МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского»

Д.В. Капитанов, О.В. Капитанова

Введение в MatLab

Лабораторный практикум

Рекомендовано методической комиссией
Института информационных технологий, математики и механики
для студентов ННГУ, обучающихся по специальности
01.05.01 «Фундаментальная математика и механика»
и направлению подготовки 38.03.05 «Бизнес – информатика»

Нижний Новгород
2016

УДК 004.42
ББК Ж/О 973.26-018.2
К 20

К 20 Капитанов Д.В., Капитанова О.В. Введение в MatLab: Лабораторный практикум. - Нижний Новгород: Нижегородский госуниверситет, 2016. – 65 с.

Рецензент: кандидат физ.-мат. наук **Чубаров Г.В.**

В данном лабораторном практикуме изложены основные принципы работы с системой компьютерной математики MatLab; перечислены наиболее важные команды и функции, а также приведены основные подходы к решению разнообразных вычислительных задач и построению математических моделей с помощью данного программного средства. Также в пособии представлен ряд заданий для самостоятельного выполнения.

Содержание пособия соответствует программам дисциплин «Применение систем компьютерной математики в экономико-математических исследованиях», преподаваемой студентам Нижегородского госуниверситета, обучающимся по направлению «Бизнес – информатика» (профиль подготовки «Информатика и математика в анализе экономических систем и бизнеса») и «Современные математические пакеты», преподаваемой студентам, обучающимся по специальности «Фундаментальные математика и механика» (специализация «Вычислительная математика и вычислительная механика»). Практикум также адресован всем желающим освоить работу и программирование в системе MatLab.

УДК 004.42
ББК Ж/О 973.26-018.2

Содержание

Введение

Целью освоения дисциплины «Применение систем компьютерной математики в экономико-математических исследованиях», преподаваемой студентам бакалавриата по направлению «Бизнес – информатика» (профиль подготовки «Информатика и математика в анализе экономических систем и бизнеса»), является получение студентами основных навыков работы с системами компьютерной математики, например, MatLab, для выполнения расчетов и проведения исследований в различных областях математики и экономики. Процесс изучения дисциплины направлен на формирование у выпускника следующих компетенций:

- способность осуществлять разработку и исследование математических моделей поддержки принятия решений в экономике и бизнесе (ДПК-1);
- способность к конструированию программ и применению компьютерных моделей поддержки принятия решений в экономике и бизнесе (ДПК-2).

Цель курса «Современные математические пакеты» у студентов, обучающихся в специалитете «Фундаментальные математика и механика» (специализация «Вычислительная математика и вычислительная механика»), заключается в изучении и получении навыков работы в математических вычислительных пакетах (в частности, MatLab). Процесс изучения дисциплины направлен на формирование у выпускника следующих компетенций:

- способность к самоорганизации и самообразованию (ОК-7);
- способность решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий с учетом основных требований информационной безопасности (ОПК-2);
- способность находить, анализировать, реализовывать программно и использовать на практике математические алгоритмы, в том числе с применением современных вычислительных систем (ОПК-4);
- способность к самостоятельному анализу поставленной задачи, выбору корректного метода ее решения, построению алгоритма и его реализации, обработке и анализу полученной информации (ПК-1).

В настоящее время система компьютерной математики MatLab приобретает все большую популярность. Эта программа используется не только в университетской среде для работы в различных отраслях математики, машиностроения и физики, но и промышленности для высокопродуктивных исследований, разработок и анализа данных, поскольку задачи и решения выражаются в форме близкой к математической.

Данный программный пакет разработан и поставляется фирмой Math Works, Inc. Следует заметить, что название MatLab появилось в 1980 году и расшифровывается как матричная лаборатория (matrix laboratory). Такое название объясняется тем, что основным элементом данных является матрица (массив). Это позволяет значительно уменьшить время решения задач для

матриц и массивов, по сравнению с такими «скалярными» языками программирования как Си, например.

Производители позиционируют MatLab как высокопроизводительный язык для технических расчетов. Он может использоваться для: математических вычислений, создания алгоритмов, моделирования, анализа, исследования и визуализации данных, научной и инженерной графики, разработки приложений, включая создание графического интерфейса.

В MatLab важная роль отводится специализированным наборам инструментов Toolboxes, которые позволяют изучать и применять специализированные методы: обработка сигналов, системы управления, идентификация систем, построение и анализ нейронных систем, поиск решений на основе нечеткой логики и т.д.

В настоящем лабораторном практикуме приводятся основы работы с системой компьютерной математики MatLab, базовые принципы и функции работы с матрицами, описаны способы построения различных двумерных и трехмерных графиков и диаграмм, перечислены типы данных, основы программирования на языке MatLab, функции для работы с символьными данными, правила работы с файлами, а также основы символьных вычислений. Каждая тема содержит набор заданий для самостоятельного решения в четырех вариантах.

Тема 1. Введение в MatLab

По умолчанию после запуска пакета MatLab на экране появляется комбинированное окно (рис.1), включающее 3 наиболее важные панели – Command Window, Workspace и Current Directory. Граница трех окон системы изменяют свои размеры и перемещаются вместе с главным окном. Для того чтобы они занимали автономную позицию надо нажать кнопку Undock.

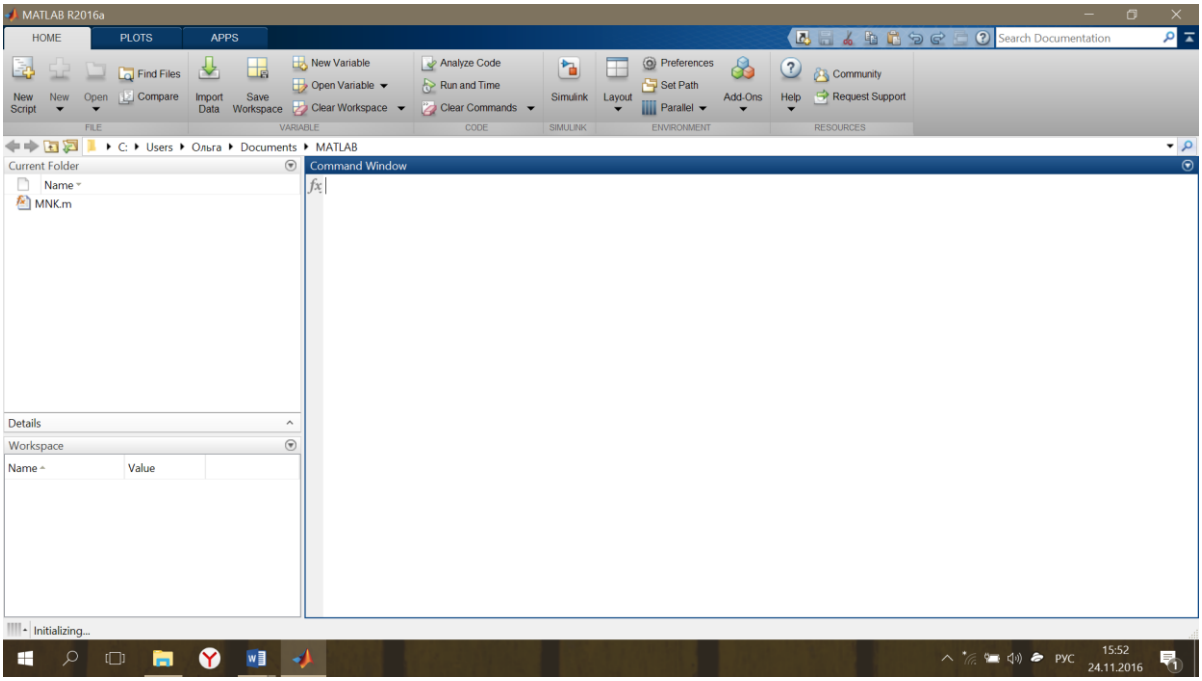

Рис.1. Рабочее окно программы MatLab

Основной панелью является Command Window. В ней набираются команды пользователя, подлежащие немедленному выполнению. Здесь же выдаются результаты исполняемых команд.

