ВВЕДЕНИЕ

MATLAB – это высокопроизводительная инструментальная система для математической поддержки научно-технической деятельности. Система обеспечивает выполнение вычислений и визуализацию их результатов с использованием программирования в удобной интегрированной среде в форме, которая близка к математической.

Одним из важнейших компонентов системы является язык программирования, также называемый MATLAB. Данный язык позволяет легко оперировать с векторами и матрицами. Являясь высокоуровневым динамично развивающимся языком, в настоящее время он содержит практически все элементы, свойственные другим языкам высокого уровня – функции пользователя, структуры данных и др. Существенная роль отводится средствам объектной технологии. В связи с этим, язык программирования MATLAB можно рассматривать в качестве базового языка программирования для студентов инженерной подготовки.

Предлагаемый цикл лабораторных работ направлен на освоение основных конструкций языка MATLAB, которые в том или ином виде присутствуют в большинстве языков программирования. Сведения, которые приводятся для выполнения работ, относятся к базовым встроенным средствам (ядру) системы. Хотя ядро системы является довольно стабильным, некоторые изменения с каждой новой версией MATLAB все-таки появляются, поэтому некоторые рассматриваемые типы данных, функции и операторы могут оказаться недоступными в некоторых версиях. Такие функции в методических указаниях помечены сносками.

Порядок выполнения для всех лабораторных работ одинаковый. При выполнении следует обратить внимание на рекомендации. По итогам лабораторной работы выпускается отчет. ***Содержание отчета*** содержит следующие обязательные разделы:

1. Задание на лабораторную работу.
2. Блок-схемы алгоритмов решения задачи
3. Тексты программ
4. Результаты выполнения программ (графики, листинг командного окна, содержимое файлов)
5. Выводы.

ОСНОВЫ РАБОТЫ В ИНТЕГРИРОВАННОЙ СРЕДЕ MATLAB

Система MATLAB является интерактивной средой, в которой основной математической формой представления данных служит матрица. Подобная ориентация системы делает ее особо значимой для решения прикладных задач с использованием матричных методов.

Элементарные операции с матрицами

Представление чисел. Числа в MATLAB могут быть целыми, дробными, с фиксированной и плавающей точкой, комплексными. Ниже приведены примеры представления чисел:

вещественные числа	*комплексные числа*
1	3i
-2	-2j
4.321	2+4i
0.000000001	-0.0001j
987.6543e-21	-5.4321i
-123.567e10	-123.45+1.2e-4i

Переменные. Система MATLAB выполняет различные действия над переменными, каждая из которых имеет некоторое имя. Имена назначаются пользователем или системой и формируются как произвольные последовательности латинских букв и цифр (допустим также символ подчеркивания), начинающиеся обязательно с буквы. При этом система различает прописные и строчные буквы. Указание имени новой переменной в левой части оператора присваивания приводит к её автоматическому размещению в оперативной памяти, трактуемой в среде как «рабочее пространство» (WorkSpace). Удалить переменную из рабочего пространства можно только с помощью команды clear.

Системные переменные. В системе по умолчанию определены некоторые часто используемые переменные, основными из которых являются следующие:

i или j – мнимая единица $\sqrt{-1}$;

pi – число π=3.141592653589793…;

inf – значение машинной бесконечности;

ans – переменная, в которой хранится результат выполнения последней операции (создается автоматически, когда не определены выходные аргументы какой-либо команды);

NaN – неопределенность, указание на нечисловой характер данных (Not-a-Number);

realmin и realmax – минимальное и максимальное вещественные числа.

Для очистки рабочего пространства используются следующие варианты команды clear:

clear – уничтожение всех переменных;

clear x – уничтожение переменной x;

clear a b c – уничтожение нескольких указанных переменных.

Задание матриц. Для задания некоторых матриц стандартного вида в систему MATLAB включены функции:

1. zeros(n,m) - нулевая матрица размера $n \times m$;

2. eye(n) - единичная матрица размера $n \times n$;

3. ones(n,m) - матрица размера $n \times m$, все компоненты которой равны 1.

4. rand(n,m) - матрица размера $n \times m$ со случайными компонентами, которые равномерно распределены на отрезке [0,1].

Прочие вектора и матрицы задаются вручную по следующим правилам: значения элементов матрицы перечисляются в квадратных скобках, элементы в строке разделяются пробелами или запятыми, строки разделяются точкой с запятой (;). Ниже приведены примеры формирования векторов и матриц.

V1=[1 2 3] – вектор-строка из трех элементов со значениями 1, 2 и 3.

V2=[1;4;7] – вектор-столбец из трех элементов со значениями 1, 4 и 7.

M=[1 2 3; 4 5 6; 7 8 9] – матрица размера 3×3 следующего вида:

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Совершенно аналогично задаются блочные матрицы (состоящие из векторов и матриц).

Функция репликации матрицы repmat(A,N,M) создает блочную матрицу NxM, каждым элементом которой является матрица A. Вызов функции repmat(0,n,m) эквивалентен вызову функции zeros(n,m).

Для определения размерности вектора служит функция length(V), размерность матрицы можно определить при помощи функции size(M).

Для указания отдельной компоненты вектора или матрицы используются выражения вида V(i) или M(i, j).

Для упрощения манипуляций с матрицами широко используется о*ператор перечисления (:)*. Например:

t = t1:h:t2 – формирование вектора-строки, представляющего собой числовую последовательность от t1 до t2 с шагом h (по умолчанию шаг равен единице, для убывающей последовательности шаг должен быть отрицательным);

A(n1:n2,m1:m2) – выделение блока из состава матрицы *A*, ограниченного строками n1 и n2 и столбцами m1 и m2 (в случае, если перечисляются все строки или столбцы, можно указать просто оператор перечисления без чисел). Например, выражение A(4,:) выделяет 4-ю строку матрицы *A*. Выражение t(n1:end) позволяет выделить часть вектора-строки t от элемента n1 до конца.

При возникновении необходимости удаления отдельных строк или столбцов матрицы, используется пустая матрица ([]). Например, выражение M(2,:)=[] удаляет 2-ю строку матрицы *M*.

Простейшие операции с матрицами. Операции сложения, вычитания и умножения матриц выполняются в соответствии с обычными правилами линейной алгебры в естественной форме записи с операторами (+, –, *). Размерности матриц должны быть согласованы. Согласования не требуют только арифметические операции между матрицей и числом. Знак точки (.) перед знаком операции означает, что операция над матрицами будет производиться над каждым элементом матрицы отдельно – поэлементно.

Операция сопряженного транспонирования матрицы выполняется с использованием символа ('). При сопряженном транспонировании у комплексных чисел меняется знак мнимой части. Для несопряженного транспонирования матрицы нужно использовать признак поэлементной операции (.').

Для вычисления определителя квадратной матрицы используется команда (функция) det. Операция обращения квадратной матрицы выполняется с помощью команды inv. Для поиска собственных чисел и собственных векторов квадратной матрицы используется команда eig:

[D] = eig(A) – возвращает только вектор собственных чисел;

[V,D] = eig(A) – возвращает матрицу V, состоящую по столбцам из собственных векторов, и диагональную матрицу D, с собственными значениями на диагонали.

Функция diag(A) возвращает в виде вектора главную диагональ матрицы A. Если параметром функции diag указать вектор V, она возвращает матрицу

с элементами вектора V на главной диагонали. Вторым параметром функции diag можно задать смещение относительно главной диагонали.

Операции с полиномами

Полиномы $P(s) = p_n s^n + p_{n-1} s^{n-1} + ... + p_1 s + p_0$ в среде MATLAB представляются в виде векторов-строк $\mathbf{p} = [p_n \;\; p_{n-1} ... \; p_1 \;\; p_0]$, содержащих коэффициенты полиномов, начиная с коэффициента при старшей степени.

Сложение и вычитание полиномов выполняется по обычным правилам сложения матриц, поэтому предварительно полиномы должны быть приведены к одинаковой размерности путем добавления нулей слева к полиному меньшей степени (Например, p = [0 0 p]).

Умножение полиномов обеспечивается функцией conv с двумя аргументами полиномами сомножителями.

Деление полиномов выполняется с помощью функции deconv, которая имеет два полинома аргумента (делимое и делитель) и возвращает также два полинома: частное и остаток от деления.

Вычисление значения полинома в точке обеспечивается функцией polyval, которая имеет два аргумента: вектор коэффициентов полинома и точку, в которой он вычисляется. Второй аргумент функции может быть матрицей. Тогда функция возвратит матрицу той же размерности, компоненты которой будут равны значениям полинома в точках, определяемых компонентами второго аргумента. В общем случае аргументы могут принимать комплексные значения.

Функция polyvalm работает аналогично polyval, но вычисляет значение матричного полинома. Вторым аргументом функции должна быть указана квадратная матрица. Вычисление значения полинома осуществляется с учетом правил произведения матриц.

Вычисление производной от полинома выполняется функцией polyder с одним аргументом. Аналогично функция polyint(p, C) вычисляет интегральный полином с постоянной интегрирования C.

Формирование полинома по его корням можно выполнить с использованием функции poly с единственным аргументом – вектором, содержащим заданные корни, в общем случае комплексные. Функция poly также позволяет определить характеристический полином квадратной матрицы, если она указана в качестве аргумента функции.

