Титульный лист методических рекомендаций и указаний, методических рекомендаций, методических указаний

Форма
Ф СО ПГУ 7.18.3/40

Министерство образования и науки Республики Казахстан

Павлодарский государственный университет им. С. Торайгырова

Кафедра Вычислительная техника и программирование

МЕТОДИЧЕСКИЕ УКАЗАНИЯ И РЕКОМЕНДАЦИИ

к лабораторным работам

по дисциплине Компьютерное моделирование

для студентов специальности 050704 Вычислительная техника и программное обеспечение

Павлодар

Лист утверждения методических рекомендаций и указаний, методических рекомендаций, методических указаний

Форма
Ф СО ПГУ 7.18.3/41

УТВЕРЖДАЮ
Проректор по УР
_____ Пфейфер Н.Э.
(подпись) (Ф.И.О.)
«___»_____201_г.

Составитель: ст. преподаватель _____Павлюк Ин.И.

Кафедра Вычислительная техника и программирование

Методические указания и рекомендации

к лабораторным работам

по дисциплине Компьютерное моделирование

для студентов специальности 050704 Вычислительная техника и программное обеспечение

Рекомендовано на заседании кафедры
«_____»_____201__г., протокол №__

Заведующий кафедрой _____ Потапенко О.Г. «___» _____201__г
(подпись) (Ф.И.О.)

Одобрено УМС Физики, математики и информационных технологий
(наименование факультета)
«____»_____201__г., протокол №____
Председатель УМС _____ Муканова Ж.Г. «___» _____201__г
(подпись) (Ф.И.О.)

ОДОБРЕНО ОПиМОУП**:**
Начальник ОПиМОУП _____ Варакута А.А. «___» _____201__г
(подпись) (Ф.И.О.)

Одобрена учебно-методическим советом университета
«_____»_____201_г. Протокол №____

Лабораторная работа №1

Тема: Элементы языка MATLAB (MATrix LABoratory)

Среда MATLAB включает интерпретатор команд на языке высокого уровня, графическую систему, пакеты расширений и реализована на языке С. Изначально MATLAB был реализован на Фортране, и язык MATLAB или М-язык конструкциями и отчасти синтаксисом напоминает Фортран. М-язык является языком высокого уровня и предоставляет достаточные возможности для реализации разнообразных вычислений, задач обработки данных и т.д. Этот язык сконструирован для решения математических задач и содержит специальные средства для эффективного выполнения математических операций.

В этой работе представлены общие сведения о работе с матрицами, арифметических и логических операциях, структурах и операторах MATLAB.

1. Теоретическая часть

1.1. Синтаксис и данные в MATLAB

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), при помощи которых создаются пользовательские объекты (user object).

Числовые массивы состоят из комплексных чисел с двойной точностью (тип double) и могут храниться целиком или в упакованном виде в случае разреженной матрицы (тип sparse). Двумерный массив - это матрица, одномерный - вектор, а скаляр - матрица размера 1× 1. Кроме того, существует несколько форматов записи данных в файлы (int8, uint8, …), отличающихся количеством используемых байтов (схема типов MATLAB приведена на рис.1).

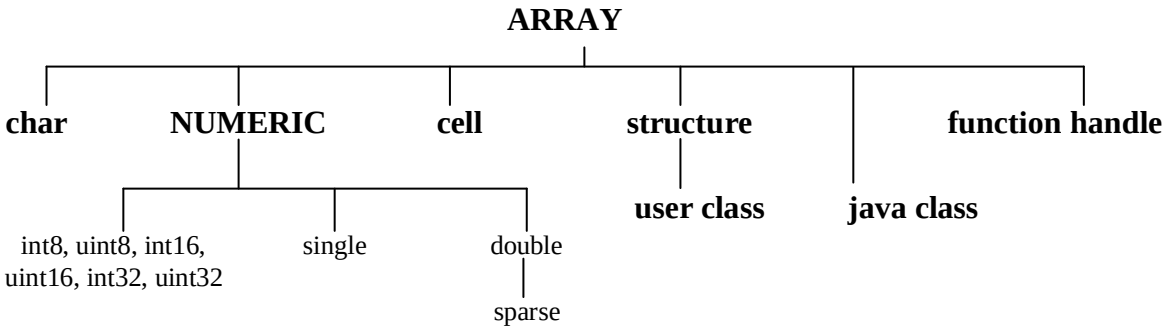

Рис.1. Типы MATLAB

Имя переменной должно начинаться с буквы, за ней могут идти буквы, цифры и символ подчеркивания. Допустимы имена любой длины, но MATLAB идентифицирует их по первым 31 символам и различает большие и малые буквы.

В MATLAB имеется ряд констант, описанных в таблице 1:

Таблица 1. Зарезервированные имена констант

Имя	Описание
ans	результат последней операции
i, j	мнимая единица
pi	число π
eps	машинная точность
realmax	максимальное вещественное число
realmin	минимальное вещественное число
inf	Бесконечность
NaN	нечисловая переменная
end	наибольшее значение индекса размерности массива

(информацию об имеющихся константах можно получить, вызвав справку о разделе команд elmat)

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

```
” 0/0
Warning: Divide by zero.
ans =
   NaN
” 0*inf
ans =
   NaN
” inf-inf
ans =
   NaN
```

Буквы i и j можно использовать как обычные переменные, например в качестве счетчика цикла. В этом случае мнимую единицу можно задать заново:

```
” jj=sqrt(-1)
jj =
      0 + 1.0000i
```

(заметим, что для обозначения мнимой единицы применяется символ i)

Выражения и команды в MATLAB записываются с помощью различных символов, краткое описание которых дано в таблице 2.

Приведем пример использования ряда зарезервированных слов и специальных символов для формирования матрицы размера 2 × 3:

```
” A = [i*realmax,pi:5;-realmin*eps -inf inf/inf]   % пример
A =
  Column 1
     0 + 1.79769313486232e+308i
```

-4.94065645841247e-324

Column 2

3.14159265358979

-Inf

Column 3

4.14159265358979

NaN

В этом примере точка с запятой отделяет строки матрицы, в качестве мнимой единицы фигурируют символы i и j, а в правой части строки знаком % отмечено начало комментария. Для вывода матрицы на экран использован формат short e , а если задать формат short, то матрица A будет выглядеть следующим образом:

” format short, A

A =

*1.0e+308 **

0 + 1.7977i 0.0000 0.0000

0.0000 + -Inf NaN

Обратите внимание, что второй и третий элементы первой строки и первый элемент второй строки не равны нулю - произошло округление при выводе матрицы на экран.

Таблица 2. 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 операторов 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 строке
%	Знак процента обозначает начало комментария
!	Восклицательный знак отмечает начало команды MS DOS, например команда !dir выводит оглавление текущего каталога
‘	Апостроф указывает на символьные строки, а для включения самого апострофа в символьную строку нужно поставить два апострофа подряд

1.2. Задание матриц

MATLAB - это интерактивная система, в которой основным элементом данных является массив. В MATLAB прямоугольный массив чисел - матрица. Особое внимание придается матрицам 1x1, которые называются скалярами, и матрицам, имеющим один столбец или одну строку, - векторам.

По умолчанию все числовые переменные MATLAB считаются матрицами с комплексными числами (так что скалярная величина есть матрица первого порядка, векторы являются матрицами, состоящими из одного столбца или одной строки).

MATLAB использует различные способы для хранения численных и не численных данных, однако вначале лучше рассматривать все данные как матрицы.

Ввод матриц можно выполнять нескольким способами: 1) вводить полный список элементов; 2) генерировать матрицы, используя встроенные функции; 3) загружать матрицы из внешних файлов; 4) создавать матрицы с помощью собственных функций в М-файлах.

1 способ задания матрицы - введение полного списка элементов. Вводя матрицу как список элементов, необходимо следовать нескольким условиям: 1) отделять элементы строки пробелами или запятыми; 2) использовать точку с запятой, **;**, для обозначения окончания каждой строки; 3) окружать весь список элементов квадратными скобками

Например, задание квадратной матрицы (n=4):

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

или

задание вектора-столбца (т.е. матрицы, вторая размерность которой равна 1)

а) вводом одной строки:

```
>> a=[7e-6+5i; 4; 3.2e1]     %ввод вектора-столбца
 a =
      0.0000 + 5.0000i
      4.0000
     32.0000
```

б) вводом нескольких строк:

```
>> a = [                    %ввод вектора-столбца
7e-6+5i
4
3.2e1];
```

2 способ задания матрицы - генерация матрицы с помощью встроенных функций.

Приведем несколько примеров:

1. Задание матрицы командами linspace и logspace. Они позволяют создавать векоры со значениями, меняющимися соответственно в арифметической и геометрической прогрессии. Три параметра команды linspace

задают соответственно первый и последний члены арифметической последовательности, а также число членов. Например:

>> linspace(1, -2, 4)

ans =

1 0 -1 -2

Параметры команды logspace задают соответственно десятичные порядки первого и последнего членов геометрической прогрессии, а также число членов. Например:

>> logspace(1, -1, 5)

ans =

10.0000 3.1623 1.0000 0.3162

0.1000

2. Векторы могут быть сформированы как диапазоны - при помощи двоеточий, разделяющих стартовое значение, шаг и предельное значение. Если величина шага отсутствует, то по умолчанию его значение равно единице: n:m:k. В результате будет сформирован вектор, последний элемент которого не больше k для положительного шага m, и не меньше - для отрицательного: [n, n+m, n+m+m, ...]. Например:

>> a=1:-2:-4

a =

1 -1 -3

3. Для задания ряда матриц специального вида имеются команды MATLAB, аргументами которых являются размерности создаваемых матриц. Если указано одно число N, то формируется квадратная матрица NxN. Например, команда rand(n,m) создаст случайную матрицу размерности n*m, а команда hilb(n) создаст гильбертову матрицу n-го порядка.

Приведем таблицу функций описания матриц (таблица 3).

Таблица 3. Функции описания матриц

Имя	Назначение
eye	единичная матрица
zeros	нулевая матрица
ones	матрица из единиц
rand	случайная матрица со значениями из интервала [0, 1]
hilb	гильбертова матрица
magic	матрица магического квадрата
diag	создание диагональной матрицы или выделение диагонали
triu	выделение верхней треугольной части матрицы
tril	выделение нижней треугольной части матрицы

В MATLAB имеется возможность, обратившись к команде gallery, получить матрицу из подготовленного разработчиками набора стандартных матриц:

[out1, out2, ...]=gallery(NAME, PAR1, PAR2, ...),

здесь NAME - имя матрицы из списка gallery, а PAR1, PAR2 - дополнительные параметры.

Например, разреженная матрица для задачи Пуассона при сетке 6x6 будет получена по команде:

>>Poi=gallery("poisson",6):size(Poi)

ans =

36 36

(для просмотра списка матриц достаточно набрать help gallery).