Окно Workspace отражает текущий набор переменных, заведенных пользователем в командном окне. Здесь можно увидеть их имена, размерность и тип представленных данных. Такую же информацию можно получить в командном окне после выполнения команды whos.

Также можно открыть окно Command History, которое хранит все команды, набираемые пользователем. Однако в отличие от Command Window в нем не выводятся сообщения системы и результаты вычислений.

Сеанс работы с MatLab обычно называют сессией (session).

В левом верхнем углу командного окна находятся два знака >>, которые символизируют начало строки. В этой строке можно набирать формулы или команды, удовлетворяющие синтаксису языка MatLab и завершающиеся нажатием клавиши Enter. Если строка не убирается, то для перехода на новую строку надо набрать … Если нужно набрать набор команд, а только потом запустить их на выполнение, то для перехода на новую строку можно воспользоваться комбинацией Shift + Enter.

Значения всех промежуточных переменных MatLab сохраняет в рабочем пространстве. На выбор имен накладываются следующие ограничения:

- Можно использовать латинские буквы, цифры и символ подчеркивания;
- Большие и малые буквы в именах различаются;
- Имя должно начинаться с буквы;
- Максимальную длину имени переменной, которая должна обеспечить оригинальность, можно определить с помощью команды *namelengthmax*.

В памяти компьютера числовые данные представляются вещественными или комплексными значениями в формате double, то есть занимают 8 и 16 байт соответственно. Количество значимых десятичных цифр в вещественном числе достигает 16-17. При отображении числовых результатов на дисплее часть значащих цифр отбрасывается в соответствии с форматом вывода.

Для установки формата представления чисел используется команда:

>> format style

где *style* — имя формата. Для числовых данных *style* может быть следующим (табл. 1):

Табл. 1. Форматы числовых данных

Формат	**Результат**	**Пример**
short (default)	Короткое представление в фиксированном формате (4 знака после запятой), используется по умолчанию	3.1416
long	Длинное представление в фиксированном формате (15 знаков после запятой для значений типа double и 7 знаков – для значений типа single).	3.141592653589793
shortE	Короткое представление в экспоненциальном формате (4 знака после десятичной точки).	3.1416e+00
longE	Длинное представление в экспоненциальном формате (15 знаков после запятой для значений типа double и 7 знаков – для значений типа single).	3.141592653589793e+00
shortG	Short или shortE, в зависимости от того, что более компактно.	3.1416
longG	Long или longE, в зависимости от того, что более компактно.	3.14159265358979
shortEng	Короткое представление в инженерном формате (показатель кратен 3) с 4 знаками после запятой.	3.1416e+000

Формат	Результат	Пример
longEng	Длинное представление в инженерном формате (показатель кратен 3) с 15 значащими цифрами.	3.14159265358979e+000
+	Отображается +, – или пробел для положительных, отрицательных или нулевых элементов.	+
bank	Представление для денежных единиц с двумя знаками после запятой. Для комплексных чисел отображается только действительная часть.	3.14
hex	Шестнадцатеричное представление двоичных чисел с двойной точностью числа.	400921fb54442d18
rat (rational)	Представление чисел в виде рациональной дроби с минимально возможными числителем и знаменателем	355/113

В MatLab существует ряд системных констант, которые задаются системой при загрузке, но могут быть переопределены:

- *i* или *j* – мнимая единица (корень квадратный из -1);
- *pi* –число π - 3.1415926...;
- *eps* – погрешность операций над числами с плавающей точкой (2^{-52});
- *realmin* – наименьшее число с плавающей точкой (2^{-1022});
- *realmax* – наибольшее число с плавающей точкой (2^{1023});
- *flintmax* – наибольшее порядковое целое число в формате с плавающей точкой (2^{53});
- *inf* – значение машинной бесконечности;
- *ans* – переменная, хранящая результат последней операции и обычно вызывающая его отображение на экране дисплея;
- *NaN* – указание на нечисловой характер данных (Not-a-Number).

Символьная константа — это цепочка символов, заключенных в апострофы, например, 'Hello my friend!' или '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. Однако с помощью специальных функций преобразования символьные выражения могут быть преобразованы в вычисляемые.

Запись комплексных величин, используемых в формулах, напоминает общепринятый математический стандарт. Мнимая часть сопровождается либо буквой *i*, либо буквой *j*. Когда полные комплексные числа используются в операциях умножения, деления или возведения в степень, то для устранения неоднозначности их заключают в круглые скобки.

В таблицах 2, 3 и 4 приведены основные функции для работы с комплексными числами, арифметические и логические операции и элементарные математические функции.

Табл. 2. Функции для работы с комплексными числами

Функция	**Действие**
real(a+bi)	Выделение вещественной части комплексного числа
imag(a+bi)	Выделение мнимой части комплексного числа
complex(a1,a2)	Построение комплексного числа по паре вещественных чисел
conj(a+bi) или *(a+bi)'*	Вычисление сопряженного комплексного числа
abs(a+bi)	Модуль комплексного числа
angle(a+bi)	Угол наклона радиус-вектора (в радианах)

Табл. 3. Арифметические и логические операции

Символ	**Выполняемое действие**
Операции над числовыми величинами	
+	покомпонентное сложение числовых массивов одинаковой размерности; добавление скалярной величины к каждому элементу массива;
-	покомпонентное вычитание числовых массивов одинаковой размерности; вычитание скалярной величины из каждого элемента массива;
*	умножение матриц в соответствии с правилами линейной алгебры; умножение всех компонент массива на скаляр;
.*	покомпонентное умножение элементов массива одинаковой размерности;
/	деление скаляра на скаляр; покомпонентное деление всех элементов массива на скаляр; $A/B=A*B^{-1}=A*inv(B)$ (A и B – квадратные матрицы одного порядка)
./	покомпонентное деление элементов массива одинаковой размерности;
\	левое матричное деление $A\backslash B=A^{-1}*B$ (A – квадратная матрица)
.\	A.\B – покомпонентное деление элементов B на A (левое поэлементное деление)
^	возведение скаляра в любую степень; вычисление целой степени квадратной матрицы;
.^ или *power()*	Покомпонентное возведение в степень
'	вычисление сопряженной матрицы;
.'	транспонирование матрицы;