Поиск корней полинома реализуется с помощью функции roots. Аргументом при ее вызове является полином, а найденные корни возвращаются в виде компонентов выходного вектора-столбца.

Типы данных языка программирования MATLAB

MATLAB является языком со слабой динамической типизацией, т. е. объявление переменных до их использования не требуется, а преобразования между типами данных осуществляются свободно, часто автоматически. Переменная, созданная без указания типа, по умолчанию принимает тип данных double – вещественное число двойной точности. Структура типов данных языка MATLAB приведена в таблице 1. Объект любого типа создается в виде матрицы, или двумерного массива.

Таблица 1

Наименование	Характеристика	Примечание
logical	логический тип	имеет два значения: true, false, объем памяти – 1 байт
char	символьный тип	объем памяти – 1 байт
int8, int16, int32, int64*	знаковое целое число	объем памяти соответственно 1, 2, 4, 8 байт
uint8, uint16, uint32, uint64*	целое число без знака	
single, double (по умолчанию)	вещественное число	единичная и двойная точность числа, объем памяти соответственно 4 и 8 байт
struct	структура	данные разных типов, упорядочены в виде пар «поле» – «значение»
cell	ячейка	данные разных типов, упорядочены в виде матрицы
table*	таблица	данные разных типов, упорядочены в виде таблиц
function_handle	указатель на функцию	тип данных для косвенного вызова функций (признак – символ @)

К базовым типам относятся логический, символьный и числовые типы данных. Логический тип logical используется для вычисления логических выражений. В основном он применяется (неявно) в операторах условий и циклов, либо при индексации матриц. Константы символьного типа задаются ключевыми словами true (истина) и false (ложь).

Символьный тип char также часто называется строковым (в MATLAB нет принципиальной разницы между операциями с одним символом, векто-

* не реализовано в MATLAB версии 6.5

ром или матрицей символов). Константы типа char создаются путем помещения текста в одинарные кавычки, например ‘string’.

Числовые типы делятся на целочисленные и вещественные. В целочисленных типах различают типы без знака и со знаком. Использование целочисленных типов позволяют экономить память, однако некоторые стандартные операции, даже арифметические, для них не определены. Между числовыми типами данных существует иерархия старшинства, от младшего типа uint8 до старшего типа double. Иерархия числовых типов играет важную роль при преобразовании типов. При создании матриц числовых типов с помощью стандартных функций можно указать тип дополнительным строковым параметром, например:

```
y = rand(6,5, ‘single’)
```

Контейнерные типы (ячейка, структура, таблица) формируются по разным принципам из базовых типов. В ячейке cell разнотипные данные имеют индексы, как в обычных матрицах. Ячейки формируются с помощью фигурных скобок {}, например:

```
A_cell = {eye(6) ‘text’; 14 -7+2j}
```

В приведенном примере A_cell – матрица ячеек размером 2×2. Элементы матрицы – ячейки, но в них «запакованы» данные различных типов. Для «распаковки» данных нужно указывать индексы в фигурных скобках: A_cell{1,1}.

В структуре данные хранятся в виде совокупности именованных полей и значений. Общая форма создания структур имеет вид:

```
s = struct(‘fieldName1’, value1, ..., ‘fieldNameN’, valueN).
```

В качестве значений полей value с именами fieldName могут быть числа, вектора и матрицы любых типов, а также ячейки. К полям структуры можно обращаться также, используя символ десятичной точки. Эквивалентный способ создания структуры имеет вид:

```
s.fieldName1 = value1
…
s.fildNameN = valueN
```

Имя типа данных является также функцией преобразования данных к этому типу. Таким образом, для создания переменной определенного типа требуется указать значение с функцией преобразования, например:

```
y = single(2.1).
```

Преобразование типов, как правило, выполняется автоматически. Если при создании матрицы в качестве ее элементов указаны данные разных числовых типов, MATLAB осуществит приведение типов к младшему в иерархии. Если среди элементов матрицы будут символы, матрица в результате будет также символьной. Между логическим и символьным типом преобразования запрещены. Для преобразований между базовыми и контейнерными типами данных предусмотрены также специальные функции:

CELL2MAT – преобразование содержимого матрицы ячеек в матрицу вещественных чисел (тип single);

MAT2CELL, NUM2CELL – преобразование матрицы вещественных чисел в матрицу ячеек;

CELL2STRUCT – преобразование матрицы ячеек в матрицу структур;

STRUCT2CELL – преобразование матрицы структур в матрицу ячеек;

Для просмотра подробного описания функции следует в командной строке MATLAB ввести команду help имя_функции.

Кроме базовых и контейнерных типов данных, MATLAB поддерживает специальный тип для работы с функциями пользователя – указатель на функцию (function_handle). Это особый тип, для которого не предусмотрены операции преобразования типов или матричная обработка (хотя есть возможность создать матрицу указателей на функции, обрабатывать эту матрицу придется поэлементно).

ЛАБОРАТОРНАЯ РАБОТА 1. МАТЕМАТИЧЕСКИЕ ФУНКЦИИ, ОПЕРАТОРЫ И ЭЛЕМЕНТАРНАЯ ГРАФИКА В СРЕДЕ MATLAB

Цель работы: освоение работы с простейшими математическими функциями и средствами графики в среде MATLAB.

Основные сведения

Ориентированность системы MATLAB на обработку многомерных массивов (матриц) позволяет с помощью простых функций производить математические вычисления над большими объемами данных без применения операторов цикла.

В основе вычислений в системе MATLAB лежит принцип векторизации. Вычисления проводятся над векторами как над скалярными объектами, за счет встроенных в MATLAB алгоритмов быстрой обработки циклов. Поэтому базовая математическая обработка данных сводится к использованию

простейших функций. Эти функции также являются встроенными. К ним относятся:

SIN(X), COS(X), TAN(X), ASIN(X), ACOS(X), ATAN(X) – тригонометрические функции, соответственно синус, косинус, тангенс, арксинус, арккосинус, арктангенс. Прямые функции принимают, а обратные возвращают значение угла в радианах. Для тригонометрических вычислений с углами, заданными в градусах, можно применять функции SIND, COSD, TAND, ASIND, ACOSD, ATAND.

ATAN2(Y, X) – функция вычисления четырехквадрантного арктангенса, т.е. угла поворота от оси *Ox* на плоскости до точки с координатами X, Y. Значение функции лежит в диапазоне $[-\pi; \pi]$.

SINH(X), COSH(X), TANH(X) – гиперболические функции, соответственно, синус, косинус и тангенс.

LOG(X), LOG10(X), LOG2(X) – логарифмические функции, соответственно, натуральный, десятичный логарифм и логарифм по основанию 2.

EXP(X) – экспонента – функция, обратная натуральному логарифму.

POWER(X, Y) – показательная функция, вычисляющая $f(x, y) = x^y$. Если X – вектор, операция вычисления степени осуществляется поэлементно.

POW2(X) или POW2(Y, X) – показательная функция, вычисляющая $f(x) = 2^x$ или $f(y, x) = y2^x$.

SQRT(X) – функция поэлементного вычисления квадратного корня. Для нахождения квадратного корня матрицы, то есть решения матричного уравнения $X * X = A$, нужно применять функцию SQRTM(A).

ABS(X) – вычисление модуля аргумента.

Комплексные числа. MATLAB поддерживает операции над комплексными числами. Для комплексных чисел используются следующие функции.

COMPLEX(X, Y) – формирование комплексного числа или вектора чисел. Вместо функции можно использовать запись в виде X + iY или X + jY.

REAL(X), IMAG(X) – функции, возвращающие вещественную и мнимую часть комплексного числа.

ABS(X), PHASE(X) – функции, возвращающие модуль и фазу (в радианах) комплексного числа. В поздних версиях MATLAB для вычисления фазы нужно использовать функцию ANGLE(X).

Приоритет операций. Для вычислений можно использовать не только функции, но и обычные операторы математики и логики. Следует отметить,

что система MATLAB поддерживает двойственность использования функций и операторов, т. е. каждый оператор языка MATLAB может быть реализован посредством вызова функции. Например, операцию сложения можно реализовать либо с использованием оператора: X + Y; либо с использованием функции: PLUS(X, Y).

При формировании сложных математических выражений с использованием различных операторов необходимо соблюдать приоритет операций.

1. Самый низкий приоритет имеет оператор логического ИЛИ с упрощением (||). Упрощение состоит в следующем: если левый операнд логического ИЛИ имеет значение (true), то правый не вычисляется. Данный оператор можно применять только со скалярными логическими операндами.

2. Следующий по приоритету – оператор логического И с упрощением (&&). Если левый операнд имеет значение false, то правый не вычисляется. Данный оператор также требует скалярные логические операнды.

3. Поэлементный оператор логического ИЛИ (|) может быть использован с матричными или векторными операндами. Матрицы (векторы) должны быть одинакового размера либо один из операндов должен быть скаляром. Оператор ИЛИ также может быть реализован через вызов функции OR(X, Y).

4. Поэлементный оператор логического И (&) функционирует по аналогии с предыдущим. Он может быть реализован через вызов функции AND(X, Y).