Из стандартных блоков можно определить новую матрицу:

>> B=[triu(ones(2)), zeros(2,1):ones(1,2),eye(1)]

B =

1 1 0
0 1 0
1 1 1

В результате получается матрица размерности 3x3.

3 способ задания матрицы - загрузка матрицы из внешнего файла.

Большую матрицу можно определить поэлементно, набрав её в обычном редакторе в виде ASCII-файла (например, data.ext), а затем считать при помощи команды

>>load data.ext

4 способ задания матрицы - создавать матрицы с помощью собственных функций в М-файлах мы рассмотрим при выполнении работы № 4.

1.3. Обращение к элементам матрицы

Обращение к элементам матрицы производится по естественному правилу, - в круглых скобках после имени матрицы даются индексы, которые должны быть положительными целыми числами. Например, A(2,1) означает элемент из второй строки первого столбца матрицы A.

Для дальнейших примеров введем матрицу 2x2:

>>A=[1 2+5*i; 4.6e-7 3];

Если в качестве индекса задать комплексное число с дробной вещественной частью, то MATLAB выведет предупреждение, отбросит мнимую составляющую, произведет округление дроби и попытается выполнить операцию:

>>A(3/2+4*i)

Warning: Complex part of array subscript is ignored.

Warning: Subscript indices must be integer values.

ans =

4.6000e-007

Данный пример показывает, что числа хранятся по столбцам и при обращении к данному двумерному массиву элемент A(2) есть то же самое, что и A(2,1). Чтобы изменить элемент матрицы, ему нужно присвоить новое значение:

>>A(2,3)=sin(1) %Третий элемент второй строки

A =

1.0000 2.0000+5.0000i 0

0.0000 3.0000 0.8415

Заметим, что изначально матрица А состояла из двух строк и столбцов. Расширение матрицы (добавление третьего элемента во вторую строку) не потребовало никаких дополнительных действий, при этом третий элемент в первой строке был обнулен автоматически.

Если обратиться к отсутствующему элементу матрицы, то будет выведено сообщение об ошибке:

>>A(3,1)^2 % у матрицы А только две строки

Index exceeds matrix dimensions.

Размер матрицы можно уточнить по команде size, а результат команды size можно использовать для организации новой матрицы. Например, нулевая матрица того же порядка, что и матрица А, будет сформирована по команде:

>>A2=zeros(size(A))

A2 =

0 0 0

0 0 0

Для преобразования матрицы в матрицу с другим числом строк и столбцов служит команда reshape

>>A3=reshape(A2,3,2)

A3 =

0 0

0 0

0 0

С помощью двоеточия легко выделить часть матрицы. Например, вектор из первых двух элементов третьего столбца матрицы А задается выражением

>>A(1:2,3)

ans =

0

0.8415

Двоеточие само по себе означает строку или столбец целиком. Работа с индексами в MATLAB очень удобна. Например, чтобы выделить несколько столбцов массива, достаточно организовать вектор из номеров столбцов. Это можно сделать явным перечислением, а можно указать нужный диапазон. Для выделения матрицы, составленной из нечетных столбцов матрицы А, используем команду:

>> A(:,1:2:3)

ans =

1.0000 0

0.0000 0.8415

Здесь конструкция 1:2:3 означает изменение второго индекса от единицы до трех с шагом два. Двоеточие применяется также для замещения элементов матрицы. Следующая команда позволяет переставить первую и вторую строки матрицы А:

```
>> A([1,2],:)=A([2,1],:)
A =
    0.0000          3.0000                  0.8415
    1.0000          2.0000+5.0000i     0
```

Здесь в качестве индекса выступают векторы [1,2] и [2,1], а для оформления использованы равносильные разделители: запятая и пробел. Для удаления элемента вектора достаточно присвоить ему пустой массив - пару квадратных скобок []. Чтобы вычеркнуть одну или несколько строк (столбцов) матрицы нужно указать диапазон удаляемых строк (столбцов) для одной размерности и поставить двоеточие для другой размерности. Например, для удаления двух последних столбцов матрицы A достаточно ввести команду

```
>>A(:,2:end)=[]   % вырезание строк
A =
    0.0000
    1.0000
```

Обратите внимание, что вместо числового значения индекса указано зарезервированное имя end - максимальное значение индекса.

В списке аргументов size второй параметр позволяет определить соответствующую размерность матрицы, например найти число столбцов матрицы. Для нахождения длины вектора можно воспользоваться также командой length. Число столбцов матрицы A1 равно 3, не зависимо от того, каким способом пользоваться:

```
>>[size(A1,2), length(A1(1,:))]
ans =
    3     3
```

Вместо двоеточия можно также использовать функцию-синоним colon.

1.4. Числовые форматы MATLAB

Все расчеты в MATLAB выполняются с двойной точностью.

Для представления чисел на экране имеются разные форматы. Числовой формат может быть определен в меню Файл - Свойства | Общие (File - Preferences | General) или в командной строке (format <формат>).

Существуют следующие способы представления чисел, устанавливаемые командой format:

Формат	Представление
short	число отображается с 4 цифрами после десятичной точки или в формате short e
shot e	число в экспотенциальной форме с мантиссой из 5 цифр и показателем из 3 цифр
bank	число с любым количеством цифр до десятичной точки и двумя цифрами после
rat	представление в виде рационального дробного числа
long	число с 16 десятичными цифрами
long e	число в экспотенциальной форме с мантиссой из 16 цифр и показателем из 3 цифр
hex	число в шестнадцатиричной форме
plus	символическое отображение числа (плюс - положительное число, минус - отрицательное и пробел для нуля)

По умолчанию действуют формат short и стиль loose, который добавляет пустые строки при выводе результата. Стиль compact подавляет вывод пустых строк. Команда format без параметров восстанавливает значения по умолчанию.

1.5. Арифметические и логические операции

Набор арифметических операций в MATLAB состоит из стандартных операций (сложения-вычитания, умножения-деления, операции возведения в степень и специальных матричных операций. Запись операций сложения (вычитания) и умножения матриц естественна (таблица 4). Если операция применяется к матрицам, размеры которых не согласованы, то будет выведено сообщение об ошибке. Для поэлементного выполнения операций умножения, деления и возведения в степень применяются комбинированные знаки (точка и знак операции). Например, если за матрицей стоит знак (^), то она возводится в степень, а комбинация (.^) означает возведение в степень каждого элемента матрицы. При умножении (сложении, вычитании, делении) матрицы на число соответствующая операция всегда производится поэлементно.

Таблица 4. Знаки операций

Символ	Назначение
+, -	Символы плюс и минус обозначают знак числа или операцию сложения и вычитания матриц, причем матрицы должны быть одной размерности
*	Знак умножения обозначает матричное умножение; для поэлементного умножения матрицы применяется комбинированный знак (.*)
'	Апостроф обозначает операцию транспонирования (вместе с комплексным сопряжением); транспонирование без вычисления сопряжения обозначается при помощи комбинированного знака (.')
/	Левое деление
\	Правое деление
^	Оператор возведения в степень; для поэлементного возведения в степень применяется комбинированный знак (.^)

Операторы отношения (см.табл.5) и логические операторы (см.табл.6), а также соответствующие им команды (см.табл.7) позволяют проводить сравнение массивов одинакового размера. Результатом таких операций являются матрицы из нулей и единиц (1 - истинность, 0 - ложь).

Таблица 5. Операции отношения

Символ	Назначение	Имя функции
<	Меньше	lt
>=	Больше или равно	ge
>	Больше	gt
<=	Меньше или равно	le
==	Равно	eq
~=	Не равно	ne

Таблица 6. Логические операции

Символ	Назначение	Имя команды
&	Логическое "и"	and
|	Логическое "или"	or
~	Отрицание	not
	Исключительное или	xor

Таблица 7. Команды проверки и сравнения

Имя	Назначение
find	Поиск значения согласно заданному условию; определение индексов
all	Проверка того, что все элементы не равны нулю
any	Проверка того, что хотя бы один элемент не равен нулю
isempty	Выявление пустого массива
isequal	Проверка равенства матриц
issparse	Проверка матрицы на разреженность
nonzeros	Вывод ненулевых элементов массива
finite	Выявление ограниченных элементов массива
isnumeric	Проверка, является ли массив числовым
isinf	Выявление бесконечных элементов массива
isnan	Выявление элементов нечислового типа
isletter	Проверка на символ
isstr	Проверка на строковую переменную
isglobal	Проверка на глобальную переменную
strcmp	Сравнение двух строк

При попытке сравнения векторов или матриц различной размерности будет введено сообщение об ошибке. При сравнении скаляра с матрицей сначала из скалярной переменной создается матрица нужного размера и затем происходит сравнение.

Операции (==, ~=) проводят сравнение вещественных и мнимых частей компексных чисел, а операции (>, <, >=, <=) - только вещественных частей.

Для логических операций ненулевое число отождествляется с единицей.

Логические операции можно записывать в виде функций.

Элементарные логические операции дополнены набором функций MATLAB, позволяющих проверить для матриц некоторые условия (см.табл.7). Проверка для матриц проводится по столбцам; результатом проверки для вектора является число.

1.6. Математические функции

Перечислим список элементарных математических функций MATLAB, применяемых к скалярным величинам (матрицы размера 1x1) или к каждому элементу матрицы-аргумента (таблица 8).

Таблица 8. Основные математические функции

Имя	Описание
abs	Абсолютная величина
log	Натуральный логарифм
log10	Десятичный логарифм
real	Вещественная часть комплексного числа
conj	Сопряженное число
fix	Округление до ближайшего целого в сторону нуля
floor	Округление до ближайшего целого в сторону -∞
gcd	Наибольший общий делитель
mod	Остаток от деления, вычисляемый по формуле mod(x,y)=x-y.*floor(x./y)
exp	Экспонента
sign	Знак числа
sqrt	Корень квадратный
imag	Мнимая часть комплексного числа
lcm	Наименьшее общее делимое
ceil	Округление до ближайшего целого в сторону +∞
round	Округление до ближайшего по абсолютному значению целого числа
rem	Остаток от деления, вычисляемый по формуле rem(x,y)=x-y.*fix(x./y)

Перед вызовом любой функции полезно опробовать обращение к ней, пользуясь тем, что MATLAB интерпретатор. Например:

```
>>sin([0 1; 2 pi])
ans =
    0         0.8415
    0.9093    0.0000
```

Реализация одной операции для всех элементов массива называется векторизацией.

Список всех имеющихся математических функций может быть получен по команде help elfun.

