ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
КЫРГЫЗСКО-РОССИЙСКИЙ СЛАВЯНСКИЙ УНИВЕРСИТЕТ

ЕСТЕСТВЕННО-ТЕХНИЧЕСКИЙ ФАКУЛЬТЕТ

Кафедра «Приборостроение»

Г.С. Воронова, М.А. Духанин

ОСНОВЫ MATLAB

**Учебник для студентов
по направлению 12.03.01 – «Приборостроение»**

Бишкек 2019

УДК 004.421
В 75

Рецензенты:
М.М. Шамсутдинов, д-р техн. наук, профессор, КРСУ,
А.А. Самсалиев, канд. техн. наук,
доцент, КГТУ им. И. Раззакова

Рекомендовано к изданию Ученым советом ГОУВПО КРСУ

Воронова Г.С., Духанин М.А.
В 75 ОСНОВЫ MATLAB: учебник для студ. по направлению 12.03.01 – «Приборостроение». – Бишкек: Изд-во КРСУ, 2019. – 94 с.

Учебник содержит сведения по использованию системы MATLAB и применению для моделирования систем автоматического управления. Издание разделено на три части, описывающие базовые возможности MATLAB для анализа систем автоматического управления и использования инструментария SIMULINK.

Предназначен для студентов второго курса дневного отделения, изучающих дисциплину «Применение пакета прикладных программ MATLAB и КОМПАС».

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ 5
1.1. Назначение матричной системы MATLAB 5
1.2. Основные работы с MATLAB 6
1.3. Основные теоретические сведения 7
1.4. Вещественные числа и тип данных double 14
1.5. Комплексные числа и комплексные функции 16
1.6. Числовые массивы 17
1.7. Вычисления с масивами 21

2. ПОСТРОЕНИЕ ГРАФИКОВ ФУНКЦИИ 24
2.1. Векторы и графики 24
2.2. Специальная графика 38
Графики в логарифмическом масштабе 39
Графики в полулогарифмическом масштабе 40
ГИСТОГРАММЫ И ДИАГРАММЫ 41
Столбцовые диаграммы 41
Построение гистограмм 43
Трехмерная графика 44
Создание массивов данных для трехмерных графиков 45
Лестничные графики 48
Графики с зонами погрешности 49
График дискретных отсчетов функции 50
Графики в полярной системе координат 52
Угловые гистограммы 53
График проекции векторов на плоскость 54
Контурные графики 55
Построение графиков поверхностей 56
Сетчатые 3D-графики с окраской 59
Сетчатые 3D-графики с проекциями 61
Построение поверхности столбцами 62

Построение поверхности с окраской 63
Построение сферы 65
Построение цилиндра 66
Объемные круговые диаграммы 67
Окраска плоских многоугольников 67
Трехмерная графика с треугольными плоскостями 68
3. СЦЕНАРИИ И М-ФАЙЛЫ 70
4. SIMULINK 71
4.1. Система моделирования Simulink 71
4.2. Состав библиотеки Simulink 71
4.3. Создание модели 84
4.4. Выполнение расчета 86

УПРАЖНЕНИЯ 89

САМОСТОЯТЕЛЬНЫЕ ЗАДАНИЯ
ДЛЯ ЛАБОРАТОРНЫХ РАБОТ 90

КОНТРОЛЬНЫЕ ВОПРОСЫ 92

ЛИТЕРАТУРА 93

1. ВВЕДЕНИЕ

MATLAB – система многоцелевого назначения, которая вышла на рынок программных продуктов почти двадцать лет назад и с тех пор непрерывно совершенствовалась. Но первоначально ее основу составляли алгоритмы решения систем линейных уравнений и задач на собственные значения, откуда и произошло ее название «матричная лаборатория». Теперь она представляется наиболее эффективной при проведении прикидочных расчетов и при разработке новых алгоритмов. Сейчас уже существует несколько десятков специальных приложений к MATLAB'у, посвященных более узким проблемам. Это обработка сигналов и изображений, инженерное программирование в виде блок-схем, решение экономических задач и многое другое. Но любое из этих приложений можно изучать только после первоначального освоения MATLAB.

При выполнении первой лабораторной работы студенты осваивают стандартные программные структуры и команды MATLAB: числа, матрицы, функции, графическое представление функций, действия с массивами.

1.1. Назначение матричной системы MATLAB

MATLAB – одна из старейших, тщательно проработанных и проверенных временем систем автоматизации математических и научно-технических расчетов, построенная на расширенном представлении и применении матричных операций. Это нашло отражение в названии системы MATrix LABoratory (матричная лаборатория). Применение матриц как основных объектов системы способствует резкому уменьшению числа циклов, которые очень распространены при выполнении матричных вычислений на обычных языках программирования высокого уровня и облегчению реализации параллельных вычислений.

Одной из основных задач при создании системы MATLAB всегда было предоставление пользователям мощного языка программирования, ориентированного на технические и математические расчеты и способного превзойти возможности традиционных языков программирования.

1.2. Основы работы с MATLAB

Среда MATLAB включает интерпретатор команд на языке высокого уровня, графическую систему, пакеты расширений и реализована на языке С. Вся работа организуется через командное окно *(Command Window),*которое появляется при запуске программы matlab.exe. В процессе работы данные располагаются в памяти *(Workspace),* для изображения кривых, поверхностей и других графиков создаются графические окна.

В командном окне в режиме диалога проводятся вычисления. Пользователь вводит команды или запускает на выполнение файлы с текстами на языке MATLAB. Интерпретатор обрабатывает введенное и выдает результаты: числовые и строковые данные, предупреждения и сообщения об ошибках. *Строка ввода* помечена знаком >>.

В командном окне показываются вводимые с клавиатуры числа, переменные, а также результаты вычислений. Имена переменных должны начинаться с буквы. Знак = соответствует операции присваивания. Нажатие клавиши *Enter* заставляет систему вычислить выражение и показать результат. Наберите с клавиатуры в строке ввода:

» a=2+51-37.

Нажмите клавишу *Enter,* на экране в *зоне просмотра* появится результат вычисления:

a = 16.

Все значения переменных, вычисленные в течение текущего сеанса работы, сохраняются в специально зарезервированной области памяти компьютера, называемой *рабочим пространством системы* MATLAB *(Workspace).* Командой *clc* можно стереть содержимое командного окна, однако это не затронет содержимого рабочего пространства. Когда исчезает необходимость

в хранении ряда переменных в текущем сеансе работы, их можно стереть из памяти компьютера командой *clear* или *clear (имя1, имя2, ...*). Первая команда удаляет из памяти все переменные, а вторая – переменные с именами *имя1* и *имя2*. Командой *who* можно вывести список всех переменных, входящих в данный момент врабочее пространствосистемы. Для просмотра значения любой переменной из текущего рабочего пространства системы достаточно набрать ее имя и нажать клавишу *Enter*.

После окончания сеанса работы с системой MATLAB все ранее вычисленные переменные теряются. Чтобы сохранить в файле на диске компьютера содержимое рабочего пространства системы MATLAB, нужно выполнить команду меню *File I Save Workspace As ...* По умолчанию расширение имени файла *mat,* поэтому такие файлы принято называть MAT-*файлами.* Для загрузки в память компьютера ранее сохраненного на диске рабочего пространства нужно выполнить команду меню:

File I Load Workspace ...

1.3. Основные теоретические сведения

Исторически MATLAB разрабатывался как диалоговая среда для матричных вычислений (MATrix LABoratory). Со временем пакет был оснащен хорошей графической системой, дополнен средствами компьютерной алгебры от Maple и усилен библиотеками команд (или Toolboxes), предназначенными для эффективной работы со специальными классами задач.

В состав MATLAB входят интерпретатор команд, графическая оболочка, редактор-отладчик, библиотеки команд, компилятор, символьное ядро пакета Maple для проведения аналитических вычислений, математические библиотеки MATLAB на C/C++, генератор отчетов и богатый инструментарий (Toolboxes).

Интерфейс MATLAB вполне отвечает современным канонам (см. рисунок 1.1). Он многооконный и имеет ряд средств прямого доступа к различным компонентам системы. Следует обратить внимание на следующие кнопки панели инструментов:

New M-file – выводит пустое окно редактора m-файлов.

Open file – открывает окно для загрузки файлов Matlab.

Simulink – открывает окно браузера библиотек Simulink.

Help – открывает окно справки.

Эти функции дублируются в очень простом меню системы MATLAB.

В левой части окна системы появились окна со вкладками **Launch Pad/Workspace** доступа к компонентам системы и вкладками текущей директории **Current Directory** и истории сессии **History**. Они обеспечивают оперативный контроль за состоянием системы. Выводимые на экран окна интерфейса MATLAB могут быть включены или отключены из пункта меню View.

Вся работа организуется через командное окно (**Command Window**), которое появляется при запуске программы. В процессе работы данные располагаются в памяти (**Workspace**) в виде матриц.

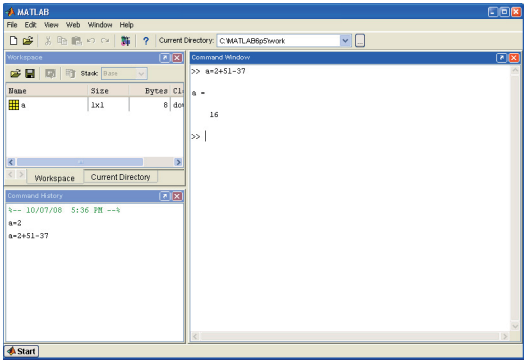


Рисунок 1.1 – Интерфейс программы Matlab

Все расчеты в MATLAB выполняются с двойной точностью, а для представления чисел на экране имеются разные форматы. Нужный формат может быть определен в меню (**File/Preferences**) либо при помощи команды **format**. Существуют следующие способы представления чисел (таблица1.1).

Переменные в MATLAB не нужно предварительно описывать, указывая их тип. Все данные хранятся в виде массивов: числовые переменные (внутренний тип numeric), текстовые строки (char), ячейки (cell) и структуры (struct). Двумерный массив – это

матрица, одномерный – вектор, а скаляр – матрица размера 1x1. Имя переменной должно начинаться с буквы, за ней могут идти буквы, цифры и символ подчеркивания.

Таблица 1.1 – Форматы вывода на экран

Формат	Представление
short	Число отображается с 4 цифрами после десятичной точки или в формате short e
short e	Число в экспоненциальной форме с мантиссой из 5 цифр и показателем из 3 цифр
rat	Представление в виде рационального дробного числа
long	Число с 16 десятичными цифрами
long e	Число в экспоненциальной форме с мантиссой из 16 цифр и показателем из 3 цифр
hex	Число в шестнадцатеричной форме

Допустимы имена любой длины, но MATLAB идентифицирует их по первым 31 символам и различает большие и малые буквы. В MATLAB имеется ряд констант (таблица 1.2) и специальные символы (таблица 1.3).

Таблица 1.2 – Зарезервированные имена констант

Имя	Описание
ans	Результат последней операции
i, j	Мнимая единица
pi	Число p
eps	Машинная точность
realmax	Максимальное вещественное число
realmin	Минимальное вещественное число
inf	Бесконечность
NaN	Нечисловая переменная
end	Наибольшее значение индекса размерности массива

Отметим, что имя NaN (Not-a-Number) зарезервировано для результата операций 0/0, 0*inf, inf-inf и т. п.

Таблица 1.3 – Специальные символы

Символ	Назначение
[]	Квадратные скобки используются при задании матриц и векторов
	Пробел служит для разделения элементов матриц
,	Запятая применяется для разделения элементов матриц и оператора в строке ввода
;	Точка с запятой отделяет строки матриц, а точка с запятой в конце оператора (команды) отменяет вывод результата на экран
:	Двоеточие используется для указания диапазона (интервала изменения величины) и в качестве знака групповой операции над элементами матриц
()	Круглые скобки применяются для задания порядка выполнения математических операций, а также для указания аргументов функций и индексов матриц
.	Точка отделяет дробную часть числа от целой его части, а также применяется в составе комбинированных знаков (.*, .^, ./, .\)
…	Три точки и более в конце строки отмечают продолжение выражения на следующей строчке
%	Знак процента означает начало комментария
'	Апостроф указывает на символьные строки, а для включения самого апострофа в символьную строку нужно поставить два апострофа подряд

В командном окне в режиме диалога проводятся вычисления. Пользователь вводит команды или запускает на выполнение файлы с текстами на языке MATLAB. Интерпретатор обрабатывает

введенное значение и выдает результаты: числовые и строковые данные, предупреждения и сообщения об ошибках. Строка ввода помечена знаком >>.

При работе с MATLAB в командном режиме действует простейший строчный редактор. Обратите особое внимание на применение клавиш **Up** и **Down** (стрелки курсора «Вверх» и «Вниз»). Они используются для подстановки после маркера строки ввода >> ранее введенных строк из специального стека, например, для их исправления, дублирования или дополнения. При этом указанные клавиши обеспечивают перелистывание ранее введенных строк снизу вверх или сверху вниз.

Имена переменных должны начинаться с буквы. Знак = соответствует операции присваивания. Нажатие клавиши *Enter* заставляет систему вычислить выражение и показать результат. Если запись оператора не заканчивается символом «;», то результат выводится в командное окно, в противном случае – не выводится. Если оператор не содержит знака присваивания «=», то значение результата присваивается системной переменной *ans* (см. рисунок 1.2).

Все значения переменных, вычисленные в течение текущего сеанса работы, сохраняются в специально зарезервированной области памяти компьютера, называемой рабочим пространством системы MATLAB (**Workspace**).

Для просмотра значения любой переменной из текущего рабочего пространства системы достаточно набрать ее имя и нажать клавишу Enter.

После окончания сеанса работы с системой MATLAB все ранее вычисленные переменные теряются. Чтобы сохранить в файле на диске компьютера содержимое рабочего пространства системы MATLAB, нужно выполнить команду меню ***File\Save Workspace As***… По умолчанию расширение имени файла mat, поэтому такие файлы принято называть MAT-файлами.

Система MATLAB работает как с действительными, так и с комплексными числами. Перед использованием операций с комплексными числами необходимо определить переменную

i = sqrt(–1) или j = sqrt(–1). В арифметических выражениях применяются следующие знаки операций:

+, – – сложение, вычитание;

*– умножение;

/ – деление слева направо;

\ – деление справа налево;

^ – возведение в степень.

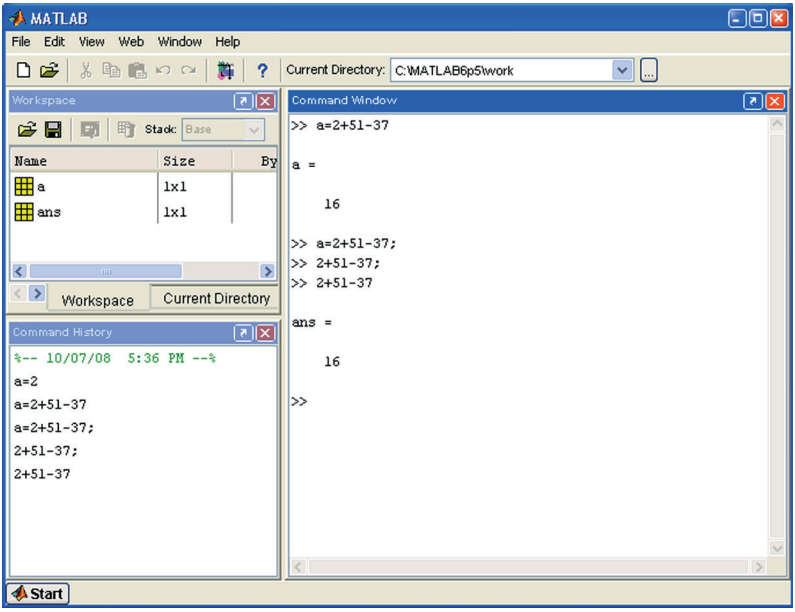


Рисунок 1.2 – Демонстрация
выполнения команды присваивания

Система MATLAB позволяет вычислять различные математические функции. Следующие элементарные алгебраические функции имеют в качестве аргумента одно или два действительных (x, y) или одно комплексное (z) число (таблица 1.4).

Таблица 1.4 – Элементарные алгебраические функции

Функция	Описание
abs(*z*), **abs**(*x*),	Вычисление модуля комплексного числа *z* или абсолютного значения действительного числа *x*.
angle(*z*)	Вычисление аргумента *z*.
sqrt(*z*), **sqrt**(*x*)	Вычисление квадратного корня чисел *z* и *x*
real(*z*)	Вычисление действительной части комплексного числа *z*.
imag(*z*)	Вычисление мнимой части комплексного числа *z*.
round(*x*)	Округление до целого.
fix(*x*)	Округление до ближайшего целого в сторону нуля.
rem(*x*, *y*)	Вычисление остатка от деления *x* на *y*.
exp(*z*)	Вычисление *e* в степени *x*.
log(*z*)	Вычисление натурального логарифма числа *x*.
log10(*z*)	Вычисление десятичного логарифма числа *x*.

Система MATLAB предоставляет возможности для вычисления следующих тригонометрических и обратных тригонометрических функций переменной *x* (таблица 1.5).

Таблица 1.5 – Тригонометрических функций

Функция	Описание
sin(*x*)	Вычисление синуса
cos(*x*)	Вычисление косинуса
tan(*x*)	Вычисление тангенса
asin(*x*)	Вычисление арксинуса
acos(*x*)	Вычисление арккосинуса
atan(*x*)	Вычисление арктангенса
atan2(*y*, *x*)	Вычисление арктангенса по координатам точки

1.4. Вещественные числа и тип данных double

Система MATLAB представляет на машинном уровне все действительные числа заданные мантиссой и показателем степени, например, 2.85093E+11, где буквой E обозначается основание степени равное 10. Этот основной тип данных носит название *double.* MATLAB по умолчанию использует формат *short* для вывода вещественных чисел, при котором показываются только четыре десятичных цифры после запятой.