5. Операторы сравнения (см. табл. 2) имеют одинаковый приоритет, в отличие от языка программирования C/C++, где операторы равенства и неравенства имеют более низкий приоритет, чем прочие. Операторы сравнения могут быть заменены функциями, приведенными в таблице. Если в выражении есть несколько операторов, равных по приоритету, они будут выполняться слева направо, за малым исключением.

6. Оператор перечисления (:) подробно описан в предыдущем разделе. Выражения X1:X2 или X1:h:X2 могут быть заменены соответственно вызовом функций COLON(X1, X2) и COLON(X1, h, X2).

7. Следующие по приоритету – аддитивные операторы: сложения (+) и вычитания (–). Соответствующие им функции – PLUS(X, Y), MINUS(X, Y).

8. Мультипликативные операторы – умножение (*), правое (/) и левое (\) деление, выполняемые над матрицами. Поэлементные операторы имеют обозначение (.*, ./, .\). Операторы матричного деления служат для решения сис-

тем линейных алгебраических уравнений (СЛАУ). Функции, соответствующие мультипликативным операторам, приведены в таблице 2.

9. Унарные операторы: плюс (+), минус (–), логическое отрицание (~). Унарные операторы имеют только один операнд, справа от знака оператора.

10. Операторы степени с унарным плюсом (^+), минусом (^–) и дополнением (^~) для матриц и соответственно (.^+, .^–, .^~) – для элементов матриц. Эти операторы имеют нестандартный порядок выполнения – от второго справа операторы выполняются справа налево, самый правый выполняется последним.

11. Следующий приоритет имеют операции транспонирования (сопряженного – ‘ и несопряженного – .’) и возведения в степень матрицы (^) и элементов матрицы или вектора (.^). Соответствующие операторам функции приведены в таблице 2.

12. Самый высокий приоритет имеет оператор (). С помощью круглых скобок можно устанавливать в сложных выражениях желаемую последовательность выполнения действий.

Таблица 2

Оператор	Функция	Описание
<	LT(X, Y)	строго меньше (X < Y)
>	GT(X, Y)	строго больше (X > Y)
<=	LE(X, Y)	меньше либо равно (X ≤ Y)
>=	GE(X, Y)	больше либо равно (X ≥ Y)
==	EQ(X, Y)	равенство
~=	NE(X, Y)	неравенство
*	mtimes(A, B)	матричное произведение
.*	times(A, B)	поэлементное произведение
/	mrdivide(A, B)	решение СЛАУ вида $xA = B$
./	rdivide(A, B)	поэлементное правое деление
\	mldivide(A, B)	решение СЛАУ вида $Ax = B$
.\	ldivide(A, B)	поэлементное левое деление
‘	ctranspose(A)	сопряженное транспонирование
.’	transpose(A)	несопряженное транспонирование
^	mpower(A, B)	матричное возведение в степень
.^	power(A, B)	поэлементное возведение в степень

Графика в системе MATLAB. Одно из достоинств системы MATLAB – большое количество графических средств, начиная от команд построения простых графиков функций одной переменной и кончая комбинированными и презентационными графиками с элементами анимации. Для начала рассматривается построение простейших графиков.

Вывод графика на экран в декартовой системе осуществляется командой plot. Эта команда может использоваться в нескольких вариантах.

При вызове функции с двумя аргументами plot(X, Y), где X, Y – векторы одинаковой размерности, будет построен график зависимости *Y(X)* в отдельном окне, именуемом figure. Если один из параметров X, Y является вектором размером $n \times 1$, а другой – матрицей размером $n \times m$, в графическое окно одновременно будут выведено m графиков.

Вызов функции с несколькими параметрами в виде plot(x1,y1,x2,y2, …) приведет к построению нескольких графиков в одних декартовых осях, каждый из которых задан вектором аргументов и вектором значений.

Когда в графическое окно одновременно выводится несколько графиков, они различаются по цветам согласно принятой по умолчанию последовательности: синий – зеленый – красный и т.д. Общий формат функции plot(x1, y1, S1, x2, y2, S2…) позволяет управлять типом линии, ее цветом и типом маркера за счет символьной константы S. Возможные значения константы можно посмотреть в таблице с помощью команды help plot. Пример использования функции plot для управления графиком приведен ниже.

```
x1 = 0:0.01:10;
x2 = 0:10;
y1 = sin(x1);
y2 = sin(x2);
plot (x1,y1, 'r--', x2, y2, 'k*'), grid
```

В примере первый график построен красным цветом и пунктирной линии, второй выводится в виде изолированных черных звездочек.

Далее рассмотрим ряд вспомогательных графических функций:

GRID (GRID ON| OFF) – координатная сетка (см. рисунок 1);

FIGURE (n) – смена текущего графического окна, *n* – номер окна. Также данная функция может применяться для создания новых графических окон.

SUBPLOT(KLM) – разбиение графического окна для отрисовки графиков в его части. Индексы K, L указывают на размерность матрицы осей, индекс M – на порядковый номер осей в окне.

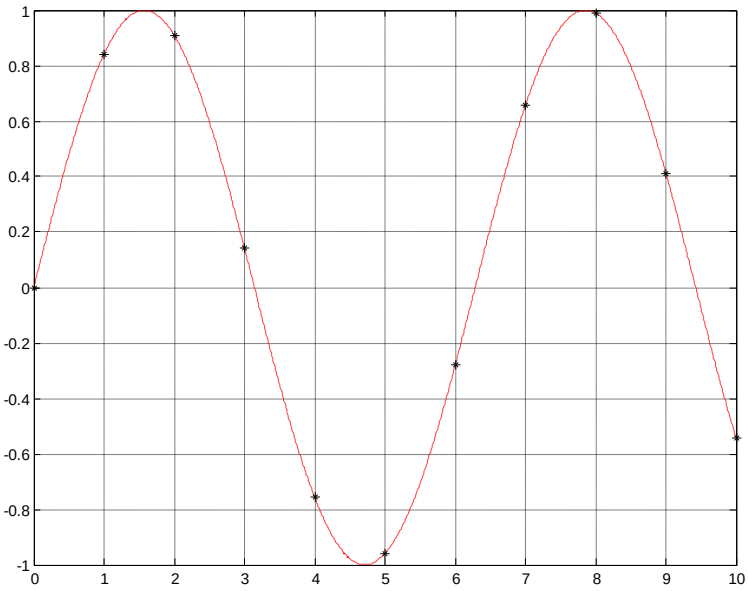

Рисунок 1

HOLD (HOLD ON| OFF) – наложение графиков в осях текущего графического окна. Наложение будет производиться до тех пор, пока не встретится команда HOLD OFF или пользователь не сменит текущее графическое окно.

TITLE('текст') – добавление заголовка к графику;

XLABEL('текст'), YLABEL('текст') – добавление подписей к осям;

В случае, когда одна из координат функции, выводимой на экран, меняется в большом диапазоне, для построения лучше использовать логарифмический или полулогарифмический масштаб. При этом имеет смысл задавать вектор аргумента в виде логарифмической, а не равномерно возрастающей последовательности. Это выполняется командой LOGSPACE(X1, X2, N), где искомый вектор меняется в диапазоне от 10^{x1} до 10^{x2} и содержит N точек.

Построение графиков в полулогарифмических осях обеспечивается функциями SEMILOGX(X, Y), SEMILOGY(X, Y). Функции используются вместо PLOT по аналогичным правилам. Для построения графика в логарифмических осях предназначена функция LOGLOG(X, Y). Однако при использовании функций с логарифмической шкалой необходимо учитывать, что значения координат должны быть положительны.

Методика выполнения работы

1. Написать программу для расчета значений двух функций и построения графиков согласно таблице 4 по номеру варианта, указанному препода-

вателем. В таблице через $A(\omega)$ и $\Phi(\omega)$ обозначены модуль и фаза функции комплексной переменной.

2. Диапазон изменения переменной *x* для первой функции выбрать самостоятельно; выбранный диапазон должен обеспечить плавное построение функции.

3. Если функция $f(x)$ в выбранном диапазоне изменяется слишком резко, скорректировать диапазон.

4. Обеспечить вывод графиков двух функций в разных графических окнах; оформить графические окна поясняющими подписями.

Требования к программе и результатам работы

1. Программа не должна иметь циклов.

2. График функции от переменной ω должен быть построен в полулогарифмических осях (с логарифмической шкалой по аргументу), значения в диапазоне ω должны быть заданы в виде логарифмической последовательности.

3. Расчет значений дробно-рациональной функции должен осуществляться с использованием функций для работы с полиномами.

4. Вместо операторов языка MATLAB можно использовать заменяющие их функции.

Рекомендации к оформлению и выполнению программы

1. Основной комментарий, указываемый в первой строчке программы, позволяет индексировать файл программы. Проиндексированные файлы могут быть найдены с помощью команды lookfor. Комментарий в MATLAB начинается с символа % и продолжается до конца строки.

2. После основного комментария рекомендуется указать команды clear и clc – для очистки переменных из рабочей области и командного окна. Это позволит избежать ошибок программирования, возникающих из-за использования в программе переменных, созданных вне программы и существующих в рабочей области.

