 1,5 = 1,375 \\ x_3^{(1)} = -0,5 \cdot x_1^{(1)} + 0,25 \cdot x_2^{(1)} + 0 \cdot x_3^{(0)} - 0,25 = -0,5 \cdot 1,5875 + 0,25 \cdot 1,375 + 0 \cdot (-0,25) - 0,25 = -0,7 \end{cases}$$

$$\varepsilon_1 \le \frac{\| \beta \|_\infty}{1 - \| \beta \|_\infty} \cdot \| x^{(1)} - x^{(0)} \| =$$

$$= \frac{0,75}{1 - 0,75} \cdot \max \{ |2 - 1,5875| ; |1,5 - 1,375| ; |-0,25 + 0,7| \} =$$

$$= 3 \cdot \max \{0,4125; 0,125; 0,45\} = 1,35.$$

2-е приближение:

$$\begin{cases} x_1^{(2)} = 0 \cdot x_1^{(1)} - 0,33 \cdot x_2^{(1)} - 0,33 \cdot x_3^{(1)} + 2 = 0 \cdot 1,5875 - 0,33 \cdot 1,375 - 0,33 \cdot (-0,7) + 2 = 1,773 \\ x_2^{(2)} = 0 \cdot x_1^{(2)} + 0 \cdot x_2^{(1)} + 0,5 \cdot x_3^{(1)} + 1,5 = 0 \cdot 1,773 + 0 \cdot 1,375 + 0,5 \cdot (-0,7) + 1,5 = 1,15 \\ x_3^{(2)} = -0,5 \cdot x_1^{(2)} + 0,25 \cdot x_2^{(2)} + 0 \cdot x_3^{(1)} - 0,25 = -0,5 \cdot 1,773 + 0,25 \cdot 1,15 + 0 \cdot (-0,7) - 0,25 = -0,8511 \end{cases}$$

$$\varepsilon_2 \le \frac{\| \beta \|_\infty}{1 - \| \beta \|_\infty} \cdot \| x^{(2)} - x^{(1)} \| =$$

$$= 3 \cdot \max \{ |1,773 - 1,5875| ; |1,15 - 1,375| ; |0,8511 - 0,7| \} =$$

$$= 3 \cdot \max \{0,1898; 0,2250; 0,1511\} = 0,675.$$

3-е приближение:

$$\begin{cases} x_1^{(3)} = 0 \cdot x_1^{(2)} - 0,33 \cdot x_2^{(2)} - 0,33 \cdot x_3^{(2)} + 2 = 0 \cdot 1,773 - 0,33 \cdot 1,15 - 0,33 \cdot (-0,8511) + 2 = 1,9014 \\ x_2^{(3)} = 0 \cdot x_1^{(3)} + 0 \cdot x_2^{(2)} + 0,5 \cdot x_3^{(2)} + 1,5 = 0 \cdot 1,9014 + 0 \cdot 1,15 + 0,5 \cdot (-0,8511) + 1,5 = 1,0744 \\ x_3^{(3)} = -0,5 \cdot x_1^{(3)} + 0,25 \cdot x_2^{(3)} + 0 \cdot x_3^{(2)} - 0,25 = -0,5 \cdot 1,9014 + 0,25 \cdot 1,0744 + 0 \cdot (-0,8511) - 0,25 = -0,9321 \end{cases}$$

$$\varepsilon_3 \le \frac{\| \beta \|_\infty}{1 - \| \beta \|_\infty} \cdot \| x^{(3)} - x^{(2)} \| =$$

$$= 3 \cdot \max\{ |1{,}9014 - 1{,}773| ; |1{,}0744 - 1{,}15| ; |-0{,}9321 + 0{,}8511| \} =$$

$$= 3 \cdot \max\{0{,}1241;\ 0{,}0756;\ 0{,}081\} = 0{,}3723.$$

Итак, полученное решение $\boldsymbol{x} = [1{,}9014,\ 1{,}0744,\ -0{,}9321]^T$. Ошибка решения $\varepsilon_3 \le 0{,}3723$.

Проверка: подставим найденный вектор $\boldsymbol{x}$ в исходную СЛАУ:

$$\begin{cases} 3 \cdot 1{,}9014 + 1 \cdot 1{,}0744 + 1 \cdot (-0{,}9321) = 5{,}8465 \\ 2 \cdot 1{,}9014 - 1 \cdot 1{,}0744 + 4 \cdot (-0{,}9321) = -1{,}0000 \\ 0 \cdot 1{,}9014 - 2 \cdot 1{,}0744 + 1 \cdot (-0{,}9321) = -3{,}0809 \end{cases}$$

Полученный вектор свободных членов близок к заданному вектору $\boldsymbol{b} = \begin{bmatrix} 6 \\ -1 \\ -3 \end{bmatrix}$.

Замечание: Точное решение данной системы уравнений равно $\boldsymbol{x}^* = [2,\ 1,\ -1]^T$.

Ответ: решение исходной системы уравнений равно

$$\boldsymbol{x} = [1{,}9014,\ 1{,}0744,\ -0{,}9321]^T.$$

Задача 7

Вычислить методом секущих корень уравнения в интервале [–3, 0]. Сделать три итерации.

$$f(x) = x^2 + 2x - 1 = 0$$

Решение:

Выполним отделение корней на заданном интервале. Для этого рассчитаем значение заданной функции в интервале от –3 до 0 с шагом 0,5. Результат показан в таблице ниже.

x	–3,0	–2,5	–2,0	–1,5	–1,0	–0,5	0
$f(x) = x^2 + 2x - 1$	2,0	0,25	–1,0	–1,75	–2,0	–1,75	–1,0

Из таблицы видно, что искомый корень лежит в интервале [–2,5, –2,0]. Дальнейшее уточнение корня выполним методом секущих. Итерационная формула метода секущих имеет следующий вид

$$x^{(k+1)} = x^{(k)} - f(x^{(k)}) \cdot \frac{x^{(k)} - x^{(k-1)}}{f(x^{(k)}) - f(x^{(k-1)})}.$$

В качестве начального приближения возьмем левую границу и середину найденного интервала, т.е. $x^{(0)}$= –2,5, $x^{(1)}$= (–2,5+(–2,0))/2 = –2,25.

1-я итерация:

$$x^{(2)} = x^{(1)} - f(x^{(1)}) \cdot \frac{x^{(1)} - x^{(0)}}{f(x^{(1)}) - f(x^{(0)})} = -2{,}25 - 0{,}4375 \frac{-2{,}25 - 2{,}5}{-0{,}4375 - 0{,}25} = -2{,}4091.$$

2-я итерация:

$$x^{(3)} = x^{(2)} - f(x^{(2)}) \cdot \frac{x^{(2)} - x^{(1)}}{f(x^{(2)}) - f(x^{(1)})} = -2{,}4091 - 0{,}0145 \frac{-2{,}4091 - 2{,}25}{-0{,}0145 - 0{,}4375} = -2{,}4145.$$

3-я итерация:

$$x^{(4)} = x^{(3)} - f(x^{(3)}) \cdot \frac{x^{(3)} - x^{(2)}}{f(x^{(3)}) - f(x^{(2)})} = -2{,}4145 - 8{,}1025 \cdot 10^{-4} \frac{2{,}4145 - 2{,}4091}{8{,}1025 \cdot 10^{-4} - 0{,}0145} = -2{,}4142.$$

Проверка: $f(x^{(4)}) = (-2{,}4142)^2 + 2 \cdot (-2{,}4142) - 1 \approx -3{,}84 \cdot 10^{-5}$.