Введите с клавиатуры пример:

» res=5.345*2.868/3.14-99.455+1.274.

Получите результат вычисления:

res = -93.2990.

Если требуется полное представление вещественного числа *res,* введите с клавиатуры команду:

» format long

и далее наберите имя переменной

» res

нажмите клавишу *Enter* и получите более подробную информацию:

res = -93.29900636942675.

Теперь все результаты вычислений будут показываться с такой высокой точностью в течение данного сеанса работы в среде системы MATLAB. Если требуется до прекращения текущего сеанса работы вернуться к старой точности визуального представления вещественных чисел в командном окне, нужно ввести и исполнить (нажав клавишу *Enter)* команду:

» format short.

Целые числа показываются системой в командном окне в виде целых чисел.

Над вещественными числами и переменными типа *double* производятся арифметические операции: сложения +, вычитания -, умножения *, деления / и возведения в степень ^. Приоритет в выполнении арифметических операций обычный. Операции одинакового приоритета выполняются в порядке слева направо, но круглые скобки могут изменить этот порядок.

Если нет необходимости видеть в командном окне результат вычисления некоторого выражения, то в конце введенного выражения следует поставить точку с запятой и только после этого нажать *Enter*.

В системе MATLAB присутствуют все основные элементарные функции для вычислений с вещественными числами. Любая функция характеризуется своим именем, списком входных аргументов (перечисляются через запятую и стоят внутри круглых скобок, следующих за именем функции) и вычисляемым (возвращаемым) значением. Список всех имеющихся в системе элементарных математических функций может быть получен по команде *help elfun*. В приложении 1 перечислены стандартные функции вещественного аргумента.

Вычислите выражение, включающее вычисление функции арксинус:

» 2*asin. (1).

Убедитесь, что получился следующий результат:

ans = 3.1416,

соответствующее числу «пи». В системе MATLAB для вычисления числа «пи» есть специальное обозначение: *pi.*(Список системных переменных MATLAB находится в приложении 2).

MATLAB имеет также логические функции, функции, связанные с целочисленной арифметикой (округления до ближайшего целого: *round,* усечение дробной части числа: *fix).* Есть еще функция *mod* – остаток от деления с учетом знака, *sign* – знак числа, *lcm* – наименьшее общее кратное, *perms* – вычисление числа перестановок и *nchoosek* – числа сочетаний и много других. Многие из функций имеют область определения, отличную от множества всех действительных чисел.

Помимо арифметических операций над операндами типа *double* выполняются еще операции отношения и логические операции. Операции отношения сравнивают между собой два операнда по величине. Эти операции записываются следующими знаками или комбинациями знаков (таблица 1.6):

Таблица 1.6 – Символьные обозначения операций отношения

<	<=	>	>=	~=	==
Меньше	Меньше или равно	Больше	Больше или равно	Не равно	Равно

В случае истинности операции отношения ее величина равна 1, а в случае ложности – 0. Операции отношения имеют более низкий приоритет, чем арифметические операции.

Наберите с клавиатуры выражение с операциями отношения и вычислите его:

» a=1; b=2; c=3;
» res=(a<p=»»><>

Вы получите следующий результат: res = 2.

Логические операции над вещественными числами обозначаются знаками, перечисленными в таблице 1.7.

Таблица 1.7 – Символьные обозначения логических операций

&	|	~
и	или	НЕ

Первые две из этих операций являются бинарными (двухоперандными), а последняя – унарной (однооперандной). Логические операции трактуют свои операнды как «истинные» (не равные нулю) или «ложные» (равные нулю). Если оба операнда операции «И» истинны (не равны нулю), то результат этой операции равен 1 («истина»); во всех остальных случаях операция «И» вырабатывает значение 0 («ложь»). Операция «ИЛИ» вырабатывает 0 («ложь») только в случае, когда являются ложными (равными нулю) оба операнда. Операция «НЕ» инвертирует «ложь» на «истину». Логические операции имеют самый низкий приоритет.

1.5. Комплексные числа и комплексные функции

Комплексные переменные, как и вещественные автоматически имеют тип *double* и не требуют никакого предварительного описания. Для записи мнимой единицы зарезервированы буквы i или j. В случае, когда коэффициентом перед мнимой единицей

является не число, а переменная, между ними следует обязательно использовать знак умножения. Итак, комплексные числа можно записывать следующим образом:

» 2+3i; -6.789+0.834e-2*i; 4-2j; x+y*i.

Почти все элементарные функции допускают вычисления с комплексными аргументами. Вычислите выражение:

» res=sin(2+3i)*atan(4i)/(1-6i).

Получится результат:

-1.8009 - 1.9190i.

Специально для работы с комплексными числами предназначены следующие функции: *abs* (абсолютное значение комплексного числа), *conj* (комплексно сопряженное число), *imag* (мнимая часть комплексного числа), *real* (действительная часть комплексного числа), *angle* (аргумент комплексного числа), *isreal* («истина», если число действительное). Функции комплексного переменного перечислены в приложении 1.

В отношении арифметических операций ничего нового для комплексных чисел (по сравнению с вещественными) сказать невозможно. То же самое отноится и к операциям отношения «равно» и «не равно». Остальные операции отношения вырабатывают результат исходя только из действительных частей этих операндов.

Введите выражение, получите результат и объясните его:

» c=2+3i; d=2i;

» c>d.

Логические операции трактуют операнды как ложные, если они равны нулю. Если же у комплексного операнда не равна нулю хотя бы одна его часть (вещественная или мнимая), то такой операнд трактуется как истинный.

1.6. Числовые массивы

Для создания одномерного массива можно использовать операцию конкатенации, которая обозначается с помощью квадратных скобок [ ]. Элементы массива помещаются между скобками и отделяются друг от друга пробелом или запятой:

» al=[1 2 3]; d=[1+2i,2+3i,3-7i].

Для доступа к индивидуальному элементу массива нужно применить операцию индексации, для чего после имени элемента указать в круглых скобках индекс элемента.

Можно изменять элементы уже сформированного массива путем применения операций индексации и присваивания. Например, введя:

» al(3)=789;

мы изменим третий элемент массива. Или, после введения:

» al(2)= (al(1)+al(3))/2;

второй элемент массива станет равным среднему арифметическому первого и третьего элементов. Запись несуществующего элемента вполне допустима -она означает добавление нового элемента к уже существующему массиву:

» al(4)=7.

Применяя после выполнения этой операции к массиву *a1* функцию *length,* находим, что количество элементов в массиве возросло до четырех:

» length(al)
ans = 4.

Тоже самое действие – «удлинение массива *a1»* – можно выполнить и с помощью операции конкатенации:

» al=[al 7].

Можно задать массив, прописывая все его элементы по отдельности:

» al(1)=67; al(2)=7.8; al(3)=0.017.

Однако этот способ создания не является эффективным.

Еще один способ создания одномерного массива основан на применении специальной функции, обозначаемой двоеточием (операция формирования диапазона числовых значений). Через двоеточие следует набрать первое число диапазона, шаг (приращение) и конечное число диапазона. Например:

» diap=3.7:0.3:8.974.

Если не нужно выводить на экран весь получившийся массив, то в конце набора (после конечного числа диапазона) следует набрать точку с запятой. Чтобы узнать, сколько элементов в массиве, следует вызвать функцию *length (имя массива).*

Для создания двумерного массива (матрицы) также можно использовать операцию конкатенацию. Элементы массива набираются один за другим согласно их расположению в строках, в качестве разделителя строк используется точка с запятой.

Введите с клавиатуры:

» a=[1 2; 3 4; 5 6].

Нажмите *ENTER,* получим:

a =
1 2
3 4
5 6

Полученную матрицу *a* размером 3x2 (первым указывается число строк, вторым – число столбцов) можно сформировать также вертикальной конкатенацией вектор-строк:

» a=[[1 2];[3 4];[5 6]];

или горизонтальной конкатенацией вектор-столбцов:

» a=[[1;3;5],[2;4;6]].

Структуру созданных массивов можно узнать с помощью команды *whos(имя массива),* размерность массива – функцией *ndims,* а размер массива -*size.*

Двумерные массивы можно задать также с помощью операции индексации, прописывая по отдельности его элементы. Номер строки и столбца, на пересечении которых находится задаваемый элемент массива, указываются через запятую в круглых скобках. Например:

» a(1,1)=1; a(1,2)=2; a(2,1)=3;
» a(2,2)=4; a(3,1)=5; a(3,2)=6.

Однако будет намного эффективнее, если до начала прописывания элементов массива, создать массив нужного размера функциями *ones (m,n)* или *zeros(m,n),* заполненный единицами или нулями (m – число строк, n – число столбцов). При вызове этих функций предварительно выделяется память под заданный размер массива, после этого постепенное прописывание элементов нужными значениями не требует перестройки структуры памяти, отведенной под массив. Использование этих функций возможно и при задании массивов других размерностей.

Если после формирования массива *X* потребуется, не изменяя элементов массива, изменить его размеры, можно воспользоваться функцией *reshape (X, M, N),* где *M и N* – новые размеры массива X.

Объяснить работу этой функции можно, только исходя из способа, каким система MATLAB хранит элементы массивов в памяти компьютера. Она хранит их в непрерывной области памяти упорядоченно по столбцам: сначала располагаются элементы первого столбца, вслед за ними расположены элементы второго столбца и т. д. Помимо собственно данных (элементов массива) в памяти компьютера хранится также управляющая информация: тип массива (например, *double),* размерность и размер массива, другая служебная информация. Этой информации достаточно для определения границ столбцов. Отсюда следует, что для переформирования матрицы функцией *reshape* достаточно изменить только служебную информацию и не трогать собственные данные.

Поменять местами строки матрицы с ее столбцам можно операцией транспортирования, которая обозначается знаком .’ (точка и апостроф). Например:

```
» A=[1 1 1; 2 2 2; 3 3 3];
» B=A’
B =
1 2 3
1 2 3
1 2 3
```

Операция ‘ (апостроф) выполняет транспонирование для вещественных матриц и транспонирование с одновременным комплексным сопряжением для комплексных матриц.

Объекты, с которыми работает MATLAB, **являются массивами!!!** Даже одно заданное число во внутреннем представлении MATLAB является массивом, состоящим из одного элемента. MATLAB позволяет делать вычисления с огромными массивами чисел также легко как и с одиночными числами, и это является одним из самых заметных и важных преимуществ системы MATLAB над другими программными пакетами, ориентированными на вычисления и программирование. Помимо памяти,

необходимой для хранения числовых элементов (по 8 байт на каждый в случае вещественных чисел и по 16 байт в случае комплексных чисел), MATLAB автоматически при создании массивов выделяет еще и память для управляющей информации.

1.7. Вычисления с массивами

В традиционных языках программирования вычисления с массивами осуществляются поэлементно в том смысле, что нужно запрограммировать каждую отдельную операцию над отдельным элементом массива. В М-языке системы MATLAB допускаются мощные групповые операции над всем массивом сразу. Именно групповые операции системы MATLAB позволяют чрезвычайно компактно задавать выражения, при вычислении которых реально выполняется гигантский объем работы.

Операции сложения и вычитания матриц (знакомые вам из линейной алгебры) обозначаются стандартными знаками + и -.

Задайте матрицы *A и B и* выполните операцию сложения матриц:

```
» A=[1 1 1; 2 2 2; 3 3 3]; B=[0 0 0; 7 7 7; 1 2 3];
» A+B
```

Если используются операнды разных размеров, выдается сообщение об ошибке, за исключением случая, когда один из операндов является скаляром. При выполнении операции *A + скаляр (A* – матрица) система расширит *скаляр* до массива размера A, который и складывается далее поэлементно *с A*.

```
» A+5
ans = 6 6 6
      7 7 7
      8 8 8
```

Для поэлементного перемножения и поэлементного деления массивов одинаковых размеров, а также поэлементного возведения в степень массивов, применяются операции, обозначаемые комбинациями двух символов: .* , ./, и .Ù. Использование комбинаций символов объясняется тем, что символами * и / обозначены специальные операции линейной алгебры над векторами и матрицами.

Кроме операции ./, называемой операцией правого поэлементного деления, есть еще операция левого поэлементного деления .\. Объясним разницу между этими операциями. Выражение A ./ B приводит к матрице с элементами A *(k, m) /B (k, m),* а выражение A.\ B приводит к матрице с элементами *B (k, m) /A (k, m).*

Знак * закреплен за перемножением матриц и векторов в *смысле линейной алгебры.*

Знак \ закреплен в системе MATLAB за решением довольно сложной задачи линейной алгебры – нахождением корней системы линейных уравнений. Например, если требуется решить систему линейных уравнений

Ay = b,

где *A* – заданная квадратная матрица размера *N* x *N, b* – заданный вектор-столбец длины *N,* то для нахождения неизвестного вектор-столбца *y* достаточно вычислить выражение A*b* (это равносильно операции: $A^{-1} \cdot B$).

Типичные задачи аналитической геометрии в пространстве, связанные с нахождением длин векторов и углов между ними, с вычислением скалярного и векторного произведений, легко решаются разнообразными средствами системы MATLAB. Например, для нахождения векторного произведения векторов предназначена специальная функция *cross,* например:

```
» u=[1 2 3]; v=[3 2 1];
» cross(u,v)
ans =
-4 8 -4
```

Скалярное произведение векторов можно вычислить с помощью функции общего назначения *sum,* вычисляющей сумму всех элементов векторов (для матриц эта функция вычисляет суммы для всех столбцов). Скалярное произведение, как известно,равно сумме произведений соответствующих координат (элементов) векторов. Таким образом, выражение:

```
» sum(u.*v).
```

.’ вычисляет скалярное произведение двух векторов u и v. Скалярное произведение можно также вычислить как: u*v

Длина вектора вычисляется с помощью скалярного произведения и функции извлечения квадратного корня, например:

» sqrt(sum(u.*u)).

Ранее рассмотренные для скаляров операции отношения и логические операции выполняются в случае массивов поэлементно. Оба операнда должны быть одинаковых размеров, при этом операция возвращает результат такого же размера. В случае, когда один из операндов скаляр, производится его предварительное расширение, смысл которого уже был пояснен на примере арифметических операций.

Среди функций, генерирующих матрицы с заданными свойствами, упомянем здесь функцию *eye,* производящую единичные квадратные матрицы, а также широко применяемую на практике функцию *rand,* генерирующую массив со случайными элементами, равномерно распределенными на интервале от 0 до 1. Например, выражение

» F=rand(3) порождает массив случайных чисел размером 3x3 с элементами, равномерно распределенными на интервале от 0 до 1.

Если вызвать эту функцию с двумя аргументами, например *R=rand(2,3),* то получится матрица *R* 3. При вызове функции´случайных элементов размером 2 *rand* с тремя и более скалярными аргументами производятся многомерные массивы случайных чисел.

Определитель квадратной матрицы вычисляется с помощью функции *det.*

Среди функций, производящих простейшие вычисления над массивами, помимо рассмотренной выше функции *sum,* упомянем еще функцию *prod,* которая во всем аналогична функции *sum,* только вычисляет она не сумму элементов, а их произведение. Функции *max* и *min* ищут соответственно максимальный и минимальный элементы массивов. Для векторов они возвращают единственное числовое значение, а для матриц они порождают набор экстремальных элементов, вычисленных для каждого столбца. Функция *sort* сортирует в возрастающем порядке

элементы одномерных массивов, а для матриц она производит такую сортировку для каждого столбца отдельно.

Наконец, рассмотрим уникальную возможность М-языка системы MATLAB производить групповые вычисления над массивами, используя обычные математические функции, которые в традиционных языках программирования работают только со скалярными аргументами. В результате с помощью крайне компактных записей, удобных для ввода с клавиатуры в интерактивном режиме работы с командным окном системы MATLAB, удается произвести большой объем вычислений. Например, всего два коротких выражения

```
» x=0:0.01:pi/2; y=sin(x);
```

вычисляют значения функции *sin* сразу в 158 точках, формируя два вектора *x* и *y* со 158 элементами каждый.

2. ПОСТРОЕНИЕ ГРАФИКОВ ФУНКЦИИ

2.1. Векторы и графики

Основные принципы построения графиков в MATLAB:

(i) выберите последовательность x-значений аргумента;

(ii) вычислите y = /(x), т. е. получите соответствующий вектор y-значений;

(iii) нарисуйте график y от x.

Прежде чем проделать это, необходимо узнать, как работает MATLAB с векторами.

Векторы

Примеры, в которых все результаты будут векторами.

```
» u=[2,2,3]
» u=[2 2 3]
» v=[1,0,-1]
>>w=u-2*v
» range=1:13
» odd=1:2:13
» dovn=20:-0.5:0
```

» even=odd+1
» xgrid=0:.05:1; x=xgrid*pi
» y=sin(x)

Первые две строки показывают, что элементы вектора могут разделяться пробелами или запятыми. Таким образом, [1+1 2 3] означает то же, что и [2, 2, 3], а [1 +1 2 3] – то же, что и [1,1,2,3].

Векторы могут быть любой длины. Они могут бытьстроками, как выше, или подобными

>>w’
ans =

O2
5

где апостроф обозначает транспонирование.

Графики

Элементарные графические функции системы MATLAB позволяют построить на экране и вывести на печатающее устройство следующие типы графиков: линейный, логарифмический, полулогарифмический, полярный.

Для каждого графика можно задать заголовок, нанести обозначение осей и масштабную сетку.

