Лабораторная работа №1

Знакомство с пакетом Matlab

Цель работы: изучение пакета Matlab; приобретение навыков программирования; приобретение навыков использования стандартных средств Matlab.

Теоретические сведения

Описание пакета. Matlab (сокращение от англ. «Matrix Laboratory») - пакет прикладных программ для решения задач технических вычислений, а также используемый в этом пакете язык программирования. Matlab работает на большинстве современных операционных систем, включая Linux, Mac OS, Solaris и Windows. Matlab как язык программирования был разработан в конце 1970-х годов деканом факультета компьютерных наук в Университете Нью-Мексико Кливом Моулером. Целью разработки служила задача дать студентам факультета возможность использования программных библиотек Linpack и EISPACK без необходимости изучения Фортрана. В 1983 году пакет был переработан и переписан на языке С Кливом Моулером, Джоном Литтлом и Стивом Бангертом, они и основали в 1984 году компанию The MathWorks для дальнейшего развития программного продукта. Эти переписанные на С библиотеки долгое время были известны под именем JACKPAC. Первоначально Matlab предназначался для проектирования систем управления, но быстро завоевал популярность во многих других научных и инженерных областях. Он также широко используется в образовании для преподавания линейной алгебры и численных методов. Язык Matlab представляет собой высокоуровневый язык программирования, включающий основанные на матрицах структуры данных, широкий спектр функций, интегрированную среду разработки, объектно-ориентированные возможности и интерфейсы к программам, написанным на других языках программирования. Основной особенностью языка Matlab является его широкие возможности по работе с матрицами, которые создатели языка выразили в лозунге «Думай векторно» (англ. Think vectorized).

При вызове Matlab на дисплей выводится заставка, которая сменяется Командным окном (Command Window) и окнами История команд (Command History) и Рабочая область (Workspace), в верхней части экрана размещена панель управления - меню с пунктами Файл (File), Правка (Edit), Вид (View), Окно (Window), Web и Помощь (Help) и панель инструментов. В окне Command History отображаются все введенные в окне Command Window за время работы команды, пролистывая это окно можно повторить ввод той или иной команды. Пошагово эти команды можно отображать в Командном окне при помощи клавиш ↑ и ↓. В окне Workspace отображаются все переменные и массивы, которые были использованы за время работы Matlab, и их характеристики. В Command Window выводится командная строка (начинается символом ») с предварительными предложениями вызвать перечень разделов, войти в справочник, открыть окно помощи, приступить к демонстрации и др. В командной строке в режиме диалога можно набрать команду (оператор) или выражение и, нажав клавишу Enter, получить ответ (answer). Например, после набора команды (оператора присваивания) $a = 5.3$ в последующих строках появится $a = 5.3000$ (переменной a присвоено значение 5.3). При вводе переменных необходимо учитывать, что их имена представляют собой последовательности латинских букв и цифр, начинающихся с буквы (знак _ относится к буквам), а также и то, что строчные и заглавные буквы не тождественны. Если имеется необходимость выполнить команду без вывода результата, то в конце команды ставится символ **;**. Если команда не помещается полностью в видимой части одной строки экрана, ставится многоточие (хотя бы две точки), после чего надо нажать клавишу Enter и продолжить ввод в следующей строке. Для очистки Command Window необходимо выполнить команду **clr**.

Для очистки Workspace полностью используется команда **clear**, для частичной очистки - команда **clear** <список имен>. В командном окне всегда можно обратиться к помощи, набрав команду **»help**. Выбрав **»help** <раздел (topic)>, можно получить помощь по командам соответствующего раздела. Для помощи в синтаксисе необходимо набрать **»help** <команда (функция)>.