Символ	Выполняемое действие
Логические операции	
& или *and()*	логическое умножение скаляров; логическое покомпонентное умножение массивов одинаковой размерности; логическое умножение массива на скаляр;
| или *or()*	логическое сложение скаляров; логическое покомпонентное сложение массивов одинаковой размерности; логическое сложение массива со скаляром;
~ или *not()*	логическое отрицание скаляра или всех элементов массива4
== или *eq()*	проверка на равенство;
~= или *ne()*	проверка на неравенство;
> или *gt()*	проверка на «больше»
>= или *ge()*	проверка на «больше или равно»
< или *lt()*	проверка на «меньше»
<= или *le()*	проверка на «меньше или равно»

Табл. 4. Элементарные математические функции

Категория функций	Наименования функций
Тригонометрические (для аргументов в радианах)	*cos(), cot(), csc(), sec(), sin(), tan()*
Обратные тригонометрические (для аргументов в радианах)	*acos(), acot(), acsc(), asec(), asin(), atan(), atan2()*
Тригонометрические (для аргументов в градусах)	*cosd(), cotd(), cscd(), secd(), sind(), tand()*
Обратные тригонометрические (для аргументов в градусах)	*acosd(), acotd(), acscd(), asecd(), asind(), atand(), atan2d()*
Гиперболические	*cosh(), coth(), csch(), sech(), sinh(), tanh()*
Обратные гиперболические	*acosh(), acoth(), acsch(), asech(), asinh(), atanh()*
Конвертация градусов в радианы и наоборот	*deg2rad(), rad2deg()*
Экспонента, логарифмы, корень	*exp(), log(), log2(), log10(), sqrt()*
Округления	*ceil*() – округляет до ближайшего большего числа, *fix*() – просто отбрасывает дробную часть, *floor*() – до ближайшего меньшего числа, *round*() – округление по математическим правилам
Наибольший общий делитель	*gcd()*
Наименьшее общее кратное	*lcm()*
Модуль числа	*abs()*
Знак числа	*sign()*
Остаток от деления	*rem()*

Выбрать рабочую папку в MatLab можно с помощью кнопки *Set Path* с панели инструментов.

Имена и значения переменных рабочего пространства можно сохранить в файл либо с помощь команды главного меню *File → Save Workspace As*, либо набрать аналогичную команду в текущей строке:

>>save qq

MatLab добавит к имени файла расширение .mat и сохранит все переменные и их значения в файле qq.mat. Также в файл можно сохранить только часть переменных при помощи той же команды save, но после имени файла указываются имена переменных:

>>save qq x y или

>>save qq x

В начале следующей сессии достаточно выполнить команду load:

>>load qq

Если команда load будет использована в середине сессии, то текущие значения переменных изменятся на загружаемые.

Для того чтобы очистить все рабочее пространство используется команда clear без параметров, но если указать список переменных, то она удалит только заданные переменные.

>>clear

>>clear x y

Команда whos позволяет получить подробную информацию о переменных рабочего стола. Также может быть использована команда who, которая выводит только список имен переменных. Для просмотра значения любой переменной достаточно просто набрать ее имя и нажать клавишу Enter.

Стандартная команда save не позволяет сохранить на диск содержание сессии, поэтому если существует такая необходимость, то следует воспользоваться специальной командой для ведения так называемого дневника сессии:

>>diary filename — ведет запись на диск всех команд в строках ввода и полученных результатов в виде текстового файла с указанным именем;

>>diary off — приостанавливает запись в файл;

>>diary on — вновь начинает запись в файл.

Таким образом, чередуя команды diary off и diary on, можно сохранять нужные фрагменты сессии в их формальном виде. Команду diary можно задать и в виде функции diary('file'), где строка 'file' задает имя файла.

Для просмотра текста дневника сессии используется команда:

>>type filename

Закончить работу с программой можно с помощью команды

>>exit

Задание 1

Вариант 1

1. Вычислить:

$$(-0.6)\cdot\left(0.0081^{\frac{1}{2}}+\left(1\frac{1}{9}\right)^{-2}\right):1\frac{5}{13}$$

$$\sqrt{0.04}-(\sqrt{7}-2\sqrt{2})(\sqrt{8}+\sqrt{7})$$

2. С помощью команды *format* вывести значение выражения *f* на экран в различных форматах.

$$f=2\cdot\pi$$

3. Чему равно значение *eps*?

4. Задать значения комплексных чисел *x* (используя *i*) и *y* (с помощью функции *complex()*). Найти сумму, разность, произведение, сопряженное (двумя способами), а также модуль и угол наклона радиус-вектора.

$$x=1.5+0.5i$$

$$y=2.5-7i$$

5. Подсчитать значения функций *cos*(), *sin*(), *tan*(), *cot*(), *sec*(), *csc*() для 8π/5.

6. Округлить числа всеми возможными способами: 1.1, 1.5, 1.8.

7. Вычислить остаток от деления для чисел 8 и -5.

8. Вычислить НОД и НОК для чисел: 12 и 27.

9. Сохранить переменные рабочего пространства в файле *tt.mat*

10. Очистить сначала значения переменных *x* и *y*, а затем все остальные переменные.

11. Загрузить переменные из файла *tt.mat*

12. Просмотреть информацию о переменных с помощью команд *whos* и *who*.

13. Записать в дневник (файл *mydiary.m*) следующие действия:

$$a=2;$$

$$b=3;$$

$$c=(a-b)/b$$

14. Посмотреть текст файла *mydiary.m*

Вариант 2

1. Вычислить:

$$3.5\cdot\left(0.027^{\frac{1}{3}}-\left(1\frac{3}{7}\right)^{-1}\right):\left(-2\frac{6}{11}\right)$$

$$\sqrt{0.09}-(\sqrt{11}-2\sqrt{3})(\sqrt{11}+\sqrt{12})$$

2. С помощью команды *format* вывести значение выражения *f* на экран в различных форматах.

$$f=e^2$$

3. Чему равно значение *realmin*?

4. Задать значения комплексных чисел *x* (используя *i*) и *y* (с помощью функции *complex*()). Найти сумму, разность, произведение, сопряженное (двумя способами), а также модуль и угол наклона радиус-вектора.

$$x=8+5i$$

$$y=7+0.8i$$

5. Подсчитать значения функций *cos*(), *sin*(), *tan*(), *cot*(), *sec*(), *csc*() для 7π/11.

6. Округлить числа всеми возможными способами: -1.1, -1.5, -1.8.

7. Вычислить остаток от деления для чисел: -23 и 4.

8. Вычислить НОД и НОК для чисел: 14 и 35.

9. Сохранить переменные рабочего пространства в файле *tt.mat*

10. Очистить сначала значения переменных *x* и *y*, а затем все остальные переменные.

11. Загрузить переменные из файла *tt.mat*

12. Просмотреть информацию о переменных с помощью команд *whos* и *who*.