Изучим наиболее простые в использовании возможности (высокоуровневую графику).

Сформируйте два вектора *x* и *y:*

» x=0:0.01:2; y=sin(x).

Вызовите функцию:

» plot(x,y)

и вы получите на экране график функции (рисунок 2.1).

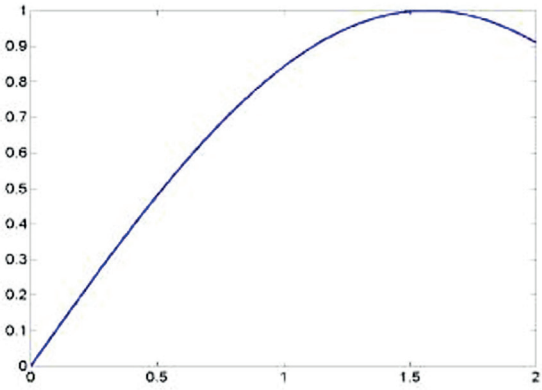


Рисунок 2.1 – График функции y=sin(x)

Пример 1

x=0:0.1:10;

plot (sin(x))

Построим графики сразу трех функций: sin(x), cos(x) и sin(x)/x. Прежде всего, отметим, что эти функции могут быть обозначены переменными, не имеющими явного указания аргумента в виде y (x) (рисунок 2.2).

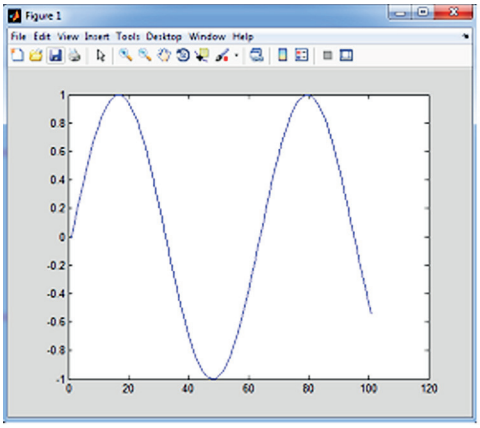


Рисунок 2.2 – Построение в одном окне графиков нескольких функций

Такая возможность обусловлена тем, что эти переменные являются векторами, как и переменная x. Теперь можно использовать одну из ряда форм команды

plot: plot(a1, f1, a2, f2, a3, f3,...),

где a1, a2, a3,… – векторы аргументов функций,

f1, f2, f3,... – векторы значений функций, графики которых строятся в одном окне.

MATLAB показывает графические объекты в специальных графических окнах, имеющих в заголовке слово *Figure.*

Не убирая с экрана дисплея первое графическое окно, введите с клавиатуры выражения

```
» z=cos(x);
» plot(x,z)
```

и получите новый график функции в том же самом графическом окне (при этом старые оси координат и график пропадают – этого также можно добиться командой *clf,* командой *cla* удаляют только график с приведением осей координат к их стандартным диапазонам от 0 до 1).

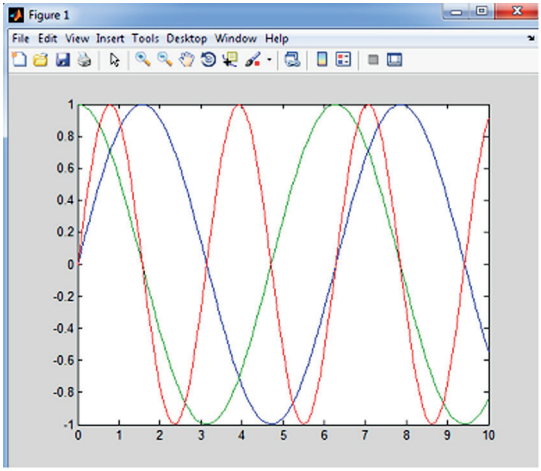


Рисунок 2.3 – Построение графика функции y = sin(x)/x

Пример 2
y1=sin(x); y2=cos(x); y3=sin(2*x);
plot (x,y1,x,y2,x,y3) (см. рисунок 2.3).

Если нужно второй график провести «поверх первого графика», то перед вторичным вызовом графической функции *plot* нужно выполнитькоманду *hold on,* которая предназначена для удержания текущего графического окна:

» x=0:0.01:2; y=sin(x);
» plot(x,y)
» z=cos(x);
» holdon
» plot(x,z).

Практически тоже самое получится (рисунок 2.4), если набрать:

» x=0:0.01:2; y=sin(x); z=cos(x);
» plot(x,y,x,z).

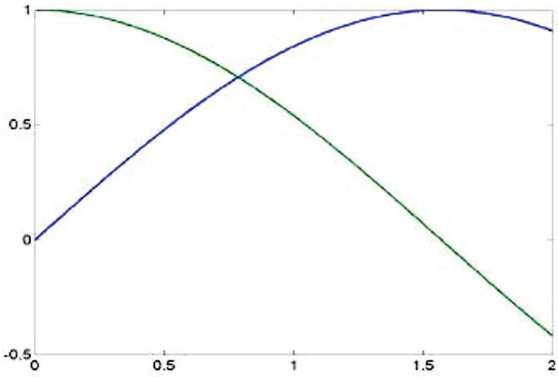


Рисунок 2.4 – Графики функций y=sin(x), z=cos(x), построенные в одном графическом окне

Если нужно одновременно визуализировать несколько графиков так, чтобы они не мешали друг другу, то это можно сделать двумя способами. Первым решением является построение их в разных графических окнах. Для этого перед вторичным вызо-

вом функции *plot* следует набрать команду *figure,* которая создает новое графическое окно и заставляет все последующие за ней функции построения графиков выводить их туда.

Вторым решением показа нескольких графиков без конфликта диапазонов осей координат является использование функции *subplot.* Эта функция позволяет разбить область вывода графической информации на несколько подобластей, в каждую из которых можно вывести графики различных функций.

Например, для ранее выполненных вычислений с функциями *sin и cos* постройте графики этих двух функций в первой подобласти, а график функции *exp(x)* – во второй подобласти одного и того же графического окна (рисунок 2.5):

```
» w=exp(x);
» subplot(1,2,1); plot(x,y,x,z)
» subplot(1,2,2); plot(x,w).
```

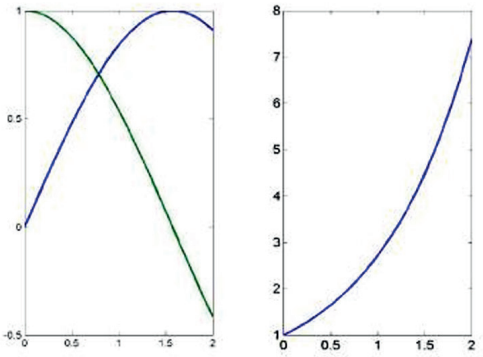


Рисунок 2.5 – Графики функций y=sin(x), z=cos(x) и w=exp(x), построенные в двух подобластях одного графического окна

Диапазоны изменения переменных на осях координат этих подобластей независимы друг от друга. Функция *subplot* принимает три числовых аргумента, первый из которых равен числу рядов подобластей, второй равен числу колонок подобластей, а третий аргумент – номеру подобласти (номер отсчитывается вдоль

рядов с переходом на новый ряд по исчерпании). Снять действие функции *subplot* можно командой:

» subplot(1,1,1)

Если для одиночного графика диапазоны изменения переменных вдоль одной или обеих осей координат слишком велик, то можно воспользоваться функциями построения графиков в *логарифмических масштабах.* Для этого предназначены функции *semilogx, semilogy* и *loglog.*

Построить график функции в полярных координатах (рисунок 2.6) можно с помощью графической функции *polar*:

» phi=0:0.01:2*pi; r=sin(3*phi);

» polar(phi,r).

Рассмотрим дополнительные возможности, связанные с управлением внешним видом графиков – задание цвета и стиля линий, а также размещение различных надписей в пределах графического окна.

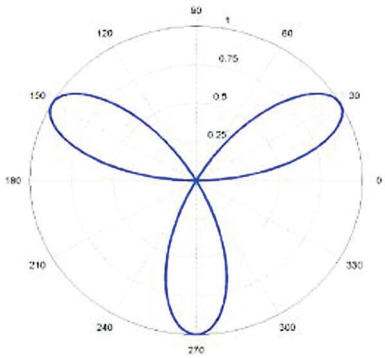


Рисунок 2.6 – График функции r = sin(3*phi)
в полярных координатах

Например, команды

» x=0:0.1:3; y=sin(x);

» plot(x,y,’r-’,x,y,’ko’)

позволяют придать графику вид красной сплошной линии (рисунок 2.6), на которой в дискретных вычисляемых точках проставляют черные окружности. Здесь функция *plot* дважды строит гра-

фик одной и той же функции, но в двух разных стилях. Первый из этих стилей отмечен как *'r-'*, что означает проведение линии красным цветом (буква r), а штрих означает проведение сплошной линии. Второй стиль, помечен как *'ko'*, означает проведение черным цветом (буква *k)* оружностей (буква *o)* на месте вычисляемых точек (рисунок 2.7).

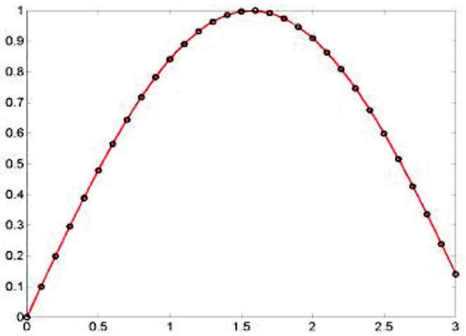


Рисунок 2.7 – Построение графика функции
y = sin(x) в двух разных стилях

В общем случае функция plot (x1, y1, s1, x2, y2, s2, ...) позволяет объединить в одном графическом окне несколько графиков функций y1(x1), y2(x2), ... проведя их со стилями s1, s2, ... и т. д.

Стили *s1, s2,...* задаются в виде набора трех *символьных маркеров,* заключенных в одиночные кавычки (апострофы). Один из этих маркеров задает тип линии (таблица 2.1). Другой маркер задает цвет (таблица 2.2). Последний маркер задает тип проставляемых «точек» (таблица 2.3). Можно указывать не все три маркера. Тогда используются маркеры, установленные по умолчанию. Порядок, в котором указывают маркеры, не является существенным, то есть 'r+-' и *'-+r'* приводит к одинаковому результату.

Таблица 2.1 – Маркеры, задающие тип линии

Маркер	-	--	:	-.
Тип линии	Непрерывная	Штриховая	Пунктирная	Штрих-пунктирная

Таблица 2.2 – Маркеры, задающие цвет линии

Маркер	Цвет линии	Маркер	Цвет линии
c	Голубой	g	Зеленый
m	Фиолетовый	b	Синий
y	Желтый	w	Белый
r	Красный	k	Черный

Таблица 2.3 – Маркеры, задающие тип точки

Маркер		+	*	o	**X**
Тип точки	Точка	Плюс	Звездочка	Кружок	Крестик

Если в строке стиля поставить маркер на тип точки, но не проставить маркер на тип линии, то тогда отображаются только вычисляемые точки, а непрерывной линией они не соединяются.

MATLAB имеет средства для построения графиков и таких функций, как sin(x)/x, которые имеют устранимые неопределенности. Это делается с помощью другой графической команды –

fplot: fplot('f(x)', [xmin, xmax]).

Она позволяет строить функцию, заданную в символьном виде, в интервале изменения аргумента x от xmin до xmax без фиксированного шага изменения x. Хотя в процессе вычислений предупреждение об ошибке (деление на 0) выводится, график строится правильно, при x=0 sinx/x=1.

Команда gridon (сетка) включает отображение сетки, которая строится пунктирными линиями.

Пример 4

fplot('sin(x)/x', [-15,15]); gridon

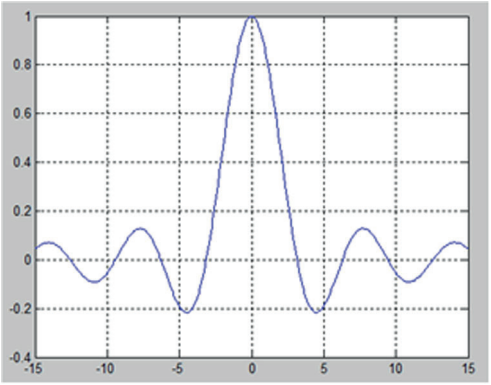


Рисунок 2.8 – Построение графиков отрезками прямых

Для отображения функции одной переменной y (x) используются графики в декартовой (прямоугольной) системе координат. При этом обычно строятся две оси: горизонтальная X и вертикальная Y, и задаются координаты x и y, определяющие узловые точки функции y(x) (рисунок 2.8).

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

plot (X, Y)

строит график функции y(x), координаты точек (x, y) которой берутся из векторов одинакового размера Y и X.

Если X или Y – матрица, то строится семейство графиков по данным, содержащимся в колонках матрицы.

Приведенный ниже пример иллюстрирует построение графиков двух функций – sin(x) и cos(x), значения, функции которых содержатся в матрице Y, а значения аргумента x хранятся в векторе X:

Пример 5

```
x=[0 1 2 3 4 5];
y1=sin(x); y2=cos(x); plot(x,y1,x,y2)
```

На рисунке 2.9 показан график функций из этого примера.

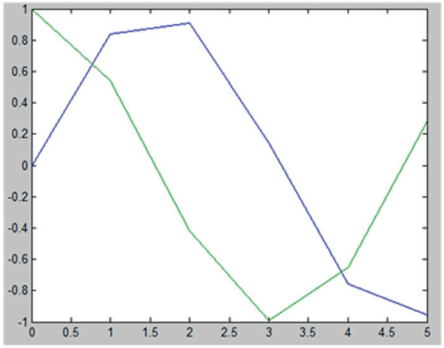


Рисунок 2.9 – Пример функций

В данном случае отчетливо видно, что график состоит из отрезков, и если вам нужно, чтобы отображаемая функция имела вид гладкой кривой, необходимо увеличить количество узловых точек. Расположение их может быть произвольным.

plot(Y) – строит график y(x), где значения y берутся из вектора Y, а x представляет собой индекс соответствующего элемента.

Если Y содержит комплексные элементы, то строится график

plot (real (Y), imag(Y)).

Во всех других случаях мнимая часть данных игнорируется.

Пример 6

x = -2*pi:0.02*pi:2*pi; y = sin(x)+i*cos(3*x);

plot(y) (см. рисунок 2.10)

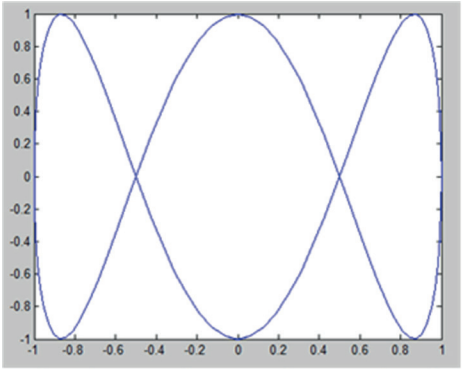


Рисунок 2.10 – Иллюстрация к примеру 6

plot (X,Y,S) – аналогична команде plot(X,Y), но тип линии графика можно задавать с помощью строковой константы S.

Таким образом, с помощью строковой константы S можно изменять цвет линии, представлять узловые точки различными отметками (точка, окружность, крест, треугольник с разной ориентацией вершины и т. д.) и менять тип линии графика.

Значениями константы S могут быть следующие символы:

Цвет линии	Тип точки	Тип линии
Желтый y	Точка .	Сплошная -
Фиолетовый m	Окружность 0	Двойной пунктир ;
Голубой c	Крест x	Штрих-пунктир -.
Красный r	Плюс +	Штриховая --
Зеленый g	Звездочка *	
Синий b		
Белый w		
Черный k		

Квадрат	s
Ромб	d
Треугольник (вверх)	^
Треугольник (вниз)	v
Треугольник (вверх)	^
Треугольник (влево)	<
Треугольник (вправо)	>
Пятиугольник	p
Шестиугольник	h

plot (X1,Y1, S1, X2, Y2, S2, X3,Y3,S3,...) – эта команда строит на одном графике ряд линий, представленных данными вида (X.,Y.,S.), где X. и Y. – векторы или матрицы, а S. – строки. С помощью такой конструкции возможно построение, например, графика функции линией, цвет которой отличается от цвета узловых точек. Так, если надо построить график функции линией синего цвета с красными точками, то вначале надо задать построение

графика с точками красного цвета (без линии), а затем графика только линии синего цвета (без точек).

При отсутствии указания на цвет линий и точек он выбирается автоматически из таблицы цветов (белый исключается). Если линий больше шести, то выбор цветов повторяется. Для монохромных систем линии выделяются стилем.

Пример 7

```
x=-2*pi:0.1*pi:2*pi;
y1=sin(x);y2=sin(x).^2;
y2=sin(x).^2; y3=sin(x).^3;
plot(x,y1,'-m',x,y2,'-.+r',x,y3,'--ok') (см. рисунок 2.11)
```

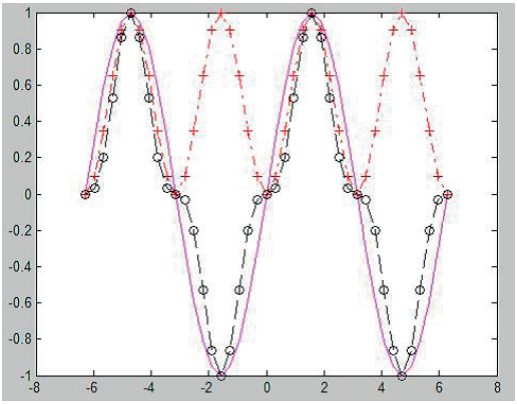


Рисунок 2.11

Здесь график функции y1 строится сплошной фиолетовой линией, график y2 строится штрихпунктирной линией с точками в виде знака «плюс» красного цвета, а график y3 строится штриховой линией с кружками черного цвета.

