ТАШКЕНТСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
имени АБУ РАЙХОНА БЕРУНИЙ

факультет «Энергетика»

Кафедры «Гидравлика ва гидроэнергетика»

на тему: «Основы программирования на Matlab»

Дипломная работа

для получения ученой степени

бакалаврията

Выполнил: ***Максудов Т.Р.***

Проверил: ***Шокиров А.О***

Ташкент – 2012

Введение

Одним из наиболее мощных и универсальных пакетов прикладных программ, обеспечивающих решение типовых математических задач, возникающих в различных областях человеческой деятельности, является пакет MATLAB фирмы MathWorks.

Систему MATLAB разработал Молер (C.B. Moler) в 70-х г. г. XX века, которая использовалась на больших ЭВМ. В начале 80-х г. г. Джон Литл (John Little) из фирмы Math Works, Inc. Модернизировал эту систему для персональных компьютеров типа IBM PC, VAX и Macintosh. Далее к расширению системы были привлечены крупнейшие ученые и научные школы в математике, программировании и естествознании. Это позволило MATLAB стать признанным лидером в решении различных проблем науки и техники среди других подобных систем.

Спектр численных методов математического анализа, которые реализованы в пакете MATLAB, весьма широк и охватывает методы численного интегрирования, интерполяции и приближения функций, линейной алгебры, решения систем нелинейных уравнений и обыкновенных дифференциальных уравнений, уравнений математической физики, задач оптимизации, нечеткой логики, искусственных нейронных сетей и др.

Пакет MATLAB обладает хорошо развитыми возможностями интерпретации двумерных и трехмерных массивов данных, снабжен встроенным языком программирования, который вобрал в себя преимущества традиционных языков и позволяющим сравнительно легко создавать собственные программы. Пользователь пакета MATLAB может также в процессе работы совершенствовать свои знания как в области компьютерного моделирования и численных методов, так и программирования и визуализации результатов расчета. Более подробно о преимуществах и возможностях системы MATLAB можно узнать в специализированных изданиях указанных в списке литературы.

1. MATLAB в режиме командной строки

Система MATLAB создана таким образом, что любые (подчас весьма сложные) вычисления можно выполнять как в режиме прямых вычислений (то есть без подготовки программы), так и посредством создания программы (*m-файла*). Работа с системой в режиме прямых вычислений носит диалоговый характер и происходит по правилу "задал вопрос, получил ответ". Пользователь набирает на клавиатуре вычисляемое выражение,

редактирует его (если нужно) в командной строке и завершает ввод нажатием клавиши ***ENTER***.

Ниже перечислены некоторые особенности работы в режиме командной строки:

a) диалог происходит в стиле "задал вопрос - получил ответ";
b) для указания ввода исходных данных используется символ **>>**;
c) данные вводятся с помощью простейшего строчного редактора;
d) для блокировки вывода результата вычислений некоторого выражения после него надо установить знак **;** (точка с запятой)
e) если не указана переменная для значения результата вычислений, то в системе MATLAB по умолчанию назначается переменная с именем ***ans***;
f) знаком присваивания является знак равенства **=**;
g) результат вычислений выводится в строках вывода (без знака);
h) встроенные функции (***sin, cos*** и др.) записываются строчными буквами, и их аргументы указываются в круглых скобках, например ***sin(a)*** или ***cos(30)***;
i) длинное выражение для удобства восприятия можно перенести на другую строку с помощью многоточия **…** .

2. Основные объекты MATLAB

2.1 Переменные и присваивание им значений

Переменные - это имеющие имена объекты, способные хранить некоторые, обычно разные по значению, данные. В зависимости от этих данных переменные могут быть числовыми или символьными, векторными или матричными. В системе MATLAB можно задавать переменным определенные значения. Для этого используется операция присваивания, вводимая знаком равенства ***Имя переменной = Выражение***

Например, ***a = 5*** ; ***b = 4 + c; s = “ текст”; v = [2, 3];***
m_1 = [3.2, 4.1; 2, 3] .

Типы переменных заранее не декларируются. Они определяются выражением, значение которого присваивается переменной. Так, если это выражение - вектор или матрица, то переменная будет векторной (***v***) или матричной (***m_1***).

Исторически MATLAB создавалась как матричная лаборатория, что и отразилось в названии (***MATrix LAB****oratory*) и, поэтому, MATLAB - система, специально предназначенная для проведения сложных вычислений с векторами, матрицами (массивами). При этом она по умолчанию предполагает, что каждая заданная переменная - это вектор или матрица. Все определяется конкретным значением переменной. Например, если задано ***a = 5***, то это значит, что ***a*** - это вектор с одним единственным элементом, имеющим значение 5 (т.е. если быть точнее MATLAB рассматривает все переменные как матрицы, так что в данном случае ***a*** отвечает матрице размером *1×1*).

Имя переменной (ее идентификатор) может содержать сколько угодно символов, но запоминается и идентифицируется только 31 начальный символ, в пределах этих 31 символов имя любой переменной должно быть уникальным, т.е. не должно совпадать с именами других переменных, функций и процедур системы.

Имя должно начинаться с буквы, может содержать буквы, цифры и символ подчеркивания _ (в приведенном выше примере ***m_1***). Недопустимо включать в имена переменных пробелы и специальные знаки, например + . - * / и т. д., поскольку в этом случае правильная интерпретация выражений становится невозможной.

Как и большинство языков высокого уровня, система MATLAB различает регистры - т.е. имя ***A*** отличается от имени ***a***.

2.2 Математические выражения

Математическое выражение задает то, что должно быть вычислено в численном (реже символьном) виде. В системе MATLAB математические выражения строятся на основе чисел, констант, переменных, операторов, функций и разных спецзнаков.

Например, ***2.301*sin(x); 4+exp(3)/5; sqrt(y)/2; sin(pi/2); 2i+3; 3*b+5***.

2.3 Действительные (вещественные) числа

Число - простейший объект языка MATLAB, представляющий количественные данные. Числа можно считать константами, имена которых совпадают с их значениями.

Основным типом данных, с которым производятся вычисления в среде MATLAB, являются конечные десятичные дроби, приближающие с заданной точностью произвольные вещественные числа. Последние в общем случае представимы лишь в виде бесконечных десятичных дробей. Можно сказать, что MATLAB работает с вещественными числами приближённо. Вещественное число задаётся в ***MATLAB***е мантиссой и показателем степени:

Например, **2.851038e+12; -456.384569; 0.0045692e0; 0.93185e-1; 4.5; -12**

и т.д. Пробелы между символами в числах не допускаются.

У целых чисел отсутствуют дробные части, но они все равно представляются системой MATLAB на машинном уровне в той же форме, что и дробные числа. Этот основной тип данных называется ***double***. Именно этот тип данных подразумевается "*по умолчанию*" для любой переменной. Под мантиссу и показатель степени (на машинном уровне используется двоичная система записи чисел) отводится 8 байт памяти. В результате достигается точность представления десятичных чисел порядка 15 значащих цифр. При этом максимальным по модулю представимым в системе MATLAB вещественным числом является

1.797693134862316e+308

а минимальным по модулю является следующее вещественное число:

2.225073858507202e-308

Для этих чисел даже зарезервированы имена: **realmax** и **realmin**.

После запуска среды MATLAB в её командном окне появляется знак приглашения **>>**, после которого можно вводить с клавиатуры числа, имена переменных, знаки операций (в частности, знак **=** соответствует операции присваивания), что в совокупности составляет некоторое выражение. Нажатие клавиши ***ENTER*** заставляет систему MATLAB вычислить значение выражения и показать результат, как это показано на рис.2.1.

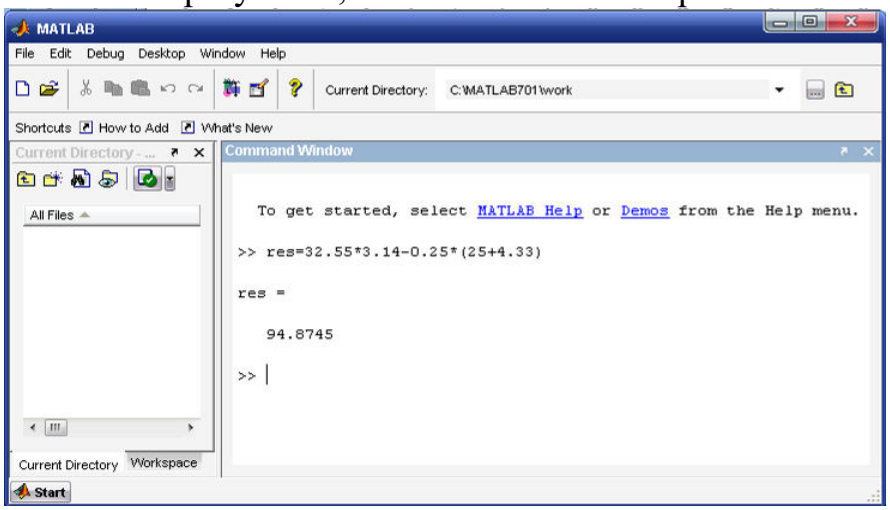


Рис.2.1. Ввод и результат математического выражения

Чтобы не перегружать излишними подробностями своё командное окно, MATLAB по умолчанию использует формат ***short*** для вывода вещественных чисел, при котором показываются только четыре десятичные цифры после запятой. Если требуется полное представление, то нужно ввести с клавиатуры команду

format long

после чего набрать имя переменной **res**, в которой записан результат вычислений. Нажав клавишу ***ENTER***, получим более подробную информацию (рис.2.2).