13. Записать в дневник (файл *mydiary.m*) следующие действия:

$$a=2\pi;$$

$$b=\cos a\,.$$

$$c=\sin a$$

14. Посмотреть текст файла *mydiary.m*

Вариант 3

1. Вычислить:

$$\left(0.48^0+\left(1\frac{9}{16}\right)^{\frac{3}{2}}:0.8^{-4}\right)\cdot 0.81^{-\frac{1}{2}}$$

$$(2-\sqrt{3})^2(7+4\sqrt{3})+3\sqrt{12\frac{1}{4}}$$

2. С помощью команды *format* вывести значение выражения *f* на экран в различных форматах.

$$f = \sqrt{3}$$

3. Чему равно значение *realmax*?

4. Задать значения комплексных чисел x (используя *i*) и y (с помощью функции *complex*()). Найти сумму, разность, произведение, сопряженное (двумя способами), а также модуль и угол наклона радиус-вектора.

$$x = 2 - 3i$$

$$y = -8 + 4i$$

5. Подсчитать значения функций *cos*(), *sin*(), *tan*(), *cot*(), *sec*(), *csc*() для π/13.

6. Округлить числа всеми возможными способами: 2.2, -2.5, 2.7.

7. Вычислить остаток от деления для чисел: 12 и -7.

8. Вычислить НОД и НОК для чисел: 16 и 2.

9. Сохранить переменные рабочего пространства в файле *tt.mat*

10. Очистить сначала значения переменных *x* и *y*, а затем все остальные переменные.

11. Загрузить переменные из файла *tt.mat*

12. Просмотреть информацию о переменных с помощью команд *whos* и *who*.

13. Записать в дневник (файл *mydiary.m*) следующие действия:

$$a = 2;$$

$$b = \sqrt{a}$$

$$c = a + b$$

14. Посмотреть текст файла *mydiary.m*

Вариант 4

1. Вычислить:

$$\left(0.49^{-1.5} : \left(1\frac{3}{7}\right)^4 + 0.64^{-\frac{1}{2}}\right) \cdot 3\frac{1}{13}$$

$$(\sqrt{5}-2)^2(9+4\sqrt{5}) - 2\sqrt{5\frac{4}{9}}$$

2. С помощью команды *format* вывести значение выражения *f* на экран в различных форматах.

$$f = \pi \cdot e$$

3. Чему равно значение выражения: 0/0?

4. Задать значения комплексных чисел *x* (используя *i*) и *y* (с помощью функции *complex*()). Найти сумму, разность, произведение, сопряженное (двумя способами), а также модуль и угол наклона радиус-вектора.

$$x = 4 - 8i$$

$$y = 5 - 2i$$

5. Подсчитать значения функций *cos*(), *sin*(), *tan*(), *cot*(), *sec*(), *csc*() для 14π/6.

6. Округлить числа всеми возможными способами: -2.2, 2.5, -2.7.

7. Вычислить остаток от деления для чисел: -18 и 8.

8. Вычислить НОД и НОК для чисел: 48 и 8.

9. Сохранить переменные рабочего пространства в файле *tt.mat*

10. Очистить сначала значения переменных *x* и *y*, а затем все остальные переменные.

11. Загрузить переменные из файла *tt.mat*

12. Просмотреть информацию о переменных с помощью команд *whos* и *who*.

13. Записать в дневник (файл *mydiary.m*) следующие действия:

$$a = 7;$$

$$b = \log a$$

$$c = a * b$$

14. Посмотреть текст файла *mydiary.m*

Тема 2. Матрицы. Операции с матрицами в MatLab

Оператор двоеточие используется для задания последовательности чисел:

>>a:b:c – задает последовательность чисел от a до c с шагом b. Если шаг равен 1, то значение b можно пропустить и записать a:c.

Для задания значений элементов матрицы используются квадратные скобки, в которых числовые данные отделяются друг от друга пробелами или запятыми (для задания чисел в строке) и точкой с запятой (для задания столбцов). Матрица задается по строкам.

Количество пробелов, которое MatLab вставляет при выводе компонент вектора-строки, регулируется параметром *Tab size* (см. опцию *Preferences* панели *Home*, раздел *Command Window*). По умолчанию оно равно 4.

Также матрицу можно задавать или изменять, используя индексное обозначение *A(i,j)*, где *i* указывает номер строки, а j – номер столбца. Нумерация строк и столбцов начинается с 1.

Для создания типовых матриц существует ряд специальных функций:

- Функции *zeros(n,m)* и *ones(n,m)* используются для заполнения квадратных или прямоугольных матриц нулями и единицами.
- С помощью функции *eye(n,m)* формируется единичная квадратная или прямоугольная матрица.
- Функции *rand(n,m)* и *randn(n,m)* заполняют элементы матриц случайными числами с равномерным или нормальным распределением.
- Функция *magic(n)* задает магическую матрицу размера n×n, у которой сумма всех столбцов, всех строк и даже диагоналей равна одному и тому же числу.

Если в круглых скобках после задания матрицы задать только одно число, то будет создана квадратная матрица.

MatLab позволяет создавать матрицы с заданными элементами на диагонали:

>> X = diag(v,k)

где *v* – вектор, *k* – номер диагонали: *k=0* – главная диагональ; *k>0* – диагональ, расположенная выше главной; *k<0* – диагональ расположенная ниже главной. *X* – матрица, на *k*-ой диагонали которой расположен вектор *v*, остальная часть заполнена нулями.

Если эту функцию записать в виде *v = diag(X,k)*, то результатом будет вектор-столбец, состоящий из элементов k-ой диагонали матрицы X.

Ранее сформированные массивы могут участвовать в формировании новых в качестве их клеток (например, *D=[A; B+5 C]*); однако следует смотреть за соответствием размерностей матриц (блочный способ).

Функция *blkdiag(A,B,C,…)* объединяет матрицы по диагонали, остальные элементы заполняются нулями.

Конкатенацией называют объединение массивов, которое реализует функция cat:

>> C = cat (dim, A, B) – объединяет массивы A и B в соответствии со спецификацией размерности dim и возвращает объединенный массив; dim = 1 – вертикальная конкатенация, dim = 2 – горизонтальная, dim = 3 – многомерный массив размерности 3 и т. д.;

>> C = cat(dim,Al,A2,A3,A4,...) – объединяет все входные массивы (A1, A2, A3, A4 и т. д.) в соответствии со спецификацией размерности dim и возвращает объединенный массив.

Для формирования матриц и выполнения ряда матричных операций возникает необходимость удаления отдельных столбцов и строк матрицы. Для этого используются пустые квадратные скобки []. Удаление i-го столбца - M(:,i)=[]; удаление i-ой строки - M(i,:)=[].

Для поиска обратной матрицы для квадратной матрицы A используется команда:

>> inv(A)

Для вычисления определителя матрицы A используется функция *det(A)*.