Ответ: найденный корень равен $x \approx -2{,}4142$.

Задача 8

Функция $y = f(x)$ задана таблично. Найти коэффициенты интерполирующего полинома $P(x)$ непосредственно из условий интерполяции. Рассчитать значение первой производной в точке 2,5.

i	0	1	2
x	1	2	3
y	–5	–8	–12

Решение:

Интерполирующий полином имеет следующий вид:

$P_n(x) = C_0 + C_1 x + C_2 x^2$. Для нахождения коэффициентов C_0, C_1, C_2 составим следующую систему уравнений, используя условие интерполяции $y_i = P_n(x_i)$, $i = \overline{0,2}$:

$$C_0 + C_1 x_0 + C_2 x_0^2 = y_0$$
$$C_0 + C_1 x_1 + C_2 x_1^2 = y_1$$
$$C_0 + C_1 x_2 + C_2 x_2^2 = y_2$$

или в матричном виде

$$\begin{bmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \end{bmatrix} \cdot \begin{bmatrix} C_0 \\ C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} \Rightarrow \mathbf{AC}=\mathbf{B},$$

где $\mathbf{A} = \begin{bmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \end{bmatrix}$; $\mathbf{B} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix}$; $\mathbf{C} = \begin{bmatrix} C_0 \\ C_1 \\ C_2 \end{bmatrix}$ – вектор искомых коэффициентов полинома.

Подставляем значения x_i и y_i из таблицы, получим

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix} \cdot \begin{bmatrix} C_0 \\ C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} -5 \\ -8 \\ -12 \end{bmatrix}, \text{ т.е. } \mathbf{A} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix}; \ \mathbf{B} = \begin{bmatrix} -5 \\ -8 \\ -12 \end{bmatrix}.$$

Решение данной системы можно выполнить методом обратной матрицы. Вектор искомых коэффициентов найдем по формуле $\mathbf{C}=\mathbf{A}^{-1}\mathbf{B}$:

$$\begin{bmatrix} C_0 \\ C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix}^{-1} \cdot \begin{bmatrix} -5 \\ -8 \\ -12 \end{bmatrix} = \begin{bmatrix} 3 & -3 & 1 \\ -2{,}5 & 4 & -1{,}5 \\ 0{,}5 & -1 & 0{,}5 \end{bmatrix} \cdot \begin{bmatrix} -5 \\ -8 \\ -12 \end{bmatrix} = \begin{bmatrix} -3 \\ -1{,}5 \\ -0{,}5 \end{bmatrix}.$$

Таким образом, интерполирующий полином будет иметь следующий вид

$$P_n(x) = -3 - 1{,}5\,x - 0{,}5\,x^2.$$

Проверка: $P_n(x_0) = P_n(1) = -3 - 1{,}5 \cdot 1 - 0{,}5 \cdot 1^2 = -5$;

$$P_n(x_1) = P_n(2) = -3 - 1{,}5 \cdot 2 - 0{,}5 \cdot 2^2 = -8;$$

$$P_n(x_2) = P_n(3) = -3 - 1{,}5\cdot 3 - 0{,}5\cdot 3^2 = -12.$$

Видно, что полученный полином проходит через заданные точки (x_i, y_i).

Получим значение первой производной $P'_n(x) = -1{,}5 - x$.

Значение производной в точке x=2,5 равно $P'_n(2{,}5) = -1{,}5 - 2{,}5 = -4$.

Ответ: значение производной равно $f'(2{,}5) \approx P'_n(2{,}5) = -4$.

Задача 9

Для функции $y = f(x)$, заданной таблицей, с помощью метода наименьших квадратов найти коэффициенты аппроксимирующей функции в виде $P_n(x) = C_0 + C_1 x$

i	0	1	2
x	1	2	3
y	2	3	6

Решение:

В методе наименьших квадратов минимизируется сумма квадратов разностей между значениями функции $y = \varphi(\vec{C}, x) = C_0 + C_1 x$ и измеренными значениями y_i в точках x_i, $i = \overline{0,n}$:

$$\varepsilon = \sum_{i=0}^{n} \varepsilon_i = \sum_{i=0}^{n} [\varphi(\vec{C}, x_i) - y_i]^2 = \min.$$

В нашем случае число точек равно 3, т.е. n=2. Тогда $\varepsilon = \sum_{i=0}^{2} (C_0 + C_1 x_i - y_i)^2 = \min.$

Дифференцируя функцию ошибки по переменным C_0, C_1 получим систему уравнений следующего вида

$$\left.\begin{aligned} \frac{\partial \varepsilon}{\partial C_0} &= \sum_{i=0}^{2} [C_0 + C_1 x_i - y_i] = 0 \\ \frac{\partial \varepsilon}{\partial C_1} &= \sum_{i=0}^{2} ([C_0 + C_1 x_i - y_i] x_i) = 0 \end{aligned}\right\} \rightarrow \left.\begin{aligned} 3C_0 + C_1 \sum_{i=0}^{2} x_i &= \sum_{i=0}^{2} y_i \\ C_0 \sum_{i=0}^{2} x_i + C_1 \sum_{i=0}^{2} x_i^2 &= \sum_{i=0}^{2} x_i y_i \end{aligned}\right\}$$

В матричном виде эту систему можно записать как

$$\begin{bmatrix} 3 & \sum_{i=0}^{2} x_i \\ \sum_{i=0}^{2} x_i & \sum_{i=0}^{2} x_i^2 \end{bmatrix} \cdot \begin{bmatrix} C_0 \\ C_1 \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{2} y_i \\ \sum_{i=0}^{2} x_i y_i \end{bmatrix} \text{ или } \mathbf{AC}=\mathbf{B}.$$

Для решения данной системы используем правило Крамера:

$$C_0 = \frac{\Delta_{11}}{\Delta},\ C_1 = \frac{\Delta_{22}}{\Delta},$$

где Δ – определитель матрицы коэффициентов $\mathbf{A}=\begin{bmatrix} 3 & \sum_{i=0}^{2} x_i \\ \sum_{i=0}^{2} x_i & \sum_{i=0}^{2} x_i^2 \end{bmatrix}$;

Δ_{11} – определитель матрицы $\begin{bmatrix} \sum_{i=0}^{2} y_i & \sum_{i=0}^{2} x_i \\ \sum_{i=0}^{2} x_i y_i & \sum_{i=0}^{2} x_i^2 \end{bmatrix}$, полученной из исходной матрицы коэффициентов **A** заменой первого столбца вектором свободных членов **B**.