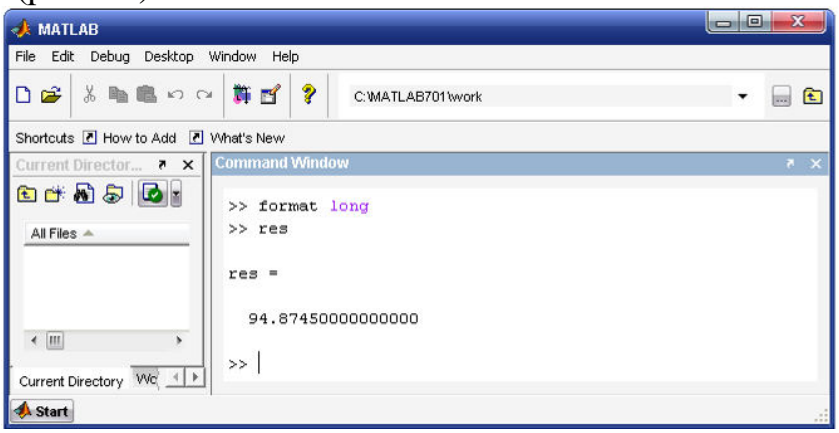


Рис.2.2. Полное представление вещественных чисел

Теперь все результаты вычислений будут показываться с такой высокой точностью в течение данного сеанса работы MATLABa. Если

требуется до прекращения текущего сеанса работы вернуться к старой точности визуального представления вещественных чисел в командном окне MATLABa, нужно ввести и исполнить (нажав клавишу ***ENTER***) команду

format short

Другим интересным форматом является показ вещественных чисел в виде обыкновенных дробей, для чего вводится команда

format rat

Ранее вычисленная переменная ***res***, будет показана в следующем виде (рис.2.3).

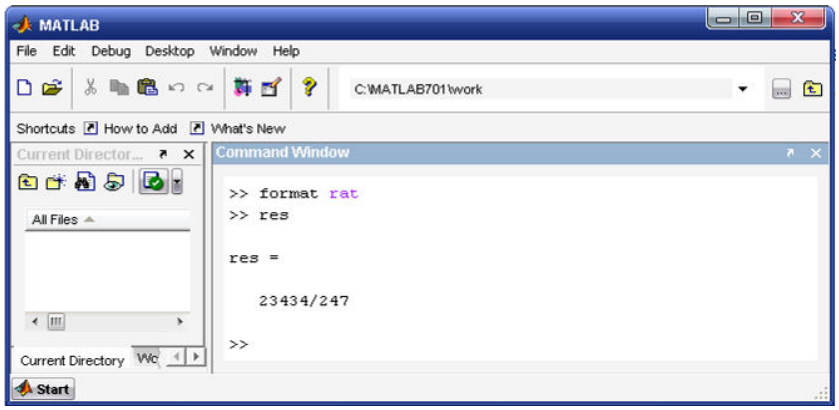


Рис.2.3 Вывод вещественных чисел в виде обыкновенных дробей

Ну и, наконец, если операнды и результаты вычислений являются целыми, то хотя они и представляются в памяти машины так же, как и дробные числа, визуально в командном окне MATLABa они показываются в виде целых чисел. Это иллюстрируется следующим примером (рис.2.4).

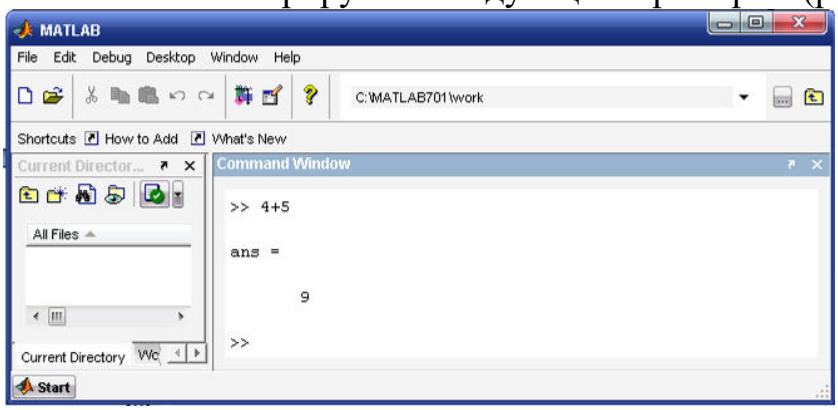


Рис.2.4 Вычисления с использованием целых чисел

В данном примере специальным именем ***ans*** обозначен (это стандартное обозначение) результат вычисления выражения, если он не был присвоен какой-либо переменной с другим именем.

2.4 Операции над вещественными числами

Над вещественными числами производятся *арифметические операции* сложения, вычитания, умножения и деления, для которых используются знаки + - * и /. Кроме того, есть ещё операция возведения в степень,

обозначаемая значком [^]. Результаты применения этой операции показаны на рис.2.5.

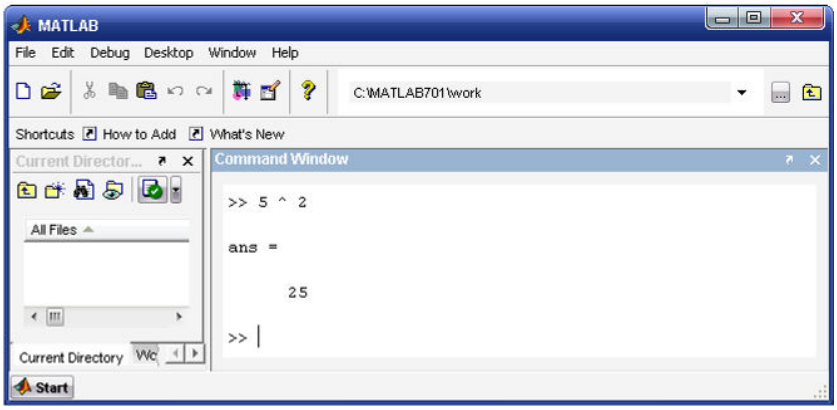


Рис.2.5. Использование операции возведения в степень (^)

Приоритет в выполнении арифметических операций обычный: сначала - возведение в степень, затем - умножение и деление, и потом - сложение и вычитание. Операции одинакового приоритета выполняются в порядке слева направо, но круглые скобки могут изменить этот порядок.

Для примера вычислим значения падения напряжения и потери мощности в активном сопротивлении, величина которого равна $\boldsymbol{r} = 5$ *Ом* при протекании по нему тока величиной $\boldsymbol{I} = 100$ *А* (рис.2.6).

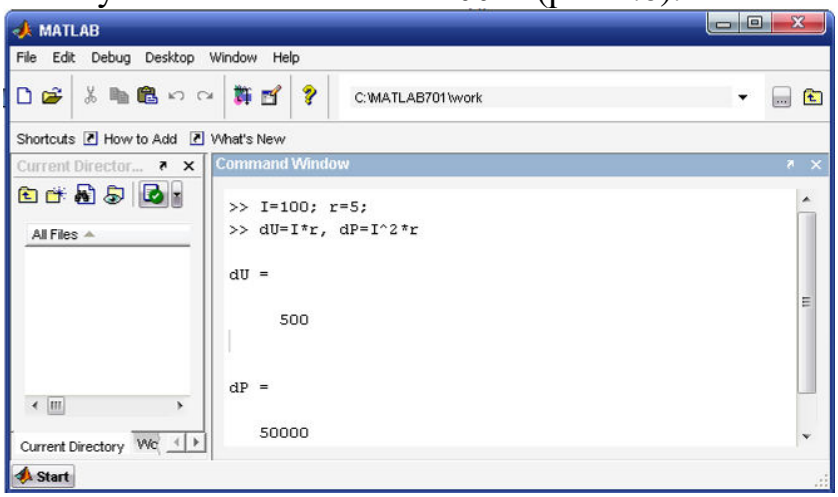


Рис.2.6 Результаты расчетов падения напряжения и потери мощности

Помимо арифметических операций используются ещё *операции отношения* и *логические операции*.