Транспонировать матрицу *A* можно с помощью функции *transpose(A)* или команды:

>> A'

Рассмотрим способы решения систем линейных уравнений в MatLab. Традиционно, система линейных уравнений в матричном виде записывается:

$$A*X=B$$

где *A* - матрица коэффициентов; *X* - вектор-столбец неизвестных; *B* - вектор-столбец правых частей. Тогда решение системы линейных уравнений может быть записано в виде:

$$X=A^{-1}*B$$

В MatLab систему линейных уравнений можно решить двумя способами:

*>> X=inv(A)*B*

или

>> X=A\B

Задание 2

Вариант 1

1. Задать матрицу $A=\begin{pmatrix} 2 & 1 & 3 \\ 5 & 10 & 2 \\ 1 & 4 & 3 \end{pmatrix}$.

2. Изменить значение элемента матрицы *A*, используя индексы: *A(2,2)=3*.

3. Используя функции *zeros*(), *ones*(), *eye*(), *rand*(), *randn*(), *magic*(), создать квадратные матрицы размерности 4.

4. Задать матрицу *X*, где вектор *v* будет *k*-ой диагональю матрицы.

v=(1 2 3 4), k=2

5. Объединить в матрицах *M*, *N*, *L* матрицы *A* и *B* блочным способом, а также с помощью функций *blkdiag*() и *cat*().

$$B=\begin{pmatrix}5 & 2 & 0\\ 7 & 3 & 0\end{pmatrix}$$

6. Удалить из матрицы B третий столбец, из матрицы *L* вторую строку, а матрицу *M* очистить.

7. Посчитать определители матриц *A* и *B*.

8. Решить систему линейных уравнений:

$$2x_1+3x_2+5x_3=10$$
$$3x_1+7x_2+4x_3=3$$
$$x_1+2x_2+2x_3=3$$

Вариант 2

1. Задать матрицу $A=\begin{pmatrix}1 & 10 & 1\\ 1 & 2 & 3\\ 1 & 3 & 6\end{pmatrix}$.

2. Изменить значение элемента матрицы *A*, используя индексы: *A(1,2)=1*.

3. Используя функции *zeros*(), *ones*(), *eye*(), *rand*(), *randn*(), *magic*(), создать квадратные матрицы размерности 2.

4. Задать матрицу *X*, где вектор *v* будет *k*-ой диагональю матрицы.

v=(65 8 7), k=3

5. Объединить в матрицах *M*, *N*, *L* матрицы *A* и *B* блочным способом, а также с помощью функций *blkdiag*() и *cat*().

$$B=\begin{pmatrix}3 & 2 & 0\\ 8 & 5 & 0\end{pmatrix}$$

6. Удалить из матрицы *B* третий столбец, из матрицы *L* вторую строку, а матрицу *M* очистить.

7. Посчитать определители матриц *A* и *B*.

8. Решить систему линейных уравнений:

$$5x_1-6x_2+4x_3=3$$
$$3x_1-3x_2+2x_3=2$$
$$4x_1-5x_2+2x_3=1$$

Вариант 3

1. Задать матрицу $A=\begin{pmatrix}1 & 1 & 1\\ 4 & 5 & 10\\ 16 & 25 & 81\end{pmatrix}$.

2. Изменить значение элемента матрицы *A*, используя индексы: *A(2,3)=9*.

3. Используя функции *zeros*(), *ones*(), *eye*(), *rand*(), *randn*(), *magic*(), создать квадратные матрицы размерности 5.

4. Задать матрицу *X*, где вектор v будет *k*-ой диагональю матрицы.

v=(5 3), k=4

5. Объединить в матрицах *M*, *N*, *L* матрицы *A* и *B* блочным способом, а также с помощью функций *blkdiag*() и *cat*().

$$B=\begin{pmatrix}6 & 9 & 0\\ 8 & 12 & 0\end{pmatrix}$$

6. Удалить из матрицы *B* третий столбец, из матрицы *L* вторую строку, а матрицу *M* очистить.

7. Посчитать определители матриц *A* и *B*.

8. Решить систему линейных уравнений:

$$4x_1-3x_2+2x_3=-4$$
$$6x_1-2x_2+3x_3=-1$$
$$5x_1-3x_2+2x_3=-3$$

Вариант 4

1. Задать матрицу $A=\begin{pmatrix}1 & 5 & 25\\ 1 & 7 & 49\\ 1 & 8 & 10\end{pmatrix}$.

2. Изменить значение элемента матрицы *A*, используя индексы: *A(3,3)=64*.

3. Используя функции *zeros*(), *ones*(), *eye*(), *rand*(), *randn*(), *magic*(), создать квадратные матрицы размерности 3.

4. Задать матрицу *X*, где вектор *v* будет *k*-ой диагональю матрицы.

v=(15 7 6 12), k=1

5. Объединить в матрицах *M*, *N*, *L* матрицы *A* и *B* блочным способом, а также с помощью функций *blkdiag*() и *cat*().

$$B=\begin{pmatrix}1 & 2 & 0\\ 3 & 4 & 0\end{pmatrix}$$

6. Удалить из матрицы *B* третий столбец, из матрицы *L* вторую строку, а матрицу *M* очистить.

7. Посчитать определители матриц A и B.
8. Решить систему линейных уравнений:

$$5x_1 + 2x_2 + 3x_3 = -2$$
$$2x_1 - 2x_2 + 5x_3 = 0$$
$$3x_1 + 4x_2 + 2x_3 = -10$$

Тема 3. Построение графиков на плоскости и в пространстве

Для построения графика функции одной переменной используется команда:

>> plot(x,y)

где x - вектор значений аргументов, y - вектор соответствующих значений функции.

Чтобы построить график нескольких функций в одном окне, можно воспользоваться двумя способами:

- перечислить в одной команде все ряды данных - *plot(x1,y1,x1,y2,…)*
- заблокировать режим создания нового окна с помощью процедуры *hold on*, отключить который можно с помощью процедуры *hold off*.

В команде *plot(x1,y1,'style1',x1,y2,'style2',…)* вместо параметров *style* в одинарных кавычках указываются дополнительные параметры для оформления линий:

- цвет линии (табл. 5);
- тип маркеров (табл. 6);
- стиль линии (табл. 7).

Табл. 5. Цвет линии на графике

Символ цвета	Цвет графика	Символ цвета	Цвет графика
y	желтый (Yellow)	g	зеленый (Green)
m	малиновый (Magenta)	b	синий (Blue)
c	бирюзовый (Cyan)	w	белый (White)
r	красный (Red)	k	черный (lack)

Табл. 6. Тип маркера на графике

Символ	Маркер	Символ	Маркер
.	жирная точка	d	ромбик
o	кружок	v	треугольник вершиной вниз
x	косоугольный крестик	^	треугольник вершиной вверх
+	прямоугольный крестик	<	треугольник вершиной влево
*	восьмиконечная снежинка	>	треугольник вершиной вправо
s	квадратик	p	пятиконечная звезда
none	отсутствие маркера	h	шестиконечная звезда

Табл. 7. Стиль линии на графике

Символ	Стиль линии	Символ	Стиль линии
-	Сплошная линия (по умолчанию)	-.	Штрих пунктирная линия
:	Пунктирная линия	--	Штриховая линия

Также можно снабдить график заголовком (процедура *title('name')*), подписать оси (процедуры *xlabel('X')*, *ylabel('Y')*), нанести координатную сетку (процедура *grid on*) и поместить легенду (процедура *legend('graph1', 'graph2',Name,Value)*).