Δ_{22} – определитель матрицы $\begin{bmatrix} 3 & \sum_{i=0}^{2} y_i \\ \sum_{i=0}^{2} x_i & \sum_{i=0}^{2} x_i y_i \end{bmatrix}$, полученной из исходной матрицы коэффициентов **A** заменой второго столбца вектором свободных членов **B**.

Следовательно, решением данной системы уравнений будет

$$C_0 = \frac{\sum_{i=0}^{2} x_i^2 \sum_{i=0}^{2} y_i - \sum_{i=0}^{2} x_i y_i \sum_{i=0}^{2} x_i}{3\sum_{i=0}^{2} x_i^2 - \sum_{i=0}^{2} x_i \sum_{i=0}^{2} x_i};\ C_1 = \frac{3\sum_{i=0}^{2} x_i y_i - \sum_{i=0}^{2} x_i \sum_{i=0}^{2} y_i}{3\sum_{i=0}^{2} x_i^2 - \sum_{i=0}^{2} x_i \sum_{i=0}^{2} x_i}$$

Замечание: При преобразовании формул следует правильно выполнять действия с величинами, входящими под знак суммы. В частности, необходимо иметь в виду, что $\sum_{i=0}^{n} x_i^2 \cdot \sum_{i=0}^{n} \frac{1}{x_i} \neq \sum_{i=0}^{n} x_i$, $\sum_{i=0}^{n} x_i y_i / \sum_{i=0}^{n} y_i \neq \sum_{i=0}^{n} x_i$ и т.д. В случае

сомнений рекомендуется проверить преобразования, раскрывая знаки сумм при небольшом числе слагаемых. Например, раскрывая первую из приведенных формул при числе слагаемых n=2, легко убедиться, что $(x_1^2 + x_2^2)(\frac{1}{x_1^2} + \frac{1}{x_2^2}) \neq x_1 + x_2$.

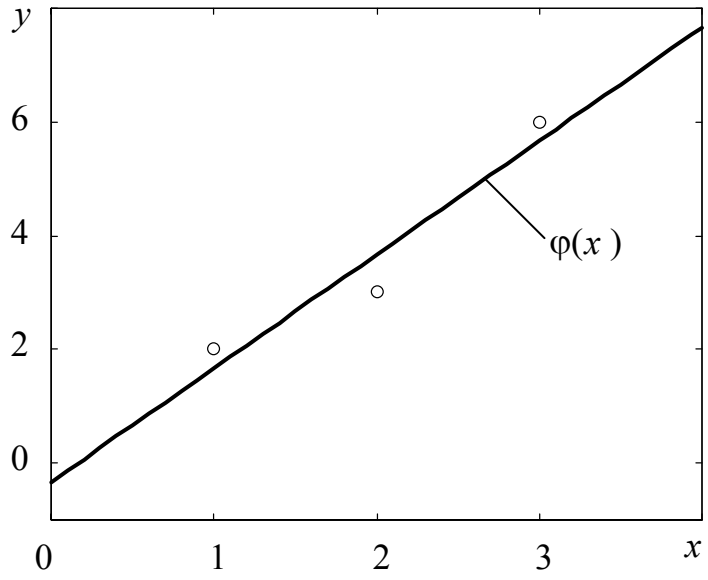
Подставляя в полученные выражения значения x_i и y_i из таблицы, получим:

$$C_0 = \frac{(1+4+9)(2+3+6) - (1\cdot 2 + 2\cdot 3 + 3\cdot 6)(1+2+3)}{3(1+4+9) - (1+2+3)(1+2+3)} = -0{,}3333;$$

$$C_1 = \frac{3(1\cdot 2 + 2\cdot 3 + 3\cdot 6) - (1+2+3)(2+3+6)}{3(1+4+9) - (1+2+3)(1+2+3)} = 2.$$

Таким образом, аппроксимирующая функция будет иметь следующий вид $\varphi(x) = -0{,}3333 + 2x$.

Проверка: для проверки построим график функции $\varphi(x) = -0{,}3333 + 2x$ и нанесем на него точки (x_i, y_i).



Ответ: аппроксимирующая функция имеет вид $\varphi(x) = -0{,}3333 + 2x$.

Задача 10

Функция $y = f(x)$ задана таблицей. В точке x=0,2 найти значения $f'(x)$ по формулам левой, правой и центральной разностных производных, а также $f''(x)$.

x	0,15	0,20	0,25
y	0,0034	0,0080	0,0156

Решение:

Аппроксимация первой производной по формуле левых разностных производных (ЛРП) имеет следующий вид:

$$y'_{\text{ЛРП}}(x_i) = \frac{\Delta y_i}{\Delta x_i},\ \Delta x_i = x_{i-1} - x_i = -h,\ \Delta y_i = y_{i-1} - y_i,$$

$$y'_{\text{ЛРП}}(x_i) = \frac{y_i - y_{i-1}}{h}.$$

Отсюда получим

$$y'_{\text{ЛРП}}(0{,}2) = \frac{0{,}008 - 0{,}0034}{0{,}20 - 0{,}15} = \frac{0{,}0046}{0{,}05} = 0{,}092.$$

Аппроксимация первой производной по формуле правых разностных производных (ПРП) имеет следующий вид:

$$y'_{\text{ПРП}}(x_i) = \frac{\Delta y_i}{\Delta x_i},\ \Delta x_i = x_{i+1} - x_i = h,\ \Delta y_i = y_{i+1} - y_i,$$

$$y'_{\text{ПРП}}(x_i) = \frac{y_{i+1} - y_i}{h}.$$

Отсюда получим

$$y'_{\text{ПРП}}(0{,}2) \approx \frac{0{,}0156 - 0{,}008}{0{,}25 - 0{,}20} = \frac{0{,}0076}{0{,}05} = 0{,}152.$$

Аппроксимация первой производной по формуле центральных разностных производных (ЦРП) имеет следующий вид:

$$y'_{\text{ЦРП}}(x_i) = \frac{\Delta y_i}{\Delta x_i},\ \Delta x_i = x_{i+1} - x_{i-1} = 2h,\ \Delta y_i = y_{i+1} - y_{i-1},$$

$$y'_{\text{ЦРП}}(x_i) = \frac{y_{i+1} - y_{i-1}}{2h}.$$

Отсюда получим

$$y'_{\text{ЦРП}}(0{,}2) \approx \frac{0{,}0156 - 0{,}0034}{0{,}25 - 0{,}15} = \frac{0{,}0122}{0{,}10} = 0{,}122.$$

Приближенное значение производной второго порядка получим следующим образом. Представим вторую производную с помощью правой разности:

$$y''(x_i) \approx \frac{\Delta y'_i}{\Delta x_i}, \Delta x_i = x_{i+1} - x_i = h, \Delta y'_i = y'_{i+1} - y'_i,$$

а производные первого порядка y'_{i+1} и y'_i – с помощью левых разностей:

$$y'_{i+1} = y'(x_{i+1}) \approx \frac{y_{i+1} - y_i}{h}, y'_i = y'(x_i) \approx \frac{y_i - y_{i-1}}{h}$$

и окончательно получим

$$y''(x_i) \approx \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2}.$$

Для нашей задачи вторая производная равна

$$y''(0{,}2) \approx \frac{0{,}0156 - 2 \cdot 0{,}008 + 0{,}0034}{0{,}05^2} = 1{,}2.$$

Замечание: Существенные отличия левой, правой и центральной разностных производных связаны с большой величиной шага *h*.