МАТРИЧНЫЕ ВЫЧИСЛЕНИЯ

Команды работы с данными

Имя	Назначение
max	Определение максимальных элементов массива
min	Определение минимальных элементов массива
sum	Суммирование элементов массива
cumsum	Суммирование элементов массива с накоплением
prod	Произведение элементов массива
cumprod	Произведение элементов массива с накоплением
median	Определение медиан (срединных значений)
mean	Определение средних значений массива

Команды сортировки

Имя	Назначение
sort (X,N)	Сортировка по возрастанию (упорядочивание по модулю) элементов массива X по размерности N
sortrows (X,N)	Сортировка строк с упорядочиванием по элементам столбца с номером N
cplxpair (X, TOL, N)	Сортировка комплексно-сопряженных пар (упорядочивание по вещественной части) с точностью TOL по размерности N

Функции от матриц

Имя	Описание
expm	Матричная экспонента
sqrtm	Квадратный корень из матрицы
logm	Логарифм от матрицы
funm	Функция от матрицы

Матричные характеристики

Имя	Описание
det	Вычисление определителя
trace	Вычисление следа матрицы
cond	Вычисление числа обусловленности
condest	Оценка числа обусловленности
rcond	Параметр обусловленности
rank	Определение ранга матрицы
norm	Вычисление нормы матрицы (вектора)
normest	Оценка нормы матрицы (вектора)

Команды работы с данными

Имя	Описание
inv	Вычисление обратной матрицы
pinv	Вычисление псевдообратной матрицы
null	Определение ядра (нуль-пространства) матрицы
orth	Вычисление ортонормального базиса

2. Задания для самостоятельного выполнения лабораторной работы №1.

Задание 1:

Дана матрица Дюрера:

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

1.1. Введите в рабочее пространство MATLAB матрицу Дюрера поэлементно.

1.2. С помощью функций MATLAB покажите замечательные свойства данной матрицы: а) сумму элементов главной (*побочной) диагонали; б) сумму элементов каждого столбца; в) сумму элементов каждой строки.

1.3. Создайте единичную матрицу A размер 3x3; Создайте матрицу нулей Z размер 3x3.

1.4. Создайте магический квадрат с помощью функции magic и преобразуйте его в квадрат Дюрера.

1.5. Создайте симметричную матрицу (добавление матрицы к её транспонированной дает симметричную матрицу).

1.6. Дана матрица A=[1 2 3; 4 5 6; 7 8 9] выполните выделение: а) нижней треугольной части матрицы A; б) верхней треугольной части матрицы A.

Задание 2:

2.1. Создайте вне MATLAB текстовый файл data1.dat, содержащий следующие строки:

-0.4326	-1.1465	0.3273	-0.5883
-1.6656	1.1909	0.1746	2.1832
0.1253	1.1892	-0.1867	-0.1364

2.2. Прочитайте этот файл из среды MATLAB и создайте переменную data1.

2.3. Найдите сумму элементов главной и побочной диагоналей.

Задание 3:

3.1. Введите матрицу H размера 2x2 и матрицу O из единиц той же размерности.

3.2. Перемножьте матрицы, используя обычное умножение, после примените поэлементную операцию.

3.3. Задайте вектор b=[1, 2, 3]. Выполните поэлементное возведение в квадрат.

3.4. Образуйте матрицу C умножением вектора-столбца, полученного транспонированием из строки, на исходный вектор-строку.

3.5. Образуйте матрицу D: прибавьте в матрице C единичную матрицу той же размерности, умноженную на комплексное число π+i. Вычтите из полученного результата число 2.

Задание 4:

4.1.Дана матрица A=[1 2; 3 4]выполнить число обусловленности матрицы A.

4.2. Дана матрица B=[7 2 1; 4 11 6; 7 3 9] выполнить следующее:

а) определить ранг матрицы B;

б) вычислить норму матрицы B;

в) вычислить ортогональный базис матрицы B;

г) отсортировать в порядке возрастания элементы каждого столбца матрицы B.

Задание 5:

5.1 После завершения работы отобразите на экране содержимое рабочего пространства MATLAB: наберите в командной строке >>whos (данная команда осуществляет вывод на экран текущее состояние рабочего пространства: имена переменных, размер и используемую память). Перепишите содержимое рабочей области в лист отчета по лабораторной работе и предоставьте его преподавателю.

5.2 Выполните очистку рабочего пространства (команда clear);

5.3 Завершите работу с MATLAB.

3. Литература

1. Гультяев А.К. MATLAB5.3. Имитационное моделирование в среде Windows. Практическое пособие. - СПб.: КОРОНА принт, 2001.

2. Дьяконов В.П. Справочник по применению системы PC MATLAB. М.:Физматлит, 1993.

3. Карпелевич И.К., Садовский Л.Е. Элементы линейной алгебры и линейного программирования. М.: Физматлит, 1963.

4. Потемкин В.Г. Система MATLAB. М.:Диалог-МИФИ, 1997.

Лабораторная работа №2

Тема: Элементы программирования в MATLAB

Среда MATLAB включает стандартный набор конструкций языка программирования высокого уровня. В этой работе представлены общие сведения о работе с условными операторами, оператором выбора, циклами, а также программирование собственных функций.

1. Теоретическая часть

1.1.Многомерные массивы

Массивы с числом размерностей более двух считаются многомерными. Такие массивы могут быть считаны из файла или созданы при помощи команд, таких как zeros, ones, rand. Число параметров при обращении к этим командам должно соответствовать размерности вводимого массива. Например, трехмерный массив 2x4x2 из нулей будет организован по команде

```
>>S=zeros(2,4,2);
```

Обращение к элементам многомерного массива производится по обычным правилам работы с массивами, так что действуют двоеточия для указания диапазона, а end означает максимальное значение данной размерности. Чтобы изменить какой-нибудь элемент, достаточно присвоить ему значение точно так же, как для обычных массивов, а если элемента не было, то произойдет увеличение размерности массива:

```
>>S(3,1,2)=13; size(S(:,:,1))
ans =
     3     4
```

Заметим, что пополнение массива означает дополнительные затраты времени на переписывание данных. Поэтому для повышения скорости расчета рекомендуется описывать максимальную размерность массива сразу (резервировать память), если это возможно. Когда многомерный массив организуется для хранения нескольких матриц одинакового размера, то первые два индекса удобнее отвести под строки и столбцы матриц, а последний индекс - для номера матрицы. Тогда в результате получим:

```
>>S2=S(:,:,end)
S2 =
     0     0     0     0
     0     0     0     0
    13     0     0     0
```

Резервирование памяти ускоряет работу в среде MATLAB, поскольку не расходуется время на пополнение массивов.

1.2. Массивы ячеек

Для хранения разнородных объектов (массивов разных размерностей, разнотипных данных) удобно пользоваться массивами ячеек, которые создаются двумя способами:

- по команде cell;
- заключением объектов в фигурные скобки.

Например,

```
>>C = {sum(S) min(max(S)) sum(sum(sum(S)))}
C =
    [1x4x2 double]    [1x1x2 double]    [13]
```

Для указания элементов массива ячеек используются фигурные скобки, так что в результате обращения к третьему элементу массива C получим число 13:

```
>>C{3}
ans =
    13
```

а содержимое первого элемента массива ячеек C есть:

```
>>C1=C{1}
C1(:,:,1) =
    0    0    0    0
C1(:,:,2) =
    13   0    0    0
```

Для превращения структуры C1 в обычный массив можно воспользоваться командами squeeze или shiftdim, которые удаляют равные единице размерности (матрица в один столбец или одну строку превращается в вектор):

```
>>C2 = shiftdim(C1)'
C2 =
    0    0    0    0
    13   0    0    0
```

Нужно помнить, что составные части массива ячеек представлены копиями, так что при изменении исходного массива S в объекте C никаких изменений не последует. Иными словами, это не указатель на массив, а сам массив.

Для преобразования массива символов в массив ячеек применяется команда cellstr, а обратная процедура реализуется командой char.

Если потребуется создать текст из нескольких строк, то обычный прием отделения строк точкой с запятой может не сработать, поскольку все строки должны быть одной длины.

В таких случаях можно подготовить массив ячеек:

```
>>C={"One";"Three";"Seven"}
C =
    "One"
    "Three"
    "Seven"
```

и затем преобразовать его в массив символов при помощи команды char:

>>S=char(C)

S =

One

Three

Seven

Для получения строк одной длины MATLAB добавляет нужное число пробелов, что легко увидеть, затребовав данные о размерностях массива S:

>>size(S)

ans =

3 5

1.3.Структуры

Структурами MATLAB (тип struct) являются многомерные массивы. Доступ к ним осуществляется путем указания индексов-имен. Например, можно создать скалярную структуру из двух полей:

>>S.name='mpan'; S.order =2

S =

name: "mpan"

order: 2

Расширение структуры производится по тому же правилу, что и добавление строк или столбцов в массив:

>>S(2).name='Симп'; S(2).order=4;

Для пополнения структуры можно также использовать специальную команду struct. Организуем третий элемент структуры S, присвоив полю name значение прям, а полю order - значение 2.

>>S(3)=struct("name", 'прям', 'order', 2)

S =

1x3 struct array with fields:

name

order

Чтобы вывести содержимое отдельных полей структуры, нужно использовать фигурные либо квадратные скобки в зависимости от того, символьные или числовые данные связаны с данным полем:

>>{S.name}

ans =

"mpan" "Симп" "прям"

>>[S.order]

ans =

2 4 2

Преобразование данных в массив осуществляется по команде char:

>>char(S.name)

ans =

трап
Симп
Прям

1.4.Управление потоками

MATLAB имеет пять видов структур управления потоками:

- оператор **if**;
- оператор **switch**;
- оператор **for**;
- циклы **while**;
- оператор **break**.

IF

Оператор ***if*** вычисляет логическое выражение и выполняет группу операторов, если выражение истинно. Необязательные ключевые слова elseif и else служат для выполнения альтернативных групп операторов. Ключевое слово end, которое согласуется с if, завершает последнюю группу операторов. Таким образом, все группы операторов заключены между четырех ключевых слов, без использования фигурных или обычных скобок.

Алгоритм MATLAB для создания магического квадрата порядка n включает три разных случая: n - нечетное; n - четное, но не делится на 4; n - четное и делится на 4. Приведем пример соответствующего кода:

```
if rem(n,2)~=0
    M=odd_magic(n)
elseif rem(n,4)~=0
    M=single_even_magic(n)
else
    M=double_even_magic(n)
end
```

(в этом примере три случая являются взаимно исключающими, но если бы это было не так, то выполнялось бы первое истинное условие)

Важно понять, как оператор отношения и оператор if работают с матрицами.

Когда Вы хотите узнать, равны ли две переменные, нужно использовать конструкцию ***if A==B, …***- A и B являются скалярами. Но когда A и B - матрицы, A==B не работает, если они не равны. Равенство матриц означает поэлементное равенство. Фактически, если A и B имеют различные размеры, MATLAB выдаст ошибку.

