Министерство образования и науки РФ
Сибирская государственная автомобильно-дорожная академия (СибАДИ)
Кафедра «Дорожные машины»

ОСНОВЫ НАУЧНЫХ ИССЛЕДОВАНИЙ

Методические указания
к курсу лабораторных работ

для специальностей 170900 «Подъемно-транспортные, строительные, дорожные машины и оборудование» и 230100 «Сервис и техническая эксплуатация транспортных и технологических машин и оборудования»

Составитель В. А. Мещеряков

Омск
Издательство СибАДИ
2004

УДК 629.113
ББК

Рецензент: канд. техн. наук, доцент В. П. Денисов

Работа одобрена методической комиссией факультета ТТМ в качестве методических указаний к курсу лабораторных работ по дисциплине «Основы научных исследований» для специальностей 170900 и 230100.

Основы научных исследований: Методические указания к курсу лабораторных работ для специальностей 170900 «Подъемно-транспортные, строительные, дорожные машины и оборудование» и 230100 «Сервис и техническая эксплуатация транспортных и технологических машин и оборудования»/ Сост. В. А. Мещеряков. – Омск: Изд-во СибАДИ, 2004. – 28 с.

Методические указания содержат задания к лабораторным работам до дисциплине «Основы научных исследований» и материалы, направленные на изучение системы инженерных и научных расчетов MATLAB. Основное внимание уделено разделам MATLAB, ориентированных на обработку и визуализацию числовых данных, построение регрессионных моделей, решение алгебраических и дифференциальных уравнений.

Библиогр.: 5 назв.

ВВЕДЕНИЕ

Автоматизация научных расчетов, ориентированных на создание новых технических объектов, базируется на применении современных программных средств. Курс лабораторных работ направлен на изучение системы инженерных и научных расчетов MATLAB, разработанной фирмой The MathWorks, Inc. Эта система представляет собой язык программирования высокого уровня для технических расчетов. Она включает в себя проведение вычислений, визуализацию и программирование в удобной среде. Наиболее распространенные научные задачи, решаемые с помощью MATLAB – это:

• математические вычисления;
• моделирование;
• анализ и визуализация данных;
• научная и инженерная графика;
• разработка приложений, включая создание графического интерфейса.

MATLAB (**mat**rix **lab**oratory – матричная лаборатория) – это интерактивная система, в которой основным элементом данных является массив. Это позволяет эффективно решать различные задачи, связанные с техническими вычислениями, особенно те, в которых используются матрицы и вектора. MATLAB фактически представляет собой стандартный инструмент для применения математических численных методов в исследованиях технических объектов.

В MATLAB важная роль отводится специализированным группам программ, называемым toolboxes. Они очень важны для большинства пользователей MATLAB, так как позволяют изучать и применять специализированные методы. Toolboxes – это всесторонняя коллекция функций – программ, написанных на языке MATLAB (M-файлов), которые позволяют решать частные классы задач. Toolboxes применяются для обработки экспериментальных данных, моделирования механических систем, решения задач оптимизации, исследования систем управления и т. д.

Работа с системой MATLAB начинается с запуска программы из главного меню Windows или с помощью ярлыка на рабочем столе. В результате на дисплее открывается главное окно MATLAB: `окно управления` Command Window. Основную часть окна занимает `рабочее поле`, предназначенное для ввода команд с клавиатуры и просмотра результатов. Оно является основным средством взаимодействия пользователя с системой и обеспечивает ведение диалога посредством командного языка MATLAB. Это означает, что любая информация, вводимая пользователем в командной строке, воспринимается системой как команда, подлежащая исполнению. Командная строка начинается с символа приглашения: **>>**.

Вывод информации в рабочее поле производится в режиме прокрутки, т.е. при выводе на экран очередного сообщения более старая информация сдвигается вверх. Информация, введенная пользователем или выведенная в рабочее поле системой, сохраняется в течение всего сеанса работы. При необходимости ее можно просмотреть, используя полосы вертикальной и горизонтальной прокрутки.

Команды, вводимые пользователем в течение сеанса работы, сохраняются в буфере команд. Выполнявшуюся ранее команду можно выполнить без повторного набора, вызвав ее на экран с помощью клавиш управления курсором (↑ или ↓).

Все переменные, создаваемые пользователем, содержатся в памяти системы MATLAB – в рабочей области (Workspace). MATLAB способен использовать всю доступную оперативную память ЭВМ, что делает возможной обработку больших объемов данных.

ВВОДНОЕ ЗАНЯТИЕ

Цель занятия: ознакомиться с основными возможностями системы инженерных и научных расчетов MATLAB 5.x или MATLAB 6.5 с помощью демонстрационной программы `introrus.m`.

Следует законспектировать команды создания переменных, векторов, матриц, арифметические операции, функции построения графиков и т.д.

Для этого необходимо запустить из командной строки в рабочем поле MATLAB русифицированную программу `introrus.m`, написанную на языке MATLAB 4 версии:

`>> introrus` (нажать Enter)

Смена экранов осуществляется нажатием любой клавиши. При выполнении графических команд появляется окно `Figure1`. Для просмотра графиков после каждой смены экрана можно переключаться между окнами `Figure1` и главным окном MATLAB с помощью мыши или клавиш `Alt+Tab`.

Прервать выполнение программы `introrus` можно нажатием клавиш `Ctrl+C`.

По окончании работы с программой `introrus` запустите из командной строки программу `demo`. Особое внимание следует обратить на разделы, посвященные 2D- и 3D-визуализации.

По окончании занятия выписанные команды и функции MATLAB с пояснениями проверяются преподавателем.

Лабораторная работа № 1
ГЛАВНОЕ МЕНЮ MATLAB

Главное меню окна Command Window по форме аналогично меню пользователя любого Windows-приложения. Главное меню MATLAB 5.2 содержит следующие разделы:

- File (команды работы с файлами и опции настройки системы);
- Edit (команды редактирования информации, отображенной в рабочем поле);
- Window (список открытых окон приложения);
- Help (команды вызова доступных средств помощи).

Ниже строки основного меню расположена панель кнопок, обеспечивающих быстрый доступ к наиболее часто используемым командам из разделов меню.

Из перечисленных выше разделов меню наибольший интерес представляет раздел File. Содержащиеся в нем команды разбиты на несколько групп:

- команды работы с файлами (New, Open, Run Script);
- команды работы с рабочей областью (памятью) MATLAB;
- опции настройки системы (Set Path, Preferences);
- команды вывода на печать (Print Setup, Print);
- список файлов, открывавшихся в последнее время;
- команда выхода из MATLAB (Exit MATLAB).