3. Для автоматического закрытия всех графических окон можно использовать команду close.

Примечание – данных рекомендаций следует придерживаться при оформлении программ в дальнейшем.

ЛАБОРАТОРНАЯ РАБОТА 2. ОПЕРАТОРЫ УСЛОВИЯ И ЦИКЛА.

Цель работы: освоить основные конструкции языка MATLAB для реализации ветвящихся и циклических алгоритмов.

Основные сведения

Организация ветвящихся алгоритмов. Отличительной особенностью алгоритмов с разветвлением является принятие решения в зависимости от хода выполняемых действий. На программном уровне такие алгоритмы реализуются при помощи условных операторов и операторов переключения. Простейшая конструкция условного оператора включает оператор IF и записывается следующим образом:

```
IF условие
    операторы
end
```

Операторы будут выполняться в том случае, если условие истинное. В качестве условия может быть использовано логическое выражение, либо переменная логического типа. Логические выражения строятся при помощи операторов сравнения и логических операторов (см. лаб. раб. 1).

При помощи оператора IF можно реализовывать и более сложные конструкции, с проверкой нескольких условий. В общем случае формат записи оператора IF выглядит следующим образом:

```
IF условие1
    операторы1
ELSEIF условие2
    операторы2
…..
ELSE
    операторы
END
```

Операторы, находящиеся после ELSE, выполняются в том случае, если все условия, описанные в IF и ELSEIF, являются ложными. Все составные операторы языка программирования MATLAB заканчиваются ключевым словом end.

Самостоятельное задание: составить типовые блок-схемы разветвляющихся алгоритмов.

Оператор переключения будет рассмотрен в следующей лабораторной работе.

Организация циклических алгоритмов. Циклические алгоритмы реализуют многократное повторение определенных действий. Соответствующая программная конструкция называется циклом, а повторяемое действие – итерацией. В программном описании цикла можно выделить тело – набор повторяемых действий, а также условие продолжения цикла. Как правило, в цикле явно или неявно присутствует внутренняя переменная, входящая в условие, и изменяющая значение в ходе выполнения цикла.

В зависимости от очередности условия и действий различают циклы с предусловием и с постусловием. Кроме того, циклы классифицируют по количеству повторений: циклы-счетчики, циклы с неопределенным числом повторений.

В языке программирования MATLAB существуют только циклы с предусловием. Цикл-счетчик реализуется при помощи оператора FOR и имеет следующий формат записи.

```
for переменная = вектор
    тело цикла
end
```

Количество итераций цикла-счетчика задается числом элементов вектора в первой строке цикла FOR. Переменная по очереди принимает значения, соответствующие компонентам вектора. Обычно для циклов используют вектора-строки, заданные с помощью оператора перечисления, хотя можно использовать любой произвольный вектор.

MATLAB поддерживает также матричный вариант цикла FOR:

```
for переменная = матрица
    тело цикла
end
```

В этом случае количество итераций цикла задается числом столбцов матрицы в первой строке цикла. Переменная по очереди принимает значения столбцов матрицы.

Цикл с неопределенным числом повторений реализуется при помощи оператора WHILE:

```
WHILE условие
    тело цикла
END
```

Цикл выполняется до тех пор, пока условие истинное. Необходимо следить за тем, чтобы в ходе выполнения итераций параметры, заданные в условии продолжения цикла менялись, иначе возникает бесконечный цикл, который может привести к зависанию программы.

Примечание: для прерывания выполнения программы можно использовать комбинации клавиш CTRL+C или CTRL+BREAK.

Для досрочного продолжения и прерывания цикла служат ключевые слова CONTINUE и BREAK.

Ключевое слово CONTINUE в теле цикла осуществляет переход к следующей итерации с начало цикла. В случае, если CONTINUE находится внутри вложенных циклов, выполняется продолжение с новой итерации для ближайшего внутреннего цикла.

Ключевое слово BREAK осуществляет досрочное прерывание цикла и переход на следующую строку после него. Если BREAK находится внутри вложенного цикла, выход осуществляется из ближайшего внутреннего цикла.

Вне циклов слова CONTINUE и BREAK не применяются.

Индексация матриц. Типовое обращение к элементам многомерных-матриц предполагает использование количества индексов, равного числу размерностей, например, для двумерной матрицы обращение к элементу a_{ij} будет выглядеть A(i, j).

В то же время MATLAB позволяет обращаться к элементам многомерных матриц также как к элементам векторов, с использованием одного индекса (subscript). При этом направление увеличения индекса соответствует движению по столбцам (индекс 2 будет у элемента матрицы a_{21}, индекс 3 – у a_{31}, и т. д.). Для перехода от одномерного индекса к многомерному и обратно служат взаимообратные функции sub2ind и ind2sub.

Функция [SUB] = SUB2IND(MSIZE, I1, I2, I3…) определяет одномерный индекс многомерной матрицы с размерами, заданными вектором MSIZE и индексами I1, I2, I3… Количество индексов должно быть равно количеству размерностей матрицы. Если в параметрах индекса указаны не числа, а матрицы одинакового размера, функция рассчитает матрицу одномерных индексов указанных элементов. Например, для того, чтобы определить одномерный индекс элемента a_{ij} двумерной матрицы *A*, можно использовать команду:

```
sub = sub2ind(size(A). i, j)
```

Функция [I1, I2, I3, …] = IND2SUB(MSIZE, SUB) определяет многомерный индекс элемента по его одномерному индексу SUB в матрице размера MSIZE. Число выходных переменных должно соответствовать размерности матрицы. Следует отметить, что если число выходов не соответствует размерности матрицы, функция IND2SUB автоматически приведет матрицу к заданной числом выходов размерности и вернет соответствующий индекс без ошибки выполнения программы. Как и SUB2IND, функция может принимать матрицу одномерных индексов и рассчитывает соответствующие матрицы индексов.

Методика выполнения работы

1. В соответствии с заданием из таблицы 5 составить блок-схему алгоритма вычисления функции двух переменных $x(t, T)$, заданной по интервалам.

2. Написать и отладить программу-сценарий вычисления функции в точке (не пользуясь стандартными математическими функциями).

3. Построить график функции $x(t)$ при заданном значении T. в диапазоне $t \in [-2T, 2T]$.

4. В соответствии с заданием из таблиц 6 и 7 составить блок-схемы алгоритмов.

5. Написать и отладить программы вычисления суммы ряда (табл. 6) и поиска элементов матрицы (табл. 7)

Требования к программам и результатам работы

1. Для функции двух переменных $x(t, T)$ аргумент T должен быть программно задан произвольным положительным числом.

2. График $x(t)$ нужно построить по 20 – 25 точкам на заданном интервале.

3. При вычислении суммы или произведения ряда следует выводить конечный результат (без промежуточных значений) и число итераций. Можно использовать математические функции MATLAB.

4. При поиске минимального/максимального элемента матрицы стандартными функциями min, max пользоваться не разрешается.

Рекомендации к выполнению программы

1. При вычислении точек для построения графика $x(t)$ можно не использовать цикл, а использовать операторы сравнения для векторов.

2. При поиске минимального/максимального значения матрицы целесообразно использовать один цикл вместо двух вложенных, совместно с функциями sub2ind и ind2sub.

ЛАБОРАТОРНАЯ РАБОТА 3. ОПЕРАТОР ПЕРЕКЛЮЧЕНИЯ И ФУНКЦИЙ ВВОДА-ВЫВОДА.

Цель работы: освоение оператора переключения языка программирования MATLAB, реализация различных способов ввода данных с клавиатуры и вывода на экран.

Основные сведения

Оператор переключения. Оператор переключения позволяет реализовать алгоритмы с ветвлением, как и оператор IF, рассмотренный в предыдущей лабораторной работе. Конструкция проверки нескольких условий операторами IF, ELSEIF является громоздкой. В случае, когда в проверках нет операторов сравнения (только операторы равенства и неравенства) целесообразнее использовать оператор переключения SWITCH. Форма записи оператора выглядит следующим образом:

```
SWITCH переменная_выбора
    CASE значение,
        операторы
    CASE {зн1, зн2, зн3,...}
        операторы
    ...
    OTHERWISE,
        операторы по умолчанию
END
```

Принцип работы оператора переключения: если переменная выбора принимает одно из значений, указанных после ключевого слова CASE, выполняются операторы, следующие за данным фрагментом кода. Операторы выполняются до тех пор, пока в коде не встретится следующее ключевое слово CASE, либо OTHERWISE, либо END. Необязательное ключевое слово OTHERWISE определяет операторы, выполняемые по умолчанию – когда пе-

ременная выбора не равна ни одному из предложенных вариантов. Если одни и те же действия требуется выполнить для нескольких значений переменной, они объединяются после CASE в вектор ячеек. При одном прохождении оператора SWITCH может выполниться не более одного блока операторов. После выполнения блока программа переходит на строку, следующую за словом END.

Переменная выбора может быть скаляром или символьной строкой. Скалярная переменная выбора может быть вещественным числом (в отличии от языка программирования С, где тип переменной строго ограничен символьным или целочисленным типом). Вместо переменной выбора и предлагаемого значения можно указать любое выражение, результатом которого является скаляр или строка символов.