Надписи и пояснения к графикам

- TITLE – заголовки для двух- и трехмерных графиков;
- XLABEL, YLABEL, ZLABEL – обозначение осей;
- CLABEL – маркировка линий уровня;
- TEXT – добавление к текущему графику текста;

- GTEXT – размещает заданный текст на графике с использованием мыши;
- LEGEND – пояснение к графику;
- COLORBAR – шкала палитры.

Теперь перейдем к оформлению осей координат, к надписям на осях. Система MATLAB устанавливает пределы на горизонтальной оси равными тем значениям, что указаны пользователем для независимой переменной. Для зависимой переменной по вертикальной оси MATLAB самостоятельно вычисляет диапазон изменения значений функции. Если мы хотим отказаться от этой особенности масштабирования при построении графиков в системе MATLAB, то мы должны явным образом навязать свои пределы изменения переменных по осям координат. Это делается с помощью функции

axis ([xmin, xmax, ymin, ymax]).

Для проставления различных надписей на полученном рисунке применяют функции *xlabel, ylabel, title* и *text.* Функция *xlabel* создает подпись у горизонтальной оси, функция *ylabel* – тоже для вертикальной оси (причем эти надписи ориентированы вдоль осей координат). Если требуется разместить надпись в произвольном месте рисунка, применяем функцию *text.* Общий заголовок для графика создается функцией *title.* Кроме того, используя команду *grid on,* можно нанести измерительную сетку на всю область построения графика (рисунок 2.12).

```
» x=0:0.1:3; y=sin(x);
» plot(x,y,'r-',x,y,'ko')
» title('Function sin(x) graph');
» xlabel('xcoordinate'); ylabel('sin(x)');
» text(2.1, 0.9, '\leftarrowsin(x)'); grid on
```

Надпись функцией *text* помещается начиная от точки с координатами, указанными первыми двумя аргументами. По умолчанию координаты задаются в тех же единицах измерения, что и координаты, указанные на горизонтальной и вертикальной осях. Специальные *управляющие символы* вводятся внутри текста после символа \ (обратная косая черта).

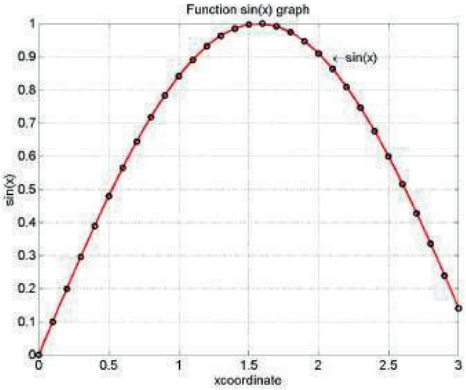


Рисунок 2.12 – График функции y=sin(x), построенный
в двух стилях, с надписями на координатных осях и на рисунке

2.2. Специальная графика

Раздел специальной графики включает графические команды и функции для построения столбцовых диаграмм, гистограмм, средств отображения векторов и комплексных элементов, вывода дискретных последовательностей данных, а также движущихся траекторий как для двумерной, так и для трехмерной графики. Этот раздел получил свое дальнейшее развитие в версии системы MATLAB 5.0, где специальные графические средства улучшены и существенно расширены.

- BAR – столбцовые диаграммы.
- ERRORBAR – график с указанием интервала погрешности.
- HIST – построение гистограммы.
- STEM – дискретные графики.
- STAIRS – ступенчатый график.
- ROSE – гистограмма в полярных координатах.
- COMPASS, FEATHER – графики векторов.
- QUIVER – поле градиентов функции.
- COMET – движение точки по траектории.
- FILL – закраска многоугольника.

- COMET3 – движение точки по пространственной траектории.

Двумерные графики

- PLOT – график в линейном масштабе.
- LOGLOG – график в логарифмическом масштабе.
- SEMILOGX, SEMILOGY – график в полулогарифмическом масштабе.
- POLAR – график в полярных координатах.

Графики в логарифмическом масштабе

Для построения графиков функций со значениями x и y, изменяющимися в широких пределах, нередко используются логарифмические масштабы. Рассмотрим команды, которые используются в таких случаях.

- loglogx(...) – синтаксис команды аналогичен ранее рассмотренному для функции plot(...). Логарифмический масштаб используется для координатных осей X и Y.

Построим график функции exp(x)/x в логарифмическом масштабе. Командой grid on строится координатная сетка. Неравномерное расположение линий координатной сетки указывает на логарифмический масштаб осей.

Пример 8

```
x=logspace(-1,5);
loglog(x,exp(x)./x)
grid on (см. рисунок 2.13)
```

Функция x = logspace (d1, d2) формирует вектор-строку, содержащую 50 равноотстоящих в логарифмическом масштабе точек, которые покрывают диапазон от 10d1 до 10d2.

Функция x = logspace (d1, d2, n) формирует вектор-строку, содержащую n равноотстоящих в логарифмическом масштабе точек, которые покрывают диапазон от 10d1 до 10d2.

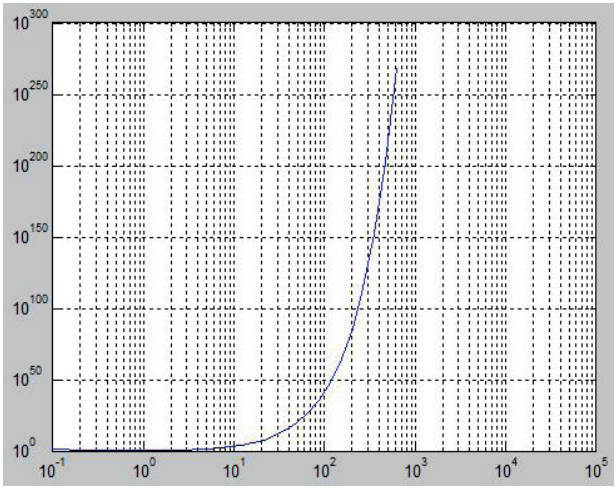


Рисунок 2.13

Графики в полулогарифмическом масштабе

В некоторых случаях предпочтителен полулогарифмический масштаб графиков, когда по одной оси задается логарифмический масштаб, а по другой – линейный.

Для построения графиков функций в полулогарифмическом масштабе используются следующие команды:

semilogx (...) – строит график функции в логарифмическом масштабе (основание 10) по оси X и линейном по оси Y;

semilogy (...) – строит график функции в логарифмическом масштабе по оси Y и линейном по оси X.

Запись параметров (...) выполняется по аналогии с функцией plot (...).

Пример 9

```
x=0:0.3:826;
semilogy(x,exp(x)) (рисунок 2.14).
```

Нетрудно заметить, что при таком масштабе график экспоненциальной функции выродился в прямую линию. Масштабной сетки теперь уже нет.

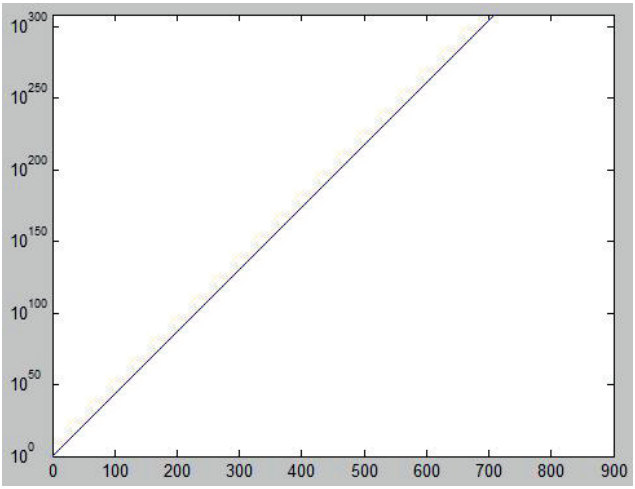


Рисунок 2.14

ГИСТОГРАММЫ И ДИАГРАММЫ

Столбцовые диаграммы

Столбцовые диаграммы широко используются в литературе, посвященной финансам и экономике, а также в математической литературе. Ниже представлены команды для построения таких диаграмм.

- bar (x, Y) – строит столбцовый график элементов вектора Y (или группы столбцов для матрицы Y) со спецификацией положения столбцов, заданной значениями элементов вектора x, которые должны идти в монотонно возрастающем порядке;
- bar (Y) – строит график значений элементов матрицы Y так же, как указано выше, но фактически для построения графика используется вектор x=1:m;
- bar (x,Y,WIDTH) или bar (Y,WIDTH) – команда аналогична ранее рассмотренным, но со спецификацией ширины столбцов (при WIDTH> 1 столбцы в одной и той же позиции перекрываются). По умолчанию задано WIDTH = 0.8.

Возможно применение этих команд и в следующем виде: bar (.... ‘Специфика- ция’) для задания спецификации графиков, например типа линий, цвета и т. д., по аналогии с командой plot. Спецификация ‘stacked’ задает рисование всех n столбцов

в позиции m друг на друге.

Пример 10

Пример построения столбцовой диаграммы матрицы размером 12x3

subplot (2,1,1),

bar(rand (12,3), ‘stacked’),

colormap (cool) (см. рисунок 2.15).

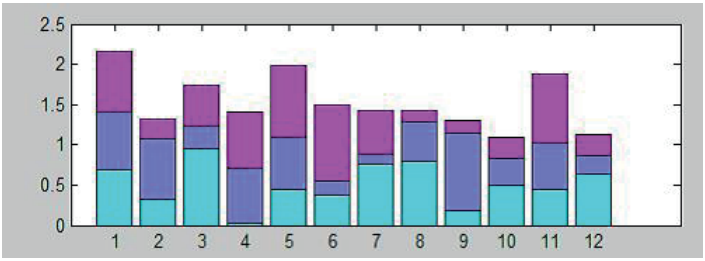


Рисунок 2.15

Помимо команды bar (...) существует аналогичная ей по синтаксису команда

barh (...), которая строит столбцовые диаграммы с горизонтальным расположением столбцов.

Пример 11

subplot (2,1,1), barh (rand (5,3),

‘stacked’),colormap(cool) (см. рисунок 2.16)

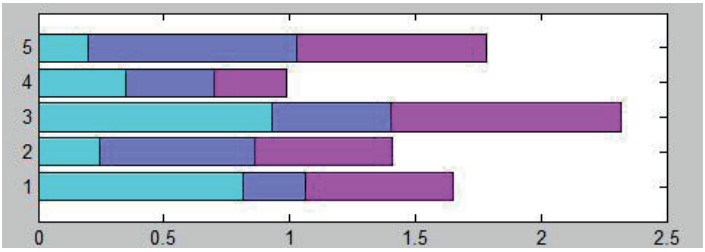


Рисунок 2.16

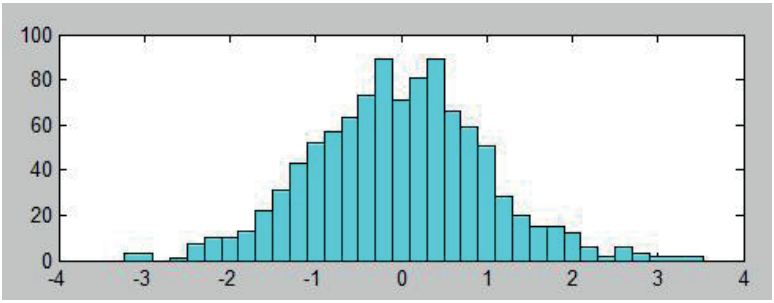
Построение гистограмм

Классическая гистограмма характеризует числа попаданий значений элементов вектора Y в M интервалов с представлением этих чисел в виде столбцовой диаграммы. Для получения данных для гистограммы служит функция hist, записываемая в следующем виде:

- N=hist(Y) – возвращает вектор чисел попаданий для 10 интервалов, выбираемых автоматически. Если Y – матрица, то выдается массив данных о числе попаданий для каждого из ее столбцов;
- N=hist(Y,M) – аналогична вышерассмотренной, но используется M интервалов (M- скаляр);
- ? N=hist(Y,X) – возвращает числа попаданий элементов вектора Y в интервалы, центры которых заданы элементами вектора X;
- [N,X]=HIST(...) – возвращает числа попаданий в интервалы и данные о центрах интервалов.

Команда hist(...) с синтаксисом, аналогичным приведенному выше, строит график гистограммы.

В следующем примере строится гистограмма для 1000 случайных чисел и выводится вектор с данными о числах их попаданий в интервалы, заданные вектором x:




```
h =
3 0 1 7 10 10 13 22 31 43 52 57 63 73 89 71 81 89 66 59 51 28 20 15 15 12
6 2 6 3 2
```

Рисунок 2.17

Пример 12

```
x = -3:0.2:3;
y = randn(1000,1);
hist(y,x)
h = hist(y,x) (рисунок 2.17)
```

Нетрудно заметить, что распределение случайных чисел близко к нормальному закону. Увеличив их количество, можно наблюдать еще большее соответствие этому закону.

Трехмерная 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. Уровень изложения данной книги не требует от читателя знания объектно-ориентированного программирования. Ее объем не позволяет в полной мере описать графическую подсистему, которая построена на таком подходе.

Возможности отображения трехмерных графических объектов в системе MATLAB весьма обширны. Мы сосредоточимся на изображении пространственных линий и на построении графиков функций двух вещественных переменных, которые представляют поверхности в пространстве.

Создание массивов данных для трехмерных графиков

Трехмерные поверхности обычно описываются функцией двух переменных

$$z(x,y).$$

Специфика построения трехмерных графиков требует не просто задания ряда значений x и y, то есть векторов x и y. Она требует определения для X и Y двумерных массивов – матриц.

Для создания таких массивов служит функция meshgrid.

В основном она используется совместно с функциями построения графиков трехмерных поверхностей.

Функция meshgrid записывается в следующих формах:

- [X,Y] = meshgrid(x) – аналогична [X,Y] = meshgrid(x, x);
- [X,Y,Z] = meshgrid(x, y, z) – возвращает трехмерные массивы, используемые для вычисления функций трех переменных и построения трехмерных графиков;
- [X,Y] = meshgrid(x,y) – преобразует область, заданную векторами x и y, в массивы X и Y, которые могут быть использованы для вычисления функции двух переменных и построения трехмерных графиков. Строки выходного массива X яв- ляются копиями вектора x; а столбцы Y – копиями вектора y.

Пример 13

```
[X,Y]=meshgrid(1:4, 13:17)

X =     1 2 3 4
        1 2 3 4
        1 2 3 4
        1 2 3 4
        1 2 3 4

Y =     13 13 13 13
        14 14 14 14
        15 15 15 15
        16 16 16 16
        17 17 17 17
```

Функция ndgrid является многомерным аналогом функции meshgrid:

- [X1, X2, X3,...] = ndgrid(x1,x2,x3....) – преобразует область, заданную векторами x1,.x,x3..., в массивы X1,X2,X3..., которые могут быть использованы для вычисления функций нескольких переменных и многомерной интерполяции, i-я размерность выходного массива Xi является копией вектора xi;
- [XI, X2....] = ndgrid(x) – аналогична [XI,X2....] = ndgrid(x,x,...).

Пример 14

```
[X1, X2] = ndgrid(-2:0.2:2,-2:0.2:2);
Z = X1.*exp(-X1.^2-X2.^2);
mesh(Z)
```

Каждая точка в пространстве характеризуется тремя координатами. Набор точек, принадлежащих некоторой линии в пространстве, нужно задать в виде трех векторов, первый из которых содержит первые координаты этих точек, второй вектор – вторые их координаты, ну а третий вектор – третьи координаты. После чего эти три вектора можно подать на вход функции *plot3,* которая и осуществит проектирование соответствующей трехмерной линии на плоскость и построит результирующее изображение (см. рисунок 2.18). Введите с клавиатуры:

```
» t = 0:pi/50:10*pi;
» x = sin(t);
» y = cos(t); plot3(x,y,t); grid on.
```

Убедитесь, что получилась винтовая линия (рисунок 2.19).

Эту же функцию *plot3* можно применить и для изображения поверхностей в пространстве, если, конечно, провести не одну линию, а много. Наберите с клавиатуры:

```
» u = -2:0.1:2; v = -1:0.1:1;
» [X,Y] = meshgrid(u,v);
» z = exp(-X.Ù2-Y.Ù2);
» plot3(X,Y,z)
```

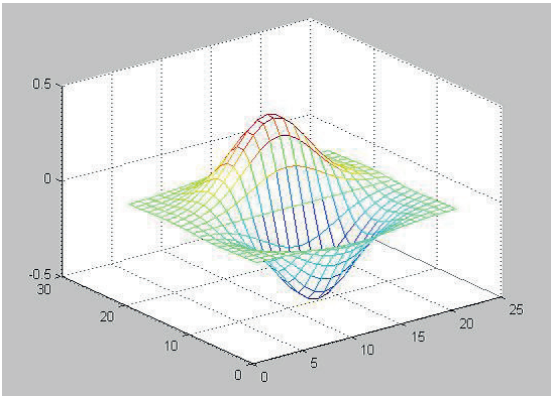


Рисунок 2.18

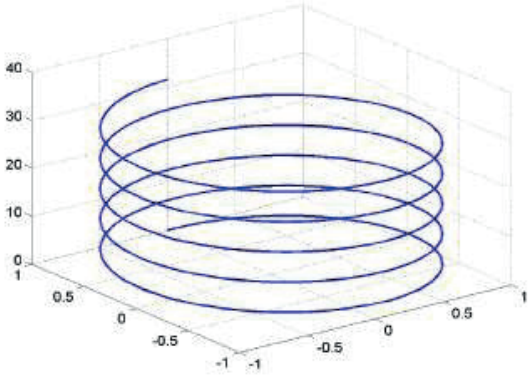


Рисунок 2.19 – График винтовой линии,
построенный с помощью функции *plot3*

Получите трехмерное изображение графика функции (рисунок 2.20).