Ответ: левая разностная производная: $y'(0{,}2) = 0{,}092$;
правая разностная производная: $y'(0{,}2) = 0{,}152$;
центральная разностная производная: $y'(0{,}2) = 0{,}122$;
вторая производная: $y''(0{,}2) = 1{,}2$.

Задача 11

Вычислить интеграл $\int_0^1 (2x^3 + 3x^2 + 2)\,dx$ по формулам прямоугольников, трапеций, Симпсона (взять 5 значений подынтегральной функции, т.е. число подинтервалов n=4). Рассчитать оценки погрешности вычислений интеграла, сравнить с действительными погрешностями.

Решение:

Выполним разбиение интервала интегрирования на 4 части и рассчитаем значение функции в узлах. Результаты представим в виде таблицы.

i	0	1	2	3	4
x_i	0	0,25	0,5	0,75	1,0
$y_i = f(x_i)$	2,0	2,2188	3,0	4,5312	7,0

а) Метод средних прямоугольников. Формула вычисления интеграла функции $f(x)$ по методу средних прямоугольников имеет следующий вид

$$I_{\text{пр}} = h\sum_{i=0}^{n-1} f(x_{\text{ср}i}),$$

где $x_{\text{ср}i} = \dfrac{x_i + x_{i+1}}{2}$; $y_{\text{ср}i} = f(x_{\text{ср}i})$; $i = \overline{0, n-1}$.

Вычислим значения функции в середине выбранных интервалов и результаты сведем в таблицу

$x_{\text{ср}i}$	0,125	0,375	0,625	0,875
$y_{\text{ср}i} = f(x_{\text{ср}i})$	2,0508	2,5273	3,6602	5,6367

Значение искомого интеграла по формуле прямоугольников будет равно:

$$I_{\text{пр}}=0,25\cdot(2,0508+2,5273+3,6602+5,6367)=3,4688.$$

Погрешность метода прямоугольников можно оценить по формуле

$$\varepsilon_{\text{пр}} \le \frac{b-a}{24}\cdot M_2 \cdot h^2,$$

где $M_2 = \max\limits_{a\le x\le b} |f''(x)|$.

Для нашего случая имеем: h=0,25; $f'(x)=6x^2+6x$; $f''(x)=12x+6$; $M_2 = \max\limits_{0\le x\le 1} |f''(x)| = \max\limits_{0\le x\le 1} (12x+6) = 18$. Следовательно

$$\varepsilon_{\text{пр}} \le \frac{1-0}{24}\cdot 18\cdot 0,25^2 = 0,0469.$$

б) Метод трапеций. Формула вычисления интеграла функции $f(x)$ по методу трапеций имеет следующий вид

$$I_{\text{тр}} = h\left(\frac{f(x_0) + f(x_n)}{2} + \sum_{i=1}^{n-1} f(x_i) \right) =$$

$$= 0{,}25\cdot((2{,}0+7{,}0)/2+2{,}2188+3{,}0+4{,}5312)=3{,}5625.$$

Погрешность метода трапеций можно оценить по формуле

$$\varepsilon_{\text{тр}} \le \frac{b-a}{12} \cdot M_2 \cdot h^2 = \frac{1-0}{12} \cdot 18 \cdot 0{,}25^2 = 0{,}0938.$$

в) Метод Симпсона. Формула вычисления интеграла функции $f(x)$ по методу Симпсона имеет следующий вид (n – должно быть четным числом)

$$I \approx \frac{h}{3}\left(f(x_0) + f(x_n) + 4\sigma_1 + 2\sigma_2 \right),$$

где $\sigma_1 = f(x_1) + f(x_3) + \ldots + f(x_{n-1})$; $\sigma_2 = f(x_2) + f(x_4) + \ldots + f(x_{n-2})$

Для нашей задачи формула Симпсона примет следующий вид

$$I_c \approx \frac{h}{3}(f(x_0) + f(x_4) + 4[f(x_1) + f(x_3)] + 2f(x_2)) =$$

$$= 0{,}25/3\cdot(2{,}0+7{,}0+4\cdot(2{,}2188+4{,}5312)+2\cdot 3{,}0)=3{,}5.$$

Погрешность метода Симпсона можно оценить по формуле

$$\varepsilon_{\text{с}} \le \frac{b-a}{180} \cdot M_4 \cdot h^4 = \frac{1-0}{180} \cdot 0 \cdot 0{,}25^4 = 0,$$

где $M_4 = \max\limits_{a \le x \le b} |f^{\text{IV}}(x)|$. Четвертая производная заданной функции $f(x)=2x^3 + 3x^2 + 2$ равна нулю, т.е. $M_4=0$.

Проверка: Точное значение интеграла равно

$$\int_0^1 (2 \cdot x^3 + 3x^2 + 2)dx = \left(\frac{1}{2}x^4 + x^3 + 2x \right)\Big|_0^1 = \frac{1}{2} + 1 + 2 - 0 = 3{,}5\,.$$

Следовательно, истинные значения ошибок будут равны:

а) метод прямоугольников $\varepsilon = | I - I_{\text{пр}} | = | 3{,}5 - 3{,}4688 | = 0{,}0312$;

б) метод трапеций $\varepsilon = | I - I_{\text{тр}} | = | 3{,}5 - 3{,}5625 | = 0{,}0625$;

в) метод Симпсона $\varepsilon = | I - I_{\text{с}} | = | 3{,}5 - 3{,}5 | = 0$.

Таким образом, оценки ошибок близки к действительным ошибкам вычисления интеграла.

Задача 12

Решить систему уравнений $\begin{cases} f_1(x_1, x_2) = x_1 - x_2^2 = 0 \\ f_2(x_1, x_2) = x_1^2 + x_2^2 - 3 = 0 \end{cases}$ методом Ньютона-Рафсона. Начальная точка $x^{(0)} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$. Сделать две итерации указанным методом. Оценить ошибку решения.