Правильный способ определения равенства между двумя переменными - это использование функции isequal***: if isequal(A,B), …***

Приведем пример, который демонстрирует особенность работы с A и B, если они являются скалярами:

```
If A > B
    'max'
elseif A < B
```

```
        'min'
elseif A= = B
        'equal'
else
        error ('непредвиденная ситуация')
end
```

SWITCH и CASE

Оператор switch выполняет группу операторов, базируясь на значении переменной или выражения. Ключевые слова case и otherwise разделяют эти группы. Выполняется только первый соответствующий случай. Необходимо использовать end для согласования с switch.

Оператор switch в MATLAB "не проваливается": если первый случай является истинным, другие случаи не выполняются, т.о. нет необходимости использовать оператор break.

FOR

Цикл ***for*** повторяет группу операторов предопределенное число раз. Ключевое слово end очерчивает тело цикла.

```
for n=3:32
        r(n) = rank(magic(n));
end
r
```

Точка с запятой после выражения в теле цикла предотвращает повторения вывода результатов на экран, а r после цикла выводит окончательный результат.

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

WHILE

Цикл ***while*** повторяет группу операторов определенное число раз, пока выполняется логическое условие. Ключевое слово end очерчивает используемые операторы.

Приведем программу, иллюстрирующую работу операторов while, if, else, end, которая использует метод деления отрезка пополам для нахождения нулей полинома.

```
a = 0; fa = -inf;
b = 3; fb = inf;
while b - a > eps*b
        x = (a+b)/2;
        fx = x^3-2*x-5;
        if sign(fx) == sign(fa)
```

```
        a = x; fa = fx;
    else
        b = x; fb = fx;
    end
end
x
```

Результатом будет корень полинома X^3 - 2X - 5

```
x =
    2.09455148154233
```

Для оператора while верны замечания относительно матричного сравнения, описанного выше.

BREAK

Оператор break позволяет досрочно выходить из циклов for или while. Во вложенных циклах break осуществляет выход только из самого внутреннего цикла.

Приведем улучшенный вариант предыдущего примера:

```
a = 0; fa = -inf;
b = 3; fb = inf;
while b - a > eps*b
    x = (a+b)/2;
    fx = x^3-2*x-5;
    if fx == 0
        break
    elseif sign(fx) == sign(fa)
        a = x; fa = fx;
    else
        b = x; fb = fx;
    end
end
x
```

1.5.Сценарии и функции

MATLAB - это мощный язык программирования, также как и интерактивная вычислительная среда. Файлы, которые содержат код на языке MATLAB, называются М-файлами. Вы можете создать М-файлы, используя любой текстовый редактор, а затем используете их как любую функцию или команду MATLAB.

Существует два вида М-файлов:

- сценарии, которые не имеют входных и выходных аргументов; они оперируют с данными рабочего пространства;
- функции, которые имеют входные и выходные аргументы; они оперируют с локальными переменными.

Чтобы увидеть содержание М-файла, например, myfunction.m, необходимо набрать type myfunction.m.

Сценарии

Когда Вы вызываете сценарий, MATLAB просто вызывает команды, содержащиеся в файле. Сценарии могут оперировать существующими данными в рабочем пространстве или они могут сами создавать эти данные. Хотя сценарии не возвращают значений, все переменные, которые они создают, остаются в рабочем пространстве для использования в последующих вычислениях. Сценарии могут осуществлять графический вывод, используя такие функции как plot.

Создадим файл magicrank.m, который содержит эти команды MATLAB:

```
% Investigate the rank of magic squares
r = zeros (1,32);
for n = 3:32
      r(n) = rank(magic(n));
end
r
bar(r)
```

Ввод строки magicrank.m повлечет за собой исполнение команд, вычисление ранга первых 30 магических квадратов и отображения столбиковой диаграммы результатов. После полного выполнения файла переменные n и r остаются в рабочем пространстве.

Функции

Функции - это М-файлы, которые могут иметь входные и выходные параметры. Имя М-файла и функции должно быть одним и тем же. Функции работают с переменными в пределах их собственного рабочего пространства, отделенного от рабочего пространства, с которым Вы оперируете в командной строке.

Хорошим примером является функция rank. М-файл rank.m находится в директории **C:\MATLAB\toolbox\matlab\matfun**

Вы можете просмотреть его содержимое, введя type rank.m

```
function r = rank(A,tol)
%RANK   Matrix rank.
%   RANK(A) provides an estimate of the number of linearly
%   independent rows or columns of a matrix A.
%   RANK(A,tol) is the number of singular values of A
%   that are larger than tol.
%   RANK(A) uses the default tol = max(size(A)) * norm(A) * eps.

%   Copyright (c) 1984-98 by The MathWorks, Inc.
%   $Revision: 5.7 $  $Date: 1997/11/21 23:38:49 $

s = svd(A);
if nargin==1
```

```
   tol = max(size(A)') * max(s) * eps;
end
r = sum(s > tol);
```

Первая строка функции начинается со слова function. Здесь происходит задание имени со списком аргументов. В нашем случае: до двух входных аргументов и один выходной.

Следующие несколько строк, до первой пустой или выполняемой строки, являются комментариями, которые предоставляют справочную информацию. Эти строки будут выведены на экран, если Вы наберете help rank

Первая строка справочного текста - это H1 строка, которую MATLAB отображает при использовании команды lookfor или при запросе help по всей директории. Остальное содержание файла составляет исполняемый код MATLAB. Переменная s, представленная в теле функции, также как и переменные в первой строке, r, A и tol, все являются локальными. Они отделены от других переменных в рабочем пространстве MATLAB.

Этот пример показывает важную особенность функций MATLAB, которая обычно не встречается в других языках программирования, - переменное число аргументов. Функция rank может быть использована в нескольких различных формах:

rank (A)
r = rank(A)
r = rank (A, 1.e-6)

Многие функции MATLAB работают таким образом. Если нет выходного аргумента, то результат сохраняется в переменной ans. Если нет второго входного аргумента, то функция вычисляет значение по умолчанию. Внутри тела функции присутствуют две величины nargin и nargout, которые выдают число входных и выходных аргументов при каждом использовании функции. Функция rank использует переменную nargin, но не использует nargout.

1.6. Рабочее пространство

Среда Matlab включает в себя как совокупность переменных, созданных за время работы Matlab, так и набор файлов, содержащих программы и данные, которые продолжают существовать между сеансами работы.

Рабочее пространство - это область памяти, доступная из командной строки Matlab.

Две команды, ***who*** и ***whos***, показывают текущее состояние рабочего пространства. Команда выдает краткий список, а команда размер и используемую память.

Для удаления всех существующих переменных из рабочего пространства MATLAB, введите >> **clear**.

Команда save

Команда ***save*** сохраняет содержание рабочего пространства в MAT-файле, который может быть прочитан командой load в последующих сеансах работы MATLAB. Например, ***save Nov22th*** сохраняет содержание всего рабочего пространства в файле Nov22th.mat.

Если нужно, Вы можете сохранить только определенные переменные, указывая их имена после имени файла. Обычно, переменные сохраняются в двоичном формате, который может быть быстро (и точно) прочитан MATLAB.

Если Вы хотите использовать эти файлы вне MATLAB, Вы можете указать любой другой формат:

- ***-ascii*** использует 8-значный текстовый формат
- ***-ascii -double*** использует 16-значный текстовый формат
- ***-ascii -double -tabs*** разделяет элементы массива табуляцией
- ***-append*** добавляет данные в существующий MAT-файл.

(!) Когда Вы сохраняете содержание рабочего пространства в текстовом файле, Вы должны сохранять только одну переменную в данный момент. Если Вы сохраняете более одной переменной, MATLAB создает текстовый файл, но Вы не сможете загрузить его обратно.

Маршрут поиска

MATLAB использует маршрут поиска, упорядоченный список директорий, для того, чтобы определить как выполнять функции, которые Вы вызываете. Когда Вы вызываете стандартную функцию, MATLAB исполняет M-файл на своем пути, который имеет заданное имя. Вы можете заменить поведение использованием специальных директорий и поддиректорий.

Команда ***path*** показывает маршрут поиска на всех платформах.

На PC и MAC выберите опцию Set Path… (Установить маршрут…) из меню File (Файл) для просмотра и изменения маршрута.

Операции над дисковыми файлами

Команды ***dir, type, delete*** и ***cd*** осуществляют комплекс групповых операционных системных команд для манипуляций над файлами. Нижеприведенная таблица показывает, как эти команды соответствуют операционной системе MS DOS

MATLAB	MS-DOS
dir	dir
type	Type
delete	del, erase
cd	Chdir

Для большинства из этих команд Вы можете использовать полные пути, шаблоны и указатели дисков в обычной форме.

2. Задания для самостоятельного выполнения лабораторной работы №2.

Задание 1:

1.7.Составьте таблицу значений Tabl_1 функции $Y = 2*x^3 + 4*\sin(0.5*|x|)$ на интервале [0; 2.0] с шагом 0,1.

1.8.Сохраните полученную таблицу в файле Arr_1 в формате MATLAB (mat-файл) и в файле Arr_2 в восьмизначном текстовом формате.

1.9.Очистите рабочее пространство MATLAB, установите представление чисел в виде рационального числа и загрузите последовательно созданные Вами файлы (зад. 1.2).

1.10.*Влияет ли текущий числовой формат рабочего пространства на форму представления данных, прочитанных из внешнего файла?

Задание.2 :

2.1. Создайте m-файл, содержащий функцию вычисления среднего арифметического элементов заданной матрицы S_fun (в качестве входных параметров определите: а) размерность обрабатываемой матрицы и ее имя; б) только имя матрицы).

2.2. Сохраните его в собственной папке.

Задание 3:

Исследовать систему уравнений и решить её, если она совместна.

Совместность системы определяется согласно теореме Кронекера-Капелли: *Для совместности системы линейных уравнений необходимо и достаточно, чтобы ранг матрицы системы равнялся рангу расширенной матрицы этой системы.*

3.1. $\begin{cases} 2x_1 + x_2 - x_3 = 1 \\ x_1 - 3x_2 + 4x_3 = 2 \\ 11x_1 - 12x_2 + 17x_3 = 3 \end{cases}$ 3.2. $\begin{cases} x_1 + x_2 + x_3 = 3 \\ 2x_1 - x_2 + x_3 = 3 \\ x_1 - x_2 + 2x_3 = 5 \\ 3x_1 - 6x_2 + 5x_3 = 6 \end{cases}$ 3.3. $\begin{cases} x_1 + 2x_2 + 3x_3 + 4x_4 = 5 \\ x_1 - x_2 + x_3 - x_4 = 1 \end{cases}$

3.4. Сохраните полученные решения в текстовом файле.