Функция *plot3* строит график в виде набора линий в пространстве, каждая из которых является сечением трехмерной поверхности плоскостями, параллельными плоскости *yOz*. Помимо этой простейшей функции система MATLAB располагает еще рядом функций, позволяющих добиваться большей реалистичности в изображении трехмерных графиков.

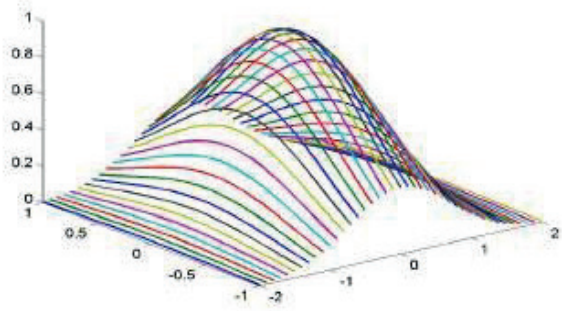


Рисунок 2.20 – График поверхности в пространстве, построенный с помощью функции *plot3*

Лестничные графики

Лестничные графики визуально представляют собой ступеньки с огибающей, представленной функцией y(x). Такие графики используются, например, для отображения процессов квантования функции y(x), представленной рядом своих отсчетов. При этом в промежутках между отсчетами значения функции считаются постоянными и равными величине последнего отсчета.

Для построения лестничных графиков в системе MATLAB используются команды группы stairs:

- stairs(Y) – строит лестничный график по данным вектора Y;
- stairs(X.Y) – строит лестничный график по данным вектора Y с координатами x переходов от ступеньки к ступеньке, заданными значениями элементов вектора X;
- stairs(...,S) – аналогична по действию вышеописанным командам, но строит график линиями, стиль которых задается строками S.

Следующий пример иллюстрирует построение лестничного графика:

Пример 15

x=0:0.25:10; stairs(x,x.^2) (рисунок 2.21).

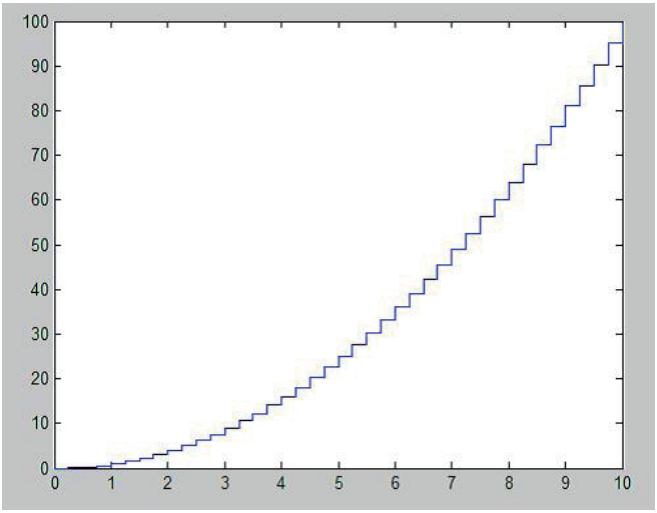


Рисунок 2.21

Обратите внимание на то, что отсчеты берутся через равные промежутки по горизонтальной оси. Если, к примеру, отображается функция времени, то stairs имеет вид квантованной по времени функции.

Функция H=stairs(X,Y) возвращает вектор дескрипторов графических объектов.

Функция [XX,YY] = stairs(X.Y) сама по себе график не строит, а возвращает векторы XX и YY, которые позволяют построить график с помощью команды plot(XX.YY).

Графики с зонами погрешности

Если данные для построения функции определены с заметной погрешностью, то используют графики функций типа errorbar с оценкой погрешности каждой точки путем ее представления в виде буквы I, высота которой соответствует заданной погрешности представления точки. Команда errorbarиспользуется в следующем виде:

- errorbar(X,Y,L.U) – строит график значений элементов вектора Y в зависимости от данных, содержащихся

в векторе X, с указанием нижней и верхней границ значений, заданных в векторах L и U;

- errorbar(X,Y,E) Herrorbar(Y.E) – строит графики функции Y(X) с указанием этих границ в виде [Y-EY+E], где E – погрешность;
- errorbar(..., 'LineSpec') – аналогична описанным выше командам, но позволяет строить линии со спецификацией 'LineSpec', аналогичной спецификации, при- мененной в команде plot.

Пример 16

```
x = 2:0.1:2; y=erf(x);
e = rand(size(x))/10;
errorbar(x,y,e) (см. рисунок 2.22)
```

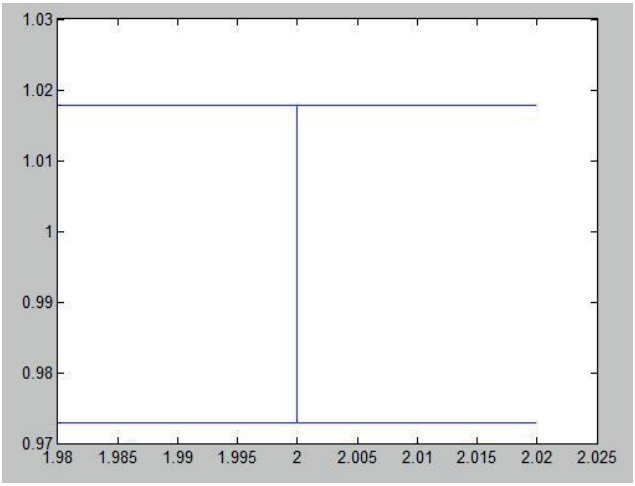


Рисунок 2.22

График дискретных отсчетов функции

Еще один вид графика функции y(x) – ее представление дискретными отсчетами. Этот вид графика применяется, например, при описании квантования сигналов. Каждый отсчет представляется вертикальной чертой, увенчанной кружком, причем высота черты соответствует y-координате точки.

Для построения графика подобного вида используются команды stem(...):

stem(X.Y) – строит график отсчетов с ординатами в векторе Y и абсциссами

в векторе X;

stem(.... ‘LINESPEC’) – дает построения, аналогичные ранее приведенным командам, но со спецификацией линий ‘LINESPEC’, подобной спецификации, приведенной для функцииplot;

stem(Y) – строит график функции с ординатами в векторе Y в виде отсчетов;

stem(.... ‘filled’) – строит график функции с закрашенными маркерами.

Следующий пример иллюстрирует применение команды stem:

Пример 17

```
x = 0:0.1:4;
y = sin(x.^2).*exp(-x);
stem(x,y) (рисунок 2.23).
```

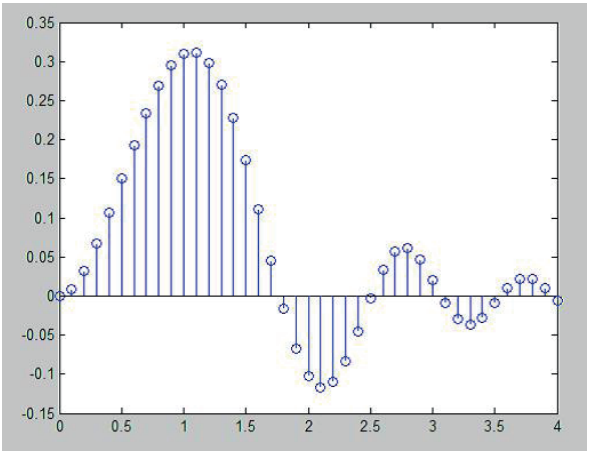


Рисунок 2.23

Графики в полярной системе координат

В полярной системе координат любая точка представляется как конец радиусвектора, исходящего из начала системы координат, имеющего длину RHO и угол THETA.

Угол THETA обычно меняется от 0 до 2*pi. Для построения графиков функций в полярной системе координат используются команды типа polar(...):

polar(THETA, RHO) – строит график в полярной системе координат, представляющий собой положение конца радиус-вектора с длиной RHO и углом THETA;

polar(THETA,RHO.S) – аналогична предыдущей команде, но позволяет задавать стиль построения с помощью строковой константы S по аналогии с командой plot.

Пример 18

t = 0:pi/50:2*pi; polar(t,sin(5*t)) (см. рисунок 2.24)

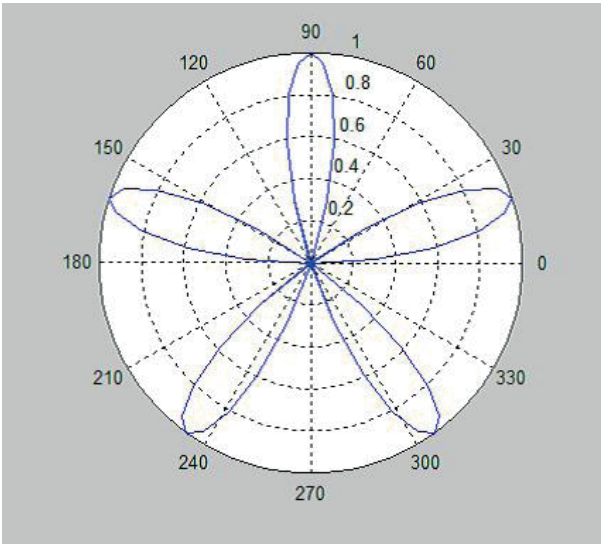


Рисунок 2.24

Угловые гистограммы

Угловые гистограммы находят применение в индикаторах радиолокационных станций, для отображения «роз» ветров и при построении других специальных графиков. Для этого используется ряд команд типа rose(...):

rose(THETA) – строит угловую гистограмму для 20 интервалов по данным

вектора THETA;

rose(THETA, N) – строит угловую гистограмму для N интервалов в пределах угла от 0 до 2*pi по данным вектора THETA;

rose(THETA. X) – строит угловую гистограмму по данным вектора THETA со спецификацией интервалов, указанной в векторе X.

Функция H=rose(...) строит график и возвращает вектор дескрипторов графических объектов, а функция [T,R]=rose(...) сама по себе график не строит, но возвращает векторы T и R, которые нужны команде polar (T. R) для построения подобной гистограммы

Пример 19

rose(1:100.12) (см. рисунок 2.25)

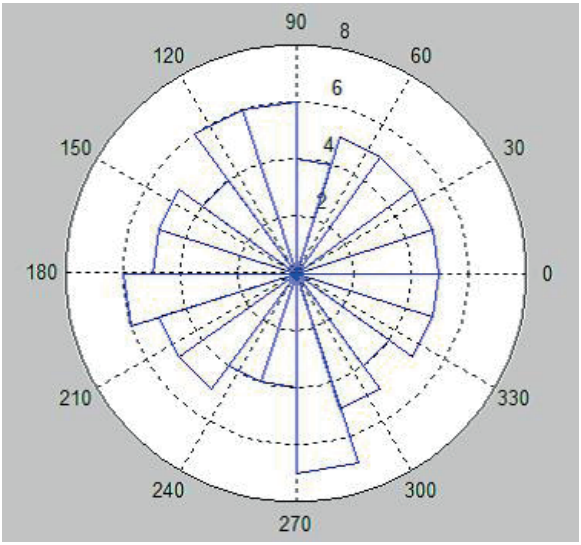


Рисунок 2.25

График проекций векторов на плоскость

Иногда полезно отображать комплексные величины вида z = x + yi в виде проекции радиус-вектора на плоскость. Для этого используется семейство графических команд класса feather:

feather(U.V) – строит график проекции векторов, заданных компонентами

U и V, на плоскость;

feather (Z) – для вектора Z с комплексными элементами дает построения, аналогичные feather(REAL(Z),IMAG(Z));

feather(..., S) – дает построения, описанные выше, но со спецификацией линий, заданной строковой константой S по аналогии с командой plot.

Пример 20

```
x=0:0.1*pi:3*pi;
y=0.05+i;
z=exp(x*y);
feather(z) (см. рисунок 2.26)
```

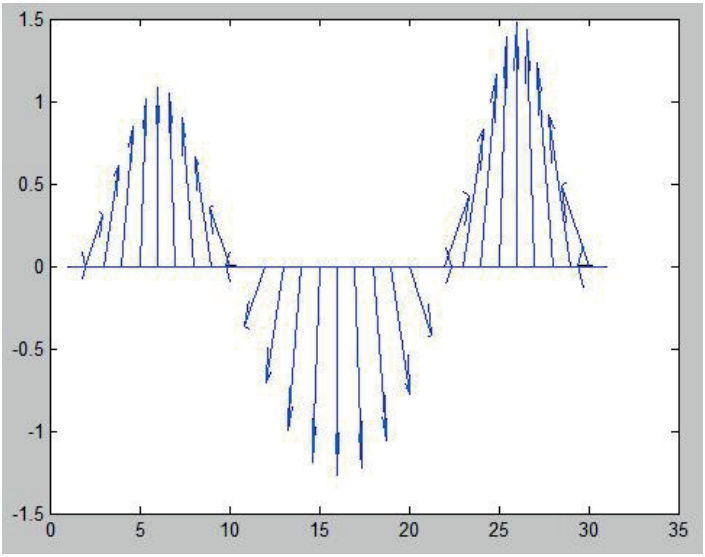


Рисунок 2.26

Контурные графики

Контурные графики служат для представления на плоскости функции двух переменных вида z(x, y) с помощью линий равного уровня. Они получаются, если трехмерная поверхность пересекается рядом плоскостей, расположенных параллельно друг другу. При этом контурный график предтавляет собой совокупность спроецированных на плоскость (x, y) линий пересечения поверхности r(x, y) плоскостями.

Для построения контурных графиков используются команды contour:

contour(Z) – строит контурный график по данным матрицы Z с автоматическим заданием диапазонов изменения x и y;

contour(X.Y.Z) – строит контурный график по данным матрицы Z с указанием спецификаций для X и Y;O contour(Z.N) и contour(X.Y.Z.N) – дает построения, аналогичные ранее описанным командам, с заданием N линий равного уровня (по умолчанию N=10);

contour(Z, V) и contour(X,Y,Z,V) – строят линии равного уровня для высот, указанных значениями элементов вектора V;

contour(Z,[vv]) или contour(X.Y,Z.[vv]) – вычисляет одиночный контур для уровня v;

[C.H] = contour (...) – возвращает дескрипторы – матрицу C и вектор-столбец H. Они могут использоваться как входные параметры для команды clabel;

contourC... ‘LINESPEC’) – позволяет использовать перечисленные выше команды с указанием спецификации линий, которыми идет построение.

Пример построения контурного графика поверхности, заданной функций peaks:

Пример 21

z = peaks(27); contour(z,15) (см. рисунок 2.27)

Построенный в этом примере график показан ниже (рисунок 2.27). Заметим, что объект – функция peaks – задан в системе в готовом виде.

Графики этого типа часто используются в топографии для представления на листе бумаги (как говорят математики – на плоскости) объемного рельефа местности. Для оценки высот контурных линий используется их функциональная окраска.

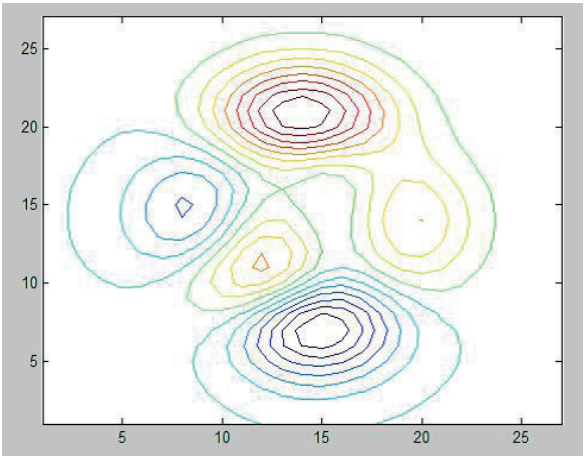


Рисунок 2.27

Построение графиков поверхностей

Команда plot3(...) является аналогом команды plot (...), но относится к функции двух переменных z(x, y). Она строит аксонометрическое изображение трехмерных поверхностей и представлена следующими формами:

plot3(x, y, z) – строит массив точек, представленных векторами x, y и z соединяя их отрезками прямых. Эта команда имеет ограниченное применение;

plot3(X, Y, Z), где X, Y и Z – три матрицы одинакового размера, строит точки с координатами X(i,:), Y(i,:) и Z(i,:) и соединяет их отрезками прямых;

plot3 (X, Y, Z, S) – обеспечивает построения, аналогичные рассмотренным ранее, но со спецификацией стиля линий и точек, соответствующей спецификации команды plot.

Пример 22

```
[X,Y]=meshgrid(-3:0.15:3);
Z=X.^2+Y.^2;
plot3(X,Y,Z,'o') (см. рисунок 2.28)
```

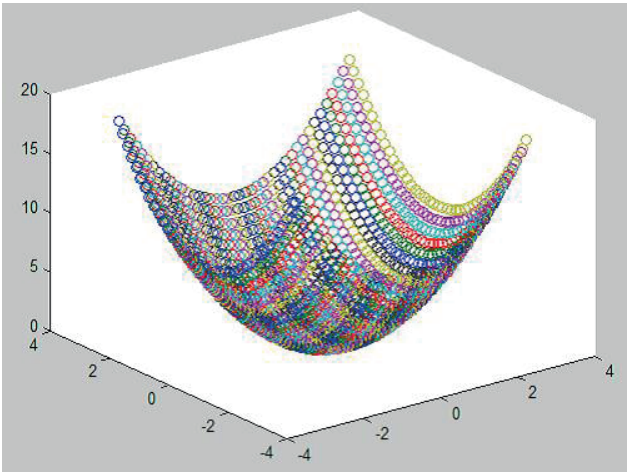


Рисунок 2.28

Команда plot3(...) является аналогом команды plot (...), но относится к функции двух переменных z(x, y).