При выполнении лабораторной работы следует законспектировать назначение подпунктов меню; создать М-файл с помощью редактора-отладчика программ Editor/Debugger (пункт меню File→New→M-File); создать графическое окно (пункт меню File→New→Figure); открыть существующий М-файл (пункт меню File→Open); запустить на выполнение программу introrus.m (пункт File→Run Script, Browse); просмотреть содержимое рабочей области, т.е. какие переменные имеются в памяти MATLAB (пункт File→Show Workspace).

Для того, чтобы запустить на выполнение из командной строки какую-либо программу, написанную на языке MATLAB, т.е. созданный пользователем М-файл (например, introrus.m), предварительно нужно указать системе путь к этому файлу. При этом используется пункт меню File→Set Path. Появляется окно MATLAB Path, которое обеспечивает выбор каталога (папки), в котором лежат М-файлы. Например, для добавления рабочей папки с М-файлами пользователя d:\Study\31SM

необходимо нажать на кнопку Add to Path... и выбрать эту папку в дереве каталога. Ту же операцию можно выполнить из командной строки в рабочем поле, выполнив команду

```
>> path(path, 'd:\Study\31SM')
```

Добавьте к списку путей каталог, содержащий М-файлы.

File→Preferences. Опция обеспечивает выбор форматов представления числовой и текстовой информации в командном окне, формата копирования данных в буфер обмена и настройку ряда других параметров системы.

Окно Preferences имеет три вкладки: General (общие параметры системы), Command Window Font (выбор шрифта для командного окна MATLAB) и Copying Options (опции копирования).

Вкладка General дает возможность произвести настройку следующих параметров.

1. Формат вывода числовых данных (Numeric Format); пользователю предлагается выбрать один из 10 вариантов:

- Short – вывод в формате с фиксированной точкой, выводится 4 знака после десятичной точки (используется по умолчанию);
- Long – формат с фиксированной точкой, выводится 14 знаков после точки;
- Hex – шестнадцатеричный формат;
- Bank – формат для вывода денежных сумм;
- Plus – вывод только знака положительных и отрицательных чисел;
- Short E – вывод дробных чисел в формате с плавающей запятой (в экспоненциальной форме), мантисса содержит 5 значащих цифр;
- Long E – вывод дробных чисел в экспоненциальной форме, в дробной части

мантиссы выводится 15 значащих цифр, для вывода порядка числа в двух последних форматах используется три знакоместа;

- Short G – «улучшенный» короткий формат, в дробной части числа выводится одна дополнительная значащая цифра;
- Long G – «улучшенный» длинный формат, в дробной части числа также выводится одна дополнительная значащая цифра;
- Rational – формат, при использовании которого дробные числа представляются в виде простых дробей (числитель/знаменатель).

2. Формат использования рабочего поля командного окна. Может быть указан либо Loose – «свободный» (используется по умолчанию), либо Compact. При использовании формата Loose выводимые системой сообщения (ответы) разделяются одной пустой строкой. Формат Compact обеспечивает вывод сообщений в каждой строке рабочего поля.

Выбирая разные форматы, вводите в командной строке дробные числа. Обратите внимание на формат вывода результата.

Вкладка `Copying Options` позволяет установить параметры для операций копирования графиков (например, для последующей вставки в документы MS Word). `Clipboard Format` – выбор формата копирования в буфер обмена:

- `Windows Metafile` – WMF-формат; используется для хранения векторных рисунков (при увеличении размеров изображения его качество не ухудшается);
- `Windows Bitmap` – BMP-формат; обеспечивает хранение растровых (точечных) изображений.

Пункт меню `Edit` аналогичен пункту «Правка» MS Word. Можно копировать выделенное содержимое рабочего поля (`Edit→Copy`), вставить текст в командную строку (`Edit→Paste`), вырезать текст из активной командной строки (`Edit→Cut`).

Очистка `рабочего поля` производится с помощью команды `Edit→Clear Session`. Важно помнить, что переменные в `рабочей области` (в памяти) при этом сохраняются.

Пункт меню `Window` позволяет переключаться между открытыми окнами MATLAB.

Пункт `Help→MATLAB Help` вызывает справку MATLAB. В поле `MATLAB Help` необходимо набрать имя интересующей функции.

Вызовите справку по функции построения плоских графиков `plot`.

Арифметические операции в MATLAB

Основными объектами MATLAB, применяемыми в научных исследованиях, являются числа, вектора и массивы. Создание объектов такого типа можно выполнить с помощью соответствующих команд:

```
>> a = 2
>> a = [2 5 6 8 9]      или      >> a = [2; 5; 6; 8; 9]
>> a = [2 5 6; 8 9 1; 4 3 0]
```

Простейшие арифметические операции – **сложение и вычитание** – реализуются с помощью операторов «+» и «–»:

```
>> z = x + y              или            >> z = x – y
```

Массивы или вектора `x` и `y` должны иметь одинаковую размерность.

Создайте переменные следующих типов и выполните над ними операции сложения и вычитания:

число	число
вектор-строка	вектор-строка
вектор-столбец	вектор-столбец
массив	массив
вектор	число
массив	число

Умножение в MATLAB может быть двух типов: поэлементное «**.***» и матричное «*****». Для чисел эти операции равнозначны. Поэлементное произведение массивов или векторов одинаковой размерности `x` и `y` (массивы или вектора как бы накладываются друг на друга, и определяются произведения соответствующих элементов массивов):

```
>> z = x .* y
```

Порядок поэлементного перемножения переменных не важен.

Матричное произведение сложнее: строки матрицы `x` умножаются на столбцы матрицы `y`, и суммы этих произведений составляют соответствующие элементы матрицы `z`.

```
>> z = x * y
```

Количество столбцов матрицы `x` должно равняться количеству строк матрицы `y`. Порядок матричного перемножения переменных важен.

Создайте переменные следующих типов и выполните над ними операции поэлементного умножения:

число	число
вектор-строка	вектор-строка
вектор-столбец	вектор-столбец
массив	массив
вектор	число
массив	число

Создайте переменные следующих типов и выполните над ними операции матричного умножения:

число	число
вектор-строка	вектор-столбец
вектор-столбец	вектор-строка
массив	массив
вектор	число
массив	число

Деление в MATLAB может быть четырех типов: поэлементное правое «**./**» и левое «**.**», а также матричное правое «**/**» и левое «****». Правое деление выполняет операцию $x/y = \frac{x}{y}$, а левое – $x \backslash y = \frac{y}{x}$.