Решение:

Рассчитаем матрицу Якоби и якобиан системы уравнений

$$J = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \dfrac{\partial f_1}{\partial x_2} \\ \dfrac{\partial f_2}{\partial x_1} & \dfrac{\partial f_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 1 & -2x_2 \\ 2x_1 & 2x_2 \end{bmatrix};$$

Для начальной точки $x^{(0)} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ $\det J = 2\,x_2 - 4\,x_1 x_2 = 2\cdot(-1) - 4\cdot 1\cdot(-1) = 2,$

т.е. $\det J \neq 0$,

Итерационная формула Ньютона-Рафсона для нашего примера имеет следующий вид:

$$\boldsymbol{x}^{(k)} = \boldsymbol{x}^{(k-1)} - [\boldsymbol{J}(\boldsymbol{x}^{(k)})]^{-1}\, \mathbf{F}(\boldsymbol{x}^{(k)}).$$

1-й шаг:

$$x^{(0)} = \begin{bmatrix} 1 \\ -1 \end{bmatrix};\ J(x^{(0)}) = \begin{bmatrix} 1 & 2 \\ 2 & -2 \end{bmatrix};\ F(x^{(0)}) = \begin{bmatrix} 1-(-1)^2 \\ 1^2 + (-1)^2 - 3 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix};$$

$$[J(x^{(0)})]^{-1} = \frac{1}{-2-4}\begin{bmatrix} -2 & -2 \\ -2 & 1 \end{bmatrix} = \frac{1}{-6}\begin{bmatrix} -2 & -2 \\ -2 & 1 \end{bmatrix} = \begin{bmatrix} \dfrac{1}{3} & \dfrac{1}{3} \\ \dfrac{1}{3} & -\dfrac{1}{6} \end{bmatrix};$$

$$\boldsymbol{x}^{(1)} = \boldsymbol{x}^{(0)} - [\boldsymbol{J}(\boldsymbol{x}^{(0)})]^{-1}\,\mathbf{F}(\boldsymbol{x}^{(0)}) = \begin{bmatrix} 1 \\ -1 \end{bmatrix} - \begin{bmatrix} \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & -\frac{1}{6} \end{bmatrix} \cdot \begin{bmatrix} 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} - \begin{bmatrix} -\frac{1}{3} \\ \frac{1}{6} \end{bmatrix} = \begin{bmatrix} 1{,}3333 \\ -1{,}1666 \end{bmatrix}.$$

Получили решение $\boldsymbol{x}^{(1)} = \begin{bmatrix} 1\frac{1}{3} \\ -1\frac{1}{6} \end{bmatrix} = \begin{bmatrix} 1{,}3333 \\ -1{,}1666 \end{bmatrix}.$

Погрешность решения:

$\| \boldsymbol{x}^{(1)} - \boldsymbol{x}^{(0)} \|_1 = \max \{ | 1{,}3333 - 1|, | -1{,}1666 + 1| \} = 0{,}3333;$

$\| \boldsymbol{x}^{(1)} - \boldsymbol{x}^{(0)} \|_2 = \sqrt{(1{,}3333-1)^2 + (-1{,}1666+1)^2} = 0{,}3727.$

2-й шаг:

$$x^{(1)} = \begin{bmatrix} 1{,}3333 \\ -1{,}1666 \end{bmatrix};\ J(x^{(1)}) = \begin{bmatrix} 1 & -2\cdot(-1{,}1666) \\ 2\cdot 1{,}3333 & 2\cdot(-1{,}1666) \end{bmatrix} = \begin{bmatrix} 1 & 2{,}3333 \\ 2{,}6666 & -2{,}3333 \end{bmatrix};$$

$$F(x^{(1)}) = \begin{bmatrix} 1-(-1{,}1666)^2 \\ 1{,}3333^2 + (-1{,}1666)^2 - 3 \end{bmatrix} = \begin{bmatrix} -0{,}0277 \\ 0{,}1388 \end{bmatrix};$$

$$[J(x^{(1)})]^{-1} = \frac{1}{-8{,}5555}\begin{bmatrix} -2{,}3333 & -2{,}3333 \\ -2{,}6666 & 1 \end{bmatrix} = \begin{bmatrix} 0.2727 & 0{,}2727 \\ 0{,}3117 & -0{,}1169 \end{bmatrix};$$

$$\boldsymbol{x}^{(2)} = \boldsymbol{x}^{(1)} - [\boldsymbol{J}(\boldsymbol{x}^{(1)})]^{-1}\,\mathbf{F}(\boldsymbol{x}^{(1)}) =$$

$$= \begin{bmatrix} 1{,}3333 \\ -1{,}1666 \end{bmatrix} - \begin{bmatrix} 0{,}2727 & 0{,}2727 \\ 0{,}3117 & -0{,}1169 \end{bmatrix} \cdot \begin{bmatrix} -0{,}0277 \\ 0{,}1388 \end{bmatrix} = \begin{bmatrix} 1{,}3030 \\ -1{,}1418 \end{bmatrix};$$

Получили решение $\boldsymbol{x}^{(2)} = \begin{bmatrix} 1{,}3030 \\ -1{,}1418 \end{bmatrix}.$

Погрешность решения:

$\| \boldsymbol{x}^{(2)} - \boldsymbol{x}^{(1)} \|_1 = \max \{ | 1{,}3030 - 1{,}3333 |, | -1{,}1418 + 1{,}1666 | \} = 0{,}303;$

$\| \boldsymbol{x}^{(2)} - \boldsymbol{x}^{(1)} \|_2 = \sqrt{(1{,}3030-1{,}3333)^2 + (-1{,}1418+1{,}1666)^2} = 0{,}0392.$

Так как ошибка уменьшается, то вычислительный процесс сходится. Значение функции в точке $\boldsymbol{x}^{(2)} = \begin{bmatrix} 1{,}3030 \\ -1{,}1418 \end{bmatrix}$ будет равно $F(x^{(2)}) = \begin{bmatrix} -0{,}000619 \\ 0{,}001538 \end{bmatrix}.$

Задача 13

Решить систему уравнений $\begin{cases} f_1(x_1, x_2) = x_1 - x_2^2 = 0 \\ f_2(x_1, x_2) = x_1^2 + x_2^2 - 3 = 0 \end{cases}$ итерационным методом. Начальная точка $x^{(0)} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$. Сделать три итерации указанным методом. Оценить ошибку решения.

Решение:

Вычислим частные производные функций

$$f_1(x_1, x_2) = x_1 - x_2^2, \qquad f_2(x_1, x_2) = x_1^2 + x_2^2 - 3$$

в точке $x^{(0)}$:

$$\left.\frac{\partial f_1}{\partial x_1}\right|_{x^{(0)}} = \left.\frac{\partial (x_1 - x_2^2)}{\partial x_1}\right|_{x^{(0)}} = 1; \qquad \left.\frac{\partial f_2}{\partial x_1}\right|_{x^{(0)}} = \left.\frac{\partial (x_1^2 + x_2^2 - 3)}{\partial x_1}\right|_{x^{(0)}} = 2x_1\Big|_{x^{(0)}} = 2;$$

$$\left.\frac{\partial f_1}{\partial x_2}\right|_{x^{(0)}} = \left.\frac{\partial (x_1 - x_2^2)}{\partial x_2}\right|_{x^{(0)}} = (-2x_2)\Big|_{x^{(0)}} = 2; \quad \left.\frac{\partial f_2}{\partial x_2}\right|_{x^{(0)}} = \left.\frac{\partial (x_1^2 + x_2^2 - 3)}{\partial x_2}\right|_{x^{(0)}} = 2x_2\Big|_{x^{(0)}} = -2.$$