Она строит аксонометрическое изображение трехмерных поверхностей и представлена следующими формами:

plot3(x.y,z) – строит массив точек, представленных векторами x, y и z, соединяя их отрезками прямых. Эта команда имеет ограниченное применение;

plot3(X,Y,Z), где X, Y и Z – три матрицы одинакового размера, строит точки с координатами X(i.:), Y(i,:) и Z(i,:) и соединяет их отрезками прямых.

Ниже дан пример построения трехмерной поверхности, описываемой функцией z(x.y)=x^2+y^2.

Пример 23

```
[x,y] = meshgrid([-3:0.15:3]);
z = x.^2+y.^2;
plot3(x,y,z) (см. рисунок 2.29)
```

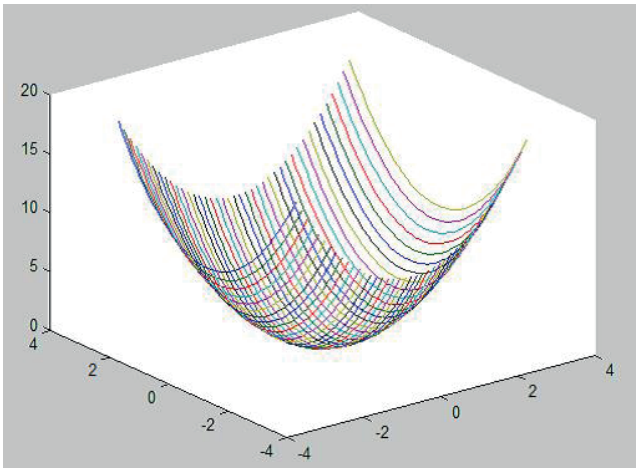


Рисунок 2.29

Пример 23 (с кружками)

```
[x,y]=meshgrid([-3:0.15:3]);
z=x.^2+y.^2;
plot3(x,y,z)
plot3(x,y,z, ‘o’) (см. рисуонк 2.30)
```

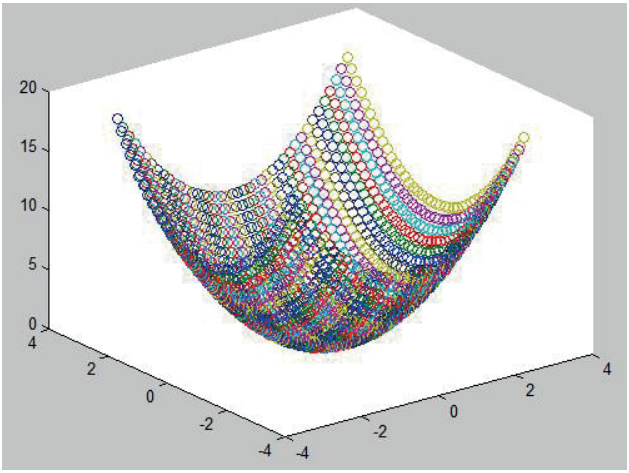


Рисунок 2.30

plot3(x1 ,y1,z1, s1,x2,y2,z2, s2, x3,y3,z3,s3,...) – строит на одном рисунке графики нескольких функций z1(x1 ,y1), z2(x2,y2) и т. д. со спецификацией линий и маркеров каждой из них.

Пример 23 (сетка)

```
[x,y]=meshgrid([-3:0.15:3]);
z=x.^2+y.^2;
plot3(x,y,z)
plot3(x,y,z, ‘o’)
plot3(x,y,z, ‘-k’, y, x, z, ‘k’) (см. рисунок 2.31)
```

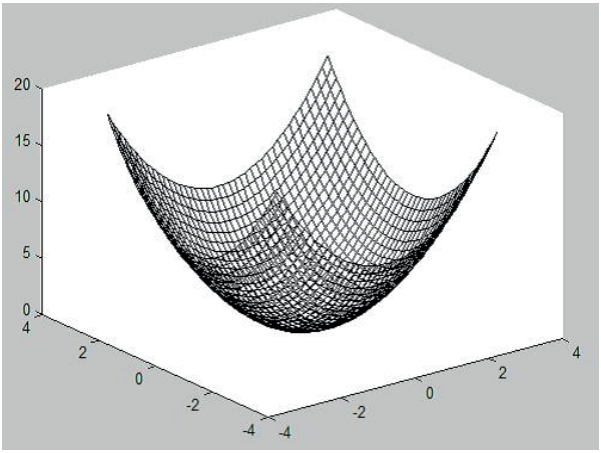


Рисунок 2.31

Сетчатые 3D-графики с окраской

Наиболее представительными и наглядными являются сетчатые графики поверхностей с заданной или функциональной окраской.

В названии их команд присутствует слово mesh.

Имеются три группы таких команд. Ниже приведены данные о наиболее полных формах этих команд. Наличие более простых форм можно уточнить, используя команду help Имя, где Имя – имя соответствующей команды.

mesh(X, Y, Z, C) – выводит в графическое окно сетчатую поверхность Z(X, Y)

с цветами узлов поверхности, заданными массивом C; mesh(X.Y.Z) – аналог предшествующей команды при C=Z. В данном случае используется функциональная окраска, при которой цвет задается высотой поверхности.

Возможны также формы команды mesh(x,y.Z), mesh(x.y.Z,C), mesh(Z) и mesh(Z.C).

Функция mesh возвращает дескриптор для объекта класса surface. Ниже приводится пример применения команды mesh.

Пример 24

```
[x,y]=meshgrid([-3:0.15:3]);
z=x.^2+y.^2;
mesh (x,y,z) (рисунок 2.32)
```

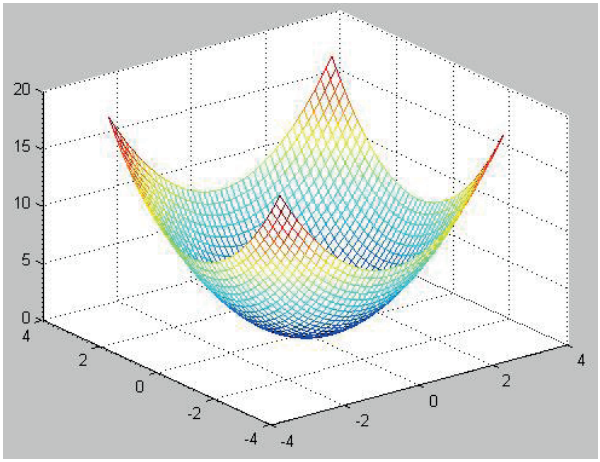


Рисунок 2.32

MATLAB имеет несколько графических функций, возвращающих матричный образ поверхностей.

Например, функция peaks(N) возвращает матричный образ поверхности с рядом пиков и впадин. Такие функции удобно использовать для проверки работы графических команд трехмерной графики.

Для упомянутой функции peaks можно привести такой пример:

Пример 25

```
z=peaks(25);
mesh(z); (рисунок 2.33)
```

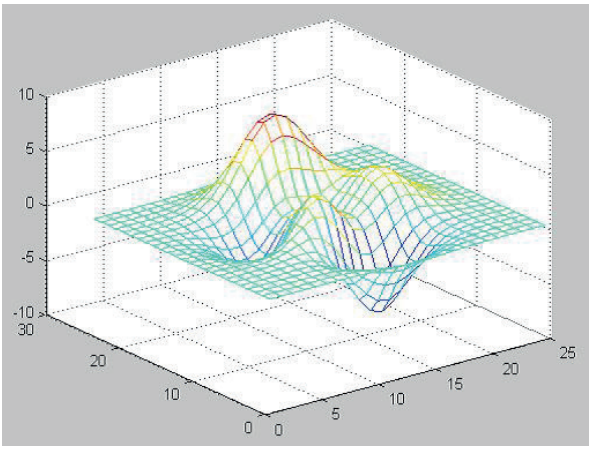


Рисунок 2.33

Сетчатые 3D-графики с проекциями

Иногда график поверхности полезно объединить с контурным графиком ее проекции на плоскость, расположенным под поверхностью.

Для этого используется команда meshc:

meshc(...) – аналогична mesh(...), но помимо графика поверхности дает изображение ее проекции в виде линий равного уровня (графика типа contour).

Пример 26

```
[x,y] = meshgrid([-3:0.15:3]);
z = x.^2+y.^2;
meshc(x,y,z) (см. рисунок 2.34)
```

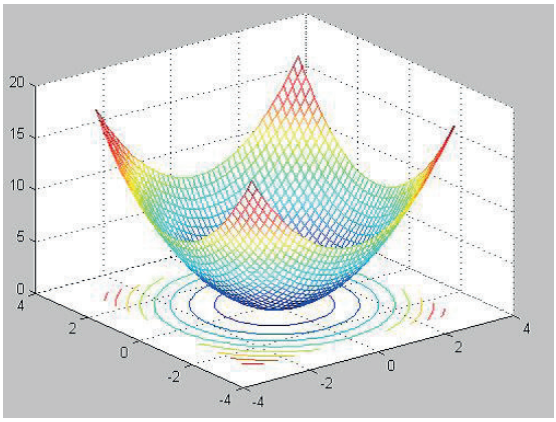


Рисунок 2.34

Построение поверхности столбцами

Еще один тип представления поверхности, когда она строится из многочисленных столбцов, дают команды класса meshz:

meshz(...) – аналогична mesh(...), но строит поверхность как бы в виде столбиков. Следующий пример иллюстрирует применение команды mesh:

Пример 27

```
[x,y] = meshgrid([-3:0.15:3]);
z = x.^2+y.^2;
meshz (x, y, z) (см. рисунок 2.35)
```

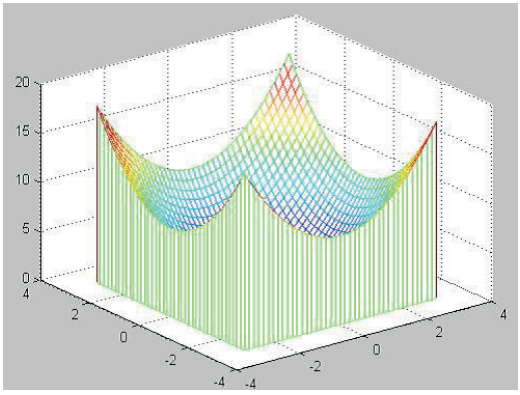


Рисунок 2.35

Построение поверхности с окраской

Особенно наглядное представление о поверхностях дают сетчатые графики, использующие функциональную закраску ячеек.

Например, цвет окраски поверхности z(x, y) может быть поставлен в соответствие с высотой z поверхности с выбором для малых высот темных тонов, а для больших – светлых.

Для построения таких поверхностей используются команды класса surf (...):

surf (X, Y, Z. C) – строит цветную параметрическую поверхность по данным

матриц X, Y и Z с цветом, задаваемым массивом C;

surf(X.Y.Z) – аналогична предшествующей команде, где C=Z, так что цвет задается высотой той или иной ячейки поверхности;

surf(x.y.Z) и surf(x.y.Z.C) с двумя векторными аргументами x и y – векторы x и y заменяют первых два матричных аргумента и должны иметь длины length(x)=n и length(y)=m, где [m.n]=size(Z). В этом случае вершины областей поверхности представлены тройками координат (x(j), yd), Z(1,j)). Заметим, что x соответствует столбцам Z, а y соответствует строкам;

surf(Z) и surf(Z.C) используют x = 1:n и y = 1:m. В этом случае высота Z – однозначно определенная функция, заданная геометрически прямоугольной сеткой;

h=surf (...) – строит поверхность и возвращает дескриптор объекта класса surface.

Команды axis, caxis, color-map, hold, shading и view задают координатные оси и свойства поверхности, которые могут использоваться для большей эффектности показа поверхности или фигуры.

Ниже приведен простой пример построения поверхности – параболоида:

Пример 28

```
[x,y] = meshgrid([-3:0.15:3]);
z = x.^2+y.^2;
surf (x, y, z) (см. рисунок 2.36)
```

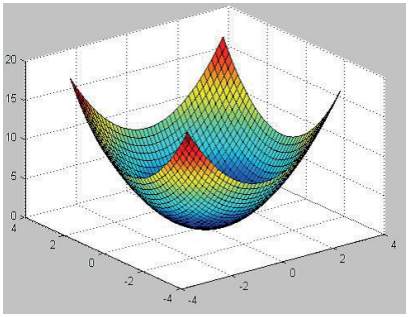


Рисунок 2.36

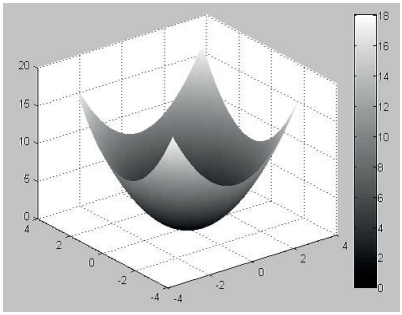


Рисунок 2.37

Можно заметить, что благодаря функциональной окраске график поверхности гораздо более выразителен, чем при построениях без такой окраски, представленных ранее (причем даже в том случае, когда цветной график печатается в черно-белом виде).

В следующем примере используется функциональная окраска оттенками серого цвета с выводом шкалы цветовых оттенков:

Пример 29

```
[x,y] = meshgrid([-3:0.15:3]);
z = x.^2+y.^2;
surf (x,y,z)
colormap(gray)
shading interp
colorbar (см. рисунок 2.37)
```

Построение сферы

Для расчета массивов X, Y и Z координат точек сферы как трехмерной фигуры используется функция sphere:

[X, Y, Z] = sphere(N) – генерирует матрицы X, Y и Z размера (N+1)x (N+1) дляпоследующего построения сферы с помощью команд surfl (X, Y,Z) или surfl(X,Y,,Z);

[X, Y, Z] = sphere – аналогична предшествующей функции при N = 20.

Пример 30

[X,Y,Z] = sphere(30);

surfl(X, Y, Z) (см. рисунок 2.38).

Хорошо видны геометрические искажения (сфера приплюснута), связанные с разными масштабами по координатным осям.

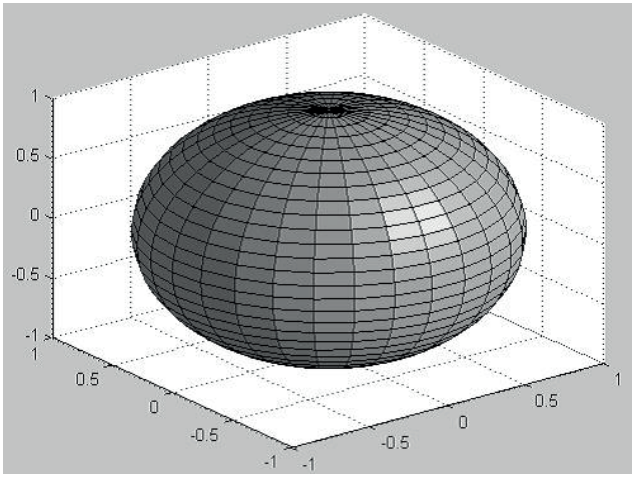


Рисунок 2.38

Построение цилиндра

Для построения цилиндра в виде трехмерной фигуры применяется функция cylinder:

[X, Y, Z] = cylinder(R,N) – создает массивы X, Y и Z, описывающие цилиндрическую поверхность с радиусом R и числом узловых точек N для последующего построения с помощью функции surf(X.Y.Z);

[X, Y, Z] = cyl1nder(R) и [X, Y, Z] = cylinder – подобны предшествующей функции для N = 20 и R = [1 1].

Пример 31

[X, Y, Z] = cylinder(10.30);

surf(X, Y, Z, X) (см. рисунок 2.39).

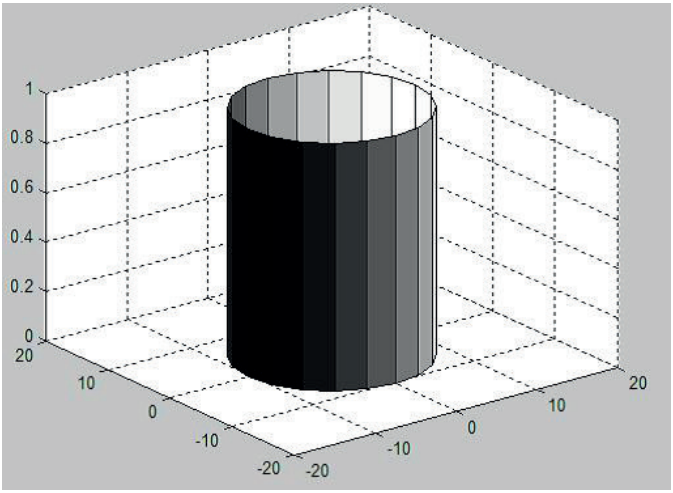


Рисунок 2.39

Объемные круговые диаграммы

Иногда используются объемные круговые диаграммы. Для их построения служит команда pie3:

pie3(...) – дает построение объемных секторов.

Пример 32

X = [1 2 3 4 5];

pie3(x,[0 0 1 0 1]) (см. рисунок 2.40).