Рассмотрим некоторые полезные пункты меню Matlab, а именно подпункт Свойства (Preference) пункта Файл (File) окна управления. В окне (Command Window) предусматривается установка формата представления чисел (Numeric Format): **short** - короткое 5-значное; **long** - длинное 15-значное; **Hex** - шестнадцатеричное; **bank** - доллары и центы; **short e** и **long e** - экспоненциальное; **rational** - отношение целых чисел и «+» - представление знаков чисел; а также формата представления межстрочного интервала (Numerical Display): с пробелом между строками

(**loose)** или без (**compact)**. Изменение формата отображения данных можно достичь использованием операторов в командной строке в виде: format <параметр>. Например, »format short e. Другим полезным пунктом является подпункт Demo пункта меню Help. В этом пункте можно запустить программу демонстрации возможностей Matlab и посмотреть некоторые наиболее употребительные приемы работы.

Как и в любой системе, в Matlab присутствует понятие переменной величины, но в роли ее значения выступает массив (array). В системе определены 6 встроенных типов данных (массивов): числа удвоенной точности (**double**); массивы символов - строки (**char**), при задании строковой константы ее символы заключают в апострофы; двумерные разреженные матрицы (**sparse**); массивы ячеек (**cell**); массивы записей (**struct**); специальные массивы целых чисел от 0 до 255 (**uint8**). Здесь ограничимся рассмотрением лишь обычных числовых массивов и строк.

Режим программирования. Для перехода в режим программирования в окне управления необходимо выбрать пункт File и войти в редактор Matlab: New (создать новый m-файл) или Open (открыть существующий файл с расширением **.m**, так называемых m-файлов). В дальнейшем могут осуществляться обычный набор текста программы или его корректура и действия в соответствии с меню (сохранение под текущим или другим именем, запуск на исполнение в обычном и отладочном режимах и др.).

Различают два вида m-файлов: m-сценарии и m-функции. m-сценарий - это файл, содержащий последовательность команд и комментариев (строк, начинающихся символом %), и пользующийся данными из рабочей области. Заголовок его начинается командой **script** или может отсутствовать. Для m-функции допускаются входные и выходные аргументы, локализация внутренних ее переменных и возможность обращения к ней из других программ. m-функции включаются в библиотеку функций системы в виде текстовых файлов. Заголовок m-функции имеет следующий вид: **function [**<список выходных переменных>**]=**<**имя**>**(**<список входных переменных>**).** Кроме использованных выше операторов присваивания, программирование в Matlab допускает и ряд других традиционных для программных сред операторов.

Условный оператор выступает в одной из следующих форм:

```
if <условие>          if <условие>          if <условие>
      <команды>             <команды>             <команды>
end                   else                  elseif <условие>
                            <команды>             <команды>
                      end                   else
                                                  <команды>
                                            end
```

В роли условия может использоваться любое логическое выражение, построенное на основе операций отношения и логических операций. Если значение этого выражения является массивом, то условие считается истинным, если все его элементы истинны (истина - 1, ложь - 0).

Оператор цикла с заданным числом повторений используется в форме:

```
for v=a:h:b           for v=a:b
      <команды>             <команды>
end                   end
```

(v - переменная/параметр цикла; a, b - начальное и конечные значения; h - приращение, по умолчанию 1). Допускаются и вложенные циклы. В заголовке цикла можно использовать одномерный массив.

Оператор цикла с предусловием имеет традиционную конструкцию:

```
while <условие>
      <команды>
end
```

и обеспечивает выполнение команд тела цикла, пока истинно проверяемое условие. Отметим, что работа цикла может быть прервана (выход из внутреннего цикла) оператором **break**:

Оператор переключения обобщает условный оператор на случай более двух условий и имеет конструкцию:

```
switch  <выражение>
case    <значение 1>
        <команды>
case    <значение 2>
        <команды>
```

. . . . .

otherwise % может отсутствовать

<команды>

end

Контрольные значения проверяются на равенство и могут задаваться списком. Выход из функции в вызывающую программу обеспечивается выполнением последнего ее оператора или командой **return**. Кроме упомянутых основных операторов, традиционных для любой системы программирования, остановимся на ряде операторов обеспечения пользовательского интерфейса. Ввод с клавиатуры реализуется командой вида

<переменная>=**input**(‘подсказка’)

Приостановка выполнения программы может быть предусмотрена включением в текст команды **pause** (приостановка до нажатия любой клавиши), **pause(n)** (приостановка на n секунд), **keyboard** (приостановка с возможностью выполнять практически любые команды и последующим возвратом в программу командой **return**).