Для чисел поэлементные и матричные операции равнозначны. Поэлементное деление массивов или векторов одинаковой размерности `x` и `y` (массивы или вектора как бы накладываются друг на друга, и определяются частные соответствующих элементов массивов):

```
>> z = x ./ y          или          >> z = x .\ y
```

Матричное деление используется при решении систем линейных уравнений и будет рассмотрено в лабораторной работе № 6.

Найдите результат деления чисел с помощью операции матричного деления. Создайте переменные следующих типов и выполните над ними операции правого и левого поэлементного деления:

число	число
вектор-строка	вектор-строка
вектор-столбец	вектор-столбец
массив	массив
вектор	число
массив	число

Возведение в степень также может быть поэлементным «**.^**» или матричным «**^**»:

```
>> z = x .^ y  (элементы массива x возводятся в степень из масс. y)
>> z = x ^ y   (квадратная матрица x умножается сама на себя y раз)
```

Вычислите $z = \sqrt{144}$, $z = \sqrt[3]{8}$, $z = 4^{-2}$, $z = 2^{10}$.

Лабораторная работа № 2
СПЕЦИАЛЬНЫЕ СИМВОЛЫ MATLAB

«**:**» – индексирование массива. Применяется для формирования векторов и массивов.

```
>> x = 1:9      («начало вектора» : «конец вектора»)
>> x = 1:2:9    («начало вектора» : «шаг» : «конец вектора»)
```

Сформируйте вектор из чисел от 5 до 55 с шагом 10; от 5 до -10 с шагом -5.

«**( )**» – круглые скобки используются для указания последовательности выполнения операций, для указания аргументов функций и при индексировании векторов и матриц.

```
>> z = (2 + 4) * 5
```

```
>> z = sin(pi)
>> z = x(3)      или     >> z = x(1,2)
```

«**[ ]**» – квадратные скобки применяются при формировании массивов (см. арифметические операции).

```
>> x = []                    (создание пустого массива)
```

Сформируйте из чисел 2, 3, 8, -6, 9, 0 **вектор, массивы из 2 и из 3 строк.**

« **.** » – десятичная точка применяется при задании десятичных дробей:

```
>> x = 1.25
```

Сформируйте вектор из чисел от 0 до 1.5 с шагом 0.1.

« **...** » – продолжение строки используется при вводе длинных команд:

```
>> x = [1 3 5 4 5 6 ...          (здесь нужно нажать Enter)
5 4 3 2 1 2 3]              (окончание команды, нажать Enter)
```

В результате формируется вектор

```
x =
   1 3 5 4 5 6 5 4 3 2 1 2 3
```

« **,** » – разделяет элементы векторов-строк, аргументы функций, матричные индексы (номера строк и столбцов) и операторы в одной командной строке:

```
>> x = [1, 3, 5, 4, 5, 6]
>> z = x(1,2)
>> y = x + 2, z = x + 3 (выполняются две операции, их результаты выводятся на экран).
```

« **;** » – разделение строк матриц, разделение операторов в командной строке и подавление вывода результата операций на экран

```
>> x = [1 3 5; 4 5 6]
>> y = x + 2; z = x + 3;   (выполняются две операции, результаты не выводятся на экран; это удобно при промежуточных вычислениях).
```

« % » – после этого символа пишется комментарий, т.е. любой поясняющий текст, игнорируемый системой:

```
>> y = 20 % координата Y в метрах
```

« **=** » – оператор присваивания

```
>> x = 5 % переменной x присвоено значение 5
```

«'» – апостроф используется для задания текстовых строк и транспонирования матриц и векторов:

```
>> a = 'Это текстовая строка'
>> x = [2 3 4 5], y = x'    % строка x, столбец y
```

Специальные переменные и константы MATLAB

`ans` (от английского слова answer – ответ) – переменная содержит результат последней арифметической операции, если результат не был присвоен какой-либо переменной:

```
>> 2 + 3 % явного присваивания нет, результат - в ans
ans =
     5
>> x = 2 + 3 % результат операции присвоен переменной
x
x =
     5
```

Переменная `ans` может быть использована при выполнении последующих команд. Например, выражение `ans+4` будет воспринято системой как команда, требующая увеличить значение переменной `ans` на 4.

`pi` – число π. **Чтобы вывести на экран значение любой переменной или константы, нужно ввести ее имя в командной строке и нажать Enter.**

`i` или `j` – мнимая единица ($\sqrt{-1}$), т.е. комплексное число. **Выведите на экран значения этих чисел.** При желании можно изменить значения этих переменных:

```
>> i = 5, j = 7
```

Для того, чтобы восстановить значения специальных переменных и констант, необходимо **выполнить очистку рабочей области** командой `clear all`. При этом удаляются все переменные из памяти MATLAB.

`Inf` – бесконечность (∞). Возникает, например, при делении на 0. С константой `Inf` можно выполнять такие же операции, что и с числами.

`NaN` – неопределенность (Not a Number). Не является численным объектом. Возникает при делении 0/0 или ∞/∞.

Элементарные функции MATLAB

В системе MATLAB имеется обширная библиотека математических функций, благодаря которым MATLAB может использоваться как мощный математический калькулятор. Каждой функции соответствует определенное имя. Функция возвращает результат в зависимости от значений аргументов. Аргументы функции – числа, вектора или массивы – указываются в круглых скобках.

`abs(x)` – возвращает модуль (абсолютное значение) `x`.
`sign(x)` – возвращает знак (абсолютное значение) `x`.
Определите модуль и знак положительных и отрицательных чисел, векторов, массивов.

`angle(x), real(x), imag(x)` – возвращают соответственно аргумент, вещественную и мнимую часть комплексных чисел. `abs(x)` для комплексных чисел возвращает модуль. **Определите значения этих функций для комплексного числа** `x = 3 + 4i`.

`ceil(x), fix(x), floor(x), round(x)` – функции округления вещественных чисел: до большего целого, отбрасывание дробной части, до меньшего целого и до ближайшего целого соответственно. **Округлите с помощью этих функций дроби:** 4.3, 4.6, –5.2, –5,8.

Тригонометрические функции:
`sin(x), csc(x), cos(x), sec(x), tan(x), cot(x)` – синус, косеканс, косинус, секанс, тангенс, котангенс. Аргумент `x` указывается в радианах. Чтобы перевести аргумент `y` из градусов в радианы, можно воспользоваться формулой `x = y / 180 * pi`.
Вычислите тригонометрические функции углов: 30°, 60°, 45°, 90°.