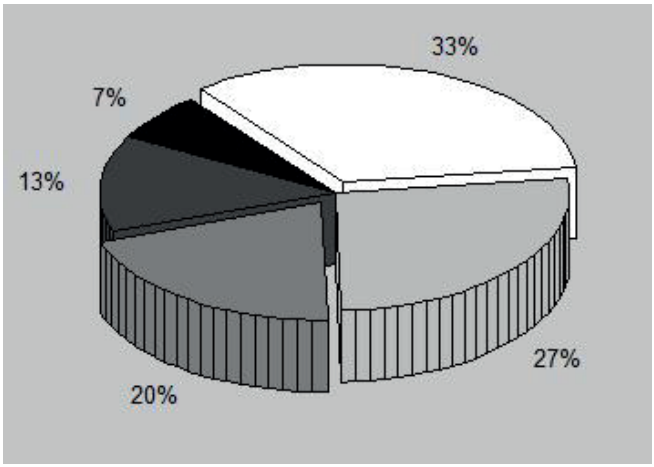


Рисунок 2.40

Окраска плоских многоугольников

Для построения окрашенных в заданный цвет плоских многоугольников может использоваться команда fill (заполнить):

fill(X.Y.C) – строит закрашенный плоский многоугольник, вершины которого задаются векторами X и Y с цветом, заданным C. Многоугольник должен быть замкнутым. Для построения нескольких прямоугольников параметры команды должны быть матрицами.

fill(Xl.Yl,Cl,X2.Y2,C2,...) – представляет собой другой способ построения нескольких закрашенных прямоугольников.

Пример 33
X = [1 2 3 2 1];
Y = [5 0.5 0 4 5];
fill(x,y,[0 0 1]) (см. рисунок 2.41).

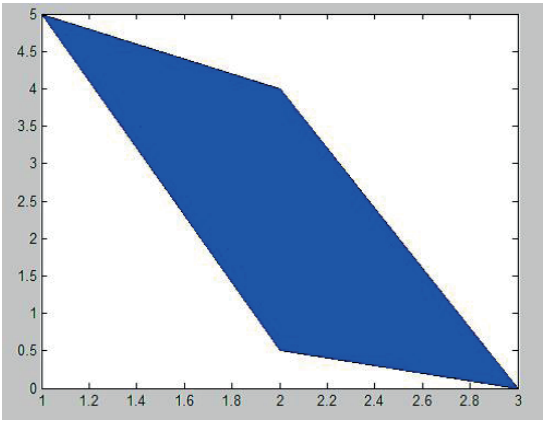


Рисунок 2.41

Трехмерная графика с треугольными плоскостями

К числу специальных видов графики относится построение объемных фигур с помощью плоских треугольников. Для построения таких фигур в виде каркаса (без окраски и отображения плоскостей) используется команда trimesh:

trimesh(TRI,X,Y,Z,C) – построение объемной каркасной фигуры с треугольниками, специфицированными матрицей поверхности TRI, каждая строка которой содержит три элемента и задает одну треугольную грань путем указания индексов, по которым координаты выбираются из векторов X, Y, Z. Цвета ребер задаются вектором C;

trimesh(TRI.X.Y.Z) – построение, аналогичное предшествующему при C = Z, т. е. с цветом ребер, зависящим от значений высоты;

H = trimesh(...) – строит график и возвращает дескрипторы графических объектов;

trimesh(..., ‘param’. ‘value’. ‘param’, ‘value’...) – добавляет значения ‘value’ для параметров ‘param’.

Следующий пример иллюстрирует применение команды trimesh для построения случайной объемной фигуры, параметры которой задаются с помощью генератора случайных чисел:

Пример 34

```
x = rand(1,40);
y = rand(1,40);
z = sin(x.^y);
tri = delaunay(x,y);
trimesh (tri,x,y,z) (см. рисунок 2.42).
```

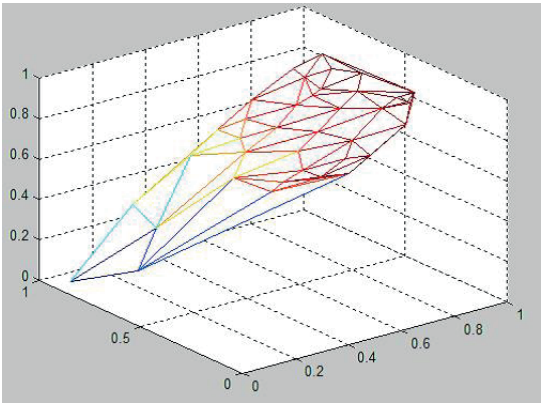


Рисунок 2.42

Другая фигура, абсолютно аналогичная, по заданию входных параметров команда – tnsurf(...) – отличается только закраской треугольных областей, задающих трехмерную фигуру. Если в приведенном выше примере заменить функцию trimesh на trisurf, то можно получить цветные графики.

Пример 35

```
x = rand(1,40); y = rand(1,40);
z = sin(x.^y);
tri = delaunay(x,y);
trisurf (tri,x,y,z) (см. рисунок 2.43).
```

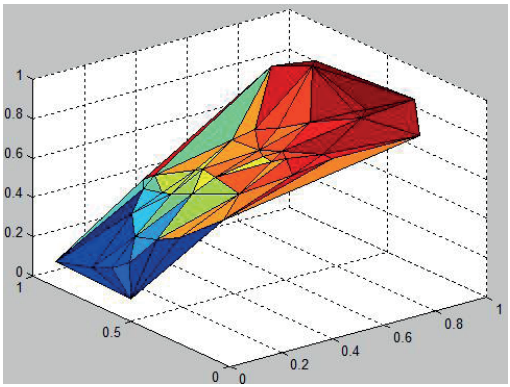


Рисунок 2.43

3. СЦЕНАРИИ И М-ФАЙЛЫ

Для простых операций удобен интерактивный режим, но если вычисления нужно многократно выполнять или необходимо реализовывать сложные алгоритмы, то следует использовать m-файлы MATLAB (расширение файла состоит из одной буквы m). Познакомимся со *script-m-файлами* (или сценариями) – текстовыми файлами, содержащими инструкции на языке MATLAB, подлежащими исполнению в автоматическом пакетном режиме. Создать такой файл удобнее с помощью редактора системы MATLAB. Он вызывается из командного окна системы MATLAB командой меню *File/New/M-file* (или самой левой кнопкой на полосе инструментов, на которой изображен чистый белый лист бумаги). Записанные в script-файлы команды будут выполнены, если в командной строке ввести имя script-файла (без расширения). Переменные, определяемые в командном окне и переменные, определяемые в сценариях, составляют единое рабочее пространство системы MATLAB, причем переменные, определяемые в сценариях, **являются глобальными**, их значения заместят значения таких же переменных, которые были использованы до вызова данного *script-файла.*

После создания текста сценария его надо сохранить на диске в удобном для вас каталоге. Путь к этому каталогу обязательно должен быть известен системе MATLAB. Командой *File/ Set Path* вызывается диалоговое окно просмотрщика путей доступа к каталогам. Для добавления нового каталога в список путей доступа необходимо выполнить далее команду меню *Path/ Add to path.*

4. SIMULINK

4.1. Система моделирования Simulink

Основным достоинством расширения **Simulink** является простота и наглядность его использования при моделировании различных устройств и систем, в том числе и электротехнических. В основном это связано с тем, что вы не имеете дело с написанием строк программы, их редактированием и отладкой, как это имеет место в **MATLAB** или любом ином языке высокого уровня. В **Simulink** используется совершенно иной подход – визуально-ориентированный. При таком моделировании используются готовые блоки, которые необходимо с помощью мыши перенести из библиотеки в окно документа **Simulink**, соединить линиями входы и выходы этих блоков. В результате получаем S-модель, т. е. **Simulink** модель, которую запускаем простым нажатием кнопки **Run**.

4.2. Состав библиотеки Simulink

Библиотека **Simulink** представляет собой набор различных визуальных блоков. Для доступа к ним необходимо нажать кнопку **Simulink Library** на панели инструментов **MATLAB** (рисунок 4.1). При этом появляется окно браузера (обозревателя, навигатора, программы просмотра) библиотеки, представленное на рисунок 4.2.

Окно содержит следующие элементы:

1) Заголовок, с названием окна – **Simulink Library Browser**.

2) Панель инструментов, с ярлыками наиболее часто используемых команд.

3) Окно комментария для вывода поясняющего сообщения о выбранном блоке.

4) Список разделов библиотеки, реализованный в виде дерева.

5) Окно содержимого раздела библиотеки (список вложенных разделов библиотеки или блоков).

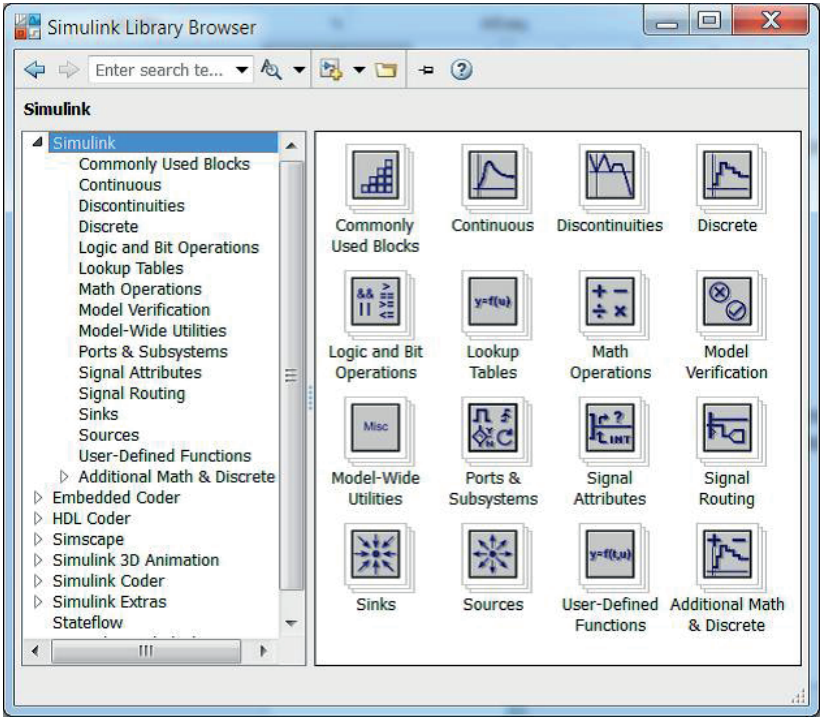


Рисунок 4.1

Список разделов библиотеки *Simulink* представлен в виде дерева. В левой части окна выделена основная библиотека, а в правой части – показаны ее разделы. При выборе соответствующего раздела библиотеки правой части окна отображается его содержимое (рисунок 4.1).

– *Continuous* – линейные блоки.
– *Discrete* – дискретные блоки.
– *Functions&Tables* – функции и таблицы.
– *Math* – блоки математических операций.
– *Nonlinear* – нелинейные блоки.
– *Signals & Systems* – сигналы и системы.
– *Sinks* – регистрирующие устройства.
– *Sources* – источники сигналов и воздействий.
– *Subsystems* – блоки подсистем.

Для того чтобы развернуть или свернуть узел дерева, достаточно щелкнуть на его пиктограмме левой клавишей мыши.

Ниже представлены некоторые разделы и блоки библиотеки **Simulink**. Полный перечень блоков доступен в **Help.** Для его получения достаточно два раза щелкнуть по любому блоку и в открытом окне параметров блока нажать кнопку **Help** (рисунок 4.2).

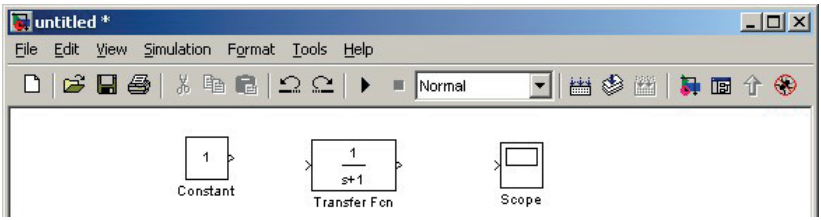


Рисунок 4.2

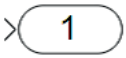
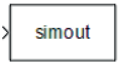
Sources – источники сигналов

1.  Band-Limited White Noise Генератор белого шума
2.  Chirp Signal Генератор линейно-изменяющийся частоты
3.  Clock Источник времени
4.  Constant Источник постоянного сигнала

5		Источник сигнала типа «счетчик»
6		Источник сигнала типа «счетчик с ограничением
7		Дискретный источник времени
8		Перечислимая константа
9		Блок считывания данных из файла
10		Блок считывания данных из рабочей области Матлаб
11		Формирователь сигнала нулевого уровня
12		Входной порт
13		Источник импульсного сигнала
14		Источник линейно-изменяющегося воздействия
15		Источник случайного сигнала с нормальным распределением
16		Источник периодического сигнала с Интерполяцией
17		Источник периодического сигнала
18		Источник ступенчатого периодического сигнала

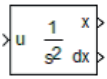
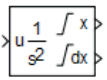
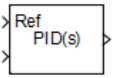
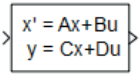
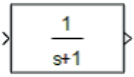
19			Конструктор сигналов
20			Генератор сигналов
21			Источник синусоидального сигнала
22			Генератор ступенчатого сигнала
23			Источник случайного сигнала с Равномерным распределением

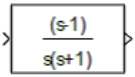
Sinks – приемники сигналов

1			Цифровой дисплей
2			«Плавающий осциллограф»
3			Выходной порт
4			Осциллограф
5			Блок остановки моделирования
6			Концевой приемник
7			Блок записи в файл
8			Блок записи в рабочую область Matlab

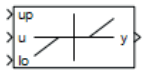
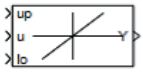
9		XY Graph	Графопостроитель

Continuous – блоки непрерывных моделей

	Derivative	Блок вычисления производной
	Integrator	Интегратор
	Integrator Limited	Интегратор с ограничением
	Integrator, Second-Order	Интегратор второго порядка
	Integrator, Second-Ord…	Интегратор второго порядка с ограничением
	PID Controller	ПИД-регулятор (контроллер)
	PID Controller (2DOF)	ПИД-регуляторы с двумя степенями свободы (two-degree-of-freedom)
	State-Space	Объект пространства состояний
	Transfer Fcn	Передаточная функция
	Transport Delay	Блок фиксированной задержки сигнала

	Variable Time Delay	Блок управляемой задержки ремени
	Variable Transport D...	Блок управляемой задержки сигнала
	Zero-Pole	Передаточная функция «нули-полюса»

Discontinuities – нелинейные блоки

	Backlash	Люфт
	Coulomb & Viscous Fric...	Блок сухого и вязкого трения
	Dead Zone	Зона нечувствительности
	Dead Zone Dynamic	Управляемая зона нечувствительности
	Hit Crossing	Блок определения момента пересечения порогового значения
	Quantizer	Квантователь
	Rate Limiter	Управляемый блок ограничения сигнала
	Rate Limiter Dynamic	Управляемый блок ограничения скорости изменения сигнала

Реле

Ограничитель

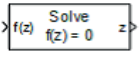
Управляемый ограничитель

Блок, выполняющий сброс сигнала до нулевого уровня

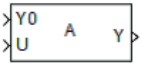
Math Operations – блоки математических операций

Блок вычисления модуля

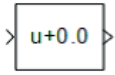
Сумматор (пиктограмма в виде прямоугольника)



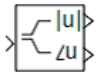
Блок алгебраического контура



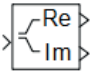
Блок присвоения элементам массива новых значений



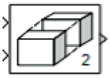
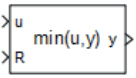
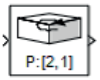
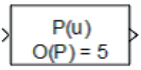
Смещение

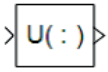
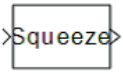


Блок вычисления модуля и (или) аргумента комплексного числа



Блок вычисления действительной и (или) мнимой части комплексного числа

	Divide	Блок деления
	Dot Product	Блок скалярного умножения векторов
	Find Nonzero Elements	Блок поиска ненулевых элементов
	Gain	Усилитель
	Magnitude-Angle to Co...	Блок расчета комплексного числа по его модулю и аргументу
	Math Function	Математическая функция
	Matrix Concatenate	Блок объединения сигналов в матрицу
	MinMax	Блок вычисления максимального или минимального значения с возможностью сброса
	MinMax Running Resett...	Блок вычисления максимального или минимального значения
	Permute Dimensions	Блок транспонирования
	Polynomial	Степенной многочлен
	Product	Умножитель

	Product of Elements	Блок вычисления произведения элементов вектора
	Real-Imag to Complex	Блок расчета комплексного числа по его действительной и мнимой части
	Reciprocal Sqrt	Блок вычисления единицы деленной на квадратный корень
	Reshape	Преобразователь размерности сигнала
	Rounding Function	Блок округления числа
	Sign	Блок определения знака числа. При 0 на входе 0 на выходе
	Signed Sqrt	Квадратный корень со знаком. Например, при u = –2 на выходе получим «– 1.4142»
	Sine Wave Function	Синусоидальная функция
	Slider Gain	Ползунковый регулятор
	Sqrt	Квадратный корень
	Squeeze	Блок сжатия
	Subtract	Блок вычисления разности
	Sum	Сумматор (круглая пиктограмма)

Блок вычисления суммы элементов вектора

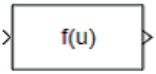
Тригонометрическая функция

Унарный минус

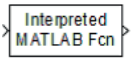
Блок объединения векторов

Блок поддержки вычислений, использующих шаг дискретизации

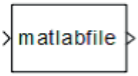
User-Defined Functions – функции, задаваемые пользователем