Можно построить выбор варианта с клавиатуры созданием меню:

<переменная>=**menu**(‘заголовок’,’выбор1’,’выбор2’,...)

Из многообразия встроенных математических функций следует отметь некоторые наиболее используемые. Аргументами большинства из приведенных ниже функций являются не только скаляры, но и массивы:

pi = 3.1415926535897..;

abs(x) - абсолютная величина;

angle(x) - аргумент комплексного числа (из диапазона [-p; p]);

real(x), **imag**(x) - действительная и мнимая часть числа;

conj(x) - комплексно-сопряженное значение;

sign(x) - знак числа (для комплексных x/|x|);

sqrt(x) - квадратный корень;

exp(x) - экспонента;

pow2(x) - двоичная экспонента 2^x;

log(x), **log2**(x), **log10**(x) - натуральный логарифм, логарифмы по основанию 2 и 10;

sin(x), **cos**(x), **tan**(x), **cot**(x), **csc**(x), **sec**(x) - тригонометрические функции (синус, косинус, тангенс, котангенс, косеканс, секанс);

asin(x), **acos**(x), **atan**(x), **acot**(x), **acsc**(x), **asec**(x) - обратные тригонометрические функции (арксинус, арккосинус и т.д.);

sinh(x), **cosh**(x), **tanh**(x), **coth**(x), **csch**(x), **sech**(x) - гиперболические функции (синус, косинус, тангенс, котангенс, косеканс, секанс);

asinh(x), **acosh**(x), **atanh**(x), **acoth**(x), **acsch**(x), **asech**(x) - обратные гиперболические функции.

Также предусмотрены операторы преобразования координат: из декартовых в полярные, из декартовой системы в цилиндрическую, из декартовой системы в сферическую, из полярной и цилиндрической в декартову, из сферической в декартову. Имеется возможность применения специальных функций (интеграл ошибок, бета- и гаммафункции, цилиндрические функции Бесселя, Неймана, Ханкеля, функции Эйри, эллиптические функции Якоби и эллиптические интегралы, интегральная показательная функция, присоединенные функции Лежандра и много других функций, полезных при изучении физических процессов), функций линейной алгебры, аппроксимации данных, численного интегрирования, поиска корней уравнений, обслуживания графики, обработки дат, множеств и др.

Особую роль в Matlab играют массивы, ввод которых осуществляется прямым (построчным) перечислением его элементов подобно A=[1 3 5 7; 4 5 6 7] (2 строки и 4 столбца), B=[1;3;5;7] (столбец с 4 элементами) или заданием диапазона значений с заданным (или умалчиваемым единичным) шагом [1:2:7], [4:7], [[1:2:7];[4:7]] и т.п. Доступ к элементам или блокам элементов массива производится указанием индексов или массива индексов: A(2,k) - элемент второй строки и k-го столбца; A(:,k) - k-й столбец; A(1:3;1:4) - подматрица из первых 3 строк и 4 столбцов матрицы; C(:,:,12) - 12-я страница трехмерного массива. Следует учесть, что хранение массивов в памяти ведется по столбцам. Поэтому возможна работа с созданным многомерным массивом как с одномерным, например A(:) - вектор-столбец из всех элементов массива A, A(4:7) - столбец из элементов с номерами от 4 до 7. Имеется возможность объединять массивы «по горизонтали» - [A,B,C] или [A B C] (массивы с одинаковым числом строк) и «по вертикали» - [A;B;C] (массивы с одинаковым числом столбцов). Из вектора можно удалить одинаковые элементы функцией

unique(x). Существует возможность объединения множеств - **union**(x,Y), пересечения - **intersect**(x,y), разности - **setdiff**(x,y). Функция **find** дает поиск по условию элементов одно- или двумерного массива в формате команд k=**find**(x<условие>), [i,j]=**find**(A<условие>) (если условия нет, отыскиваются ненулевые элементы). Для определения длины вектора используется функция **length**(x), для размеров массива - функцию **size**(A).