Обратные тригонометрические функции:
`asin(x), acos(x), atan(x), acot(x)` – арксинус, арккосинус, арктангенс, арккотангенс. Результат возвращается в радианах.
Чтобы перевести угол `y` из радиан в градусы, можно воспользоваться формулой `z = y * 180 / pi`.
Вычислите обратные тригонометрические функции 0.5, 1, –0.866, –0.5, 0.

`log(x), log10(x)` – натуральный ($\ln x$) и десятичный ($\lg x$) логарифмы. **Вычислите натуральный логарифм** 2.71828, 1/2.71828**, десятичный логарифм** 100, 10000, 0.01, –1, 0.

`exp(x)` – функция экспоненты (e^x). **Определите** значение числа e, экспоненту нуля e^0, экспоненту натурального логарифма 5 ($e^{\ln 5}$).

`sqrt(x)` – функция извлечения квадратного корня. **Найдите корень одной переменной, вектора и массива.**

Специальные массивы, матрицы и операции над ними

`zeros(n)` или `zeros(m,n)` – формирование вектора длиной `n` или массива размером `m x n`, заполненных нулями.
`ones(n)` или `ones(m,n)` – формирование вектора длиной `n` или массива размером `m x n`, заполненных единицами.
Создайте строку и столбец длиной 7, заполненные нулями и единицами.
`eye(n)` – формирование единичной матрицы размером `n x n`.
Создайте единичные матрицы размером `2 x 2, 5 x 5`.

Определение размеров массивов и векторов:
`size(a)` – определение количества строк и столбцов массива `a`.
`length(a)` – определение длины вектора `a`.

Лабораторная работа № 3
ДВУМЕРНЫЕ ГРАФИКИ MATLAB

В научных исследованиях для анализа зависимостей между изучаемыми показателями часто используются графики, показывающие связь двух величин. Двумерные графики в MATLAB строятся с помощью функции `plot()`. Формы вызова этой функции:
`plot(y)` – по оси абсцисс указываются номера точек, по оси ординат – значения из массива `y`. Если `y` – массив, графики строятся для каждого столбца.
`plot(x, y)` – график `y` в зависимости от `x`. `x` и `y` должны иметь одинаковую размерность.
`plot(x, y, '*')` – точки данных изображаются в виде звездочек. Выражение в апострофах – это `атрибут` графика. Атрибуты, позволяющие управлять стилем изображения графиков, могут быть сочетаниями следующих символов:

Тип линии		Тип точек данных		Цвет	
Непрерывная	-	Точка	.	Желтый	y
Штриховая	--	Плюс	+	Фиолетовый	m
Пунктир	:	Звездочки	* или p или h	Голубой	c
Штрих-пунктир	-.	Кружок	o (Малое латинское)	Красный	r
		Крестик	x (Малое латинское)	Зеленый	g
		Треугольник	^ или < или > или v	Синий	b
		Квадрат	s	Белый	w
		Ромб	d	Черный	k

Например:

```
y = [1 3 5 6 7 7.5 7.8 8 8.1],  plot(y, ':+g')
```

Управление цветами также осуществляется с помощью функции `whitebg`, которая меняет цвет фона и графиков на противоположный.

Команда типа `plot(x1, y1, '-*', x2, y2, '-.')` позволяет строить несколько графиков с разными атрибутами в одном окне.

Постройте графики, используя разные атрибуты, для следующих данных: `y = [2 4 5 6 5 4 3 4 5 6 7 8 7 6 7 8 9];`

```
x = [3 5 7 9 11 13 15], y = [12 11.5 10 9.5 8 7.5 4];
```

$$x = [50\ 100\ 150\ 200],\ y = \begin{pmatrix} 1 & 9 & 4 & 0 \\ 2 & 8 & 4 & 2 \\ 5 & 5 & 4 & 4 \\ 7 & 2 & 4 & 6 \end{pmatrix}.$$

Построение графиков различных функций выполняется следующим образом. Сначала задается интервал и шаг изменения аргумента `x`, при этом формируется вектор `x`. Шаг изменения `x` нужно выбирать так, чтобы обеспечить достаточную точность изображения графика. В зависимости от `x` вычисляется `y` по заданной функции, и формируется вектор `y` той же длины, что и `x`. Затем строится график командой `plot`. Например, построим график синуса на интервале от 1 до 10:

```
x = 1:0.01:10; % задаем аргумент x с шагом 0.01
y = sin(x); % вычисляем y – это вектор из 901 чисел
plot(x, y, '.r-') % строим график
```

Наложение координатной `сетки` на график производится командой `grid`: `grid` (выключить или включить сетку), `grid on` (включить), `grid off` (выключить).

Режим `увеличения` участков графика с помощью мыши включается командой `zoom`: `zoom` (выключить или включить), `zoom on` (включить), `zoom off` (выключить).

Выделить интересующий участок графика (в том числе уменьшить график) можно с помощью команды управления осями координат:
`axis([`x_{min} x_{max} y_{min} y_{max}`])` – будет показан участок, ограниченный координатами x_{min}, x_{max}, y_{min}, y_{max}. Например:
`axis([-5 5 0 20])` % Интервал по X от -5 до 5, по Y от 0 до 20.

Надписи в графическом окне могут быть добавлены командами:
`title('Заголовок')` – выводит заголовок графика;
`xlabel('Надпись по X')` – выводит надпись по оси абсцисс;
`ylabel('Надпись по Y')` – по оси ординат.

По умолчанию каждая команда `plot` заменяет старый график на новый в графическом окне `Figure1`. Режим наложения графиков в одном окне включается командой удержания окна `hold`:
`hold` (выключить или включить), `hold on` (включить), `hold off` (выключить). Например: `plot(x1, y1); hold on; plot(x2, y2)`
Наложите график функции косинуса на график синуса (разными стилями), создайте координатную сетку и включите режим увеличения графиков. Обозначьте оси, вставьте заголовок графика.

Одновременно в системе можно держать открытыми несколько графических окон. При этом график строится в активном окне, которое последний раз было выбрано с помощью мыши. Команда создания нового пустого графического окна `figure`. Окна нумеруются автоматически.
Создайте 3 графика в разных графических окнах. Покажите область по X от -10 до 10, по Y от -20 до 20.

Графическое окно также можно разбивать на независимые области – подокна, в каждом из которых можно строить отдельный график. Для этого служит команда `subplot(m, n, k)`, где `m` – количество окон по вертикали, `n` – по горизонтали, `k` – номер активного подокна, где будет построен график. Например: `subplot(2, 3, 1), plot(x1, y1)` – разбивает графическое окно на 6 частей, первое из которых активно, и где строится график. `subplot(2, 3, 2), plot(x2, y2)` – график строится во втором подокне из 6, и т. д.
Разбейте графическое окно на 4 части, и в каждом подокне постройте график с координатной сеткой.