Задание 4:

4.1.* Создайте m-файл, содержащий функцию генерации матрицы произвольной размерности, содержащей случайные целые числа в диапазоне от 0 до 10, и функцию вычисления определителя сгенерированной матрицы. Размерность матрицы и ее имя определяется в командной строке MATLAB при обращении к функции генерации (допустимо использование встроенных функций MATLAB в программном коде Вашего m-файла). Рассмотрите возможные альтернативы решения поставленной задачи. Оцените достоинства и недостатки каждой альтернативы и обоснуйте реализованный Вами алгоритм решения.

4.2.* Создайте m-файл, содержащий функцию Kramer, использование которой позволяет решить систему линейных уравнений методом Крамера. Созданная Вами функция должна быть применима только при соблюдении условий: 1) число уравнений системы равно числу неизвестных; 2) число уравнений - не более 10, в противном случае выдается сообщение об ошибке.

Задание 5:

5.4 После завершения работы (после выполнения заданий 2.1-2.4) отобразите на экране содержимое рабочего пространства MATLAB: наберите в

командной строке >>whos (данная команда осуществляет вывод на экран текущее состояние рабочего пространства: имена переменных, размер и используемую память). Перепишите содержимое рабочей области в лист отчета по лабораторной работе (или в файл с именем “Отчет_<№ работы> ”). Предоставьте преподавателю отчет и созданные m-файлы.

5.5 Выполните очистку рабочего пространства (команда clear);

5.6 Завершите работу с MATLAB.

Приложение 1

МАТРИЧНЫЕ ВЫЧИСЛЕНИЯ

Команды работы с данными

Имя	Назначение
max	Определение максимальных элементов массива
min	Определение минимальных элементов массива
sum	Суммирование элементов массива
cumsum	Суммирование элементов массива с накоплением
prod	Произведение элементов массива
cumprod	Произведение элементов массива с накоплением
median	Определение медиан (срединных значений)
mean	Определение средних значений массива

Команды сортировки

Имя	Назначение
sort (X,N)	Сортировка по возрастанию (упорядочивание по модулю) элементов массива X по размерности N
sortrows (X,N)	Сортировка строк с упорядочиванием по элементам столбца с номером N
cplxpair (X, TOL, N)	Сортировка комплексно-сопряженных пар (упорядочивание по вещественной части) с точностью TOL по размерности N

Функции от матриц

Имя	Описание
expm	Матричная экспонента
sqrtm	Квадратный корень из матрицы
logm	Логарифм от матрицы
funm	Функция от матрицы

Матричные характеристики

Имя	Описание
det	Вычисление определителя
trace	Вычисление следа матрицы
cond	Вычисление числа обусловленности
condest	Оценка числа обусловленности
rcond	Параметр обусловленности
rank	Определение ранга матрицы

norm	Вычисление нормы матрицы (вектора)
normest	Оценка нормы матрицы (вектора)

Команды работы с данными

Имя	Описание
inv	Вычисление обратной матрицы
pinv	Вычисление псевдообратной матрицы
null	Определение ядра (нуль-пространства) матрицы
orth	Вычисление ортонормального базиса

Лабораторная работа №3

Тема: Графика в MATLAB

MATLAB имеет широкие возможности для графического изображения векторов и матриц, а также для создания комментариев и печати графики.

1. Теоретическая часть

1.1. Двумерная графика

Создание графика

Функция **plot** имеет различные формы, связанные с входными параметрами.

Например, 1) plot(y) создает кусочно-линейный график зависимости элементов y от их индексов; 2) если Вы задаете два вектора в качестве аргументов, plot(x,y) создаст график зависимости y от x.

Например, для построения графика значений функции sin от нуля до 2π, сделаем следующее:

```
t = 0:pi/100:2*pi;
y = sin(t);
plot(t,y)
```

Вызов функции plot с многочисленными парами x-y создает многочисленные графики. MATLAB автоматически присваивает каждому графику свой цвет, что позволяет различать заданные наборы данных.

Например, следующие три строки отображают график близких функций, и каждой кривой соответствует свой цвет:

```
y2 = sin(t  - .25);
y3 = sin(t - .5);
plot (t, y, t, y2, t, y3)
```

Возможно изменение цвета, стиля линий и маркеров, таких как знаки плюс или кружки, следующим образом:

plot(x, y, 'цвет_стиль_маркер')

цвет_стиль_маркер - это 1-, 2-, 3-символьная строка, составленная из типов цвета, стиля линий и маркеров:

- символы, относящиеся к цвету: 'c' - голубой, 'm' - малиновый, 'y' - желтый, 'r' - красный, ' g' - зеленый, 'b' - синий, 'w' - белый, 'k' - черный;

- символы, относящиеся к типу линий: ' - ' для сплошной, ' - - ' для разрывной, ' : ' для пунктирной, ' -. ' для штрихпунктирной, 'none' для её отсутсвия;

- наиболее часто встречающиеся маркеры: ' + ', ' o ', ' * ', ' x '.

Функция plot автоматически открывает новое окно изображение, если до этого его не было на экране.

Если оно существует, то plot использует его по умолчанию.

Для открытия нового окна и выбора его по умолчанию, необходимо указать:

figure

Для того, чтобы сделать существующее окно текущим -figure(n), где n - это номер заголовка окна. В этом случае результаты всех последующих команд будут выводится в это окно.

Функция **subplot** позволяет выводить множество графиков в одном окне.

subplot(m, n, p) - разбивает окно изображений на m матрицу на n подграфиков и выбирает p-ый график текущим. Графики нумеруются вдоль первого в верхней строке, потом во второй и т.д.

Например, для того, чтобы представить графические данные в четырех разных подобластях окна необходимо выполнить следующее:

```
t  = 0 : pi / 10 : 2 * pi;
[x, y, z]  = cylinder(4*cos(t));
subplot(2,2,1)
mesh(x)
subplot(2,2,2); mesh(y)
subplot(2,2,3); mesh(z)
subplot(2,2,4); mesh(x, y, z)
```

Управление осями

Функция axis имеет несколько возможностей для настройки масштаба, ориентации и коэффициента сжатия.

Обычно находит максимальное и минимальное значение и выбирает соответствующий масштаб и маркирование осей. Функция axis заменяет значения по умолчанию предельными значениями, вводимыми пользователем:

axis([xmin xmax ymin ymax])

В функции axis можно также использовать ключевые слова для управления внешним видом осей.

Например, axis square - создает оси равной длины, а axis equal - создает отдельные отметки приращений для x и y осей одинаковой длины.

Так, функция plot(exp(i*t)) следующая за axis square, либо за axis equal превращает овал в правильный круг.

axis auto возвращает значения по умолчанию и переходит в автоматический режим.

axis on включает обозначения осей и метки промежуточных делений.

axis off выключает обозначение осей и метки промежуточных делений.
grid off выключает сетку координат, а grid on включает ее заново.

Подписи к осям и заголовки

Фукнции ***xlabel, ylabel, zlabel*** добавляют подписи к соответствующим осям, функция добавляет заголовок в верхнюю часть окна, а функция вставляет текст в любое место графика. Использование TEXT представления позволяет применять греческие буквы, математические символы и различные шрифты. Следующий пример демонстрирует эту возможность.

```
t = -pi:pi/100:pi;
y = sin(t);
plot(t,y)
axis([-pi pi -1 1])
xlabel( ' -\pi \leg \itt \leg  \pi ')
ylabel ( ' sin(t) ')
title( 'График функции sin ' )
text (-1, -1/3, ' \it {Отметьте нечетную симметрию}' )
```

MATLAB определяет поверхность как z координаты точек над координатной сеткой плоскости x-y, используя прямые линии для соединения соседних точек.

Функции mesh и surface отображают поверхность в трех измерениях. При этом mesh создает каркасную поверхность, где цветные линии соединяют только заданные точки, а функция surface вместе с линиями отображает в цвете и саму поверхность.

Визуализация функций двух переменных

Для отображения функции двух переменных, z = f (x, y) создаются матрицы X и Y, состоящие из повторяющихся строк и столбцов соответственно, перед использованием функции. Затем используют эти матрицы для вычисления и отображения функции. Функция meshgrid преобразует область определения, заданную через один вектор или два вектора x и y, в матрицы X и Y для использования при вычислении функций двух переменных. Строки матрицы X дублируют вектор x, а столбцы Y - вектор y.

Для вычисления двумерной функции sinc, sin(r)/r, в области x-y поступают следующим образом

```
[X, Y] = meshgrid ( - 8:.5:8);
R = sqrt(X.^2+Y.^2) + eps;
Z = sin(R)./R;
mesh(X, Y, Z)
```

В этом примере R - это расстояние от начала координат, которому соответствует центр. Добавление eps позволяет избежать неопределенности 0/0 в начале координат.

1.2. Трехмерная графика

Команда **plot3** для построения кривых в трехмерном пространстве похожа на рассмотренную команду двухмерной графики plot. Кривая строится в перспективной проекции по трем векторам одинаковой размерности x, y и z:

plot3 (x, y, z)

Можно нарисовать кривую при помощи массива xyz с тремя столбцами, где каждый столбец хранит данные о своей координате: plot(xyz).

Если требуется нарисовать несколько прямых, то возможны следующие решения:

- приготовим три двухмерных массива X, Y, Z, каждый из которых содержит данные об одной координате. Столбцы этих массивов отвечают разным кривым, то есть первая кривая задается первыми столбцами массивов X, Y, Z, вторая - вторыми столбцами и т.д. В этом случае обращение к команде plot3 имеет вид: plot3 (X, Y, Z) ;
- второй способ заключается в последовательном перечислении векторов, содержащих координаты кривых:

plot3(x1, y1, z1, s1, x2, y2, z2, s2, …..)

Здесь сочетаниями s1, s2, …. Обозначены маркировки кривых (тип линии, маркер, цвет), например, строка 'y -.' задает построение пунктирной кривой желтого цвета и точкой в качестве маркера.

Команды оформления axis, title, xlabel и другие действуют аналогично командам, рассмотренным для двухмерной графики.

Построение поверхностей

Чтобы построить поверхность, нужно иметь массив значений функции нескольких переменных, вычисленный по некоторой сетке. Для формирования двумерной прямоугольной и трехмерной параллелепипедальной сеток используется команда ***meshgrid***, а для функций с большим числом переменных - команда ***ndgrid***.

Пусть x, y - одномерные массивы точек, задающие абсциссы и ординаты соответственно. В результате выполнения команды

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

будут сформированы два двумерных массива X и Y, причем все строки массива X будут копиями вектора x , а столбцы Y - копиями вектора y. Теперь с помощью этих массивов можно вычислить массив значений функции двух переменных и затем нарисовать поверхность.

В MATLAB для изображения поверхности имеются команды, перечисленные в таблице 3.1.