Операции отношения сравнивают между собой два операнда по величине. Эти операции записываются следующими знаками или комбинациями знаков указанных в таблице 2.1.

Таблица 2.1. Операции отношения и логические операции

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

На рис.2.7 приведен пример использования данных операций.

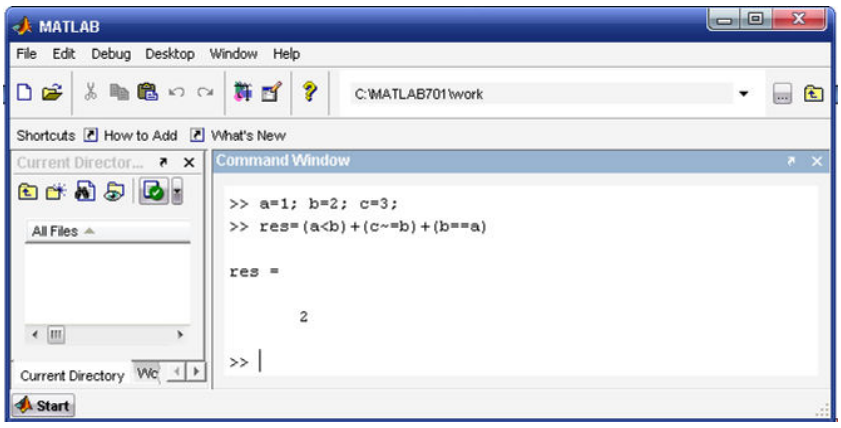


Рис.2.7 Пример использования логические операции

В примере выражение ***a*** вырабатывает единицу в силу того, что величина переменной ***a*** действительно меньше величины переменной ***b*** (истина). Выражение ***c ~= b*** является истинным, так как на самом деле ***c***, равное ***3***, не равно ***b***, которое равно двум. В итоге оно вырабатывает значение **1**. Последнее выражение, ***b == a***, не является истинным и вырабатывает **0**. В результате переменная ***res***, равная сумме значений этих трёх выражений, оказывается равной двум.

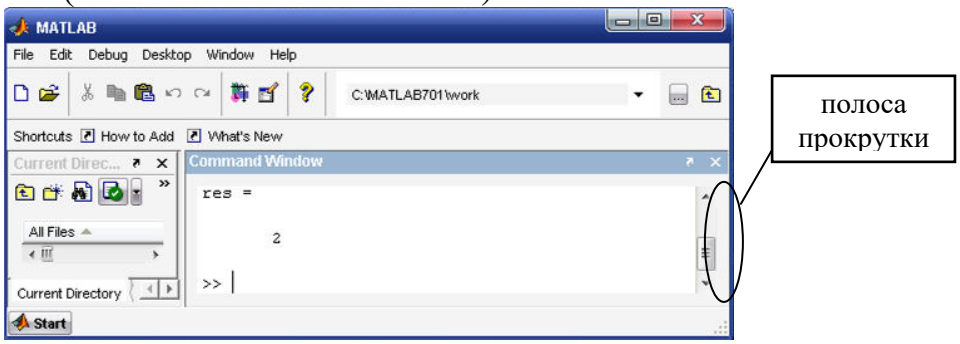
Здесь обратим внимание на роль *точки с запятой* ; в системе MATLAB. Точка с запятой может использоваться для разных целей. Когда мы вводим с клавиатуры некоторое выражение (оно расположено после знака приглашения >>) и нажимаем клавишу ***ENTER***, то MATLAB прозводит вычисление этого выражения и выводит результат в своё командное окно. Если мы не хотим тотчас же видеть результат вычислений (это характерно, например, для промежуточных результатов), то в конце введённого выражения следует поставить точку с запятой, и только после этого нажать ***ENTER***. Кроме того, если мы хотим за один раз, то есть одним нажатием клавиши ***ENTER*** вычислить несколько разных выражений, а их значения присвоить разным переменным, то эти выражения следует отделить друг от

друга точкой с запятой, как это и показано на предыдущем изображении командного окна системы MATLAB. Если же хотим выводить результат каждого из выражений, то эти выражения надо отделять запятыми [,].

3. Рабочее пространство системы MATLAB и её командное окно

Когда Вы запускаете MATLAB и начинаете производить вычисления, в командном окне показываются вводимые с клавиатуры числа, переменные (через их имена), результаты вычислений. Обычно вычисления повторяются вновь и вновь: вводятся с клавиатуры новые числовые данные и новые символьные выражения. В результате в окне MATLABa не хватает свободного места и производится "скроллирование" ("протяжка"; "прокрутка") — все строки сдвигаются на одну позицию вверх, так что самая верхняя строка покидает область видимости, а в самом низу окна появляется свободная строка для ввода новых данных. Естественно, эта строка содержит знак приглашения [>>].

Та информация, что покинула видимую часть окна, никуда не исчезает. Её всегда можно просмотреть снова, если осуществить прокрутку содержимого окна стандартным графическим средством управления — полосой прокрутки (по английски — Scrollbar).



Для этого нужно щелкнуть мышью на этой полосе, или протащить с помощью мыши ползунок полосы прокрутки в нужном направлении (вверх или вниз). Можно также осуществлять прокрутку содержимого командного окна системы MATLAB с помощью следующих клавиш клавиатуры: ***PageUp, PageDown, Ctrl+Home*** (одновременное нажатие клавиш ***Ctrl*** и ***Home***) и ***Ctrl+End***.

Клавиши "*Стрелка вверх*" [↑] и "*Стрелка вниз*" [↓], в любом текстовом редакторе осуществляющие перемещение курсора вверх-вниз и прокрутку содержимого окна, в системе MATLAB работают по-другому. Эти клавиши позволяют вернуть в строку ввода ранее введённые с клавиатуры команды и другую входную информацию, то есть вся эта информация запоминается в специальной области памяти. Эту область памяти называют *стеком команд*, так самая последняя входная информация при её прокрутке клавишой [↑] появится первой. Затем появится предпоследняя команда и так далее. Клавиша [↓] осуществляет прокрутку команд в противоположном направлении.

В итоге можно сказать, что вся видимая информация в окне системы MATLAB располагается в двух принципиально разных зонах: *зоне просмотра* и *зоне редактирования*.

В зоне просмотра уже ничего нельзя исправить, хотя в неё и можно поместить курсор, однако реакцией на ввод с клавиатуры будет автоматическое перемещение курсора (то есть точки ввода) в строку ввода, расположенную в зоне редактирования. В зоне просмотра можно выделять (селектировать) с помощью мыши любую информацию и копировать её в Буфер обмена операционной системы ***Windows*** (то есть в Clipboard), чтобы потом вставить её либо в документ текстового редактора (например, редактора Word), либо опять-таки в строку ввода.

Зона редактирования обычно занимает одну (последнюю) строку командного окна системы MATLAB, в которой показан знак приглашения `>>`. Её мы и называем строкой ввода. Однако при необходимости эту логическую "строку" можно распространить на несколько физических строк командного окна MATLABa. Для этого нельзя просто нажать клавишу ***ENTER***, так как при этом ввод информации будет закончен и MATLAB приступит к вычислениям и дальнейшему показу результата. Поэтому для продления ввода с показом вводимой информации на следующих физических строках требуется нажать ***ENTER*** только после `....` трёх или более точек, что и показано примере представленном ниже на рис.3.1.