Суммирование и умножение элементов массива можно реализовать функциями **sum**(A) и **prod**(A) (для двумерного массива выполняется поиск сумм и произведений по столбцам). С помощью функций **sum**(A,dim) и **prod**(A,dim) можно выполнить операции по измерению dim. Функцию **sum** часто используют для поиска скалярного произведения векторов. Сортировку элементов массива по возрастанию можно выполнить функцией **sort**(A,dim), причем команда [B,I]=**sort**(A) выдает и список индексов. Сортировку по убыванию можно выполнить аналогичной функцией **sortrows**.

Иногда могут быть полезными функции начального задания:

zeros(n), **zeros**(m,n), **zeros**(**size**(A)) - формирование массива нулей (одномерного, двумерного, соразмерного с массивом A); допустимо формирование массива и большей размерности **zeros**(m,n,p,...);

ones(n), **ones**(m,n), **ones**(**size**(A)) - формирование массива единиц;

rand(n), **rand**(m,n), **rand**(**size**(A)) - формирование массива чисел с равномерным законом распределения в (0,1);

randn(n), **randn**(m,n), **randn**(**size**(A)) - формирование массива чисел с нормальным законом распределения (Mx=0, Dx=1);

eye(n), **eye**(m,n), **eye**(**size**(A)) - формирование единичной матрицы (n×n, m×n, соразмерной с матрицей A).

Отметим также и некоторые полезные конструкции:

cross(x,y) - векторное произведение (x, y - трехмерные векторы);

kron(x,y) - тензорное произведение (произведение Кронекера);

meshgrid(x,y), **meshgrid**(x,y,z) - формирование двумерной (трехмерной) сетки (обычно используется при реализации графики): [x,y]=**meshgrid**(–2:0.1:2, –10:0.5:10).

Построение графиков. Здесь рассмотрим лишь небольшую часть графических команд высокого уровня, не затрагивая графические объекты (axes, line, patch, surface, text). Для построения графика создается графическое окно. Создание такого окна задается командой **figure**(n), можно выбирать некоторое из созданных окон в качестве текущего. Включение (выключение) режима сохранения текущего графика: **hold on/off**, **hold.**

Двумерная графика. Для построения графика в линейном масштабе используются операторы:

plot(y) - строит график одномерного массива в зависимости от номера элемента (для двумерного массива строятся графики для столбцов);

plot(x,y) - строит график функции y=y(x);

plot(x,y,LineSpec) - заданием строки LineSpec (до 3 символов) определяет стиль линий, форму маркера точек и цвет линий и маркера: Непрерывная **-;** Штриховая --; Двойной пунктир : ; Штрихпунктирная -.; Желтый y; Фиолетовый m; Голубой c; Красный r; Зеленый g; Синий b; Белый w; Черный k. Маркер может определяться символами: **.** **+** ***** ° × **s** (квадрат), **d** (ромб), **p** (пятиугольник), **h** (шестиугольник), **v** ^ < > (стрелки). По умолчанию выбирается непрерывная линия с точечным маркером и чередованием цветов с желтого по синий; **plot**(x1,y1,LineSpec1,x1,y1,LineSpec2,...) - строит на одном графике несколько линий (диапазон по аргументу - объединение x1 и x2.