Пара *Name*, *Value* определяет значение свойств легенды. Свойство *'Location'* определяет положение легенды (*'north'*, *'south'*, *'east'*, *'west'*, *'northeast'*, *'northwest'*, *'southeast'*, *'southwest'*, *'northoutside'*, *'southoutside'*, *'eastoutside'*, *'westoutside'*, *'northeastoutside'*, *'northwestoutside'*, *'southeastoutside'*, *'southwestoutside'*, *'best'*, *'bestoutside'*, *'none'*), а свойство *'Orientation'* – ориентацию (*'vertical'* или *'horizontal'*).

Рассмотрим пример:

```
>> x=0:pi/10:2*pi;
>> y1=sin(x);
>> y2=cos(x);
>> y3=sin(x)+cos(x);
>> plot(x,y1,':or',x,y2,'--*g',x,y3,'-.xb')
>> legend('sin(x)','cos(x)','sin(x)+cos(x)',...
    'Location','southoutside','Orientation','horizontal')
>> title('Graph')
>> xlabel('X')
>> ylabel('Y')
>> grid on
```

Результат построения представлен на рис. 2.

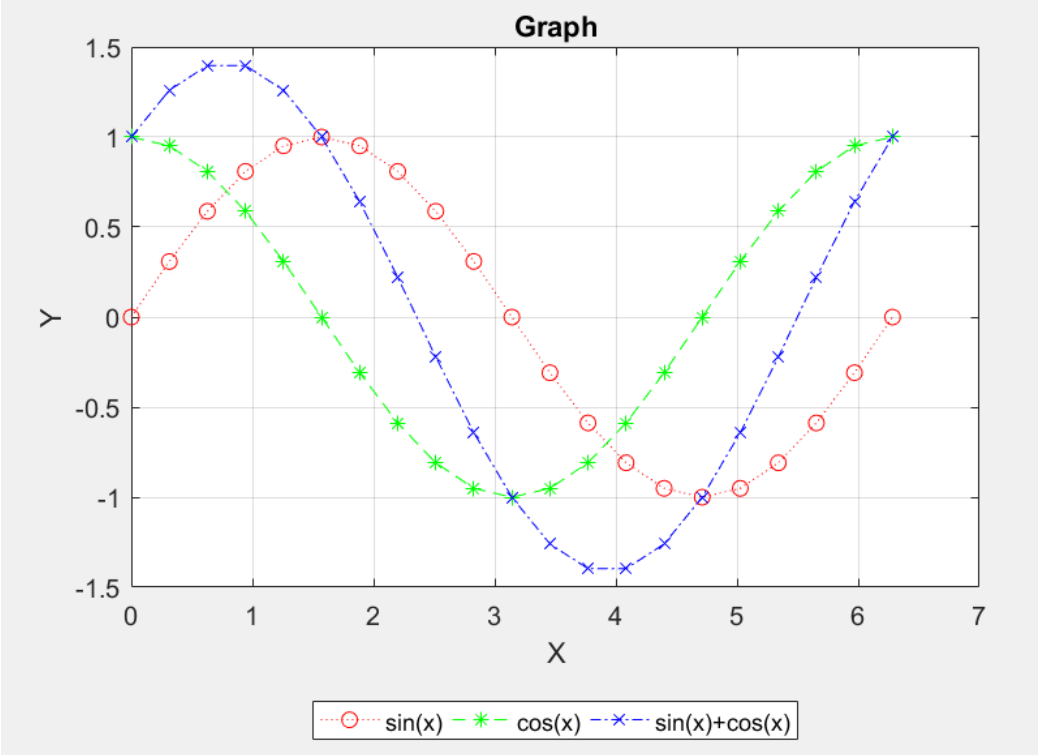

Рис. 2. Графики функций

В некоторых случаях совмещение двух графиков в одном окне может вызвать проблемы, так как могут не совпадать диапазоны изменения аргументов или значений. Для этого используют процедуру *plotyy(x1,y1,x2,y2)*, которая производит двойную оцифровку осей. Для первой функции цифруются ось *x* внизу, ось *y* слева. А для второй функции ось *x* размечается вверху, ось *y* справа. Цвет оцифровки при этом совпадает с цветом кривых.

Если в одном окне необходимо отобразить несколько графиков. Для этого используется функция *subplot()*, которая позволяет разделить область рисования на несколько прямоугольных областей равного размера:

>>subplot (row,col,cur);

Первые два аргумента задают количество рядов (*row*) и колонок (*col*), третий параметр (*cur*) объявляет порядковый номер подобласти, в котором будет построен очередной график.

Для построения графиков функций, заданных в полярной системе координат, используется процедура *polarplot()*.

>> polarplot(phi,ro)

где *phi* – угол, а *ro* – расстояние.

Единицы измерения значений ординат и абсцисс далеко не всегда соответствуют друг другу и для создания более обозримого графика вдоль одной или обеих осей приходится выбирать логарифмический масштаб. Для построения таких графиков используются процедуры: *loglog(x,y)* (логарифмический масштаб по обеим осям), *semilogx(x,y*) (логарифмический масштаб по оси x), *semilogy(x,y)* (логарифмический масштаб по оси y).

Процедура *fplot()* строит график функции *y=f(x)* без предварительного вычисления векторов *x* и *y*. Формат вызова этой функции записывается следующим образом:

>> fplot (@x f(x), [limits])

В первом случае первым аргументом является указатель на переменную и функция. Второй аргумент *limits* представляет двухкомпонентный вектор *[xmin xmax]*. Особенностью данной функции является то, что в местах резкого изменения значения функции значение аргумента *x* выбираются с более мелким шагом.

Для построения трехмерных графиков используются функции *mesh(X,Y,Z)* и *surface(X,Y,Z*). При этом *mesh*() создает каркасную поверхность, где цветные линии соединяют только заданные точки, а функция *surface()* вместе с линиями отображает в цвете и саму поверхность.

Для отображения функции двух переменных *z=f(x,y)*, создаются матрицы *X* и *Y*, состоящие из повторяющихся строк и столбцов. Затем эти матрицы используются для вычисления и отображения функции.

Перед построением трехмерных графиков следует задать сетку с помощью функции *[X,Y]=meshgrid(x,y)*, которая преобразует векторы *x* и *y* в двумерные массивы. После этого рассчитывается значение функции *Z,* используя значения *X* и *Y.*

Рассмотрим пример:

```
>> x=0:pi/10:2*pi;
>> y=0:pi/10:2*pi;
>> [X,Y]=meshgrid(x,y);
>> Z=cos(X)+sin(Y);
>> surface(X,Y,Z)
>> grid on
>> colorbar()
>> xlabel('X')
>> ylabel('Y')
>> zlabel('Z')
```

На рис. 3. представлен результат выполнения команд.