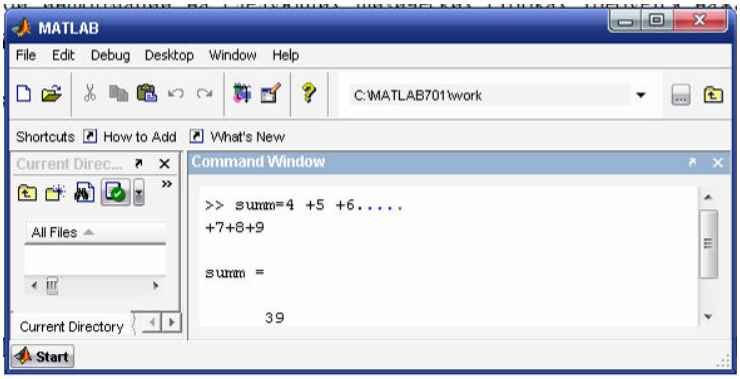


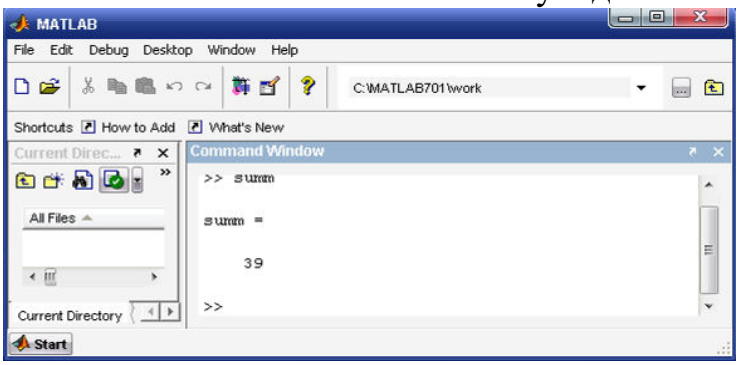
Рис.3.1 Пример использования многоточия (….)

Однако и в этом случае зона редактирования распространяется только на самую последнюю строку (теперь она уже не содержит знак приглашения `>>`), а в предыдущих физических строках логической строки ввода изменить уже ничего нельзя. Логическая строка ввода не может содержать более 256 символов.

Все значения переменных, вычисленные в течение текущего сеанса работы системы MATLAB, сохраняются в специальной области памяти компьютера, называемой *Рабочим пространством* (английскийское название -Workspace).

Всё видимое содержимое окна системы MATLAB можно стереть командой ***clc*** однако, это не затронет содержимого Рабочего пространства.

Действительно, если после этого набрать имя ранее вычисленной переменной ***summ***, то после нажатия клавиши ***ENTER*** мы снова увидим её значение:



То, что MATLAB автоматически сохраняет все предыдущие результаты (а также команды), является большим удобством. Однако тут могут обнаружиться и неприятности, если объём запомненной информации станет слишком большим (в дальнейшем мы увидим, что MATLAB может работать с данными гигантских размеров). Если Вам уже не требуется хранить некоторые переменные в данном сеансе работы, их можно стереть из памяти машины командой

clear имя1 имя2 …

удаляющей из Рабочего пространства переменные с именами **имя1** и **имя2**. Чтобы удалить сразу все переменные, нужно использовать команду

clear

Если Вы не знаете или сомневаетесь, какие переменные остались в Рабочем пространстве, Вы всегда можете выполнить команду

who

которая выведет список всех переменных, входящих на данный момент в Рабочее пространтсво системы MATLAB (Рис. 3.2).

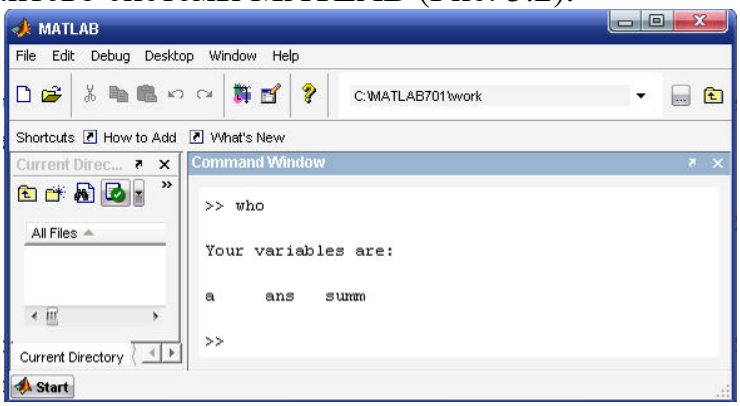


Рис.3.2 Результат выполнения команды ***who***

Для просмотра значения любой переменной из текущего рабочего пространства достаточно набрать её имя и нажать клавишу ***ENTER***.

После закрытия сеанса работы MATLABa все переменные, вычисленные в течение сеанса, теряются. Однако их можно сохранить для последующего использования в иных сеансах, сохранив содержимое Рабочего пространтсва в файле на диске командой меню

File | Save Workspace As…

после чего появляется стандартное диалоговое окно операционной системы Windows для выбора каталога на диске и имени файла. Расширение имени файла должно быть (**.mat**). Такие файлы будем называть MAT-файлами.

Вместо команды меню можно набрать команду

save путь_к_файлу\имя_MAT-файла

непосредственно в командном окне MATLABa. Например:

save c: MATфайлы \ file.mat

В новых сеансах системы MATLAB вы можете восстановить ранее сохранённое на диске Рабочее пространство командой меню

File | Load Workspace…

после которой Вы в стандартном диалоговом окне указываете нужный MAT-файл.

Более того, выполнив эту команду несколько раз с разными файлами, мы можем соединить в текущем Рабочем пространстве системы MATLAB содержимое нескольких предыдущих сеансов работы! Однако, если имена переменных из разных сеансов совпадают, то в текущем Рабочем пространстве будет представлена лишь переменная из последнего открытого MAT-файла.

Вместо команды меню можно набрать команду

load имя_MAT-файла например, ***load file.mat***

непосредственно в командном окне MATLABa. Можно также из записанного на диске MAT-файла считать в Рабочее пространство значения отдельных переменных. Для этого нужно выполнить команду

load имя_MAT-файла имя1, имя2, …

В результате из указанного MAT-файла будут считаны переменные с именами ***имя1***, ***имя2*** и т. д. При этом, если MAT-файл указан без полного пути к нему, то он должен находиться в *текущем каталоге системы* MATLAB, который всегда можно узнать с помощью команды

cd

Изменить текущий каталог можно командой

cd путь_к_новому_каталогу

3.1 Элементарные математические функции

В системе MATLAB присутствуют все основные элементарные функции: степенные, показательные, тригонометрические и обратные к ним. Любая функция характеризуется своим именем, списком входных аргументов (перечисляются через запятую и стоят внутри круглых скобок, следующих за именем функции) и вычисляемым (выходным) значением.

Помимо операции возведения в степень, реализуемой с помощью знака ^, есть ещё функция извлечения квадратного корня ***sqrt***, функция ***exp*** для возведения в степень числа ***e***, функция ***pow2*** для возведения в степень числа **2**. Также присутствуют обратные к ним функции: ***log*** — натуральный логарифм, ***log10*** — логарифм по сонованию10, ***log2*** — логарифм по

сонованию 2. В системе MATLAB можно быстро получить справочную информацию по любой элементорной функции, выполнив команду

help имя_функции например, ***help pow2***

Тригонометрические функции представлены весьма полно: ***sin***, ***cos***, ***tan*** (тангенс), ***cot*** (котангенс), ***asin*** (арктангенс), ***acos*** (арккосинус), ***atan*** (арктангенс), ***acot*** (арккотангенс). Имеются также и менее употребительные функции типа секанса, косеканса и т. д., а также гиперболические функции (являются комбинацией экспонент).

Для примера, вычислим выражение **2 * asin(1)**, включающее вычисление функции ***asin***, и получим следующий результат (Рис. 1.10) соответствующий числу π. В системе MATLAB для числа π есть специальное обозначение ***pi***.

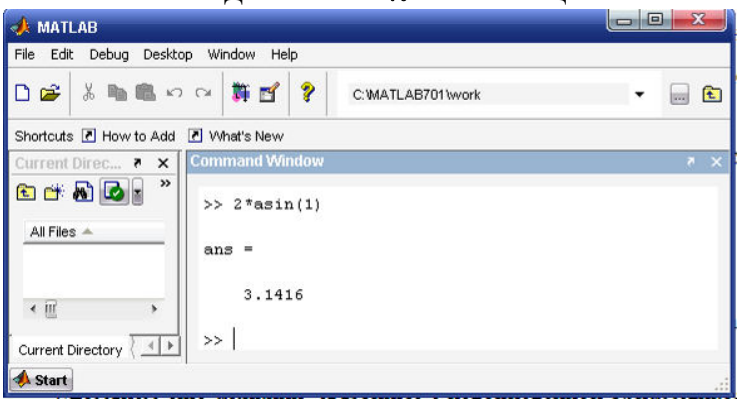


Рис.3.3 Использование встроенной тригономитрической функции ***asin***

Упомянем ещё функции, связанные с целочисленной арифметикой. Например, *функции округления*: ***round*** (округление до ближайшего целого), ***fix*** (усечение дробной части числа), ***floor*** (округление до меньшего целого), ***ceil*** (округление до большего целого).