Постройте разными стилями графики функций, наложите координатную сетку:

1) $y = 2x^2 + 3,\ x \in [-2; 2]$;
2) $y = 2(x-3)^3 - 4,\ x \in [-5; 8]$;
3) $y = e^x - 4x,\ x \in [-5; 5]$;
4) $y = (\sin x)/x,\ x \in [1; 10]$.

Обратите внимание: необходимо использовать поэлементные арифметические операции. Шаг `x` выберите самостоятельно.

Лабораторная работа № 4
ТРЕХМЕРНЫЕ ГРАФИКИ MATLAB

Взаимосвязи трех исследуемых показателей визуально отображаются в виде трехмерных графиков. Это позволяет, например, оценить поведение исследуемой величины z в зависимости от изменения факторов x и y.

Простейший трехмерный график – это линия в пространстве (например, так можно изобразить траекторию движения материальной точки). Такой график строится с помощью функции `plot3()` – трехмерного аналога функции `plot()`.

`plot3(x, y, z)`, где `x, y, z` – вектора одинаковой длины, строит точки с координатами `x(i), y(i), z(i)` и соединяет их прямыми.

`plot3(x, y, z, '*-')` – строит график с учетом заданного атрибута.

`plot3(X, Y, Z)`, где `X, Y, Z` – массивы одинаковой размерности, строит отдельную линию для каждого столбца `X(i,:), Y(i,:), Z(i,:)`, цвета линий циклически чередуются.

`plot3(X, Y, Z, '*-')` – строит семейство линий с учетом заданного атрибута.

Технология построения `объемных фигур` в MATLAB такова.

1). Задаются интервалы и шаг изменения x и y (так же, как интервал изменения аргумента для плоских графиков). Формируются вектора x и y.

2). Вычисляются координаты точек вспомогательной плоскости XY, над которой будет построена поверхность z. Для этого используется функция `meshgrid`. В результате формируются массивы координат X, Y.

3). В зависимости от значений координат вспомогательной плоскости X и Y вычисляется высотная координата поверхности z.

4). Строится график объемной фигуры.

Например, построим трехмерный график функции $z = x \cdot e^{-x^2-y^2}$ на интервале $x \in [-2; 2]$, $y \in [-2; 2]$.

```
x = -2:0.1:2;   % задаем интервал и шаг изменения x
y = -2:0.1:2;   % задаем интервал и шаг изменения y
[X, Y] = meshgrid(x,y); %Вспомогательная плоскость XY
z = X .* exp(- X.^2 - Y.^2); % z в зависимости от X,Y
plot3(X, Y, z, '*-') % график в виде семейства линий
```

Команды `grid, zoom, hold, subplot, figure, title, xlabel, ylabel` используются так же, как для двумерных графиков. Надпись по оси z можно добавить командой `zlabel('Ось Z')`.

Управление осями координат с учетом третьей оси z:

axis([x_{min} x_{max} y_{min} y_{max} z_{min} z_{max}]) – будет показан участок, ограниченный координатами x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max}. Например, axis([-1 1 -1 1 -0.1 0.1]) увеличит центральную часть графика.

Режим трехмерного вращения графика с помощью мыши включается/выключается командой rotate3d.

Проекцию трехмерной поверхности на плоскость XY, т.е. линии уровня, можно получить с помощью функции contour(x, y, z), Такие изолинии используются для исследования экстремумов z. Координата z формируется по описанной выше методике.

Линии уровня z в пространстве строятся функцией contour3(x, y, z).

Трехмерная сетчатая поверхность, цвет которой меняется в зависимости от высоты по шкале z, строится с помощью функции mesh(x, y, z).

meshc(x, y, z) – сетчатый график с линиями уровня.

meshz(x, y, z) – сетчатый график с плоскостью отсчета на нулевом уровне, ограниченный вертикальными плоскостями.

Трехмерная сплошная поверхность строится с помощью функции surf(x, y, z).

surfl(x, y, z) – сплошная поверхность с подсветкой.

surfc(x, y, z) – сплошная поверхность с линиями уровня.

Поверхности изображаются в разных цветовых гаммах в зависимости от выбранной палитры. Управление палитрой осуществляется функцией colormap('Название палитры'). Например, colormap('hot') включает яркие оттенки. Название палитры можно выбирать из следующих: hot, hsv, gray, bone, copper, pink, flag, cool.

Управление подсветкой графика обеспечивается командой shading:

shading faceted – подсветка по сегментам (по умолчанию),

shading flat – делает ребра сегментов невидимыми,

shading interp – интерполяция (сглаживание) цветов.

Постройте все виды рассмотренных трехмерных графиков. Сравните между собой различные цветовые палитры и режимы подсветки.

Постройте трехмерные графики следующих функций, наложите координатную сетку, включите режим вращения графиков:

1) $z = (x+1)^2 + (yx-2)^2 + 2$, $x \in [-5;5]$, $y \in [-5;5]$;

2) $z = 2(x-5)^2 - \frac{1}{2}(y+3)^2 - 2$, $x \in [-10;10]$, $y \in [-15;8]$;

3) $z = \sin y \cdot e^x - 4x$, $x \in [-10;10]$, $y \in [-20;20]$;

4) $z = y^2 \cdot tg\ x$, $x \in \left[\frac{\pi}{3}; \frac{\pi}{2}\right]$, $y \in [-20; 20]$.

Обратите внимание: необходимо использовать поэлементные арифметические операции. Шаг `x` и `y` выберите самостоятельно.

Лабораторная работа № 5
АППРОКСИМАЦИЯ И ИНТЕРПОЛЯЦИЯ ДАННЫХ

Аппроксимация данных полиномом

Целью анализа экспериментальных данных часто является выявление аналитической зависимости между случайными переменными. При этом какая-либо физическая величина определяется как функция другой величины: $y = f(x)$. Такую функциональную зависимость называют регрессионной моделью. Распространенным подходом к построению регрессионных моделей на основе экспериментальных данных является подбор функциональной зависимости $y = a_0 + a_1 x + a_2 x^2 + ... + a_n x^n$. Такой подход называется аппроксимацией данных полиномом (многочленом) n-й степени.

Коэффициенты полинома a_0, a_1, ..., a_n подбираются по критерию наименьших квадратов ошибки. В MATLAB коэффициенты аппроксимирующего полинома степени `n` определяются функцией

```
p = polyfit(x, y, n)
```

`p` – это вектор-строка, содержащая коэффициенты в порядке a_n, ..., a_1, a_0.