Таблица 3.1. Команды построения поверхности

Команда	Назначение

mesh	Построение сетчатой поверхности
meshc	Построение сетчатой поверхности с линиями уровня
meshz	Построение сетчатой поверхности и отсчетной плоскости
surf	Построение расцвеченной поверхности
surfc	Построение расцвеченной поверхности с линиями уровня
surfl	Построение расцвеченной поверхности с подсветкой
waterfall	Трехмерная поверхность без прорисовки ребер

Команда ***mesh*** строит в графическом окне расцвеченную сетчатую поверхность, используя различную окраску вершин и ребер. Чтобы ребра были окрашены в цвета прилегающих вершин (в процессе обхода вершин цвет ребра меняется), нужно выполнить команду ***shading flat***, а для плавного изменения цвета между вершинами (линейная интерполяция) надо запустить команду - ***shading interp.***

Приведем пример построения сетчатой поверхности (см. рис.5.1).

>> x = 0:.2:8:y = 0:.2:4; [X,Y] = meshgrid(x,y);
Z = 2 * cos (X + Y) + Y.* cos (X - Y); mesh(X, Y, Z)

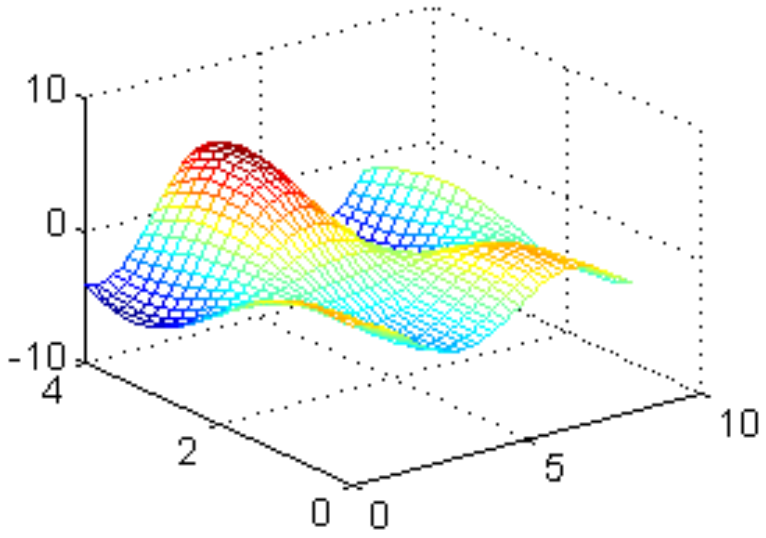

Рисунок 3.1. Построение поверхности по команде mesh

Та же сетчатая поверхность будет построена при помощи команды mesh(x, y, Z).

Имеется несколько разновидностей команды mesh. На рисунке 3.2 нарисована поверхность -Z с линиями уровня (команда meshc).

Для этого в командной строке MATLAB следует дать команду:

>> subplot(1, 2, 1), meshc(x, y, -Z), shading interp

Построение той же поверхности с отсчетной плоскостью (команда meshz):

subplot(1, 2, 2), meshz(x, y, -Z), shading flat

Ряд команд для построения поверхностей использует затенение.

Например, команда

>> ***subplot(1,3,1), surf(x,y,Z),*** дает затененную сеточную поверхность;

>> ***subplot(1,3,2), surfc(x,y,Z),***дает затененную сеточную поверхность с линиями уровня;

>> ***subplot(1,3,3), surfl(x,y,Z),*** дает поверхность с подсветкой справа.

После построения рисунков для каждого подокна следует выполнить масштабирование при помощи команды

>> ***axis ([-Inf Inf -Inf Inf -Inf Inf])***

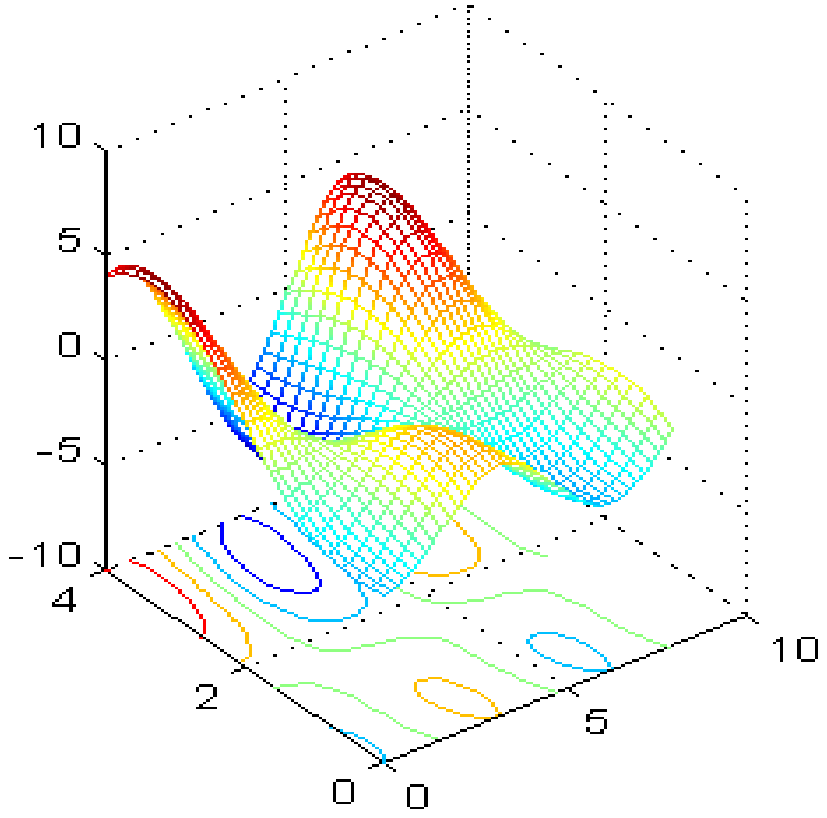

Рисунок 3.2 - Построение поверхности командой meshc

Это позволит определить действительные интервалы изменения величин по всем координатам и сделать рисунки выразительнее.

Когда поверхность построена, то для получения нужного ракурса можно воспользоваться командой ***view***, устанавливающей угол зрения наблюдателя по отношению к осям:

view (AZ, EL) или view([AZ, EL])

Здесь AZ - угол вращения в горизонтальной плоскости (азимут), EL - угол возвышения. Угол зрения можно задать также посредством декартовых координат:

view([x, y, z])

причем важны отношения величин, а не абсолютные значения. Команда view без параметров выведет матрицу поворота размера 4x4.

Имеется возможность выбрать ракурс интерактивно средствами графического окна. Нужно установить режим вращения изображения при помощи пункта Rotate 3D меню Tools или соответствующего значка, а затем, перемещая мышь при нажатой клавише, задать расположение осей. При этом в левом углу окна будут отображаться значения углов AZ и EL.

Приведем пример использования команды view и установки нужного ракурса для построенной командой waterfall поверхности без прорисовки поперечных ребер.

>> x=1:.1:4; y=0:.02:1; [X, Y]=meshgrid(x, y);

Z=.2*X+(1-Y).*sin(pi*X.*Y);

waterfall(X', Y', Z'), axis equal, view(-140, 30), grid

В этом примере установлен единый масштаб по координатам x и y и использована графическая команда grid, чтобы убрать нанесение сетки.

Функция ndgrid формирует массивы для вычисления и интерполяции функций нескольких переменных. Подобно команде meshgrid, на вход подается несколько одномерных векторов x1, x2, ..., а на выходе имеем многомерные массивы X1, X2, ..., причем k-я размерность массива Xk есть копия вектора xk. Рассмотрим использование команд ndgrid и slice для построения сечений функции трех переменных.

>> x1=-2:.2:2; x2=-2:.2:2; x3=0:.05:1;

[X1, X2, X3]=ndgrid(x1, x2, x3); V=(X3-.3).*sin(X2+X1);

slice(x1, x2, x3, V, [-.8 .6 2], 2, [0 .5])

xlabel("x1"), ylabel("x2"), zlabel("x3")

Для оформления трехмерной графики используются команды, аналогичные командам двумерной графики.

2. Задания для самостоятельного выполнения лабораторной работы №3.

Задание 1:

1.11. Постройте в одном окне изображений графики функций от n до m с шагом k: а) *sin, sinh, cos, cosh;* б) *tan, tanh, cot, coth;* в) *sec, csc*. Значения n, m и k определите самостоятельно.

Выполните подписи по осям и задайте заголовок для графика.

1.12. Выполните задание 1.1, построив графики функций в одном окне (если это возможно) с использованием различных цветов для отображения графиков и маркеров.

1.13. Сохраните полученный результат в файле MATLAB.

1.14. *Каковы особенности хранения изображений в MATLAB?

Задание 2:

2.1. Сформируйте произвольные одномерные массивы точек a и b, задающие абсциссы и ординаты соответственно, на их основе сформируйте два

двумерных массива A и B, вычислите массив значений функции двух переменных и нарисуйте поверхность.

2.2. * Найти графически начальное приближение для решения системы методом простой итерации

$$\begin{cases} f_1(x,y) = 2*x - \sin(0.5*(x-y)) = 0 \\ f_2(x,y) = 2*y - \cos(0.5*(x+y)) = 0 \end{cases}$$

Задание 3:

3.1.* Изучите самостоятельно и покажите на примерах использование цветовой гаммы для оформления рисунка (формирование цветов, подсветка).

3.2.* Изучите самостоятельно и покажите на примерах возможности специализированной графики MATLAB (построение столбцовой диаграммы, построение круговой диаграммы, рисование гистограммы).

Задание 4:

4.1 После завершения работы (после выполнения заданий 1-3) отобразите на экране содержимое рабочего пространства MATLAB: наберите в командной строке >>whos (данная команда осуществляет вывод на экран текущее состояние рабочего пространства: имена переменных, размер и используемую память). Перепишите содержимое рабочей области в лист отчета по лабораторной работе (или в файл с именем “Отчет_<№ работы> ”). Предоставьте преподавателю отчет и созданные m-файлы.

4.2 Выполните очистку рабочего пространства (команда clear);

4.3 Завершите работу с MATLAB.

Лабораторная работа №4

Тема: *Simulink и Stateflow*

В настоящее время существует три различных подхода к моделированию сложных систем, каждый из которых поддерживается соответствующей группой пакетов:

а) Simulink и Stateflow;

б) Omsim и Omola, Dymola и Modelica;

в) Model Vision Studium.

ЧАСТЬ I

1. Теоретическая часть

1.1. Подсистема Simulink пакета MatLab. Основные свойства подсистемы Simulink.

Подсистема Simulink - это интерактивная среда для моделирования и анализа широкого класса динамических систем, использующая графический язык блок-диаграмм.

Подсистема Simulink:

- предоставляет возможность моделирования непрерывных, дискретных и гибридных - как линейных, так и нелинейных - систем;
- включает в себя обширную библиотеку блоков (непрерывные элементы, дискретные элементы, математические функции, нелинейные элементы, источники сигналов, средства отображения, дополнительные блоки), которые можно использовать для создания новых систем;
- позволяет объединять блок-диаграммы в составные блоки, что обеспечивает иерархическое представление структуры модели;
- содержит средства для создания блоков и библиотек, определяемых пользователем;
- дает возможность проектировать подсистемы, имеющие изменяемую во времени структуру, но эти возможности весьма ограничены.