Кроме того, есть ещё функции ***mod*** (остаток от деления с учётом знака), ***rem*** (остаток в смысле модульной арифметики), ***sign*** (знак числа), ***factor*** (разложение числа на простые множители), ***isprime*** (истинно, если число простое), ***primes*** (формирование списка простых чисел), ***rat*** (приближение числа в виде рациональной дроби), ***lcm*** (наименьшее общее кратное), ***gcd*** (наибольший общий делитель).

Функции ***mod*** и ***rem*** дают одинаковый результат для положительных аргументов. В частности,

mod(7,2) == rem(7,2) == 1

Но для операций с аргументами разных знаков они вырабатывают разные значения:

mod(-7,2) = 1; rem(-7,2) = -1

В общем случае эти функции связаны с функциями округления следующим образом:

rem(x,y) == x—y*fix(x/y); mod(x,y) == x—y*floor(x/y)

И, наконец, есть функции, вычисляющие некоторые стандартные результаты из *комбинаторики*: функция ***perms*** вычисляет число

перестановок, а функция ***nchoosek*** — число сочетаний. Например, число сочетаний из 10 по 3 легко находится вызовом функции ***nchoosek(10,3)*** (Рис.3.4).

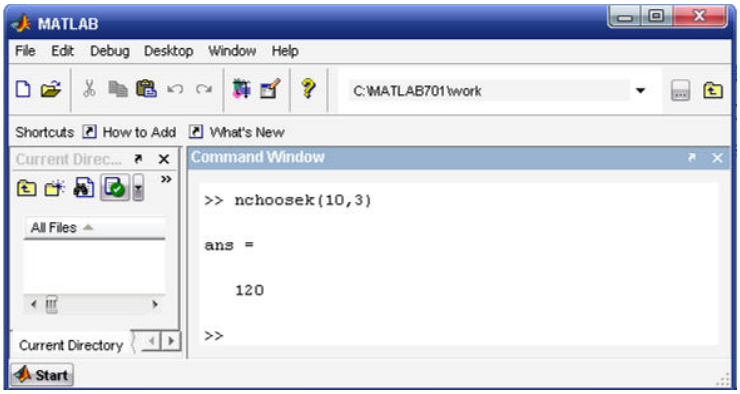


Рис.3.4 Использование встроенной функции ***nchoosek***

Многие из перечисленных функций имеют область определения, отличную от множества всех действительных чисел ***R***. В случаях, когда для функции задаётся недопустимое значение аргумента, или совершается попытка выполнить недопустимую операцию, мы получаем предупреждающее сообщение, например: **Warning: Divide by zero.**
при попытке деления на нуль. А в качестве результата выводится **ans = Inf,** где ***Inf*** символизирует бесконечность. Тот же результат получается при попытке вычислить логарифм от нуля.

Однако, чаще всего, при задании аргумента, выходящего за область определения функции действительного переменного (например, отрицательный аргумент для квадратного корня) система MATLAB автоматически выходит в область комплексных чисел, и вычисляет значение аналогичной комплексной функции.

3.2 Комплексные числа и вычисления в системе MATLAB

Числа могут быть комплексными: **z=Re(x)+Im(x)*i**. Такие числа содержат действительную **Re(z)** и мнимую **Im(z)** части. Мнимая часть должна иметь множитель **i** (или **j**), например, ***(2+3i; -3.141j; -123.456+2.7e-3i)***. Специально для работы с комплексными числами предназначены следующие функции: ***abs*** (абсолютное значение комплексного числа), ***conj*** (комплексно сопряжённое число), ***imag*** (мнимая часть комплексного числа), ***real*** (действительная часть комплексного числа), ***angle*** (аргумент комплексного числа), ***isreal*** (истина, если число действительное). Например, при вычислении функции ***sqrt(-1)*** порождается комплексное число (Рис. 3.5):

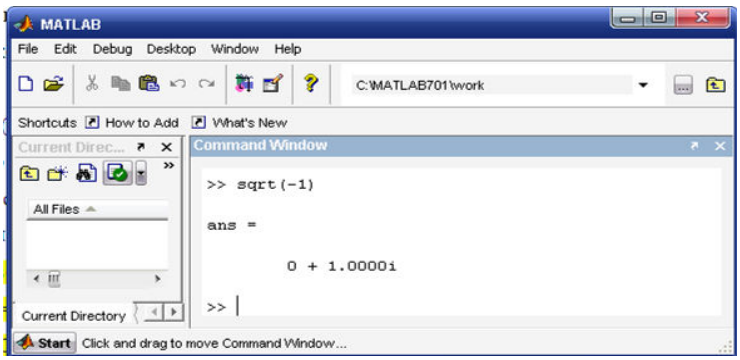


Рис.3.5 Вычисления с использованием функции ***sqrt***

Как мы видим, мнимая единица в системе MATLAB обозначается традиционной для математической литературы буквой ***i***. Помимо этого для мнимой единицы в MATLABe зарезервирована ещё и буква ***j***. Лучше не использовать эти буквы для других целей во избежание собственной путаницы.

В случае переменных нельзя писать просто ***x+iy***, а нужно обязательно использовать знак умножения, то есть ***x+i*y***.

Почти все элементарные функции допускают вычисления с комплексными аргументами. Например как это указанно на рис. 3.6.

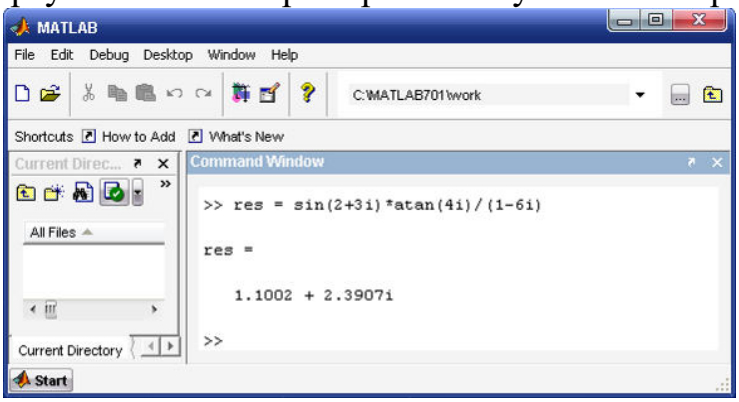


Рис.3.6 Вычисления с комплексными числами

Теперь мы можем проверить знаменитую *формулу Эйлера*:

exp(i*x) = cos(x)+i*sin(x)

Придавая вещественной переменной ***x*** различные значения, вычисляем выражения из правой и левой частей этой формулы. Например, при **x = 1** имеем как для левого, так и для правого выражения одно и то же значение: ***0.5403+0.8415i***. Для **x = 2** обе стороны этого равенства дают снова одинаковый результат ***-0.4161+0.9093i***, и так далее. Но в очевидных случаях о комплексных аргументах не может быть и речи (рис. 3.7):

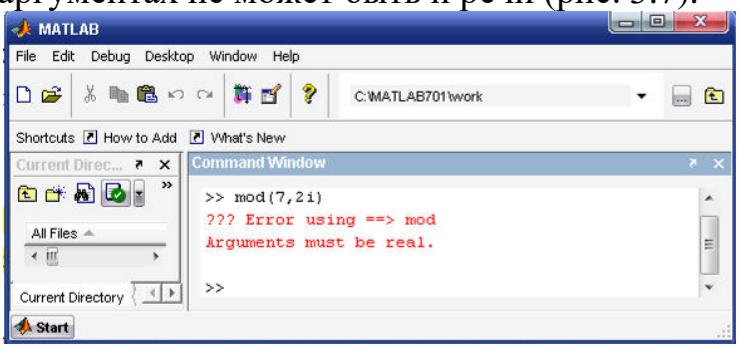


Рис.3.7 Неправильное использование функций с комплексными аргументами

Заметим, что никаких специальных соглашений об именах комплекснозначных переменных и комплексных функций не существует. Переменные не требуют никакого предварительного описания (объявления). Все вычисления перетекают из вещественной области в комплексную абсолютно автоматически. Это происходит при задании комплексных операндов (используются зарезервированные имена ***i*** или ***j*** для комплексной единицы) или при невозможности ограничиться лишь дествительными вычислениями (как в случае вычисления функции **sqrt(-1)**).

Итак, система MATLAB может прозрачно для пользователя перейти от вычислений с действительными числами к вычислениям с комплексными числами. Кроме того, MATLAB может производить вычисления с набором вещественных (или комплексных) чисел почти так же, как и с одиночными числами. В этом состоит одна из самых характерных особенностей системы MATLAB.

Рассмотрим применение комплексных чисел на примере вычисления проводимости активно-индуктивной (***RL***) ветви с активным сопротивлением R = 3 Ом и индуктивным сопротивлением X = 4 Ом (Рис. 3.8).