Примечание – хотя оператор SWITCH не допускает в качестве переменной вектора чисел, иногда это ограничение можно обойти путем явного преобразования вектора чисел в символы. При этом следует учитывать, что при попытке преобразования вещественных чисел в символы MATLAB автоматически приведет данные к целочисленному типу, при этом дробные части чисел теряются. Кроме того, диапазон чисел, преобразуемых в строковый тип, ограничен. Попытки преобразовать к символу число вне диапазона может дать неопределенный результат, отличающийся в различных версиях пакета MATLAB на различных платформах.

Функции ввода-вывода. Базовой функцией для ввода данных с клавиатуры является функция INPUT. Форматы вызова функции следующие:

N = INPUT(‘PROMPT’) или Str = INPUT(‘PROMPT’, ‘s’).

Обязательным аргументом функции INPUT является символьная строка, отображаемая в командном окне перед передачей управления пользователю. Как правило, в ней указывают информацию о вводимых данных. Пользователь может ввести с клавиатуры любое число, матрицу, строку – в соответствии с правилами языка программирования MATLAB, – либо вызвать функцию.

Если функция INPUT имеет второй параметр ‘s’, данные, введенные с клавиатуры, воспринимаются как символы. Выходной переменной при этом является символьная строка.

Основной функцией для вывода данных является функция DISP(X). Функция позволяет выводить на экран командного окна любые данные: мат-

рицы чисел, текст, ячейки. Функция выводит содержимое X без указания имени переменной.

Иногда возникает необходимость ввести числовые данные в строковом формате и затем преобразовать их в числа. Справедлива и обратная задача – вывод на экран комбинации текста и значений переменных или констант через строку символов. Для этих целей удобно использовать функции форматного чтения и записи sscanf и sprintf.

Функция sscanf предназначена для считывания данных из символьной матрицы и имеет следующую форму вызова

[A, COUNT, ERRMSG, NEXTINDEX] = SSCANF(S, FORMAT, SIZE)

Аргументами функции являются матрица символов S, формат считываемых данных, строка FORMAT, а также размер матрицы данных A, в которой формируются считанные данные. Остальные выходные параметры являются необязательными. Целое число COUNT возвращает число успешно считанных элементов; ERRMSG – строка с сообщением об ошибке либо пустая строка, если при считывании ошибок не возникло; NEXTINDEX – индекс следующего после последнего считанного символа в строке S.

Строка FORMAT определяет формат считываемых данных по аналогии с языком программирования C. Перечень форматных констант приведен в таблице 3. Признаком считываемых данных является символ %.

Формат %i по умолчанию соответствует целому десятичному числу, но если считываемое число начинается с 0x или 0, то оно интерпретируется в системе счисления с основанием соответственно 16 и 8.

Формат %[ ] позволяет считывать символы, входящие в квадратные скобки, притом допускается задавать их в виде диапазона (0–9, a–z, A-Z). Считывание происходит до первого символа, не входящего в заданный шаблон.

Если в символьном параметре FORMAT указано несколько разных типов считываемых переменных, MATLAB автоматически приведет значения к числовому типу. Вместо символов в матрице данных A будут ASCII-коды этих символов.

Параметр SIZE (необязательный) определяет размер матрицы считанных данных A. Можно указать целое число – максимальное количество считываемых элементов, можно указать inf для считывания всех возможных элементов из символьной матрицы S, либо можно явно задать размер A вектором размерности [m n].

Таблица 3

Формат	Тип данных	Примечание
%d	integer	десятичное целое число
%i	integer	целое число (система счисления определяется по контексту)
%ld, %li	long integer	длинное целое (64-разрядное)
%u	unsigned integer	десятичное целое число без знака
%o	octal integer	число в восьмеричной системе счисления без знака
%x	hexadecimal integer	число в шестнадцатеричной системе счисления без знака
%lu, %lo, %lx	long unsigned	длинное целое без знака (64-разрядное)
%f	single, double	вещественное число, естественная форма записи
%e	single, double	вещественное число, нормализованная форма записи
%g	single, double	вещественная число, автоматически определяемая форма записи
%s	char	последовательность символов (до пробела)
%c	char	единичный символ (включая пробел)
%[…]	char	заданные в строке символы

Функция sprintf предназначена для формирования символьной матрицы и имеет следующий вид вызова:

[STR, ERRMSG] = SPRINTF(FORMAT, A1, ..., AN)

Выходные параметры функции: символьная матрица STR и символьная строка, содержащая сообщение об ошибке, либо пустая, если ошибок при записи не возникло. Аргументами функции является строка символов FORMAT, определяющая вид матрицы STR и переменные A1 – AN, значения которых передаются в строку в соответствии с заданным форматом.

Формат является ключевым параметром для функции sprintf. Он может содержать любой текст и специальные последовательности символов для передачи значений параметров. Структура такой последовательности имеет вид (параметры в квадратных скобках необязательны):

% [идентификатор$][флаги][ширина][.точность][подтип]тип.

Идентификатор выделяется символом $ и выбирает по номеру переменную из списка переменных, приведенных после строки FORMAT.

Флаги устанавливают дополнительные свойства. Флаг ‘+‘ указывает на явный вывод знака числа (+ или –) и выравнивает строку по правому краю,

флаг ‘–‘ форматирует строку с выравниванием по левому краю. Флаг ‘ ‘ добавляет пробел перед выводом значения, флаг ‘0’ заполняет нулями пустые места перед значением.

Ширина (поля) – число, указывающее на общее количество знаков, отводимых под вывод значения. Для вещественных чисел ширина поля включает также знак числа и символ десятичной точки.

Точность устанавливается только для вещественных чисел и указывает на количество знаков после запятой.

Подтип указывается строго перед типом, предназначен для вывода вещественных чисел в разных системах счисления: ‘b’ – вещественных чисел двойной точности; ‘t’ – вещественных чисел одинарной точности.

Символ типа данных является единственным обязательным параметром и определяет выводимое значение согласно таблице 3. Для вещественных чисел можно дополнительно указывать %E и %G – при отображении числа в нормализованной форме символ порядка будет прописным (например, 3.14E+01).

Если символьная матрица не содержит сторонних символов, функции форматирования можно не применять. Вместо этого можно использовать функции конвертирования типов (STR2NUM, STR2DOUBLE и т. д.).

[N] = STR2NUM(STR) – функция конвертирования строковой матрицы STR в матрицу чисел N. Каждое число, заданное в строковом формате, может содержать кроме цифр символы i и j, обозначающие мнимую единицу, знак десятичной точки, признаки числа в экспоненциальном формате e и d. Функция распознает в строковой матрице специальный символ разделителя строк, знаки операций. Если в строке задано выражение вместо числа, функция STR2NUM вычислит значение выражения и запишет его в выходную переменную (вычисление выражения обеспечивается внутренним вызовом функции EVAL). Если число или выражение задано некорректно, функция возвращает пустую матрицу.

[N] = STR2DOUBLE(STR) – функция преобразования строковых матриц к типу double. В качестве параметра требуется указать матрицу, каждый элемент которой является строкой, конвертируемой в число. Соответственно, в каждом строковом элементе могут присутствовать кроме чисел знак десятичной точки, мнимые единицы, унарные плюс и минус и признак экспоненциального формата (‘e’). Если аргумент невозможно преобразовать к числу, функция возвращает значение NaN.

Методика выполнения работы

1. Изменить программу вычисления функции $x(t, T)$ из табл. 5, добавив ввод данных с клавиатуры и вывод значений функции на экран.

2. Составить блок-схемы алгоритмов эмулятора календаря и обработки строки символов (см. таблица 8) в соответствии с вариантом задания.

3. Написать и отладить программы в соответствии с алгоритмами.

Требования к программе и результатам работы

1. Для расчета функции $f(t, T)$ T нужно вводить с клавиатуры, вывод значений функции на экран должен соответствовать виду

```
f(t1, T) = f1
f(t2, T) = f2
…
```

Вместо t, T, f должны отображаться вещественные числа, строки должны быть отформатированы так, чтобы их длина была одинакова для всех значений функции.

2. Поскольку T вводится с клавиатуры, необходимо ввести проверку на знак введенного значения: если T не положительное число, следует преобразовать его в положительное число, либо вывести сообщение об ошибке (на выбор студента).

3. Исходные данные для эмулятора календаря должны вводиться с клавиатуры символьной строкой, в точности согласно таблице 8.

4. В программе эмулятора календаря должна быть проверка на ввод недопустимой даты.

5. Использование функций работы с датами языка MATLAB не разрешается.

Рекомендации к выполнению программ

1. Для искусственного прерывания программы с выводом сообщения об ошибке нужно использовать функцию ERROR(PROMPT), PROMPT – отображаемая символьная строка.

2. В программе эмулятора календаря целесообразно ввести переменную для хранения числа дней в месяце, и определить ее через SWITCH.

3. Для обработки целочисленных типов можно использовать функции REM(X, Y) (остаток от деления X на Y), IDIVIDE(X, Y) (целочисленное деление X на Y).

ЛАБОРАТОРНАЯ РАБОТА 4. ФУНКЦИИ ПОЛЬЗОВАТЕЛЯ.