Начиная с версии 3.0, в Simulink появились специализированные приложения, значительно расширившие ее возможности, в частности:

- подсистема Stateflow - дает возможность моделировать поведение гибридных или сложных событийно-управляемых систем, базируясь на картах состояния Харела. Уже созданные пользователями пакета Simulink модели рассматриваются как объекты, закон управления которыми реализуется в Stateflow (смотри приложение в работе №4 “Определение гибридного автомата”);
- подсистема Stateflow Coder предназначена для генерации С-кода при реализации диаграмм Stateflow. Применяя Stateflow и Stateflow Coder, пользователь может генерировать нужный код на алгоритмическом языке С только для управляющих моделью блоков, реализованных с помощью Stateflow;
- подсистема Real-Time Workshop дополняет Simulink и Stateflow Coder, обеспечивая автоматическую генерацию кода С для моделей, разработанных в Simulink;
- подсистема Simulink Report Generator позволяет создавать и настраивать отчеты из моделей Simulink и Stateflow в различных форматах, среди которых HTML, RTF, XML и SGML.

Мы рассмотрим только подсистему Simulink и ее самое существенное дополнение - Stateflow.

Этапы построения модели в подсистеме Simulink

Перед построением модели необходимо предварительно загрузить систему Matlab и запустить подсистему Simulink.

Запуск подсистемы Simulink выполняется из основного окна Matlab. Для этого необходимо либо щелкнуть по кнопке запуска этой подсистемы, находящейся в верхней части окна, либо набрать в командной строке ***Simulink***.

В том и в другом случае откроется окно ***Library: simulink*** и окно ***untitled*** для проектирования новой модели.

Процесс построения модели Simulink включает в себя компоновку и задание необходимых параметров. Компоновка заключается в выборе из библиотек Simulink необходимых блоков, размещение их в открывшемся окне и задание межблочных связей. Далее для каждого блока устанавливаются соответствующие параметры, отвечающие требованиям моделируемой системы. Для того чтобы построить модель Simulink необходимо знать, какие типы блоков предоставляются пользователю.

Библиотеки блоков Simulink

Основным "строительным" элементом в процессе построения модели в пакете Simulink является блок. Блок представляет собой систему типа "вход-выход-состояние" (или просто "вход-выход") и может быть как простым, так и составным.

Пользуясь терминологией, аналогичной используемой в объектно-ориентированных языках, можно сказать, что каждый блок из любой библиотеки блоков пакета Simulink является классом. Как только пользователь переносит блок из библиотеки в окно построения модели, он тем самым создает экземпляр данного класса. В этом экземпляре он может изменять значение параметров блока в зависимости от требований, предъявляемых к моделируемой системе.

С помощью связей экземпляры блоков объединяются в единую систему, которая создается в окне построения модели или может быть собрана сразу в составном устройстве-контейнере, экземпляр которого предварительно помещается в окно.

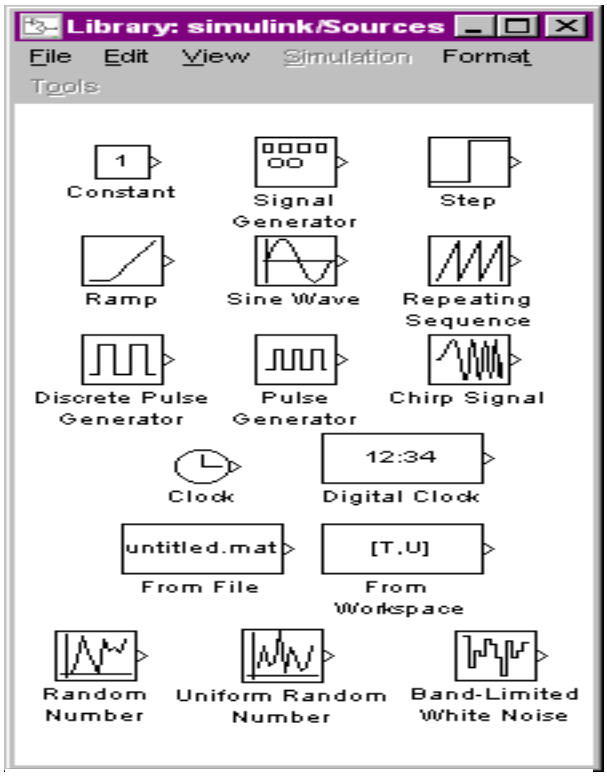

Рис.1. Содержимое раздела Sources

Процедура поиска и перемещения блоков из библиотек Simulink к окно модели во многом напоминает операции копирования и перемещения файлов в среде Windows. В частности, технология работы в ***Library: simulink*** аналогична работе с Проводником Windows. Для того чтобы переместить необходимый блок из библиотеки в окно построения модели, необходимо найти его в списке стандартных блоков Simulink. Для этого в окне ***Library: simulink*** выбрать один из пунктов (Sources, Sinks, Discrete, Linear, Nonlinear, Connection), затем выделить и раскрыть соответствующий пункт. Например, при выборе в окне ***Library: simulink*** пункта ***Sources*** откроется окно, изображенное на рис.1. Для перемещения курсор мыши устанавливается на нужный блок; затем, нажав левую кнопку мыши, блок перемещают в окно модели.

После того как блок появился в окне модели, можно установить для него соответствующие параметры. Для этого достаточно сделать двойной щелчок левой кнопкой мыши по пиктограмме блока и установить необходимые параметры.

Прежде чем перейти к рассмотрению библиотек блоков, содержащихся в Simulink, попытаемся получить самое общее представление о S-модели. Для этого воспользуемся одним из примеров, включенных авторами Matlab в раздел Simple Models.

S-модель: общее представление

Откройте окно демонстрации Matlab. Примеры из раздела Simulink помогут уяснить основные концепции Simulink.

В списке разделов выберите раздел Simulink и его подраздел Простые модели, в списке примеров Simulink выделите модель Tracking a bouncing ball (Трассировка прыгающего шара) и щелкните по кнопке Запуск Трассировка....

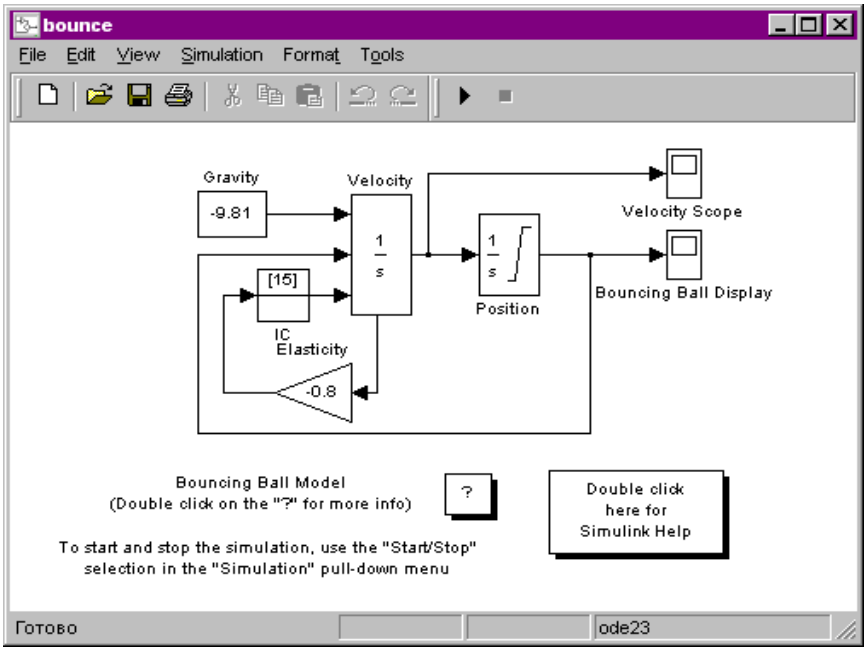

Рис.2. Блок-диаграмма модели “Трассировка прыгающего шара”

На экране появятся два окна: одно из них - bounce - содержит блок-диаграмму модели с комментариями (рис.2), другое - Bouncing Ball Display -

представляет собой пример одного из “смотровых окон”, обеспечивающих наблюдение за поведением моделируемой системы. Это окно по виду напоминает экран электронного измерительного прибора (рис.3). До начала моделирования на нем ничего нет, кроме измерительной шкалы и кнопок панели инструментов. С точки зрения структуры S-модели, такое “смотровое окно” - один из блоков диаграммы, параметрами которого можно управлять.

Блок-диаграмма модели (рис.2) представляет собой набор блоков, соединенных между собой линиями связи. Направление движения информационных и управляющих сигналов на диаграмме обозначено стрелками. Любая линия связи может иметь произвольное число ответвлений, начало каждого из которых обозна-чается точкой. Число входов и выходов блока определяется его типом и значениями параметров настройки блока. Для обеспечения наглядности диаграммы входящие в нее блоки не только различаются графическим представлением, но и снабжаются (при необходимости) индивидуальными именами, которые выбираются пользователем. На выбор имени не накладывается никаких ограничений, оно может представлять собой целую фразу.

Все блоки, входящие в блок-диаграмму, можно условно разделить на три группы: функциональные, смотровые и информационные.

В состав рассматриваемой S-модели входят два “смотровых окна”. Одно из них (Bouncing Ball Display) открывается автоматически при открытии файла модели. Второе “смотровое окно” представлено на диаграмме блоком Velocity Scope (индикатор скорости). Чтобы его открыть, нужно дважды щелкнуть на блоке. До начала моделирования это окно, так же, как и первое, пусто.

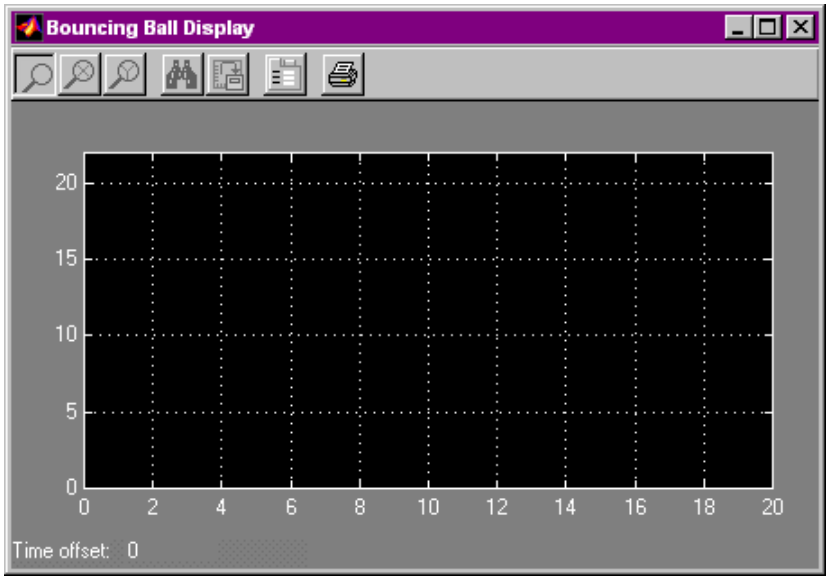

Рисунок 3 - “Смотровое окно” блока

Информационных блоков, т.е. блоков, предоставляющих пользователю дополнительную информацию о модели, также два. Первый из них помечен символом “?”, а второй имеет развернутую метку - Double click here for Simulink Help (для получения справки дважды щелкните мышью). Выполнив эту рекомендацию для каждого из информационных блоков, можно увидеть их

содержимое. Технология создания таких информационных блоков будет рассмотрена нами позже. Отметим лишь, что первый из них создан на основе механизма подсистем, а второй - с помощью инструмента компоновки интерфейса.