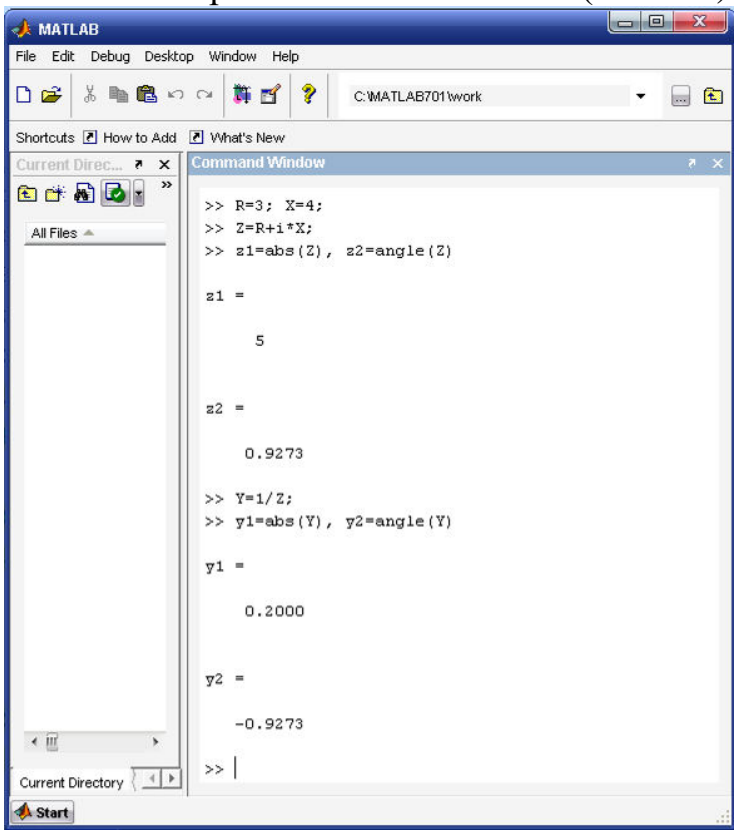


Рис.3.8 Вычисления проводимости активно-индуктивной (***RL***) ветви

В программе последовательно осуществляется присвоения переменным значений, образования комплексного числа **Z**, вычисление его модуля и угла (в радианах), вычисление комплексной проводимости **Y** и значение ее модуля и угла.

3.3 Константы и системные переменные

Константа - это предварительно определенное числовое или символьное значение, представленное уникальным именем. Константы в MATLAB принято назвать системными переменными, поскольку, с одной стороны, они задаются системой при ее загрузке, а с другой - могут переопределяться. Основные системные переменные, применяемые в системе MATLAB, указаны ниже:

a) ***i*** или ***j*** - мнимая единица (корень квадратный из -1);
b) ***pi*** - число 3.1415926. . .;
c) ***eps*** - погрешность операций над числами с плавающей точкой ($2.2204 * 10^{-16}$);
d) ***realmin*** - наименьшее число с плавающей точкой ($2.2251 * 10^{-308}$);
e) ***realmax*** - наибольшее число с плавающей точкой ($1.7977 * 10^{308}$);
f) ***inf*** - значение машинной бесконечности;
g) ***ans*** - переменная, хранящая результат последней операции и обычно вызывающая его отображение на экране дисплея;
h) ***NaN*** - указание на нечисловой характер данных (Not-a-Number).

Системные переменные могут переопределяться. Можно задать системной переменной ***eps*** иное значение, например ***eps = 0.0001***. Однако важно то, что их значения по умолчанию задаются сразу после загрузки системы. Поэтому неопределенными, в отличие от обычных переменных, системные переменные не могут быть никогда. Символьная константа - это цепочка символов, заключенных в апострофы, например, *'Hello my friend!', '2+3'*. Если в апострофы помещено математическое выражение, то оно не вычисляется и рассматривается просто как цепочка символов. Так что '2+3' не будет возвращать число 5.

3.4 Матрицы и векторы

Двумерный массив чисел или математических выражений принято называть матрицей. Одномерный массив называют вектором. MATLAB допускает так же задание и использование многомерных массивов. Векторы и матрицы могут иметь имена, например ***V*** - вектор или ***M*** - матрица. Элементы векторов и матриц рассматриваются как индексированные переменные, ***V(2)*** - второй элемент вектора ***V***; ***M(2,3)*** - третий элемент второй строки матрицы ***M***. Как уже отмечалось, даже обычные числа и переменные в MATLAB рассматриваются как матрицы размера *1×1*, что дает единообразные формы и методы проведения операций над обычными числами и массивами. Данная операция обычно называется векторизацией. Векторизация обеспечивает и упрощение записи операций, производимых одновременно над всеми элементами векторов и матриц, и существенное повышение скорости их выполнения. Это также означает, что большинство функций может работать с аргументами в виде векторов и матриц. При необходимости вектора и матрицы преобразуются в массивы, и значения вычисляются для каждого их элемента.

3.5 Задания векторов и матриц

Для задания вектора, значения его элементов следует перечислить в квадратных скобках, разделяя пробелами или запятыми. Так, например, присваивание ***V= [1 2 3]*** (или ***V= [1,2,3]***) задает вектор ***V***, имеющий три элемента со значениями 1, 2 и 3. После ввода вектора, если строка не заканчивается символом ;, система выводит его на экран дисплея. Задание матрицы требует указания нескольких строк. Для разграничения строк используется знак **;** (точка с запятой). Так, ввод ***M = [1 2 3;4 5 6;7 8 9]*** задает квадратную матрицу. Возможен ввод элементов матриц и векторов в виде арифметических выражений, содержащих любые доступные системе функции, например,

V= [2+2/(3+4) exp(5) sqrt(10)].

Для указания отдельного элемента вектора или матрицы используются выражения вида ***V(k)*** или ***M(к,1)*** (В тексте программ ***MATLAB*** лучше не использовать ***i*** и ***j*** как индексы, так как ***i*** и ***j*** - обозначение квадратного корня из -1, но можно использовать ***I*** и ***J***). Если нужно присвоить элементу ***M(к,1)*** новое значение, следует использовать выражение ***M(к, 1) = 'новое значение'***. Для вывода значения переменной на экран необходимо набрать имя данной переменной и нажать клавишу ***ENTER***.

Имеется также ряд особых функций для задания векторов и матриц. Например, функция ***magic (n)*** задает магическую матрицу размера $n \times n$, у которой сумма всех столбцов, всех строк и даже диагоналей равна одному и тому же числу, например, ***M=magic(4)*** результат использования которой представлен на рис 3.9.

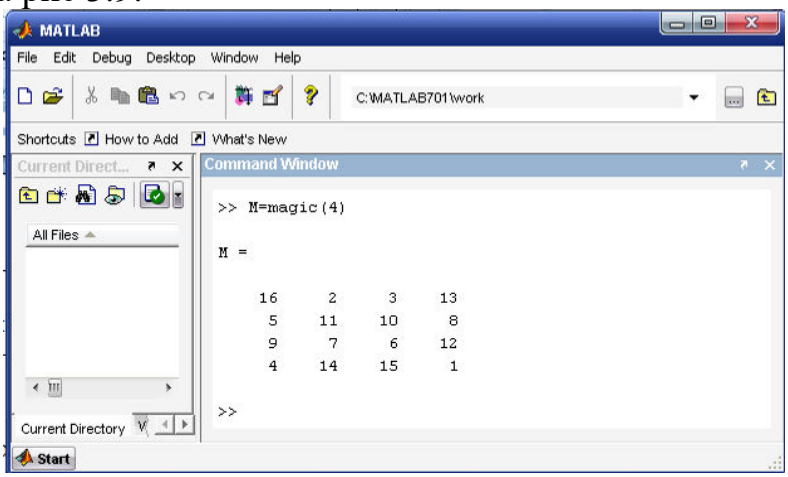


Рис.3.9 Использование функции ***magic (n)*** для вывода магической матрицы

Сушествует ряд других функций для формирования матриц. Так например для формирования единичной матрицы заданного размера ***n*** используется функция ***eye(n),*** квадратной матрицы единиц - ***ones(n),*** а для матрица состоящей из нулей - ***zeros(n)***, состоящей из равномерно распределенных случайных элементов - ***rand(n).***

Приложении

Матлаб.pas program:

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, FreeButton, StdCtrls, Clipbrd, WordXP, OleServer, ExtCtrls,
comobj,
  XPMan;

type
  TForm12 = class(TForm)
    Timer1: TTimer;
    FreeButton1: TFreeButton;
    Edit1: TEdit;
    Edit2: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    FreeButton2: TFreeButton;
    WordDocument1: TWordDocument;
    WordApplication1: TWordApplication;
    procedure FreeButton2Click(Sender: TObject);
    procedure FreeButton1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form12: TForm12;
```

```
implementation

uses Unit2;

{$R *.dfm}

procedure TForm12.FreeButton2Click(Sender: TObject);
begin
Self.Close;
end;

procedure TForm12.FreeButton1Click(Sender: TObject);
var
  worddoc: OLEVariant;
  name:string;
  i,countfile:integer;
  f1:TextFile;
  s:string;
  arr:array[1..6,1..6] of integer;
 begin
 try
Worddoc := CreateOleObject('Word.Application');
  except
    ShowMessage('Не удалось запустить MS Word');
    Exit;
  end;
  Worddoc.Visible := True;
  Worddoc.Documents.Add;


  if Worddoc.ActiveWindow.View.SplitSpecial <> 0 then
Worddoc.ActiveWindow.Panes[2].Close;
  if (Worddoc.ActiveWindow.ActivePane.View.type = 1) or
    (Worddoc.ActiveWindow.ActivePane.View.type = 2) or
    (Worddoc.ActiveWindow.ActivePane.View.type = 5) then
Worddoc.ActiveWindow.ActivePane.View.type := 3;
      Worddoc.ActiveDocument.PageSetup.LineNumbering.Active := False;
      Worddoc.ActiveDocument.PageSetup.Orientation :=
wdOrientLandscape;
      Worddoc.ActiveDocument.PageSetup.TopMargin := (30);
      Worddoc.ActiveDocument.PageSetup.BottomMargin := (30);
      Worddoc.ActiveDocument.PageSetup.LeftMargin := (30);
      Worddoc.ActiveDocument.PageSetup.RightMargin := (30);
      Worddoc.ActiveDocument.PageSetup.Gutter := (0);
```

```
            Worddoc.ActiveDocument.PageSetup.HeaderDistance := (25);
            Worddoc.ActiveDocument.PageSetup.FooterDistance := (25);
            Worddoc.ActiveDocument.PageSetup.FirstPageTray :=
wdPrinterDefaultBin;
            Worddoc.ActiveDocument.PageSetup.OtherPagesTray :=
wdPrinterDefaultBin;
            Worddoc.ActiveDocument.PageSetup.SectionStart :=
wdSectionNewPage;

Worddoc.ActiveDocument.PageSetup.OddAndEvenPagesHeaderFooter := False;
            Worddoc.ActiveDocument.PageSetup.DifferentFirstPageHeaderFooter
:= False;
            Worddoc.ActiveDocument.PageSetup.VerticalAlignment :=
wdAlignVerticalTop;
            Worddoc.ActiveDocument.PageSetup.SuppressEndnotes := False;
            Worddoc.ActiveDocument.PageSetup.MirrorMargins := False;
            Worddoc.ActiveDocument.PageSetup.TwoPagesOnOne := False;
            Worddoc.ActiveDocument.PageSetup.BookFoldPrinting := False;
            Worddoc.ActiveDocument.PageSetup.BookFoldRevPrinting := False;
            Worddoc.ActiveDocument.PageSetup.BookFoldPrintingSheets :=1;
            Worddoc.ActiveDocument.PageSetup.GutterPos := wdGutterPosLeft;


          Worddoc.ActiveWindow.ActivePane.View.SeekView := 0;
          Worddoc.Selection.ParagraphFormat.Alignment := 1;
          Worddoc.Selection.Font.Size     := 12;
          Worddoc.Selection.TypeText(Text :=Матлаб ');
          Worddoc.Selection.TypeParagraph;
          name:=Worddoc.ActiveDocument.Shapes.AddLine(30,43,820,43).Name;


        {###############TABLEAGA#################}
          Worddoc.Selection.TypeParagraph;
          Worddoc.Selection.ParagraphFormat.Alignment :=
wdAlignParagraphCenter;


        Form2.ADOQuery1.Parameters.ParamByName('ed1').Value:=strtoint(Edit1.
Text);
        Form2.ADOQuery1.Parameters.ParamByName('ed2').Value:=strtoint(Edit2.
Text);

        Form2.ADOQuery1.Close;
        Form2.ADOQuery1.SQL.Clear;
        Form2.ADOQuery1.SQL.Add('select * from student');
```

```
        Form2.ADOQuery1.SQL.Add('where');
        Form2.ADOQuery1.SQL.Add('Возраст>=ed1 and Возраст<=ed2;');
        Form2.ADOQuery1.Open;
        Form2.ADOQuery1.Active:=true;
             {
        countfile:=Form2.ADOQuery1.RecordCount;


          Worddoc.Selection.ParagraphFormat.Alignment := 0;
          Worddoc.ActiveDocument.Tables.Add (Range:=Worddoc.selection.Range,
NumRows :=countfile+1, NumColumns :=4);
          Worddoc.ActiveDocument.Tables.Item(1).Columns.Width:=150;
          Worddoc.ActiveDocument.Tables.Item(1).Rows.Height:=15;


          Worddoc.ActiveDocument.Tables.Item(1).Columns.Item(1).Width:=25;
          Worddoc.ActiveDocument.Tables.Item(1).Columns.Item(2).Width:=260;
          Worddoc.ActiveDocument.Tables.Item(1).Columns.Item(3).Width:=70;
          Worddoc.ActiveDocument.Tables.Item(1).Columns.Item(4).Width:=70;




          for i:=2 to 5 do begin

Worddoc.ActiveDocument.Tables.Item(1).Columns.Item(1).Cells.Item(i).Range.T
ext:=i-1;
          end;


Worddoc.ActiveDocument.Tables.Item(1).Columns.Item(3).Cells.Item(6).Range.T
ext:='Итого:';


Worddoc.ActiveDocument.Tables.Item(1).Columns.Item(1).Cells.Item(1).Range.T
ext:='№ ';

Worddoc.ActiveDocument.Tables.Item(1).Columns.Item(2).Cells.Item(1).Range.T
ext:='Направление ';

Worddoc.ActiveDocument.Tables.Item(1).Columns.Item(3).Cells.Item(1).Range.T
ext:='Коль-во студент ';

Worddoc.ActiveDocument.Tables.Item(1).Columns.Item(4).Cells.Item(1).Range.T
```

```
ext:='Гранты ';

Worddoc.ActiveDocument.Tables.Item(1).Columns.Item(5).Cells.Item(1).Range.T
ext:='Контракты ';

Worddoc.ActiveDocument.Tables.Item(1).Columns.Item(6).Cells.Item(1).Range.T
ext:='Мужчины ';

Worddoc.ActiveDocument.Tables.Item(1).Columns.Item(7).Cells.Item(1).Range.T
ext:='Женщины ';

          Form2.ADOQuery1.First;
          for i:=2 to countfile+1 do begin

Worddoc.ActiveDocument.Tables.Item(1).Columns.Item(2).Cells.Item(i).Range.T
ext:=Form2.ADOQuery1.FieldByName('Фамилия').Value+'
'+Form2.ADOQuery1.FieldByName('Имя').Value+'
'+Form2.ADOQuery1.FieldByName('Отчество').Value;
        Form2.ADOQuery1.Next;
          end;

               Worddoc.ActiveDocument.Tables.item(1).borders.item(-2).LineStyle
:= 1;
               Worddoc.ActiveDocument.Tables.item(1).borders.item(-4).LineStyle
:= 1;
               Worddoc.ActiveDocument.Tables.item(1).borders.item(-1).LineStyle
:= 1;
               Worddoc.ActiveDocument.Tables.item(1).borders.item(-3).LineStyle
:= 1;
               Worddoc.ActiveDocument.Tables.item(1).borders.item(-5).LineStyle
:= 1;
               Worddoc.ActiveDocument.Tables.item(1).borders.item(-6).LineStyle
:= 1;
             }

        end;

        end.
        begin

        Form2.ADOQuery1.Close;
        Form2.ADOQuery1.SQL.Clear;
        Form2.ADOQuery1.SQL.Add('select * from student where
(Академический=false and Отчислённый=false);');
        Form2.ADOQuery1.Open;
```

```
        Form2.ADOQuery1.Active:=true;