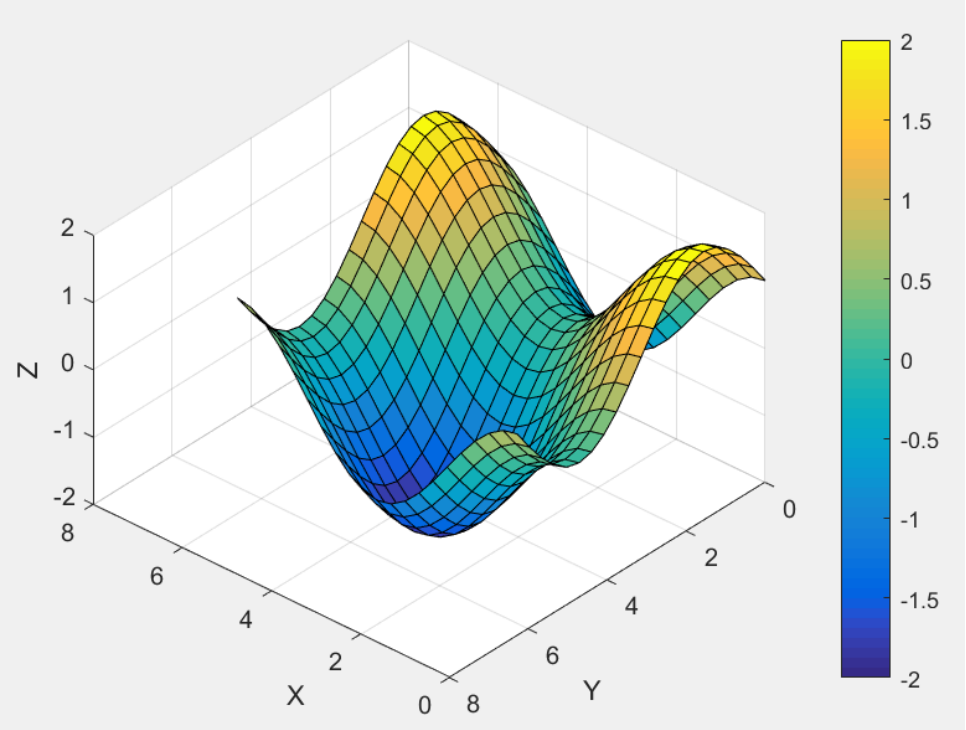

Рис. 3. Трехмерный график

Функция *colorbar()* строит рядом с полем графика столбик с цветовой гаммой, которая демонстрирует привязку цвета к значению аппликат z.

Функция *meshc(X,Y,Z)* строит поверхность, дополненную линиями уровня, а функция *meshz(X,Y,Z)* опускает из каждой граничной точки каркаса перпендикуляр на плоскость *oxy*.

Для отображения движения точки по траектории используется команда *comet()*. При этом движущаяся точка напоминает ядро кометы с хвостом. Используются следующие формы представления этой команды:

>> *comet (y)* – отображает движение «кометы» по траектории, заданной вектором y;

>> *comet (x,y)* – отображает движение «кометы» по траектории, заданной парой векторов y и x;

>> *comet (x,y,p)* – аналогична предшествующей команде, но позволяет задавать длину хвоста кометы (отрезка траектории, выделенного цветом) как p*1ength(Y), где length(Y) - размер вектора y, а p<1. По умолчанию p = 0.1

Гистограмма, отражающая значения компонент вектора y, строится с помощью функции *bar(y)* (вертикальная) и *barh(y)* (горизонтальная).

Если задать y с помощью матрицы, то можно представить несколько диаграмм одновременно. В этом случае есть два способа группировки столбцов:

>> *bar(y,'grouped') – обычная гистограмма*

>> *bar(y,'stacked') – гистограмма с накоплением*

Объемную гистограмму можно построить с помощью команды *bar3(z)*, для которой верно все вышесказанное.

Команда *colormap* осуществляет замену стандартной цветовой палитры. Названия и расцветку палитр можно найти в справке.

Круговые диаграммы, состоящие из плоских или объемных секторов, строятся с помощью функций *pie(q)* или *pie3(q)*. Чтобы выдвинуть один из секторов, надо указать выдвигаемые сектора в виде вектора (1 для выдвигаемых секторов, 0 для остальных), и указать его вторым параметром в функции. Выдвигаемым областям соответствуют ненулевые элементы в векторе.

Задание 3

Вариант 1

1. Построить график функции: $y=-2x^2+4x-1$.

2. Построить в одном окне графики функций, изменяя цвет и тип линий, а также тип маркера:

$$y1 = |x|; y2 = |4x - 2|; y3 = |x| - 5; y4 = ||x| - 8|;$$

3. На предыдущий график добавить заголовок, подписать оси, нанести координатную сетку и добавить легенду.

4. С помощью функции *plotyy()* построить графики функций:

$$y1 = \sin x;$$

$$y2 = \cos x + \sin x$$

5. Построить графики функций, разделив области рисования:

$$y1 = \tan x; y2 = \ln x - 15x; y3 = \cos|x|; y4 = (1 + \tan x)^2$$

6. Построить график функции в полярных координатах: $\rho = \sqrt{3\cos 8\varphi}$.

7. Построить график функции, используя логарифмический масштаб:

$$y = e^x$$

8. Построить график функции на заданном интервале: y=sin(x)/x.

9. Построить трехмерный график функции, используя функции *mesh*(), *meshc*(), *meshz*() и *surf*(): $z=sin(X+Y)$.

10. Построить анимированный график функции: *y=tg(x)*.

11. Построить гистограмму по следующим данным:

$$\begin{bmatrix} 3 & 5 & 9 \\ 7 & 8 & 2 \\ 1 & 4 & 10 \end{bmatrix}$$

12. Построить круговую диаграмму по данным: [1 5 7].

Вариант 2

1. Построить график функции: $y=x^2-6x$.

2. Построить в одном окне графики функций, изменяя цвет и тип линий, а также тип маркера:

$$y1 = \cos x;\ y2 = |\cos 2x|;\ y3 = \cos x + 2;\ y4 = \cos|3x|;$$

3. На предыдущий график добавить заголовок, подписать оси, нанести координатную сетку и разместить легенду.

4. С помощью функции *plotyy*() построить графики функций:

$$y1 = \cos x;$$

$$y2 = \cos x + \sin x;$$

5. Построить графики функций, разделив области рисования:

$$y1 = \cos(\frac{\pi}{6} - x);\ y2 = \tan x + ctgx;\ y3 = \sqrt{||x| - 5|};\ y4 = \sin^2 x - \cos x;$$

6. Построить график функции в полярных координатах: $\rho = 2 + 5\cos\varphi$.

7. Построить график функции, используя логарифмический масштаб.

$$y = 10^{x+2}$$

8. Построить график функции на заданном интервале: *y=tan(x)/x*.

9. Построить трехмерный график функции, используя функции *mesh*(), *meshc*(), *meshz*() и *surf*(): *z=sin(X)+tg(Y)*.

10. Построить анимированный график функции: *y=ctg(x)*.

11. Построить гистограмму по следующим данным:

$$\begin{bmatrix} 1 & 2 & 7 \\ 4 & 8 & 12 \\ 5 & 19 & 4 \end{bmatrix}$$

12. Построить круговую диаграмму по данным: [2 6 1].