График в логарифмическом масштабе задается функцией **loglog** с тем же набором параметров, что и **plot**, с той лишь разницей, что проводится масштабирование десятичным логарифмированием по обеим координатам. График в полулогарифмическом масштабе задается функциями **semilogx** и **semilogy** с тем же набором параметров, что и **plot** (проводится масштабирование логарифмированием по одной из координат).

График с двумя осями ординат (одна отображается слева, другая справа) реализуется функцией **plotyy**(x1,y1,x2,y2) и той же функцией с добавлением параметров масштабирования ‘f1’ или ‘f1’,’f2’, в роли которых могут выступать plot, semilogx, semilogx, loglog.

Также используются операторы:

fplot(<имя функции>,limits) - строит график функции (функций) в интервале limits=[xmin,xmax]. В качестве имени функции может использоваться m-файл или строка типа

‘sin(x)’, ‘[sin(x)cos(x)]’, ‘[sin(x),myfun1(x),myfun2(x)]’. Можно установить размеры графика по оси значений функции limits=[xmin,xmax,ymin.ymax].

fplot(<имя функции>,limits,eps) - строит график с относительной погрешностью eps (по умолчанию 0.002) и максимальным числом шагов (1/eps) + 1. Эту конструкцию можно дополнить четвертым параметром n (n + 1 - минимальное число точек) и параметром LineSpec.

ezplot('f(x)') - строит график f(x), заданный символьным выражением, с выводом выражения в качестве заголовка графика.

ezplot('f(x)',limits) и **ezplot**('f(x)',limits,fig) - строят график f(x) на указанном интервале и в заданном окне.

Трехмерная графика. В трехмерной графике выполняются представления функции $z = z(x, y)$, отличающиеся способом соединения точек: линия, сечения, сетчатая или сплошная поверхность: **plot3**(x,y,z) в тех же вариациях, что и **plot**, предполагает задание одномерных и двумерных массивов (строятся точки с координатами x(i,:),y(i,:),z(i,:) для каждого столбца, которые соединяются прямыми линиями. Если используется [x,y]=**meshgrid**(...), то строятся сечения; **mesh**(x,y,z,c), **mesh**(z,c), **mesh**(z) определяют задание сетчатой поверхности (массив **c** определяет цвета узлов поверхности), если x, y не указаны, то x=1:n, y=1:m, где [m,n]=**size**(z). Аналогичный оператор **meshc** в дополнение к поверхности строит проекции линий уровня, а **meshz** делает срез поверхности до нулевого уровня (своеобразный пьедестал); **surf**(x,y,z,c), **surf**(z,c), **surf**(z) определяют задание сплошной поверхности, отличаясь от mesh системой окраски; аналогичный оператор **surfc**(...) задает проекции линий уровня. Оператор **contour** рисует только линии уровня соответствующих поверхностей и выступает в многообразии синтаксических форм: **contour**(x,y,z) - для массива z = z(x,y), **contour**(x,y,z,n) - то же с указанием числа линий уровня (по умолчанию 10), **contour**(x,y,z,v) - то же для массива указанных значений; **contour**(z), **contour**(z,n**),** **contour**(z,v) - аналогичные функции без указания диапазонов для аргументов и **contour**(...,LineSpec) - аналогичные функции с указанием типа и цвета линий (см. **plot**); [C,h]=**contour (...)** возвращает массив C и вектор дескрипторов, позволяя тем самым продолжить работу с рисунком (давать оцифровку линий, заголовки и др.). Оператор **contourf(...)** закрашивает области между линиями уровня, аналогично **contourf**(...), с разницей в формате[C,h,cf]=**contour(...),** где cf определяет матрицу раскраски. Оператор **contour3**(...) по синтаксису полностью аналогичен **contour**(...), но изображает не проекции линий уровня, а рисует их в пространственной интерпретации.