        Form2.ADOQuery1.First;
        k2:=0;

        if CheckBox1.Checked=false then begin
        Gauge1.Progress:=0;
        FreeButton2.Enabled:=false;
        for i:=1 to Form2.ADOQuery1.RecordCount do
        begin
        Form2.ADOQuery1.Edit;
        Form2.ADOQuery1.FieldByName('Kypc').AsInteger:=Form2.ADOQuery1.
FieldByName('Kypc').Value+1;
        Form2.ADOQuery1.Next;
        k2:=k2+100/Form2.ADOQuery1.RecordCount;
        Gauge1.Progress:=(trunc(k2) div 1);
        end;
        Gauge1.Progress:=100;
        FreeButton2.Enabled:=true;
        end else
        begin
        FreeButton2.Enabled:=false;
        k2:=100;
        Gauge1.Progress:=100;
        for i:=1 to Form2.ADOQuery1.RecordCount do
        begin
        Form2.ADOQuery1.Edit;
        Form2.ADOQuery1.FieldByName('Kypc').AsInteger:=Form2.ADOQuery1.
FieldByName('Kypc').Value-1;
        Form2.ADOQuery1.Next;

        k2:=k2-100/Form2.ADOQuery1.RecordCount;
        Gauge1.Progress:=(trunc(k2) div 1);

        end;
        CheckBox1.Checked:=false;
        FreeButton2.Enabled:=true;
        end;


        end;

        procedure TForm9.CheckBox1Click(Sender: TObject);
        begin
```

```
if CheckBox1.Checked=true then begin Gauge1.Progress:=100; end else
begin
Gauge1.Progress:=0;
end;

end;

end.
var
  Form7: TForm7;
  colordlg:char;
  fontdlg:char;
implementation

uses Unit1, Unit2, Unit3, Unit4, Unit5, Unit6, Unit8, Unit9, Unit10;

{$R *.dfm}

procedure TForm7.FreeButton1Click(Sender: TObject);
begin
Self.Close;
end;

procedure TForm7.FreeButton3Click(Sender: TObject);
begin
FontDialog1.Execute;
fontdlg:='1';
end;

procedure TForm7.FreeButton4Click(Sender: TObject);
begin
ColorDialog1.Execute;
colordlg:='1';
end;

procedure TForm7.FreeButton2Click(Sender: TObject);
var f:textfile;
pf:textfile;
sf:textfile;
saa:string;
begin
AssignFile(f,ExtractFileDir(Application.ExeName)+'\settings\face.stg');
Rewrite(f);

if CheckBox1.Checked=true then begin
```

```
        AssignFile(sf,ExtractFileDir(Application.ExeName)+'\settings\setting.stg');
        Rewrite(sf);
        write(sf,'1');
        CloseFile(sf); end else begin
        AssignFile(sf,ExtractFileDir(Application.ExeName)+'\settings\setting.stg');
        Rewrite(sf);
        write(sf,'0');
        CloseFile(sf);

        end;

        AssignFile(pf,ExtractFileDir(Application.ExeName)+'\settings\hopaci.dll');
        Reset(pf);
        read(pf,saa);
        if LabeledEdit1.Text=saa then begin
        closefile(pf);
        begin
        AssignFile(pf,ExtractFileDir(Application.ExeName)+'\settings\hopaci.dll');
        Rewrite(pf);
        if LabeledEdit2.Text=LabeledEdit3.Text then begin Write(pf,
LabeledEdit3.Text) end
        else ShowMessage('  Подтверждение не правильно!  ');
        closefile(pf);
        end;
        end;


        if fontdlg='1' then
        begin
        Form2.Font:=FontDialog1.Font;
        Form3.Font:=FontDialog1.Font;
        Form5.Font:=FontDialog1.Font;
        Form6.Font:=FontDialog1.Font;
        Form7.Font:=FontDialog1.Font;
        Form8.Font:=FontDialog1.Font;
        Form9.Font:=FontDialog1.Font;
        Form10.Font:=FontDialog1.Font;
        writeln(f, FontDialog1.Font.Name);
        writeln(f, FontDialog1.Font.Color);
        writeln(f, FontDialog1.Font.Charset);
        writeln(f, FontDialog1.Font.Size);
        end else
        begin
        writeln(f, Form2.Font.Name);
        writeln(f, Form2.Font.Color);
```

```
writeln(f, Form2.Font.Charset);
writeln(f, Form2.Font.Size);
end;
if colordlg='1' then
begin
Form2.Color:=ColorDialog1.Color;
Form3.Color:=ColorDialog1.Color;
Form5.Color:=ColorDialog1.Color;
Form6.Color:=ColorDialog1.Color;
Form7.Color:=ColorDialog1.Color;
Form8.Color:=ColorDialog1.Color;
Form9.Color:=ColorDialog1.Color;
Form10.Color:=ColorDialog1.Color;
writeln(f, ColorDialog1.Color);
end else
begin
writeln(f, Form2.Color);
end;

CloseFile(f);
ShowMessage('            Все настройки применены!        ');
end;

procedure TForm7.FormCreate(Sender: TObject);
var sf:textfile;
ss:string;
begin
colordlg:='0';
fontdlg:='0';
AssignFile(sf,ExtractFileDir(Application.ExeName)+'\settings\setting.stg');
reset(sf);
read(sf,ss);
CloseFile(sf);
if ss='1' then CheckBox1.Checked:=true else CheckBox1.Checked:=false;

end;

procedure TForm7.FreeButton5Click(Sender: TObject);
begin

ShellExecute (Form7.Handle, nil,
PAnsichar(ExtractFileDir(Application.ExeName)+'\settings\faku.txt'), nil, nil,
SW_RESTORE);

end;
```

```
        procedure TForm7.FreeButton7Click(Sender: TObject);
        begin
        ShellExecute (Form7.Handle, nil,
PAnsichar(ExtractFileDir(Application.ExeName)+'\settings\naprv.txt'), nil, nil,
SW_RESTORE);
        end;

        procedure TForm7.FreeButton6Click(Sender: TObject);
        begin
        ShellExecute (Form7.Handle, nil,
PAnsichar(ExtractFileDir(Application.ExeName)+'\settings\grupp.txt'), nil, nil,
SW_RESTORE);
        end;

        procedure TForm7.FreeButton11Click(Sender: TObject);
        begin
        ShellExecute (Form7.Handle, nil,
PAnsichar(ExtractFileDir(Application.ExeName)+'\settings\stran.txt'), nil, nil,
SW_RESTORE);
        end;

        procedure TForm7.FreeButton10Click(Sender: TObject);
        begin
        ShellExecute (Form7.Handle, nil,
PAnsichar(ExtractFileDir(Application.ExeName)+'\settings\oblas.txt'), nil, nil,
SW_RESTORE);
        end;

        procedure TForm7.FreeButton8Click(Sender: TObject);
        begin
        ShellExecute (Form7.Handle, nil,
PAnsichar(ExtractFileDir(Application.ExeName)+'\settings\Goay.txt'), nil, nil,
SW_RESTORE);
        end;

        procedure TForm7.FreeButton9Click(Sender: TObject);
        begin
        ShellExecute (Form7.Handle, nil,
PAnsichar(ExtractFileDir(Application.ExeName)+'\settings\nasii.txt'), nil, nil,
SW_RESTORE);
        end;

        end.
```

ВЫВОДЫ

1. Проведен анализ современного состояния технологии программировании МАТЛАБА и выбраны основные направления функциональностей.
2. Установлено, что в условиях Узбекистана наиболее целесообразно коллективные использования перерабатывать методами коллективно-селективной флотацией с последующей плавкой отдельных методов по классическим технологиям.
3. Выбраны объекты исследований и установлены методики проведения экспериментов. В качестве объекта разработки выбраны алгоритмические программирования Матлаб, которые в скором времени должны быть введены в промышленную эксплуатацию.

Литература:

1. Дьяконов В.П. MATLAB 6. Учебный курс. – СПб.: Питер, 2001. – 592 с.
2. Мэтьюз Дж.Г., Финк К.Д. Численные методы. Использование MATLAB. Пер. с англ. – М.: Изд. Дом «Вильямс», 2001. – 720 с.
3. Кетков Ю. Л., Кетков А. Ю., Шульц М. М. MATLAB 7: программирование, численные методы. — СПб.: БХВ-Петербург, 2005. — 752 с.
4. Ануфриев И. Е., Смирнов А. Б., Смирнова Е. Н. MATLAB 7. - СПб.: БХВ-Петербург, 2005. - 1104 с.
5. Чен К., Джиблин П., Ирвинг А. MATLAB в математических исследованиях. — М.: Мир, 2001. — 346 с.
6. Gerald W. Recktenwald "Numerical Methods with MATLAB: Implementation and Application", 2000, Prentice-Hall.
7. Сайт фирмы MathWorks http://www.mathworks.com
8. http://matlab.exponenta.ru/