После расчета коэффициентов полинома задается интервал `x` с равномерным малым шагом, и на этом интервале рассчитывается значение величины `y` по модели с помощью функции `polyval`. Расчетные значения по модели на интервале `xm` вычисляются следующим образом:

```
ym = polyval(p, xm)
```

Затем строятся график экспериментальных значений `x` и `y` и график регрессионной модели (прогнозные значения `ym`).

Например, необходимо построить полиномиальную модель 3-го порядка $y = a_0 + a_1 x + a_2 x^2 + a_3 x^3$ для буксования y в зависимости от тягового усилия x колесной землеройно-транспортной машины и затем спрогнозировать среднее значение буксования при тяговом усилии $x = 22$ кН. Исходные экспериментальные данные:

```
% сила тяги в кН:
x = [0 1.3 4.7 5.0 7.0 9.1 9.5 10.7 11.0 11.5 11.8 12.4 ...
13.0 14.4 14.8 15.5 16.3 16.4 16.7 16.8 18.7 19.2 19.7 ...
23.0 23.6 24.7 26.0 27.85 30 30.5];
```

```
% буксование:
y = [0 0.007 0.015 0.01 0.017 0.035 0.042  0.052 0.049  0.05
...
0.038  0.051  0.04  0.048 0.06  0.073  0.09  0.13  0.17 ...
0.11   0.2  0.23  0.21  0.38  0.49  0.5  0.53  0.71  1  0.99];
plot(x, y, '.b'); grid on; hold on % график эксперим. данных
xlabel('Сила тяги, кН'); ylabel('Буксование')
% вычисляем коэффициенты полинома 3-й степени p:
n = 3;
p = polyfit(x, y, n)
% Задаем интервал x с равномерным мелким шагом:
xm = 0:0.1:30.5;
% Расчетные значения y по полиномиальной регрессионной модели:
ym = polyval(p, xm);
plot(xm, ym, 'g'); % график регрессионной модели
% Точечный прогноз буксования при силе тяги 22 кН:
xm = 22; ym = polyval(p, xm); plot(xm, ym, 'r*');
```

Постройте линейную модель парной регрессии (полином 1 степени) для тяговой мощности автогрейдера ДЗ-143-1 y (кВт) в зависимости от силы тяги x (кН) на второй передаче:

```
x=[26.1064 23.5488 22.9819 22.8450 26.0743 26.3049 24.0401 ...
   29.2168 31.9103 28.6852 28.4548 24.3532 25.0969 26.4924 ...
   23.1202 24.3695 23.0572 23.5488 23.5488 22.5676 24.7159 ...
   26.6253 24.9035 27.4736 25.1666 22.5204 25.9831 34.5612 ...
   35.7871 33.3608 33.8727 32.3796 29.4360 25.5112 27.4736 ...
   24.5300 28.0405 28.4548 32.8197 28.3894];

y=[37.2949 33.6411 31.9193 30.8716 36.2143 37.5784 31.6318 ...
   39.4822 44.3198 38.7639 38.4524 33.8239 35.8527 37.8463 ...
   30.4213 32.9317 32.0239 32.7067 32.7067 31.3439 36.3470 ...
   40.3413 36.6228 40.4024 38.1312 33.1183 35.1123 45.4752 ...
   48.3609 45.0822 47.0454 44.9717 40.8833 35.4322 38.1578 ...
   34.0694 40.0578 40.6497 45.5829 40.5562];
```

В версиях MATLAB 6.x в меню графических окон Tools есть инструмент Basic Fitting, позволяющий автоматически аппроксимировать данные полиномами, выводя уравнение регрессионной модели на экран.

Построение регрессионных моделей возможно также в системе **Microsoft Excel**. Для этого необходимо разместить исходные данные x и y в строках или в столбцах, выделить их мышью, и построить точечную диаграмму с помощью Мастера Диаграмм (меню Вставка→Диаграмма). Затем щелкнуть правой кнопкой мыши по точкам исходных данных на графике и выбрать в выпадающем подменю пункт Добавить линию тренда. Затем в окне Линия тренда выбрать тип модели (полиномиальная или линейная) и во вкладке Параметры указать Показывать уравнение на диаграмме.

Постройте для двух рассмотренных примеров регрессионные модели с помощью Мастера диаграмм MS Excel.

Интерполяция данных

Интерполяция в математике и статистике – это отыскание промежуточных значений величины по некоторым известным ее значениям. Например, отыскание значений функции $f(x)$ в точках x, лежащих между точками (узлами интерполяции) $x_0 < x_1 < ... < x_n$, по известным значениям $y_i = f(x_i)$ (где $i = 0, 1, ..., n$).

Если точки, соответствующие исходным экспериментальным данным x и y, на графике соединять прямыми отрезками, график будет иметь вид ломаной линии. Однако многие процессы в природе и технике описываются гладкими кривыми. Для построения таких графиков необходимо выполнить интерполяцию – вычисление промежуточных значений y между точками экспериментальных данных.

Одномерная табличная интерполяция

Пусть вектора x и y содержат экспериментальные (табличные) данные, показанные на графике плюсами:

```
x = 0:10;
y =[0 0.8415 0.9093 0.1411 -0.7568 -0.9589 ...
-0.2794 0.6570 0.9894 0.4121 -0.5440];
plot(x, y, '+'); grid on; hold on
```

Табличные данные x изменяются с шагом 1. Для вычисления промежуточных значений, т.е. интерполяции данных, нужно задать интервал xi с более мелким шагом:

```
xi = 0:0.1:10;
```

Промежуточные значения yi вычисляются с помощью функции `interp1`, при этом могут применяться разные методы сглаживания:

```
yi1 = interp1(x, y, xi, 'linear'); % линейная интерп.
plot(xi, yi1, 'b')
yi2 = interp1(x, y, xi, 'cubic');  % кубическая инт.
plot(xi, yi2, 'g')
yi3 = interp1(x, y, xi, 'spline'); % инт. сплайнами
plot(xi, yi3, 'r'); hold off
```

Сплайны – это алгебраические многочлены третьей степени. Сравните графики, построенные разными методами сглаживания.

Задайте произвольные вектора x и y из 7 чисел каждый (значения x должны изменяться монотонно – только возрастать или убывать). Выполните одномерную табличную интерполяцию.

Двумерная табличная интерполяция

Экспериментальные данные, которые графически могут быть представлены как точки в трехмерном пространстве с координатами X, Y и Z, могут быть интерполированы с помощью функции `interp2`.