Решаем систему

$$\begin{cases} 1 + \lambda_{11}\left.\dfrac{\partial f_1}{\partial x_1}\right|_{x^{(0)}} + \lambda_{12}\left.\dfrac{\partial f_2}{\partial x_1}\right|_{x^{(0)}} = 0; \\ \lambda_{11}\left.\dfrac{\partial f_1}{\partial x_2}\right|_{x^{(0)}} + \lambda_{12}\left.\dfrac{\partial f_2}{\partial x_2}\right|_{x^{(0)}} = 0; \\ \lambda_{21}\left.\dfrac{\partial f_1}{\partial x_1}\right|_{x^{(0)}} + \lambda_{22}\left.\dfrac{\partial f_2}{\partial x_1}\right|_{x^{(0)}} = 0; \\ 1 + \lambda_{21}\left.\dfrac{\partial f_1}{\partial x_2}\right|_{x^{(0)}} + \lambda_{22}\left.\dfrac{\partial f_2}{\partial x_2}\right|_{x^{(0)}} = 0. \end{cases} = \begin{cases} 1 + \lambda_{11} + 2\lambda_{12} = 0; \\ 2\lambda_{11} - 2\lambda_{12} = 0; \\ \lambda_{21} + 2\lambda_{22} = 0; \\ 1 + 2\lambda_{21} - 2\lambda_{22} = 0, \end{cases}$$

для нахождения коэффициентов λ_{ij}.

Получим 2 системы уравнений с разными правыми частями:

$$\begin{bmatrix} 0 & 2 \\ 2 & -2 \end{bmatrix} \cdot \begin{bmatrix} \lambda_{11} \\ \lambda_{12} \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \end{bmatrix} \quad \text{и} \quad \begin{bmatrix} 0 & 2 \\ 2 & -2 \end{bmatrix} \cdot \begin{bmatrix} \lambda_{21} \\ \lambda_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}.$$

Решив эти системы, найдем коэффициенты

$$\begin{bmatrix}\lambda_{11}\\ \lambda_{12}\end{bmatrix}=\frac{1}{-4}\begin{bmatrix}-2 & -2\\ -2 & 0\end{bmatrix}\cdot\begin{bmatrix}-1\\ 0\end{bmatrix}=\begin{bmatrix}0{,}5 & 0{,}5\\ 0{,}5 & 0\end{bmatrix}\cdot\begin{bmatrix}-1\\ 0\end{bmatrix}=\begin{bmatrix}-0{,}5\\ -0{,}5\end{bmatrix}$$

$$\begin{bmatrix}\lambda_{21}\\ \lambda_{22}\end{bmatrix}=\frac{1}{-4}\begin{bmatrix}-2 & -2\\ -2 & 0\end{bmatrix}\cdot\begin{bmatrix}0\\ -1\end{bmatrix}\begin{bmatrix}0{,}5 & 0{,}5\\ 0{,}5 & 0\end{bmatrix}\cdot\begin{bmatrix}0\\ -1\end{bmatrix}=\begin{bmatrix}-0{,}5\\ 0\end{bmatrix}$$

Условие $\lambda_{11}\lambda_{22}-\lambda_{12}\lambda_{21} = (-0{,}5)\cdot 0 - (-0{,}5)\cdot(-0{,}5) = -0{,}25 \neq 0$ выполнено.

Приведенная система уравнений имеет следующий вид:

$$\begin{cases} x_1 = g_1(x_1, x_2) = x_1 + \lambda_{11} f_1(x_1, x_2) + \lambda_{12} f_2(x_1, x_2) = x_1 - 0{,}5\left(x_1 - x_2^2\right) - 0{,}5\left(x_1^2 + x_2^2 - 3\right); \\ x_2 = g_2(x_1, x_2) = x_2 + \lambda_{21} f_1(x_1, x_2) + \lambda_{22} f_2(x_1, x_2) = x_2 - 0{,}5\left(x_1 - x_2^2\right) + 0\left(x_1^2 + x_2^2 - 3\right). \end{cases}$$

Выполним 3 шага метода простых итераций.

1-й шаг:

$$x_1^{(1)} = x_1^{(0)} - 0{,}5\left(x_1^{(0)} - (x_2^{(0)})^2\right) - 0{,}5\left((x_1^{(0)})^2 + (x_2^{(0)})^2 - 3\right) =$$
$$= 1 - 0{,}5\cdot(1 - (-1)^2) - 0{,}5\cdot(1^2 + (-1)^2 - 3) = 1{,}5$$

$$x_2^{(1)} = x_2^{(0)} - 0{,}5\left(x_1^{(0)} - (x_2^{(0)})^2\right) = -1 - 0{,}5\cdot(1 - (-1)^2) = 0$$

Получили решение: $x^{(1)} = \begin{bmatrix}1{,}5\\ 0\end{bmatrix}$.

Погрешность решения:

$$\| \boldsymbol{x}^{(1)} - \boldsymbol{x}^{(0)} \|_1 = \max\{\,|\,1{,}5 - 1{,}0\,|, |\,0 - 1{,}0\,|\,\} = 1;$$

$$\| \boldsymbol{x}^{(1)} - \boldsymbol{x}^{(0)} \|_2 = \sqrt{(1{,}5-1{,}0)^2 + (0-1{,}0)^2} = 1{,}118.$$

2-й шаг:

$$x_1^{(2)} = x_1^{(1)} - 0{,}5\left(x_1^{(1)} - (x_2^{(1)})^2\right) - 0{,}5\left((x_1^{(1)})^2 + (x_2^{(1)})^2 - 3\right) =$$
$$= 1{,}5 - 0{,}5\cdot(1{,}5 - 0^2) - 0{,}5\cdot(1{,}5^2 + 0^2 - 3) = 1{,}125$$

$$x_2^{(2)} = x_2^{(1)} - 0{,}5\left(x_1^{(1)} - (x_2^{(1)})^2\right) = 0 - 0{,}5\cdot(1{,}5 - 0^2) = -0{,}75$$

Получили решение: $x^{(1)} = \begin{bmatrix}1{,}125\\ -0{,}75\end{bmatrix}$.

Погрешность решения:

$$\| \boldsymbol{x}^{(2)} - \boldsymbol{x}^{(1)} \|_1 = \max\{\,|\,1{,}125 - 1{,}5\,|, |\,-0{,}75 - 0\,|\,\} = 0{,}75;$$

$$\| \boldsymbol{x}^{(2)} - \boldsymbol{x}^{(1)} \|_2 = \sqrt{(1{,}125-1{,}5)^2 + (-0{,}75-0)^2} = 0{,}8385.$$

3-й шаг:

$$x_1^{(3)} = x_1^{(2)} - 0{,}5\left(x_1^{(2)} - (x_2^{(2)})^2\right) - 0{,}5\left((x_1^{(2)})^2 + (x_2^{(2)})^2 - 3\right) =$$
$$= 1{,}125 - 0{,}5 \cdot (1{,}125 - (-0{,}75)^2) - 0{,}5 \cdot (1{,}125^2 + (-0{,}75)^2 - 3) = 1{,}4296$$

$$x_2^{(3)} = x_2^{(2)} - 0{,}5\left(x_1^{(2)} - (x_2^{(2)})^2\right) = -0{,}75 - 0{,}5 \cdot (1{,}125 - (-0{,}75)^2) = -1{,}03125$$

Получили решение: $x^{(1)} = \begin{bmatrix} 1{,}4296 \\ -1{,}03125 \end{bmatrix}$.