Символьными (или аналитическими) операциями называются такие операции, когда вычисления задаются в виде символьных (формульных) выражений и результаты вычислений также получаются в символьном виде. В настоящее время имеется возможность выполнять символьные операции на компьютере. Для этого разработаны различные программные системы, такие, как Reduce, Maple, Mathematica. Эти системы способны преобразовывать алгебраические выражения, находить аналитические решения систем линейных, нелинейных и дифференциальных уравнений, манипулировать полиномами, вычислять производные и интегралы, анализировать функции и находить их пределы и т.д. К символьным вычислениям относят также численные расчеты с произвольным числом цифр результатов и с отсутствующей погрешностью, поскольку это требует символьного представления чисел и особых алгоритмов выполнения операций с ними.

Создание символьных переменных. Поскольку переменные системы Matlab по умолчанию не определены и задаются как векторные, матричные, числовые и т.д., т.е. не имеющие отношения к символьной математике, то для реализации символьных вычислений нужно, прежде всего, позаботиться о создании специальных символьных переменных. В простейшем случае их можно определить как строковые переменные, заключив имена в апострофы. Например, при вводе в окне управления команды (sin('x')^2 + cos('x')^2) и нажатии клавиши Enter получим результат ans = 1.

Для создания символьных переменных или объектов используется функция **sym:**

x=sym('x') - возвращает символьную переменную с именем 'x' и записывает результат в x;

x=sym('x','real') - возвращает символьную переменную вещественного типа с именем 'x' и записывает результат в x (в общем случае символьные переменные рассматриваются как комплексные);

x=sym('x','unreal') - возвращает символьную переменную мнимого типа с именем '**x**' и записывает результат в x.

Для создания группы символьных переменных или объектов используется функция **syms**. **syms x1, x2, …** создает группу символьных объектов, подобную выражениям **x1=sym('x1'), x2=sym('x2'),…; syms x1, x2, …, real** и **syms x1, x2, …, unreal**, которые создают группы символьных объектов с вещественными (real) и невещественными (unreal) значениями. Последнюю функцию можно использовать для отмены задания вещественных объектов.

Вывод символьного выражения. Функция **pretty**(s) дает вывод выражения **s** в формате, приближенном к математическому; **pretty**(s,n) аналогична предшествующей функции, но задает вывод выражения **s** в **n** позициях строки (по умолчанию n = 79).

Упрощение выражений. Функция z=**simplify**(s) поэлементно упрощает символьные выражения массива s. Если упрощение невозможно, то возвращается исходное выражение. Дополнительные возможности обеспечивает функция **simple**(s), которая выполняет различные упрощения для элементов массива s и выводит как промежуточные результаты, так и самый короткий конечный результат. При обращении к функции **simple** в форме [R,HOW]=**simple**(s) промежуточные результаты не выводятся. Конечные результаты упрощений содержатся в векторе R, а в строковом векторе HOW указывается выполняемое преобразование.

Вычисление производных. Для вычисления в символьном виде производных от выражения s служит функция **diff**:

diff(s) - возвращает символьное значение первой производной от символьного выражения или массива символьных выражений s по независимой переменной, определенной функцией **findsym**;

diff(s,n) - возвращает символьное значение n-й производной от символьного выражения или массива символьных выражений s по независимой переменной, определенной функцией **findsym**;

diff(s,V) или **diff**(s,**sym**(V)) - возвращает символьное значение первой производной от символьного выражения или массива символьных выражений **s** по переменной v;

diff(s,V,n) или **diff**(s,n,V) - возвращает символьное значение n-й производной от символьного выражения или массива символьных выражений s по переменной V.