Вначале сформируем исходные данные – некоторую поверхность Z на двумерной сетке X, Y:

```
x = -2:0.5:2; % интервал и шаг изменения X
y = -2:0.5:2; % интервал и шаг изменения Y
[X, Y] = meshgrid(x, y); % исходная 2-мерная сетка XY
Z=X.*exp(-X.^2-Y.^2); % пусть это исходные значения Z
plot3(X, Y, Z, 'b.') % график для исходных данных
grid on, hold on
```

Задаем мелкую сетку XI, YI с шагом 0,1:

```
xi = -2:0.1:2; % интервал и мелкий шаг изменения XI
yi = -2:0.1:2; % интервал и мелкий шаг изменения YI
[XI, YI] = meshgrid(xi, yi); % мелкая сетка XI,YI
```

Затем вызываем функцию интерполяции, которая может использовать два метода сглаживания (линейная и кубическая интерполяция):

```
ZI1 = interp2(X, Y, Z, XI, YI, 'linear'); % линейная
mesh(XI, YI, ZI1); rotate3d
hidden off  % прозрачный график
```

Линейная интерполяция соединяет точки исходных данных плоскостями. Для получения гладкой поверхности используется кубическая интерполяция:

```
ZI2 = interp2(X, Y, Z, XI, YI, 'cubic'); % кубическая
mesh(XI, YI, ZI2)
hidden off, hold off
```

Лабораторная работа № 6
РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ УРАВНЕНИЙ

Широкий класс математических моделей, в особенности применяемых при решении экономических задач и планировании производства, основан на решении систем линейных уравнений:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1; \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2; \\ \ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m, \end{cases}$$

где a и b – постоянные коэффициенты, x_j – переменные.

В матричной форме система уравнений записывается следующим образом:

$$AX = B.$$

В случае $n = m$ матрица коэффициентов $A = (a_{ij})$ является квадратной, и система имеет единственное решение. Условие решения системы уравнений – определитель матрицы A должен быть отличен от нуля: $|A| \neq 0$. Определитель вычисляется с помощью функции MATLAB `det`:

```
det(A)
```

Вектор-столбец переменных $X = (x_1, x_2, ..., x_n)^T$ можно определить с помощью обратной матрицы: $X = A^{-1}B$. Обратная матрица A^{-1} вычисляется с помощью функции `inv`:

```
inv(A)
```

Матричное произведение исходной и обратной матриц дает единичную матрицу:

```
inv(A) * A,    A * inv(A)
```

Решение системы линейных уравнений можно найти следующим образом:

```
X = inv(A) * B
```

$B = (b_1, b_2, ..., b_m)^T$ – это вектор-столбец из коэффициентов правой части системы уравнений.

Найдите определители и обратные матрицы, а также произведения исходных и обратных матриц:

$$A = \begin{pmatrix} 6 & 8 \\ 7 & 8 \end{pmatrix}; \qquad A = \begin{pmatrix} 2 & 4 & 0 \\ 4 & 1 & 1 \\ 2 & 1 & 0 \end{pmatrix}.$$

Решите систему линейных уравнений с помощью обратной матрицы:

$$\begin{cases} 3x_1 - x_2 = 1; \\ 2x_1 + x_2 - 3x_3 = -5; \\ x_1 + 2x_2 + x_3 = 8. \end{cases}$$

Решение X может быть найдено с помощью `решателя` систем линейных уравнений – матричного деления:

```
X = A \ B      % X и B - это вектора-столбцы
X = B / A'     % X и B - это вектора-строки
```

Решите предыдущую систему линейных уравнений с помощью «/» и «\».

Если количество переменных превышает количество уравнений в системе $n > m$, то система имеет в общем случае бесконечное множество решений. Решатели «/» и «\» позволяют находить одно из базисных решений такой системы.

Решите систему линейных уравнений с помощью «/» и «\»:

$$\begin{cases} x_1 - 3x_2 - x_4 = 0; \\ 2x_2 + x_3 + 4x_4 = 4. \end{cases}$$

Все базисные решения этой системы можно найти, если принимать по две переменных (например, x_1 и x_2, или x_1 и x_3, и т.д.) за 0, составлять матрицы A размером 2^x2 для оставшихся переменных и затем решать систему уравнений.

Найдите несколько решений системы линейных уравнений:

$$\begin{cases} x_1 + 2x_2 + 2x_3 + 22x_4 - 4x_5 = 11; \\ x_1 + 2x_2 + x_3 + 16x_4 - 4x_5 = 9; \\ x_1 + x_2 + x_3 + 12x_4 - 2x_5 = 6. \end{cases}$$

Решение нелинейных уравнений

Решение нелинейного уравнения с одной переменной в виде $f(x)=0$ осуществляется в MATLAB с помощью функции `fzero`. Решение ищется приближенно численными методами в окрестности точки x_0. Например, найдем корни нелинейного уравнения $(x-2)^2 - 9 = 0$ в окрестности точек $x_0 = -2$ и $x_0 = 6$:

```
x = fzero('(x-2).^2-9', -2)
x = fzero('(x-2).^2-9', 6)
```

Выражение в апострофах – это текстовая строка, содержащая левую часть уравнения $f(x)=0$ с переменной `x`. Здесь также можно указывать имя функции MATLAB, стандартной или написанной пользователем в M-файле. Например, решение уравнения $\cos x = 0$ в окрестности точки $x_0 = 1{,}5$ находится так:

```
x = fzero('cos', 1.5)
```

Решите нелинейные уравнения:

$x^4 - 4x^3 + 12 = 0$; $2\sin(x-2) = -1$.

Интегрирование обыкновенных дифференциальных уравнений

Научные исследования динамических, т.е. развивающихся во времени, процессов с помощью детерминистских математических моделей основаны на решении дифференциальных уравнений. В частности, динамику механических систем, динамику рабочих процессов машин можно описать системой обыкновенных дифференциальных уравнений (ОДУ). Переменными величинами, зависящими от времени t, в этом случае могут быть координаты $x(t)$, скорости $v(t)$ элементов машин и т.п.