Цель работы: освоение работы с функциями пользователя в среде MATLAB.

Основные сведения

Скрипты и функции пользователя. Все m-файлы, содержащие исходные коды на языке MATLAB, делятся на два типа: файлы-сценарии и файлы-функции.

Файлы-сценарии (или script-файлы) являются просто записью серии команд, которые могли бы быть набраны и выполнены в командном окне MATLAB. Главная их особенность состоит в том, что файлы-сценарии не имеют входных и выходных параметров. В них используются переменные из общего рабочего пространства MATLAB. В процессе выполнения они не компилируются в двоичный код. Обычно файл-сценарий имеет следующую структуру:

```
% Основной комментарий
% Дополнительный комментарий
Тело сценария, состоящее из любой совокупности операторов.
```

Основным комментарием является первая строка текстовых комментариев, а дополнительным – последующие строки. Основной комментарий выводится при выполнении команд lookfor и «help имя_каталога». Полный комментарий выводится при выполнении команды «help имя_файла».

М-файл типа «функция» является типичным элементом языка программирования MATLAB. Файлы-функции обязательно начинаются с объявления function. Структура файла-функции (имя файла f_name.m) с одним выходным параметром выглядит следующим образом:

```
function var=f_name(список_параметров)
% Основной комментарий
% Дополнительный комментарий
Тело функции, состоящее из любой совокупности операторов
var=выражение
```

Примечание 1: при вызове функции идентификатором ее формально является имя файла.

Примечание 2: Если выходных параметров больше одного, то необходимо использовать конструкцию типа:

```
function [var1, var2, …]=f_name(список_параметров)
```

Вызов функции можно осуществить из командной строки, из скрипта или из другой функции. Необходимо, чтобы все входные параметры перед вызовом были определены. Если функция, имеющая несколько выходных параметров, входит в состав математического выражения, то для выполнения вычислений в выражении будет использоваться первый из выходных параметров.

Область видимости переменных. Все переменные, используемые в теле файла-функции, являются локальными, т.е. действуют только в пределах тела функции. При этом переменные общего рабочего пространства внутри функции не видны. После окончания вызова функции локальные переменные стираются, т. е. освобождают память.

Иногда возникает необходимость в использовании внутри функции данных, находящихся в рабочем пространстве сценария или функции, откуда осуществляется вызов. В этих случаях используется механизм глобальных переменных, имеющих неограниченную область видимости. Чтобы определить переменные как глобальные, используется команда

```
global var1 var2 …
```

Чтобы несколько программных модулей (функций) могли совместно использовать глобальную переменную, ее идентификатор должен быть объявлен как global во всех этих модулях.

Кроме локальных и глобальных переменных MATLAB позволяет определить статические переменные. Они существуют и доступны только внутри функций, и их значения не стираются после окончания вызова функции. Если функция будет вызвана повторно, в статической переменной сохранится от предыдущего вызова. Для определения статической переменной используется команда.

```
persistent stat1 stat2  …
```

Статические переменные невозможно изменить из командной строки или из других функций, так как область видимости для них ограничена функцией, в которой они определены. Статические переменные хранятся до тех пор, пока хранящая их функция не будет изменена, либо пока не закончится сессия MATLAB. Статические переменные можно использовать для контроля числа вызовов функции.

Локальные и встроенные функции. Если некоторая функция пользователя требует для своих вычислений использование другой вспомогательной функций, ее можно описать и вызвать непосредственно в файле основной

функции. При этом вспомогательные функции могут быть локальными и встроенными. Порядок описания таких функций приведен ниже

Описание локальной функции

```
function val = func(var1, var2)
   % осн. комментарий
   % доп. комментарий
   xl = sub_func(x1)
   операторы
   val = выражение
end

val = sub_func(x1)
   val = выражение
end
```

Описание встроенной функции

```
function val = func(var1, var2)
   % осн. комментарий
   % доп. комментарий
   xl = sub_func(x1)

   val = sub_func(x1)
      val = выражение
   end
   операторы
   val = выражение
end
```

При использовании локальных и встроенных функций нужно использовать ключевое слово end для обозначения окончания функций. Описание локальной функции приводится после описания основной функции. Встроенная – описывается непосредственно в теле основной функции. И локальная, и встроенная функция доступны только внутри основной функции.

Различие между локальными и встроенными функциями состоит в области видимости переменных основной функции. Встроенной функции доступны переменные основной функции, локальной – недоступны.

Функции с переменным числом параметров. Важным преимуществом MATLAB является возможность изменения конфигурации функций в зависимости от разного числа входных и выходных параметров. Например, функция EIG(A) дает разные результаты в зависимости от того, сколько при вызове указано выходных переменных. Функции sprintf, sscanf имеют необязательные входные и выходные параметры. Для программирования такого поведения функций используются ключевые слова:

nargin – определение числа входных аргументов функции;

nargout – определение числа выходных значений функции;

varargin – список входных аргументов функции;

varargout – список выходных значений функции.

Ключевые слова nargin и nargout указываются в коде функции (строчными буквами, без параметров). Ключевые слова varargin и varargout указываются при определении функции, например:

```
[V_OUT, varagout] = var_func(V_IN1, V_IN2, varargin)
```

В примере V_IN1, V_IN2, V_OUT – обязательные входные параметры функции var_func, которые должны быть указаны при вызове. Все остальные аргументы, указанные при вызове, войдут в одномерные массивы ячеек varargin и varargout.

Методика выполнения работы

1. Преобразовать алгоритмы лабораторной работы 2 к модульной структуре.

2. Составить и отладить следующие функции

– для расчета значения функции двух переменных $x(t, T)$ (табл. 5);

– для расчета суммы либо произведения элементов ряда (табл. 6);

– для поиска минимальных или максимальных элементов матрицы (табл. 7).

3. Составить и отладить программу-сценарий для вызова функций пользователя.

Требования к программе и результатам

1. Функция, реализующая расчет $x(t, T)$, должна принимать в качестве аргументов t, T числа либо векторы (по выбору студента) и возвращать, кроме значения функции, вторым параметром порядковый номер условия.

2. Функция расчета суммы либо произведения элементов ряда должна принимать два параметра – число обрабатываемых элементов и символ, указывающий на действие над ними: ‘s’ – сумма, ‘p’ – произведение.

3. Внутри функции расчета суммы либо произведения должна быть локальная функция – для расчета слагаемого / сомножителя.

4. Функция поиска элемента в матрице должна возвращать искомый элемент (строку, столбец) и его индекс. Если в ходе выполнения функции требуется найти несколько элементов, индекс должен быть матрицей индексов.

5. Функция поиска элементов должна быть адаптирована к вызовам с разным числом выходных параметров.

6. Программа-сценарий вызова функции должна обеспечить, кроме вызова функций, вывод результатов на экран в виде таблицы значений $x(t, T)$,

значений суммы и произведения элементов ряда, искомых элементов и их индексов матрицы, а также графика $x(t)$.

Рекомендации к выполнению программы.

1. Функции пользователя не должны выполнять вывод результатов на экран, если они для этой цели не предназначены. Расчетные функции, как правило, не выводят никакой информации, кроме служебной, связанной с возможным возникновениям ошибок (например, деление на ноль).

2. Задавать исходные данные через ввод с клавиатуры необязательно, достаточно указать все необходимые параметры в коде программы-сценария.

3. Использование глобальных переменных без достаточных оснований не рекомендуется.

4. Вывод результатов на экран можно осуществлять по аналогии с лабораторной работой 3.

5. Для хранения в одной переменной данных разных типов (например, элементы матрицы и ее индексы) можно использовать структуры или ячейки.

ЛАБОРАТОРНАЯ РАБОТА 5. ФУНКЦИИ РАБОТЫ С ФАЙЛАМИ

Цель работы: освоить основные функции работы с файлами в среде MATLAB.

Основные сведения

Для хранения переменных в файлах MATLAB поддерживает специальный формат – MAT-файлы. Для записи и чтения переменных в файле используются команды save и load.

save filename X Y Z – сохранение переменных X Y Z в файле с именем filename.mat. Если не указать конкретные переменные, в файл запишутся все переменные рабочего пространства. Команда load используется аналогично и осуществляет чтение переменных из файла в рабочее пространство.

MATLAB поддерживает множество других функций для работы (чтение, запись) с текстовыми файлами. Функция DLMREAD позволяет считать данные из текстового файла с разделителем. Функция имеет следующие формы вызова:

```
RESULT = DLMREAD(FILENAME)
RESULT = DLMREAD(FILENAME, DELIMITER)
RESULT = DLMREAD(FILENAME, DELIMITER, R, C)
```

RESULT = DLMREAD(FILENAME, DELIMITER, RANGE)

Данные из файла с именем, указанным в текстовой строке FILENAME считываются в матрицу RESULT. DELIMITER – символ разделителя, числа R и C указывают позицию, откуда начинается считывание из файла (по умолчанию R = 0, C = 0). RANGE = [R1 C1 R2 C2] – указание начальной и конечной позиции при считывании. Если в файле, откуда считываются данные, есть пустые поля, функция DLMREAD заполнит их нулями, что позволяет считывать из файлов несимметрично хранящиеся данные. Обратная функция DLMWRITE позволяет записать в файл данные с разделителем:

DLMWRITE(FILENAME, M, DELIMITER, R, C);

Функция записывает в файл с именем FILENAME содержимое матрицы M с разделителем DELIMITER (необязательный параметр, разделитель по умолчанию – знак запятой). Необязательные параметры R, C используются также, как и при чтении. В качестве параметра можно также указать символьную константу ‘–append’, что позволит дописывать данные в конец уже существующего файла (в противном случае содержимое файла каждый раз с вызовом функции будет обновляться).