Остальные блоки модели являются функциональными. Двойной щелчок на любом из них приводит к открытию окна настройки параметров блока.

Попытаемся провести модельный эксперимент с рассматриваемой S-моделью.

Предварительно расположите все три окна модели (два “смотровых окна”: Bouncing Ball Display, Velocity Scope; блок-диаграмму bounce) на экране так, чтобы они не перекрывали друг друга. Сделайте активным окно с блок-диаграммой и запустите модель щелчком по кнопке Start, расположенной на панели инструментов окна. После начала моделирования траектория движения прыгающего мяча отображается в “смотровом окне” Bouncing Ball Display, а изменение его скорости - в окне Velocity Scope.

Модель заканчивает свою работу, когда мяч останавливается (его скорость становится нулевой). В рассматриваемом примере условием окончания моделирования является наступление определенного события - остановка мяча. Но условием окончания модельного эксперимента может служит также истечение заданного временного интервала моделирования. Simulink позволяет использовать и такую возможность. Попробуем ею воспользоваться.

При заданных параметрах модели (упругости мяча и начальной скорости полета) мяч перестает прыгать через 20 с. Этот отрезок времени совпадает в исходной модели с временным интервалом моделирования, длительность которого отображается на горизонтальной шкале “смотровых окон”.

Изменим параметры движения мяча, оставив прежним временной интервал моделирования. Найдите на блок-диаграмме элемент в форме треугольника с именем Elasticity (упругость). Он позволяет задавать значение коэффициента упругости мяча. Текущее значение этого коэффициента, равное -0.8, выводится внутри блока. Очевидно, чем более упругий мяч нам попадется, тем дольше он будет прыгать. В данной модели увеличению упругости мяча соответствует увеличение абсолютного значения коэффициента Elasticity. Замените -0.8 на -0.9 (для этого щелкните дважды на блоке, в открывшемся окне настроек произведите соответствующие изменения и щелкните на кнопке ОК). Запустите модель на исполнение. Оцените результат.

Вернемся к рассмотрению основной библиотеки блоков Simulink.

Основная библиотека блоков разбита на несколько разделов (библиотек), содержимое которых не может изменяться пользователем. Рассмотрим последовательно каждый из них.

Раздел (библиотека) Sources

Блоки, входящие в раздел **Sources** (источники), предназначены для описания рабочей нагрузки моделируемой системы, а также для формирования

сигналов, обеспечивающих управление работой S-модели в целом или отдельных её частей.

Все блоки-источники имеют по одному выходу и не имеют входов (рис.1). Терминология, используемая авторами Simulink для описания блоков этого и других разделов библиотеки, говорит о том, что в основном имеются в виду электрические сигналы. Тем не менее, физическая интерпретация понятия “сигнал” в каждом конкретном случае различна и определяется в первую очередь физической природой моделируемой системы.

В качестве источников сигналов (входных величин) могут использоваться следующие блоки:

- **Band-Limited White Noise** (“белый шум” с ограниченной полосой) - генератор “белого шума” с ограниченной полосой;
- **Chirp Signal** (гармонический сигнал) - источник гармонических колебаний переменной частоты;
- **Clock** (часы) - генератор непрерывного временного сигнала;
- **Constant** (константа) - источник постоянной величины (скаляра, вектора или матрицы);
- **Digital clock** (цифровые часы) - источник дискретного временного сигнала;
- **Discrete Pulse Generator** (дискретный импульсный генератор) - генератор дискретных импульсных сигналов;
- **Pulse Generator** (импульсный генератор) - генератор импульсных сигналов;
- **Ramp** (возбудитель) - генератор линейно возрастающего (убывающего) сигнала;
- **Random Number** (случайное число) - источник дискретного сигнала, амплитуда которого является случайной величиной, распределенной по нормальному закону;
- **Repeating Sequence** (периодический сигнал) - генератор периодического дискретного сигнала произвольной формы;
- **Signal Generator** (генератор сигнала) - генератор непрерывного сигнала произвольной формы;
- **Step** (такт) - источник единичного дискретного сигнала с заданными параметрами;
- **Sine Wave** (генератор гармонических колебаний);
- **Uniform Random Number** (равномерное случайное число) - источник дискретного сигнала, амплитуда которого является равномерно распределенной случайной величиной.

Следующие два блока из раздела Sources отличаются от перечисленных тем, что обеспечивают использование в модели различных числовых данных, полученных ранее как с помощью Simulink, так и другими средствами Matlab:

- **From File** (ввод из файла) - ввод в S-модель данных, хранящихся в mat-файле;
- **From Workspace** (ввод из рабочей области) - ввод в модель данных непосредственно из рабочей области Matlab.

Раздел (библиотека) Sinks

Библиотека **Sinks** (получатели) включает средства отображения сигналов, возникающих на выходе блоков. Блоки, собранные в этом разделе, существенно различаются по функциональному назначению.

Условно их можно разделить на три вида:

- блоки, используемые при моделировании в качестве “смотровых окон”: Scope (индикатор), XYGraph (двумерный график), Display (экран);
- блоки, обеспечивающие сохранение промежуточных и/или выходных результатов моделирования: To File (запись в файл), To Workspace (запись в рабочую область);
- блок управления моделированием - Stop Simulation (остановка моделирования) позволяет прервать моделирование при выполнении тех или иных условий. Блок срабатывает в том случае, если на его вход поступает ненулевой сигнал.

Раздел (библиотека) Nonlinear

Несмотря на свое название, раздел Nonlinear (нелинейные системы) содержит достаточно универсальные блоки, которые могут быть использованы при построении моделей систем различных типов.

Блоки данного раздела условно можно разделить на несколько групп (более подробное рассмотрение каждой из имеющихся групп выполним в следующих частях лабораторной работы №4). Отельную группу образуют “блоки-переключатели”, т.е. блоки, управляющие направлением передачи сигнала: Switch (переключатель), Manual Switch (ручной переключатель), Multiport Switch (многовходовый переключатель), Relay (реле).

Блок Switch имеет три входа: два информационных (первый и третий) и один управляющий (второй). Логика работы блока состоит в следующем: если амплитуда сигнала, поступающего на 2-й вход, не меньше заданного порогового значения, то на выход блока передается сигнал с 1-го входа, в противном случае - сигнал с 3-го входа. Блок имеет единственный параметр настройки - Threshold (порог). Он может задаваться либо как числовая константа, либо как вычисляемое выражение. Периодичность срабатывания блока Switch определяется значением параметра блока, подсоединенного к его управляющему входу.

2. Практическая часть

Задание 1:

1.15. Одним из способов получения косинусоидального сигнала Cos t может послужить взятие интеграла от синусоиды.

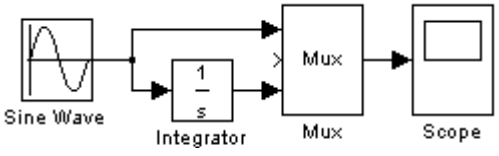

Рис.4. Блок-диаграмма получения косинусоидального сигнала

Для этого достаточно подать на вход блока Integrator сигнал с выхода блока Sine Wave (генератор синусоиды) предварительно установив для него соответствующие параметры. На выходе блока Integrator можно наблюдать косинусоиду, подключив к нему блок Scope (Осциллограф, раздел Sinks), как показано на рис.4. Блок Mux

(смеситель) позволяет объединить входные сигналы в один векторный сигнал (раздел Connections). Выполните следующие задания:

а) постройте блок-диаграмму модели получения косинусоидального сигнала;

б) запустите модель на исполнение и оцените полученный результат в “смотровом окне” Scope;

в) сохраните полученный результат в mdl-файле в собственной папке.

1.16. Блок-диаграмма, изображенная на рис.5, иллюстрирует применение блока Stop. Как только значение времени станет больше или равно 15с., модель прекращает выполняться.

Выполните следующие задания:

а) постройте блок-диаграмму, изображенную на рис.5;

б) сохраните полученный результат в mdl-файле в собственной папке.

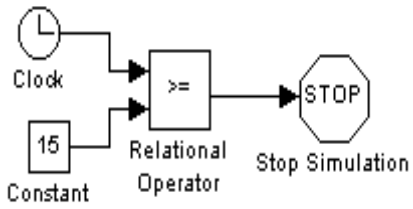

Рис.5. Прекращение моделирования с помощью блока Stop

Задание 2:

2.1. Допустим, необходимо получить сигнал, который первые 20 с. будет прямоугольной волной, а последующее время - пилообразным. Для этого необходимо использовать блок Switch. В

блоке Switch устанавливается пороговая величина 20, на первый вход подается пилообразный сигнал, на второй - сигнал часов, на третий - прямоугольная волна. Первые 20 с. значение времени будет меньше порогового, поэтому будет активным третий вход. Как только время превысит порог, активным становится первый вход и на выходе появляется пилообразный сигнал (рис. 6).

Выполните следующие задания:

а) постройте блок-диаграмму, изображенную на рис.6;

б) запустите модель на исполнение и оцените полученный результат в “смотровом окне” Scope;

в) сохраните полученный результат в mdl-файле в собственной папке.

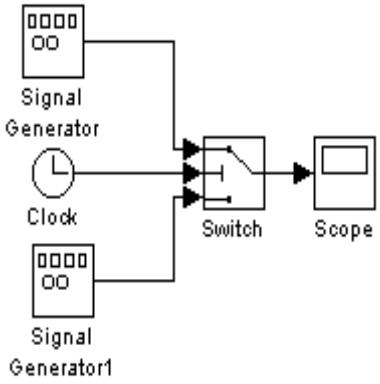

Рис.6. Изменение характера поведения системы

Задание 3:

3.1. Дано дифференциальное уравнение x’(t)=-2x(t)+u(t), где u(t) - прямоугольная волна с амплитудой 1 и частотой 1 рад/с. Постройте его численное решение.