Вычисление интегралов. Для вычисления определенных и неопределенных интегралов в символьном виде служит функция **int**:

int(s) - возвращает символьное значение неопределенного интеграла от символьного выражения или массива символьных выражений **s** по переменной, которая автоматически определяется функцией **findsum**. Если s - константа, то вычисляется интеграл по переменной 'x';

int(s,a,b) - возвращает символьное значение определенного интеграла на отрезке интегрирования [a,b] от символьного выражения или массива символьных выражений s по переменной, которая автоматически определяется функцией **findsum**. Пределы интегрирования a, b могут быть как символьными, так и числовыми;

int(s,v) - возвращает символьное значение неопределенного интеграла от символьного выражения или массива символьных выражений **s** по переменной **v**;

int(s,v,a,b) - возвращает символьное значение определенного интеграла от символьного выражения или массива символьных выражений **s** по переменной **v** с пределами интегрирования [a,b].

Вычисление определителя матрицы, обращение матрицы. Функция **det**(X) вычисляет определитель (детерминант) квадратной матрицы X в символьном виде. Для обращения (инвертирования) матрицы в символьном виде используется функция **inv**(X).

Существует также возможность вычисления пределов, разложения в ряды Тейлора и некоторые другие операции. Для получения численного результата от символьной операции используется оператор **double**.

Порядок выполнения работы

1. Изучить краткое описание пакета.
2. Выполнить задания к лабораторной работе.

Содержание отчета

Отчет должен содержать:

1) результаты, полученные по каждому заданию и сохраненные в файле name.doc (name - имя или фамилия выполняющего работу);

2) тексты созданных самостоятельно *m*-файлов на языке Matlab;

3) графики функций, построенных в соответствующих заданиях.

Задание

1. Ввести вектор *x* от 0 до 10 с шагом 0,1. Вычислить значения функций $y = x\cos x$ и $y1 = x\sin x$. Построить графики данных функций *y*(*x*) и *y*1(*x*).
2. Ввести вектор *x* от 0 до 1 с шагом 0,01, вектор *y* от 0 до 2 с шагом 0,2. Вычислить функцию $z = \exp\left(-10\left((x-0,5)^2 + (y-1)^2\right)\right)$, построить ее 3D-график.
3. Ввести матрицу A размерности 3x3, полагая A(*i*, *j*) = *i* + 2×*j* – 1. Сформировать матрицу A1, копию матрицы A.
4. Найти матрицу A2, удалив из матрицы A 1-ю строку. Найти матрицу A3 удалением из матрицы A 3-го столбца.
5. Образовать матрицу A4, добавив к матрице A 4-й столбец, равный вектору *a* = [7 8 9].
6. Найти матрицу B, транспонированную к матрице A.
7. С помощью условных операторов найти элементы матрицы A большие 5.
8. Найти вектор *d*, содержащий числа, которые находятся на главной диагонали матрицы A, сформировать диагональную матрицу D, содержащую нулевые элементы и диагональ A. Найти след, собственные числа, определитель и ранг матрицы A.
9. Найти матрицу A1, обратную матрице A. Если эта операция невыполнима, переопределите какой-либо элемент матрицы A, чтобы сделать ее допустимой.
10. Ввести символьную матрицу *F* = [*a b*; *c d*]. Найти det(*F*), F^{-1}.
11. Ввести символьную функцию *f* (*x*) = *x* cos *x*, определить $f'(x)$, $\int f(x)dx$, вычислить интеграл на отрезке [0; 1].

Контрольные вопросы

1. Чем отличается script-файл от файла-функции?
2. Как осуществляется редактирование m-файла с помощью меню Edit?
3. Что такое входные и выходные переменные файла-функции?
4. Как устроены безусловные и условные цикловые операции?
5. Указать способы формирования матрицы на языке Matlab.
6. Как осуществляется структурирование матрицы: выделение заданной подматрицы, удаление строк или столбцов, преобразование матрицы в вектор и обратно?
7. Как осуществляется конкатенация матриц, т.е. объединение нескольких малых матриц в одну большую?
8. Объяснить действие следующих функций обработки множеств: unique, intersect, union, setdiff.

Литература

Гончаров В.А., Земсков В.Н., Яковлев В.Б. Лабораторный практикум по курсу «Вычислительная математика». – М.: МИЭТ, 2008.