Погрешность решения:

$$\| \boldsymbol{x}^{(3)} - \boldsymbol{x}^{(2)} \|_1 = \max \{ | 1{,}4296 - 1{,}125 |, | -1б03125 + 0{,}75 | \} = 0{,}3046;$$

$$\| \boldsymbol{x}^{(2)} - \boldsymbol{x}^{(1)} \|_2 = \sqrt{(1{,}4296 - 1{,}125)^2 + (-1{,}03125 + 0{,}75)^2} = 0{,}4146.$$

Так как ошибка уменьшается, то вычислительный процесс сходится. Значение функции в точке $\boldsymbol{x}^{(3)} = \begin{bmatrix} 1{,}4296 \\ -1{,}03125 \end{bmatrix}$ будет равно $F(x^{(3)}) = \begin{bmatrix} 0{,}3661 \\ 0{,}1072 \end{bmatrix}$.

Заметим, что преобразование исходной системы уравнений (вычисление коэффициентов λ_{ij}) выполняется только один раз.

ПРИЛОЖЕНИЕ А

Исходный текст программы-макета Polinom3.m

```
% ---------------------------------------------------------------
%   Программа линейной полиномиальной интерполяции функций
%   Кафедра КСУП ФВС ТУСУР
% ---------------------------------------------------------------
clc;  % --- очистка главного окна
home; % --- перемещение курсора в начало
disp('  Лабораторная работа No 2 ');
disp('------------------------------------------------------------------ ');
disp(' ');
disp('    Линейная полиномиальная интерполяция функций  ');
ddisp('------------------------------------------------------------------ ');
disp(' ');
disp('  Жми любую клавишу ...')
pause

%-------------------------------------------------------------------------
key=1;
while key~=0, % Общий цикл программы
clc;          %  Очистить экран
close all;    %  Закрыть все графические окна
clear all;    %  Удаление переменных из памяти

% --- входные данные
NN=100; % -- количество точек для построения графиков

% --- запрос имени исходной фукнции
clc;
disp(' Введите имя интерполируемой функции')
disp('  Замечание: Функция должна быть оформлена в виде function    ');
disp('             т.е. должен существовать М-файл с Вашей функцией ');
disp('             подробности см. в MATLAB HELP                     ');
% --- цикл запроса имени функции
k=1;
while k~=0, txtFUN=input('  Введите имя функции ->','s');
 if length(txtFUN)==0,
  k=1;
 else
  if (exist(txtFUN)==2 | exist(txtFUN)==5),
    k=0;
   else
    disp([' Ошибка:  ',txtFUN,' - такой функции нет... ' ]);
    k=1;
  end
 end
end

% --- задание интервала интерполяции
disp(' Задайте интервал  интерполяции [a b] ');
predel=input(' Введите [a b] ->');

key1=1;
while key1~=0, %  цикл по узлам N
% -- очистка переменных для нового цикла расчетов
i=[];j=[];ii=[];x=[];x1=[];x2=[];A1=[];A2=[];n=[];k=[];
C1=[];C2=[];P1=[];P2=[];Error1=[];Error2=[];
f=[];f0=[];f1=[];f2=[];

%-------------------------------------------------------------------------
disp(' Задайте порядок полинома Pn(x) ')
disp(' (Число узлов интерполяции будет N+1): ')
```

```
N=input(' Введите порядок полинома N ->');
delta=(predel(2)-predel(1))/(NN-1);   % -- Шаг построения графика функции
delta1=(predel(2)-predel(1))/N;         % -- Шаг интерполяции

%-------------------------------------------------------------------------
j=1:NN;      % - количество точек для построения графика функции
i=1:N+1;
ii=0:N;
x(j)=predel(1)+delta*(j-1); % -- Расчет x для построения графика функции
f=feval(txtFUN,x);   % -- расчет f(x) для построения графика функции
% -- Узлы интерполяции (x-координата)
x1(i)=predel(1)+delta1*(i-1);     % -- равномерное распределение
      x2(ii+1)=(predel(1)+predel(2))/2+((predel(2)-predel(1))/2)*…
      cos(((2*ii+1)*pi)/(2*N+2)); % -- оптимальное распределение

% --- расчет функции (имя функции находится в переменной txtFUN)
f1=feval(txtFUN,x1); % -- для узлов с равномерным распределением
f2=feval(txtFUN,x2); % -- для узлов с оптимальным распределением

%-------------------------------------------------------------------------
%------- установки графического окна для отображения графика функции
[a,fig]=figflag('Программа линейной интерполяции: Графики',1);
if a==0
  figN1=figure;
end
 figure(figN1);
 set(figN1,'Name','Программа линейной интерполяции: Графики',...
          'NextPlot','add',...
          'NumberTitle','off');

% --- Построение графика функции и узлов интерполяции
plot(x,f,'-g',x1,f1,'ow',x2,f2,'+w');
title(' Вид функции f(x) в интервале интерполяции');
xlabel(' o - равномерное x - оптимальное распределение узлов')
disp('  ');
disp(' Жми любую клавишу ...')
pause

%-------------------------------------------------------------------------
% ---  Формирование матрицы для расчета коэф-тов интерполирующего полинома
for i=1:N+1,
 for n=0:N,
   A1(i,n+1)=x1(i)^n;  % -- для равномерного распределения узлов
 end
end
%---------------------------------------------------
for i=1:N+1,
 for n=0:N,
   A2(i,n+1)=x2(i)^n;  % -- для оптимального распределения узлов
 end
end

%-------------------------------------------------------------------------
% --- Решение системы уравнений и нахождение коэф-тов полинома
disp('  ');
disp(' Интерполяция функции  f(x) полиномом вида :   ');
disp('    P=C(1)*X^0+C(2)*X^1+C(3)*X^2+...+C(N+1)^N, ');
disp('     где N - порядок интерполирующего полинома ');
disp('  ');
%---------------------------------------------------
C1=inv(A1)*f1'; % -- решение системы уравнений

disp('  Коэф-ты интерполирующего полинома P1(x) ')
```

```
disp('    (при равномерном распределении узлов) ')
for k=1:N+1,
  fprintf('   C1(%g) = %g \n',k-1,C1(k));
end
%-----------------------------------------------------
C2=A2\f2'; % -- решение системы уравнений

disp('   Коэф-ты интерполирующего полинома P2(x) ');
disp('     (при оптимальном распределении узлов) ');
for k=1:N+1,
 fprintf('   C2(%g) = %g \n',k-1,C2(k));
end
%-----------------------------------------------------
disp('  ');
disp(' Жми любую клавишу ...')
pause

% --- расчет интерполирующего полинома для равномерного распределения
for j=1:NN,
P1(j)=0;
 for n=0:N,
  P1(j)=P1(j)+C1(n+1)*x(j)^n;
 end
end
% --- расчет интерполирующего полинома для оптимального распределения
for j=1:NN,
P2(j)=0;
 for n=0:N,
  P2(j)=P2(j)+C2(n+1)*x(j)^n;
 end
end

%-------------------------------------------------------------------------
% -- вызов графического окна
[a,fig]=figflag('Программа линейной интерполяции: Графики',1);
if a==0
  figN=figure;
 else
  figN=fig;
end
figure(figN);
set(gcf,'NextPlot','add');

%--- построение интерполирующих полиномов в виде графика
plot(x,f,'-w',x1,f1,'og',x,P1,'-.g',x2,f2,'ob',x,P2,':b');
title(' Интерполяция функции f(x) полиномом P(x)')
xlabel('  белый - f(x)  зеленый - P1(x)  синий - P2(x)')
pause

%-------------------------------------------------------------------------
% -- Расчет ошибки интерполяции
Error1=abs(f-P1); % -- ошибка для равномерного распределения
Error2=abs(f-P2); % -- ошибка для оптимального распределения

% -- Нахождение максимального значения ошибки
MaxErr1=max(Error1);
MaxErr2=max(Error2);
f0=zeros(1,N+1);

% --- Построение графика ошибки
[a,fig]=figflag('Программа линейной интерполяции: Ошибка',1);
if a==0
  figN2=figure;
else
```

```
 figN2=fig;
end
 figure(figN2);
set(figN2,'Name','Программа линейной интерполяции: Ошибка',...
          'NextPlot','add',...
          'NumberTitle','off');

plot(x,Error1,'-g',x,Error2,'-b',x1,f0,'og',x2,f0,'ob')
title(' Ошибка интерполяции ')
xlabel('  зеленый - P1(x)  синий - P2(x)')
pause

disp('Максимальная ошибка интерполяции функции f(x) :')
fprintf(' MaxErr1= %g  \n MaxErr2= %g \n',MaxErr1,MaxErr2)
disp(' ------------------------------------------------------------- ');
disp('Для печати графика выберите опцию -> File/Print в горизонтальном меню ')
%-------------------------------------------------------------------------
% --- Блок запроса на повторный запуск процедуры расчета коэф-тов
k=1; % --  Начало блока запроса
while k~=0,
  t=input(' Продолжим (y/n) ? ->','s');
  if isempty(t)==1, t='y'; end;
  txt=sscanf(t(1),'%s');
  if txt=='y' | txt=='n'
     k=0;
    else
     k=1;
   end
end % --  Конец блока запроса

if txt=='y'
  key1=1;
elseif txt=='n'
  key1=0;
end
end % --  Конец  цикла по узлам N
%-------------------------------------------------------------------------
% --- Блок запроса на повторный запуск программы
k=1; % --  Начало блока запроса
while k~=0,
t=input(' Конец работы с программой  (y/n) ? ->','s');
if isempty(t)==1, t='y'; end;
txt=sscanf(t(1),'%s');
if txt=='y' | txt=='n'
     k=0;
   else
     k=1;
end
end % --  Конец блока запроса

if txt=='y'
  key=0;
elseif txt=='n'
  key=1;
end
end % --  Конец общего цикла программы while key~=0

clc;
close all;
disp(' Вот и все ... ');
                            % ----------- END PROGRAM
```