Для считывания и записи данных в определенном формате удобнее использовать функции, перенесенные в MATLAB из C – fscanf и fprimtf. Предварительно нужно осуществить операцию открытия файла:

FID = FOPEN(FILENAME, PERMISSION)

Имя файла указано в строке FILENAME, параметр-строка PERMISSION устанавливает режим работы с файлом: ‘r’ (по умолчанию) – файл открыт для чтения; ‘w’ – для записи; ‘a’ – для добавления (append); дополнительный символ ‘+’ к перечисленным параметрам откроет файл сразу для чтения и записи. В случае успешного открытия файла его идентификатор FID примет значение положительного целого числа (начиная от 3 и выше), при неудачной попытке FID будет иметь значение –1.

После открытия файла операции чтения и записи осуществляются с помощью идентификатора. Функция чтения имеет вид:

[A, COUNT] = FSCANF(FID, FORMAT, SIZE)

Считанные из файла с идентификатором FID данные записываются в матрицу A. Аргументы FORMAT, SIZE и выходное значение COUNT имеют то же назначение, что и в функции SSCANF (см. описание к лаб. раб. 3).

Функция форматной записи в файл FPRINTF также имеет вид аналогичный SPRINTF: FPRINTF(FID, FORMAT, A, ...). Если идентификатор FID не

указан, функция FPRINTF выводит содержимое форматированных аргументов в командное окно.

После окончания работы с функциями необходимо закрыть файл командой FCLOSE(FID). Команда FCLOSE('all') закроет все открытые файлы.

Методика выполнения работы.

1. Составить блок-схему алгоритма обработки данных из файла.

2. Написать и отладить программу считывания данных из файла-источника, расчета значений функций согласно таблице 4 и записи вычисленных значений в файл результатов.

Требования к выполнению программы и результатам

1. Исходные данные должны быть заданы в текстовом файле-источнике, в виде двух строк. Первая строка содержит три числовых параметра: начальное и конечное значение диапазона переменной x; вторая строка содержит три параметра для переменной ω, задаваемой функцией LOGSPACE.

2. Математические функции $f(t)$, $A(\omega)$, $\Phi(\omega)$ должны быть оформлены в виде функций пользователя.

3. Результаты расчета обеих функций должны быть записаны в файл результатов в виде двух столбцов: аргумент – значение.

4. Перед выводом таблиц значений функций должны быть заголовки.

5. Запись в файл должна быть реализована с помощью функции fprintf.

Рекомендации к выполнению программ

1. Для считывания данных из файла можно использовать любую функцию.

2. Функции пользователя целесообразно реализовать так, чтобы в качестве аргумента можно задавать вектор значений.

ЛАБОРАТОРНАЯ РАБОТА 6. ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ ГРАФИКИ В MATLAB.

Цель работы: изучение методов работы с объектно-ориентированной графикой в среде MATLAB.

Основные сведения

В лабораторной работе 1 были изучены простейшие средства для работы с графикой MATLAB. Использование средств объектно-ориентированного подхода к графике позволяет раскрыть ее возможности значительно шире.

Графическое окно в MATLAB (figure) представляет собой графический объект, обладающий рядом свойств, например, координаты его расположения на экране, ширина, высота, цвет фона и др. Внутри графического окна можно разместить другие графические объекты – оси, элементы управления. Оси используются для построения графиков, элементы управления – для создания приложений GUI (Graphic User Interface). В данной работе ограничимся построением графиков.

Связь графического окна (figure) с размещенными в нем осями (axes) строго иерархическая. Для осей графическое окно является предком, родительским объектом (parent), а объект axes – потомком, или дочерним объектом (children) графического окна. Внутри осей можно размещать линии (объект line) или текст (объект text), которые также являются дочерними по отношению к осям, в которых они размещены. Все эти объекты имеют свои свойства, которые доступны изменениям с помощью универсальных функций.

Для начала необходимо обеспечить доступ к графическим объектам. Доступ реализуется через механизм графических указателей (graphic handle). Чтобы создать графический указатель, можно использовать конструкции

```
h1 = figure(2);
h2 = axes;
h3 = plot(10, 10, '*');
```

При этом все указатели h1, h2, h3 указывают на различные объекты: h1 – на окно; h2 – на оси; h3 – на объект line (так как функция plot как раз реализует отрисовку линий). Для просмотра свойств графических объектов используется команда GET(H), где H – графический указатель. Эта же команда используется для просмотра значений любого свойства:

```
PROP = GET(H, 'PropertyName')
```

Самостоятельное задание: получить указатели родительских объектов для объекта с указателем h3 из примера. Убедиться по значениям идентификаторов указателей, что полученные указатели совпадают с h1 и h2.

Для изменения свойств любого графического объекта применяется функция SET, которая имеет вид

```
SET(H, 'PropName1', PropValue1, PropName1', PropValue2, …)
```

H – графический указатель объекта, далее последовательно перечисляются пары: наименование свойства объекта – новое значение. Число пар не ограничено. При изменении свойств необходимо следить за типом данных и размерностью значения.

Ниже приводится еще несколько полезных функций для работы с графическими указателями:

GCF(), GCA() – получение указателя на соответственно активную фигуру и активные оси. Активным считается последний созданный, вызванный или модифицированный объект.

LINE(), TEXT() – функции для создания графических объектов линий и текста, размещаемых внутри осей. Подобные объекты также называются графическими примитивами. Эти функции также возвращают графические указатели, их свойства можно менять, как свойства осей и графических окон.

Методика выполнения работы

1. В соответствие с вариантом задания (табл. 9) составить блок-схему алгоритма определения произвольного решения уравнения $ax+b=f(x)$.

2. Написать и отладить программу решения уравнения графическим способом. Оценить точность решений. В случае низкой точности (более 0.01) найти численное решение.

3. Отобразить решения, удовлетворяющие требуемой точности, средствами объектно-ориентированной графики.

4. Подобрать коэффициенты таким образом, чтобы уравнение имело единственное решение.

Рекомендации к выполнению работы

1. Для получения координат графика нужно использовать функцию

[X, Y] = GINPUT(n), где n – количество точек, задаваемых пользователем с помощью мыши; X, Y – векторы координат длиной n. Если указан один выходной параметр, координаты будут сформированы в виде матрицы размером n×2.

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. https://www.mathworks.com/help/matlab/index.html

2. В. Г. Потемкин. Введение в MATLAB (v 5.3)
http://matlab.exponenta.ru/ml/book1/index.php

ПРИЛОЖЕНИЕ 1. ВАРИАНТЫ ЗАДАНИЙ К ЛАБОРАТОРНЫМ РАБОТАМ

Таблица 4

№ варианта	$f(x)$	$A(\omega), \Phi(\omega), \quad \omega \in (0.01 \ldots 100)$
1	$f(x)=\frac{e^{x}+e^{-2x}}{x^{2}+1}\sin(x)$	$\Phi(\omega)=\Phi\left(\left.\frac{s^{3}+2s^{2}+3s+2}{s^{4}+3s^{3}+2s^{2}+2s+9}\right\|_{s=j\omega}\right)$
2	$f(x)=\frac{\ln\left(x^{4}+10\right)+e^{-x\cos(x)}}{4x^{2}+1}$	$A(\omega)=\left\|\left.\frac{s^{3}+2s^{2}+3s+2}{s^{4}+3s^{3}+2s^{2}+2s+9}\right\|_{s=j\omega}\right\|$
3	$f(x)=\frac{3}{5}\ln\left(\left\|\sin^{2}(x)\right\|+1\right)tg\left(x^{3}\right)$	$\Phi(\omega)=\Phi\left(\left.\frac{s^{2}+12s+1}{s^{4}-s^{3}+2s^{2}+6}\right\|_{s=j\omega}\right)$
4	$f(x)=\frac{3}{7}10^{-x^{2}\cos(x)}-\frac{1}{6}e^{-\|4-x\|}$	$A(\omega)=\left\|\left.\frac{s^{3}+12s+1}{s^{4}-s^{3}+2s^{2}+6}\right\|_{s=j\omega}\right\|$
5	$f(x)=\frac{\sqrt{x^{2}+6}-e^{-\|xtg(x)\|}}{2x^{4}+3}$	$\Phi(\omega)=\Phi\left(\left.\frac{3s^{3}+s+10}{s^{4}+3s^{3}+6s^{2}-8s+20}\right\|_{s=j\omega}\right)$
6	$f(x)=\frac{e^{x}+\lg\left(\|x+2\|\right)}{\sqrt{x^{4}+21}}\cos(x)$	$A(\omega)=\left\|\left.\frac{3s^{3}+s+10}{s^{4}+3s^{3}+6s^{2}-8s+20}\right\|_{s=j\omega}\right\|$
7	$f(x)=\frac{1}{6}2^{-x^{3}\sin(x)}-\frac{1}{3}e^{-\|2+x\|}$	$\Phi(\omega)=\Phi\left(\left.\frac{s^{3}+2s^{2}-1}{s^{4}+2s^{2}+9s+7}\right\|_{s=j\omega}\right)$
8	$f(x)=\frac{\left(\sqrt{x^{4}+10}\right)e^{-x\sin\left(x^{2}\right)}}{4x^{4}+2}$	$A(\omega)=\left\|\left.\frac{s^{3}+2s^{2}-1}{s^{4}+2s^{3}+9s+7}\right\|_{s=j\omega}\right\|$
9	$f(x)=x\|x\|-\frac{e^{x}+tg(x)}{x^{6}+16}$	$\Phi(\omega)=\Phi\left(\left.\frac{s^{2}+6s+8}{s^{4}-2s^{3}+5s+4}\right\|_{s=j\omega}\right)$
10	$f(x)=\frac{1}{4}e^{-x^{3}\sin(x)}-\frac{1}{3}\ln\left(x^{4}+1\right)$	$A(\omega)=\left\|\left.\frac{s^{2}+6s+8}{s^{4}-2s^{3}+5s+4}\right\|_{s=j\omega}\right\|$