Для нахождения приближенного решения численными методами в MATLAB система нелинейных ОДУ должна быть сведена к системе дифференциальных уравнений первого порядка и представлена в форме задачи Коши:

$$\frac{dX}{dt} = F(X,t),$$

где $X = (x_1, x_2, ..., x_n)^T$ – вектор состояния, содержащий переменные x_j (координаты, скорости и т.п.);

$F = (f_1, f_2, ..., f_n)^T$ – вектор нелинейных функций f_j от переменных X и времени t, т.е. содержит правые части дифференциальных уравнений. В развернутой форме можно записать систему ОДУ следующим образом:

$$\begin{cases} \dfrac{dx_1}{dt} = f_1(x_1, x_2, ..., x_n, t); \\ \dfrac{dx_2}{dt} = f_2(x_1, x_2, ..., x_n, t); \\ \qquad ... \\ \dfrac{dx_n}{dt} = f_n(x_1, x_2, ..., x_n, t). \end{cases}$$

Для интегрирования ОДУ в MATLAB имеется ряд программных функций. Рассмотрим 2 из них: `ode23` и `ode45`. Они реализуют методы Рунге-Кутты 2 и 3, а также 4 и 5 порядков.

```
[t, X] = ode23('имя функции', [t0 tf], x0)    или
[t, X] = ode45('имя функции', [t0 tf], x0)
```

Аргументы решателей систем ОДУ:

`'имя функции'` – указывается имя написанной пользователем функции MATLAB, вычисляющей значения $F(X,t)$ и возвращающей вектор-столбец; эта функция должна храниться в одноименном М-файле, и путь к этому файлу должен быть добавлен к списку путей MATLAB (см. стр. 5);

`t0` и `tf` – начальное и конечное значение времени t (интервал интегрирования ОДУ);

`x0` – вектор-столбец начальных условий, т.е. значения переменных $x_1, x_2, ..., x_n$ в начальный момент времени `t0`.

В результате возвращается вектор текущих моментов времени `t` и двумерный массив `X`, где каждый столбец соответствует одной переменной, а строка – моменту времени `t`.

Рассмотрим дифференциальное уравнение 2-го порядка – уравнение Ван дер Поля: $\ddot{x} + (x^2 - 1)\dot{x} + x = 0$. Пусть x – это координата материальной точки, тогда $\dot{x}$ – это ее скорость, $\ddot{x}$ – ускорение. Это уравнение может

быть представлено в форме Коши. Для этого вводим обозначения переменных:

$$x_1 = x;\ x_2 = \dot{x}.$$

Задача в форме Коши:

$$\begin{cases} \dot{x}_1 = x_2; \\ \dot{x}_2 = x_2\left(1 - x_1^2\right) - x_1. \end{cases}$$

Правые части системы уравнений:

$$\begin{cases} f_1 = x_2; \\ f_2 = x_2\left(1 - x_1^2\right) - x_1. \end{cases}$$

Первый шаг процедуры интегрирования ОДУ – это создание М-файла для вычисления правых частей ОДУ. С помощью пункта главного меню `File→New→M-File` вызываем редактор-отладчик `Editor/Debugger`. В этом окне необходимо набрать текст программы, вычисляющей правые части ОДУ:

```
function f = fright(t, x) % Имя функции fright
f(1,1) = x(2); % правая часть первого ОДУ
f(2,1) = x(2).*(1-x(1).^2)-x(1); % правая часть 2го ОДУ
```

С помощью пункта меню редактора-отладчика `File→Save As` сохраним этот файл обязательно под тем же именем `fright.m` в рабочем каталоге, например, в папке `d:\Study\31SM`. Переключившись в окно управления MATLAB, нужно добавить путь к этому файлу, например, с помощью команды (см. стр. 5)

```
>> path(path, 'd:\Study\31SM')
```

Чтобы проинтегрировать систему на временном интервале $0 \le t \le 20$ с начальными условиями $x_1 = 0$ и $x_2 = 0{,}25$, следует вызвать `ode23`:

```
t0 = 0; tf = 20; % начальный и конечн. момент времени
x0 = [0; 0.25];  % начальные условия
[t, X] = ode23('fright', [t0 tf], x0)
plot(t, X); grid on; xlabel('t')
% синяя линия - координата x1, зеленая - скорость x2
```

Другой пример: промоделируем движение точки по заданному закону в трехмерном пространстве.

$$\begin{cases} \dot{x} = -y - z; \\ \dot{y} = x + ay; \\ \dot{z} = b + z(x - c), \end{cases}$$

где $a = 0{,}2$, $b = 0{,}2$, $c = 5{,}7$, а начальные условия – $x_0 = -0{,}7$, $y_0 = -0{,}7$, $z_0 = 1$.

Вводим обозначения переменных: $x_1 = x$, $x_2 = y$, $x_3 = z$. Создаем М-файл для вычисления правых частей ОДУ:

```
function f = fright1(t, x) % Имя функции fright1
f(1,1) = -x(2)-x(3);        % правая часть 1го ОДУ
f(2,1) = x(1) + 0.2 * x(2); % правая часть 2го ОДУ
f(3,1) = 0.2 + x(3).*(x(1)-5.7); % пр. часть 3го ОДУ
```

Сохраним этот файл под тем же именем `fright1.m` в рабочем каталоге.

Чтобы проинтегрировать систему на временном интервале $0 \le t \le 100$ вызываем `ode45`:

```
t0=0; tf=100; % начальный и конечный моменты времени
x0 = [-0.7; -0.7; 1];  % начальные условия
[t, X] = ode45('fright1', [t0 tf], x0) % интегриров.
comet3(X(:,1), X(:,2), X(:,3)); grid on; % график
xlabel('x'), ylabel('y'), zlabel('z')
```

Библиографический список

1. Потемкин В.Г. Система MATLAB. Справочное пособие. – М.: ДИАЛОГ-МИФИ, 1997. – 350 с.

2. Дьяконов В. П., Абраменкова И.В. MATLAB 5.0/5.3. Система символьной математики. – М.: Нолидж, 1999. – 640 с.

3. Дьяконов В. П. MATLAB 6: Учебный курс. – М., 2001.

4. Ануфриев И. Е. Самоучитель MatLab 5.3/6.x. – М., 2002.

5. Дьяконов В. П. MATLAB 6/6.1/6.5 + Simulink 4/5. Основы применения: Полное руководство пользователя. – М., 2002.

Учебное издание

ОСНОВЫ НАУЧНЫХ ИССЛЕДОВАНИЙ

Методические указания к курсу лабораторных работ
для специальностей 170900 «Подъемно-транспортные, строительные, дорожные машины и оборудование» и 230100 «Сервис и техническая эксплуатация транспортных и технологических машин и оборудования»

Составитель Виталий Александрович Мещеряков

Лицензия ИД № 00064 от 16.08.99
Подписано к печати 26.03.04
Формат 60×90 1/16. Бумага писчая.
Усл.п.л. 1,75, уч-изд.л. 1,75
Тираж 100 экз.
Заказ _____. Цена договорная

Издательство СибАДИ
644099, Омск, ул. П. Некрасова, 10

_____

Отпечатано в ПЦ издательства СибАДИ
644099, Омск, ул. П. Некрасова, 10