ПРИЛОЖЕНИЕ Б

Пример оформления титульного листа ПЗ

Федеральное агентство по образованию

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ
(ТУСУР)

Кафедра КСУП

РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ
УРАВНЕИЙ МЕТОДОМ ГАУССА

Пояснительная записка к курсовой работе по дисциплине
«Вычислительная математика»

Выполнил:
студент гр. № группы
Ф.И.О.
«__» _____ 2007 г.

Проверил:
Преподаватель каф. КСУП
Ф.И.О.
«___»_____ 2007 г.

ТОМСК – 2007

ПРИЛОЖЕНИЕ В

Пример оформления задания на курсовой проект (работу)

Федеральное агентство по образованию

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра компьютерных систем в управлении и проектировании (КСУП)

Утверждаю
Зав. каф. КСУП
Шурыгин Ю.А. _____
«____» _____ 2007 г.

ЗАДАНИЕ
на курсовую работу по дисциплине "Вычислительная математика"

Студентке гр. 587-1 Никитиной Ирине Владимировне

Тема работы: РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ МЕТОДОМ ГАУССА

Исходные данные к работе:

1. Разработать программу решения систем линейных алгебраических уравнений (СЛАУ) методом Гаусса
2. Программа решения СЛАУ должна быть оформлена в виде самостоятельной вычислительной процедуры
3. При прямом ходе необходимо осуществлять выбор главного элемента в матрице коэффициентов
4. Входными данными являются размер СЛАУ, матрица коэффициентов A и вектор свободных членов B. Входные данные вводятся с помощью клавиатуры и из внешнего текстового файла
5. Результат работы процедуры – вектор искомых переменных X. Вывод осуществляется на экран монитора и во внешний текстовый файл.
4. Выбор среды реализации предоставляется разработчику
5. При тестировании рассмотреть решение СЛАУ с матрицей коэффициентов близкой к вырожденной размером не менее 20x20

Литература:

1. М.В. Черкашин. Вычислительные методы. Курс лекций (часть 2). Томск, ТУСУР, 2002.
2. В.М. Вержбицкий. Численные методы (линейная алгебра и нелинейные уравнения): учебное пособие для ВУЗов. – М.: Высшая школа, 2000. –266 с.
3. Л.И. Турчак. Основы численных методов. – М., Наука, 1987. – 320с.

Задание принял к исполнению Никитина И.В.
«___» _____ 2007 г.

Руководитель Черкашин М.В.

ПРИЛОЖЕНИЕ Г

Пример оформления содержания

Содержание

ПРИЛОЖЕНИЕ Д

Пример оформления списка литературы

Список использованных источников

1. Хьюлсман Л.П., Аллен Ф.Е. Введение в теорию и расчет активных фильтров. – М.: Радио и связь, 1984. – 384с.
2. Лэм Г. Аналоговые и цифровые фильтры. - М.: Мир, 1982. – 592 с.
3. Вай Кайчень Теория и проектирование широкополосных согласующих цепей.- М.: Связь, 1979. - 288 с.
4. Тейксейра С., Паченко К. Delphi 4. Руководство разработчика. - СПб.: ”Вильямс”, 1999. - 912с.
5. Фаронов В.В. Delphi 4. Учебный курс.- М.: “Нолидж”, 1998.- 464с.
6. Программирование с среде CBuilder [электронный ресурс] / режим доступа к ресурсу: http:// www.metka.cbuilder.ru