Блок задания функции



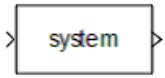
Блок ввода функций или выражений MATLAB



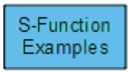
Блок задания S-функции второго уровня

Блок создания MATLAB-функции для использования в Simulink

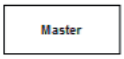
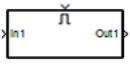
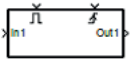
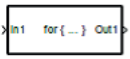
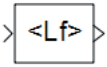
Блок для использования системных объектов

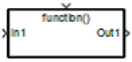
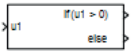
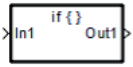
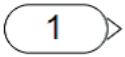
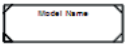
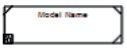
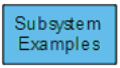
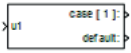
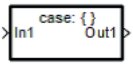
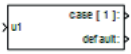
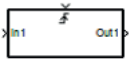


S-функция

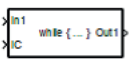
 S-Function Builder	Конструктор S-функции на языке C
 S-Function Examples	Примеры S-функции

Порты и подсистемы Ports & Subsystems

 Atomic Subsystem	Неделимая подсистемы
 CodeReuseS-ubsystem	Подсистема, имеющая общий код для всех ее экземпляров в модели
 Configurable Subsystem	Блок конфигурации подсистем
 Enable	Создание порта для управления подсистемой
 Enabled Subsystem	Блок создания управляемой подсистемы, т. е. E-подсистемы
 Enabled and Triggered S...	Управляемая уровнем и фронтом сигнала подсистема, т. е. ET-подсистема
 For Each Subsystem	Позволяет многократно выполнять алгоритм на каждом элементе или подмассиве
 For Iterator Subsystem	Блок создания подсистемы, выполняющей итерационные действия под управлением цикла типа for
 Function-Call Feedback L...	Прерывает обратную связь между блоками
 Function-Call Generator	Внешний блок управления подсистемой с запросом функци

 Function-Call Split	Обеспечивает разветвление сигнала
 Function-Call Subsystem	Блок создания подсистемы с запросом функции
 If	Блок условного оператора
 If Action Subsystem	Управляемая условием подсистема
 In1	Входной порт
 Model	Блок, обеспечивающий подключение файла к текущей модели
 Model Variants	Позволяет одну модель использовать как блок в другой модели
 Out1	Выходной порт
 Subsystem	Блок позволяет создать подсистему
 Subsystem Examples	Примеры подсистем
 Switch Case	Переключаемая подсистема
 Switch Case Action Subs...	Примеры подсистем
 Switch Case	Переключаемая подсистема
 Trigger	Создание триггерного (пускового) порта для запуска подсистемы
 Triggered Subsystem	Блок подсистемы с триггерной системой управления

Представляет собой подсистему с несколькими подсистемами



Блок создания подсистемы, выполняющей итерационные действия под управлением цикла типа While

4.3. Создание модели

Для создания модели в среде *Simulink* необходимо последовательно выполнить ряд действий:

1. Создать новый файл модели с помощью команды ***File/New/Model***.

Создание, редактирование и запуск Simulink модели (рисунок 4.3).

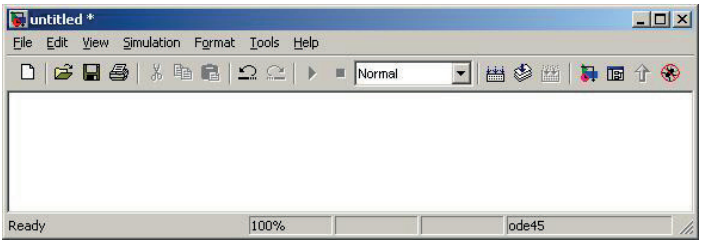


Рисунок 4.3 – Пустое окно модели

2. Расположить в окне модели необходимые блоки. Для этого открыть соответствующий раздел библиотеки и, указать курсором на блок, «перетащить» блок в созданное окно (рисунок 4.4).

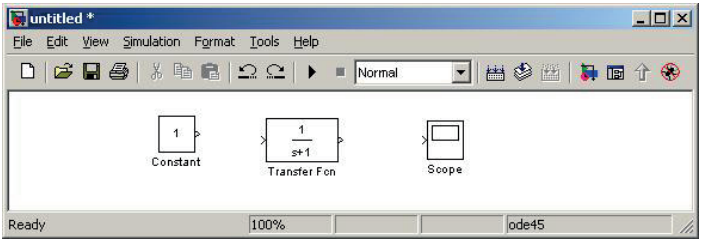


Рисунок 4.4 – Окно модели, содержащее блоки

3. Далее, если требуется, можно изменить параметры блока, установленные программой «по умалчиванию». Для этого необходимо дважды щелкнуть левой клавишей «мыши» на изображение блока. Откроется окно редактирования параметров (рисунок 4.5).

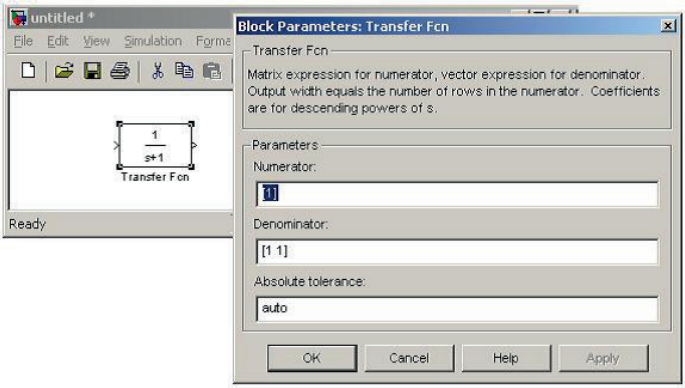


Рисунок 4.5 – Блок, моделирующий передаточную функцию и окно редактирования его параметров

4. После установки на схеме всех блоков из требуемых библиотек необходимо выполнить соединение элементов схемы (рисунок 4.6).

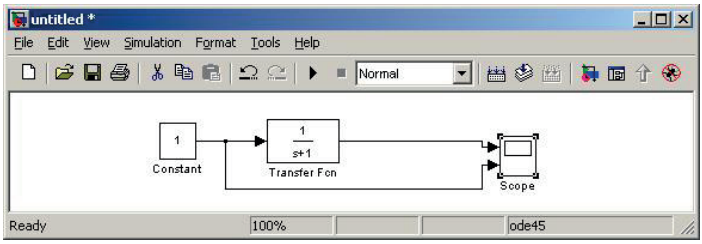


Рисунок 4.6 – Готовая схема модели

5. Для повышения наглядности модели удобно использовать текстовые надписи.

6. После составления расчетной схемы необходимо сохранить ее в виде файла на диске, выбрав пункт ***File/SaveAs…*** в окне схемы и указать папку и имя файла.

4.4. Выполнение расчета

Запуск расчета выполняется с помощью выбора пункта меню ***Simulation/Start*** или пиктограммой на панели инструментов. В ходе выполнения процесс расчета можно приостановить (***Simulation/Pause***), а затем продолжить (***Simulation/Continue***).

Для досрочного завершения расчета используется команда меню ***Simulation/Stop*** или пиктограмму на панели инструментов.

Результаты моделирования процесса отображаются с помощью блоков-приемников сигналов (***Sinks***):

- цифровой дисплей ***Display*** отображает значение сигнала в виде числа (рисунок 4.7);
- осциллограф ***Scope***строит графики исследуемых сигналов в функции
- времени и позволяет наблюдать за изменениями сигналов в процессе моделирования. Для того, чтобы открыть окно просмотра сигналов необходимо выполнить двойной щелчок левой клавишей “мыши” на изображении блока (рисунок 4.8).
- графопостроитель **XY Graph** строит график вида **Y(X)** одного сигнала в функции другого. Блок имеет два входа: верхний предназначен для подачи сигнала, который является аргументом (**X**), нижний – для подачи значений функции (**Y**) (рисунок 4.9).

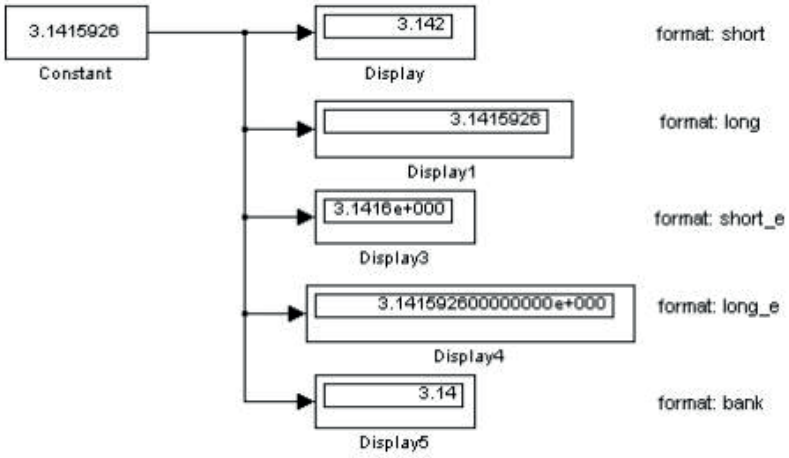


Рисунок 4.7 – Применение блока ***Display*** с использованием различных форматовотображения данных (***Format***)

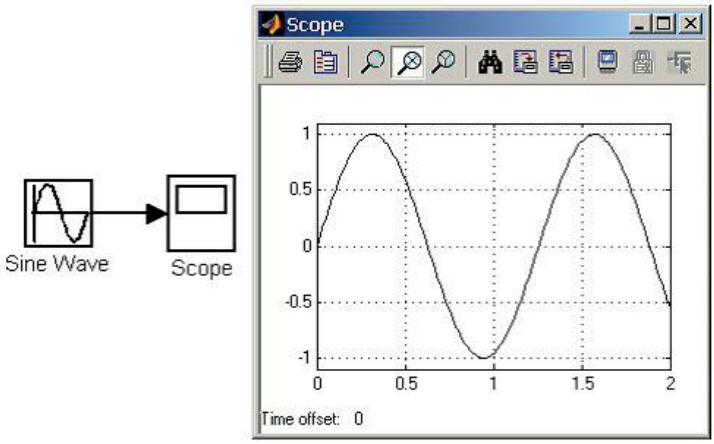
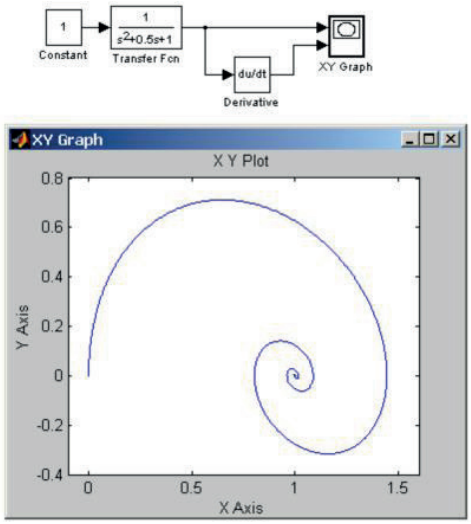


Рисунок 4.8 – Изображение блока ***Scope*** и окно для просмотра графиков



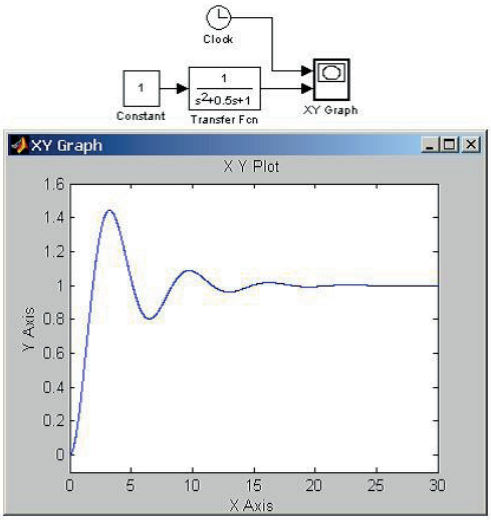


Рисунок 4.9 – Примеры использования графопостроителя

УПРАЖНЕНИЯ

1. Найдите сумму первых четырех членов последовательности

   1 2 3
   2x3 3 x 4 ‘ 4x5

2. Определите вектор t со значениями компонент, равномерно расположенными с шагом 0.2 между 0 и 6 включительно. Теперь используйте его, чтобы нарисоватькривые

   f(t) =sin(pt) и g(t) =exp(-t)sin

   на одном графике, изобразив первую зеленым, а вторую – желтым цветом. Если вы не уверены, как применить нужные вам функции MATLAB, наберите helpexp и т. д. Улучшите график, добавив белую линию, соответствующую *y*= 0.

САМОСТОЯТЕЛЬНЫЕ ЗАДАНИЯ ДЛЯ ЛАБОРАТОРНЫХ РАБОТ

Задание 1.

Задать матрицу A с помощью операции конкатенации:

$$\begin{pmatrix} 3,25 & -1,07 & 2,34 \\ 10,10 & 0,25 & -4,78 \\ 5,04 & -7,79 & 3,31 \end{pmatrix}$$

Задание 2.

Сгенерируйте массив B размером 3*3 со случайными элементами, равномерно распределенными на интервале от 0 до 1.

Задание 3.

Выполните действия:

$$A+10*B, A*T, B^T$$

Почленно умножить A на B, расположить элементы матрицы A по возрастанию (по столбцам), определить максимальный и минимальный элементы матрицы B, вычислить определитель матрицы B.

Задание 4.

Задать массив C, используя операцию индексации и одну из функций: ones или zeros:

$$\begin{pmatrix} 0 \\ 5.71 \\ -3.61 \end{pmatrix}$$

Задание 5.

Решить систему алгебраических линейных уравнений:

$$A*X=C$$

Задание 6.

Определить массив D:

$$\left[D=\sin(A)+B^{3/5} \right]$$

Задание 7.

Для двух векторов:

$\left[D = \sin(A) + B^{3/5} \right]$ и $\overline{b} = \{0,6;3,2;-4,8\}$

определите: $\overline{a} * \overline{b}$; $\overline{a} * \overline{b}; [\overline{a}]$.

Задание 8.

Постройте два графика в рамках одних осей координат:

$y = e^{-x^2}$

$z = arctg\left(x\right)^{1/2}$

$x \in [0, 4\pi]$.

Сделайте надписи на осях, заголовок для графика, пояснительную надпись на рисунке. Задайте самостоятельно тип линий и цвет.

Задание 9.

Построить графики функций *y(x)* и *z(x)* из **Задания 8** в разных подобластях одного графического окна. Интервалы изменения для *x* определите самостоятельно.

Задание 10.

Постройте поверхность:

$f(x, y) = 1n\left(x^2 + y^{2-xy}\right)$,

$x, y \in [1, 2]$.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Как изменить на экране формат вывода числа?
2. Как можно просмотреть в MATLAB список всех элементарных математических функций?
3. Какие виды функций в MATLAB Вам известны?
4. Опишите способы создания одномерных массивов в MATLAB.
5. Опишите способы создания двумерных массивов в MATLAB.
6. Перечислите и объясните действие операторов, используемых при вычислениях с массивами.
7. Опишите действие операций отношения.
8. Опишите действие логических операций.
9. Как построить декартовый и полярный графики функции одной переменной?
10. Как построить несколько графиков в одной системе координат?
11. Как построить графики в разных подобластях одного графического окна?
12. Как изменить цвет и стиль линий на графиках?
13. Как сделать надписи на осях, на полученном рисунке? Как сделать заголовок для графика?
14. Как построить график функции двух переменных? Как построить график поверхности?
15. Что такое m-файлы? Как создать, сохранить и вызвать m-файл?

ЛИТЕРАТУРА

1. *Дьяконов В.П.* MATLAB: учебный курс. – СПб.: Питер, 2001. – 560 с.
2. *Черных И.В.* Simulink: среда создания инженерных приложений. – М.: Диалог-Мифи, 2004. – 496 с.
3. Компьютер для студентов, аспирантов и преподавателей: самоучитель. – М.: Изд-во ТРИУМФ, 2001. – 656 с.
4. *Курбатов Е.А.* MATLAB 7: самоучитель. – М.: Вильямс, 2006.
5. *Черных И.В.* Моделирование электротехнических устройств в MATLAB, SimPowerSystems и Simulink. – М.: ДМК Пресс; Питер, 2008.
6. *Дьяконов В.П., Пеньков А.А.* MATLAB и Simulink в электроэнергетике: справочник. – М.: Горячая линия – Телеком, 2009.
7. *Беспалов В.Я., Котеленец Н.Ф.* Электрические машины. – М.: Академия, 2013.
8. *Копылов И.П.* Математическое моделирование электрических машин. – М.: Высш. шк., 2001.
9. *Герман-Галкин С.Г.* Matlab & Simulink: проектирование мехатронных систем на ПК. – СПб.: КОРОНА-Век, 2008.
10. *Дьяконов В.П.* Simulink 5/6/7: самоучитель. – М.: ДМК-Пресс, 2008.
11. SimPowerSystems. MATLAB. Exponenta. URL: http://matlab.exponenta.ru/simpower/book1/index.php. (дата обращения: 15.12.2014).
12. Simscape. MathWorks Центр компетенций. MathWorks. URL: http://matlab.ru/products/simscape (дата обращения: 28.12.2014).

Г.С. Воронова, М.А. Духанин

ОСНОВЫ MATLAB

Учебник для студентов
по направлению 12.03.01 – «Приборостроение»

Редактор *А. Шабалин*
Компьютерная верстка *А.Ш. Мельниковой*

Подписано в печать 29.11.2018
Формат 60×84 $^{1}/_{16}$. Печать офсетная.
Объем 6,0 п. л. Тираж 100 экз. Заказ 68

Издательство КРСУ
720000, г. Бишкек, ул. Киевская, 44

Отпечатано в типографии КРСУ
720048, г. Бишкек, ул. Анкара, 2а