Таблица 5

№ варианта	Вид функции *x(t, T)*	№ варианта	Вид функции *x(t, T)*
1	$\operatorname{sign}(t - T/2)$	6	$\left\lvert\frac{t+T}{2}\right\rvert$
2	$\begin{cases} 0, t < -T \\ 1, -T \le t \le T \\ 0, t > T \end{cases}$	7	$\begin{cases} 0, t < -T \\ -t, -T \le t \le T \\ 0, t > T \end{cases}$
3	$\begin{cases} 0, t < -T \\ 3+3t, -T \le t \le 0 \\ 3-3t, 0 \le t \le T \\ 0, t > T \end{cases}$	8	$\begin{cases} 0, t < -T \\ -3t, -T \le t \le 0 \\ 3t, 0 \le t \le T \\ 0, t > T \end{cases}$
4	$\begin{cases} 0, t < -T \\ -2-2t, -T \le t \le 0 \\ -2+2t, 0 \le t \le T \\ 0, t > T \end{cases}$	9	$\begin{cases} 0, t < -T \\ -3+3t, -T \le t \le 0 \\ -3-3t, 0 \le t \le T \\ 0, t > T \end{cases}$
5	$\begin{cases} 0, t < -T \\ 1-t, -T \le t \le T \\ 0, t > T \end{cases}$	10	$\begin{cases} 0, t < -T \\ t/4, -T \le t \le T \\ 0, t > T \end{cases}$

Таблица 6

№ варианта	Исходные данные			№ варианта	Исходные данные		
	Σ / Π	$g(x)$	условие		Σ / Π	$g(x)$	условие
1	Σ	$\frac{1}{1+\lvert x\rvert}$	$x = -6:6$	6	Π	$\frac{2x}{(x+1)^x}$	$(x+1)^x < 10^4$
2	Π	$\frac{1}{2^x}$	$2^{x+1} < 10^3$	7	Σ	$\frac{x^3}{x+6}$	$x = -5:5$
3	Σ	$\frac{x}{x+1}$	$x = 1:12$	8	Π	$\frac{2}{(2x+1)^3}$	$(2x-1)^3 < 10^3$
4	Π	$\frac{2x}{2x+1}$	$x^3 < 10^4$	9	Σ	$\frac{1}{2}\left(\frac{1}{x}+x\right)$	$x = 1:16$
5	Σ	$\frac{1}{x}+\frac{1}{x^2}$	$x = 1:10$	10	Π	$\frac{x-1.5}{(x+1)^3}$	$e^x < 10^4$

Таблица 7

№ варианта	Исходные данные	Результат поиска
1	$\begin{bmatrix} -3 & 2 & 9 & -6.5 \\ 0 & 1 & 10 & 2 \\ -3 & -4.4 & 3 & 2.4 \end{bmatrix}$	минимальный элемент
2	$\begin{bmatrix} 1 & -4.2 & 1 & 5 \\ 0 & 11 & -6.1 & 2 \\ 0 & 4.3 & 8 & 3.1 \end{bmatrix}$	строка с минимальным элементом
3	$\begin{bmatrix} -1.1 & 4 & -11 & 6 \\ 3 & 0 & 5 & 2.3 \\ 6.2 & -0.3 & 3 & -6.4 \end{bmatrix}$	столбец с минимальным элементом
4	$\begin{bmatrix} 0.4 & -4.2 & 2 & -7.5 \\ 4 & 1.5 & -5.1 & 0.6 \\ -3 & 3.7 & -5.2 & 2.6 \end{bmatrix}$	максимальный элемент
5	$\begin{bmatrix} 3 & -2 & 0.6 & 6.2 \\ 0 & 4 & 1.2 & 12 \\ 9 & -7.4 & -3 & -4.2 \end{bmatrix}$	строка с максимальным элементом
6	$\begin{bmatrix} -0.3 & 1.6 & 2 & 0 \\ 0.2 & -12 & 1.8 & 2.4 \\ -3 & -1.4 & 3.6 & -6.4 \end{bmatrix}$	столбец с максимальным элементом
7	$\begin{bmatrix} 1.9 & -4.2 & 3.4 & -6.3 \\ 10 & 4.8 & 5.2 & 8.2 \\ 6.2 & -7.2 & -9.3 & -4.6 \end{bmatrix}$	произведение столбца и строки с минимальным элементом
8	$\begin{bmatrix} 6.3 & -5 & 4 & -6.5 \\ 4.2 & -1.2 & 7.8 & 2.7 \\ 3.9 & 4.3 & 3.3 & -2.4 \end{bmatrix}$	отношение максимального и минимального элемента
9	$\begin{bmatrix} -3 & -4.2 & 5.6 & -4.2 \\ 0.7 & 6.4 & 1.8 & 1.5 \\ 2.9 & 3.5 & -1.1 & -4.4 \end{bmatrix}$	разность максимального и минимального элемента
10	$\begin{bmatrix} -3.3 & 0 & 2.2 & 0.8 \\ 7.2 & -12 & 0.8 & -2.4 \\ -2 & -1.4 & 3.8 & -1.6 \end{bmatrix}$	произведение максимального и минимального элемента

Таблица 8

№ варианта	Эмулятор календаря	Работа со строками
1	Ввести дату в формате ДД.ММ.ГГГГ. Вывести на экран дату следующего дня	Написать программу подсчета видимых символов в строке
2	Ввести дату в формате ДД.ММ.ГГГГ. Вывести на экран дату дня через неделю	Написать программу подсчета слов в строке
3	Ввести дату в формате ДД.ММ. Вывести на экран день недели *	Написать программу подсчета английских гласных букв в строке
4	Ввести дату в формате ДД.ММ. Вывести на экран номер недели *	Написать программу подсчета цифр в строке
5	Ввести дату в формате ДД.ММ.ГГ. Вывести на экран дату дня через день	Написать программу подсчета нулей и единиц в строке
6	Задать месяц, день недели в месяце и номер недели в формате ММ–Д/Н. Вывести на экран дату *	Написать программу подсчета нечетных цифр в строке
7	Задать месяц и день недели в формате ММ–Д Вывести на экран все даты, соответствующие нечетным неделям *	Написать программу, копирующую на экран цифры из символьной строки
8	Задать дату в формате ДД.ММ.ГГГГ. Вывести на экран дату предыдущего дня	Написать программу подсчета нулей и единиц в строке
9	Задать дату в формате ДД.ММ. Вывести номер недели в месяце*	Написать программу, копирующую на экран строку без пробелов
10	Задать месяц и день недели в формате ММ–Д Вывести на экран все соответствующие даты *	Написать программу подсчета знаков препинания в строке
	* – для текущего года	

Таблица 9

№ варианта	a	b	$f(x)$
1	0.1	0.5	$\sin x+\cos 2x$
2	0.3	4	$3\sin x-\cos 3x+3$
3	–0.6	4.6	$\sin x/2-\cos x+1$
4	0.3	–0.6	$\sin 5x/2-\cos x/2+1$
5	0.04	3.2	$4-0.3\sin 5x-\cos x$
6	–0.09	2.4	$2-0.3\sin 5x+\cos x/3$
7	–0.25	–1	$\cos x+\sin(2x)-2$
8	0.15	0.01	$\sin 3x/2+\sin 5x/2+0.9$
9	0.35	–5	$\sin 3x+2\cos x-4$
10	–0.4	4	$2\sin x/2-\cos 4x+1.5$

Содержание

Программирование и основы алгоритмизации

Учебно-методическое пособие к лабораторным работам

Редактор / /

Подписано в печать . Формат 60 × 84 1/16.
Бумага офсетная. Печать офсетная. Печ. л. 2,5.
Тираж 59 экз. Заказ .

Издательство СПбГЭТУ «ЛЭТИ»
197376, С.-Петербург, ул. Проф. Попова, 5