Вариант 3

1. Построить график функции: *y=-4x-1*.

2. Построить в одном окне графики функций, изменяя цвет и тип линий, а также тип маркера:

$$y1 = \log x, y2 = \log(-x); y3 = \left|\log 2x\right|; y4 = -2\log 3x;$$

3. На предыдущий график добавить заголовок, подписать оси, нанести координатную сетку и разместить легенду.

4. С помощью функции *plotyy*() построить графики функций:

$$y1 = \sin x;$$

$$y2 = \cos x - \sin x;$$

5. Построить графики функций, разделив области рисования:

$$y1 = \frac{1}{x} + \frac{2}{x^2}; y2 = \tan x + \cos x; y3 = x^2 - 4; y4 = \sqrt{\sin x};$$

6. Построить график функции в полярных координатах: $\rho = \sqrt{4\cos 2\varphi}$.

7. Построить график функции, используя логарифмический масштаб.

$$y = x^{10}$$

8. Построить график функции на заданном интервале: *y=cot(x)/x*.

9. Построить трехмерный график функции, используя функции *mesh*(), *meshc*(), *meshz*() и *surf*(): $z = \sqrt{X^2 + Y^2}$.

10. Построить анимированный график функции: *y=sec(x)*.

11. Построить гистограмму по следующим данным:

$$\begin{bmatrix} 8 & 6 & 4 \\ 5 & 7 & 13 \\ 1 & 10 & 2 \end{bmatrix}$$

12. Построить круговую диаграмму по данным: [15 7 8].

Вариант 4

1. Построить график функции: $y=-x^3$.

2. Построить в одном окне графики функций, изменяя цвет и тип линий, а также тип маркера:

$$y1 = 2^x; y2 = \left|10 - 2^x\right|; y3 = 2^{1-2x}; y4 = x^2.$$

3. На предыдущий график добавить заголовок, подписать оси, нанести координатную сетку и разместить легенду.

4. С помощью функции *plotyy*() построить графики функций:

$$y1 = \cos x;$$

$$y2 = \cos x - \sin x;$$

5. Построить графики функций, разделив области рисования:

$$y1 = \exp(x-5); y2 = \frac{x}{2} + \frac{2}{x}; y3 = \left|\cos x - \cos^2 x\right|; y4 = \log(\tan x)$$

6. Построить график функции в полярных координатах: $\rho = -1 + 7\sin\varphi$.

7. Построить график функции, используя логарифмический масштаб.

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

8. Построить график функции на заданном интервале: *y=cos(x)/x*.

9. Построить трехмерный график функции, используя функции *mesh*(), *meshc*(), *meshz*() и *surf*(): *z=ln(X+Y)*.

10. Построить анимированный график функции: *y=csc(x)*.

11. Построить гистограмму по следующим данным:

$$\begin{bmatrix} 12 & 4 & 2 \\ 3 & 18 & 15 \\ 4 & 6 & 7 \end{bmatrix}$$

12. Построить круговую диаграмму по данным: [2 6 9].

Тема 4. Типы данных

Иерархия типов данных, которая используется в системе компьютерной математики MatLab, приведена на схеме (рис. 4):

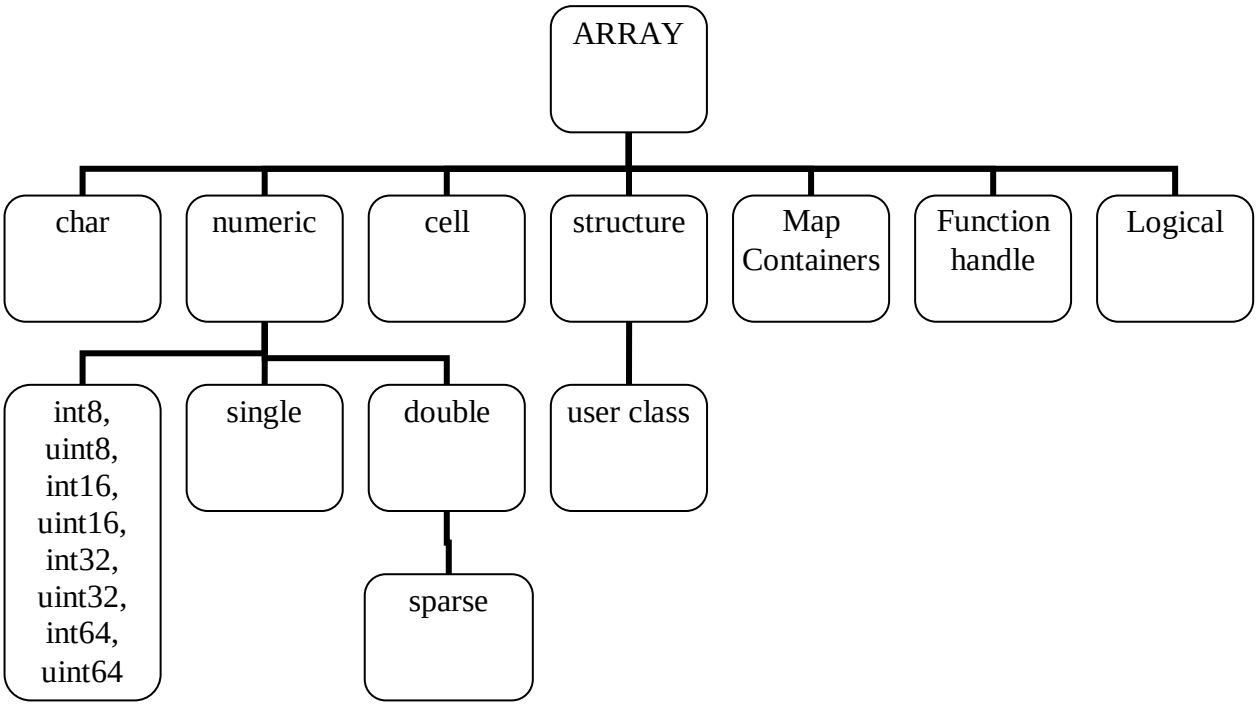

Рис. 4. Типы данных в MatLab

Символьные векторы (массивы размерности $1\times n$) представляют собой обычные строки. Для задания их значений в правой части оператора присваивания записывают цепочку символов, заключенную в одинарные кавычки.

>>q= 'Symbol'

Символьная матрица должна состоять из строк равной длины. Для определения текущих длин, размеров и количества измерений символьных массивов можно использовать функции *length (q), size (q)* и *ndims (q).*

MatLab допускает создание и хранение числовой информации в виде одно-, двух-, четырех- и восьмибайтовых форматов со знаком или без знака. В пакете предусмотрены функции конвертирования вещественных и комплексных данных типа double в любой из допустимых форматов целых чисел, а также обратные преобразования.

При конвертировании вещественных данных в формат целых чисел аргументы, выходящие за пределы допустимого диапазона, заменяются соответственно самым большим и самым маленьким числом.

Табл. 8. Числовые форматы

Функция	**Назначение**	**Диапазон допустимых значений**
int8()	Преобразование в формат однобайтовых целых чисел со знаком	от -2^7 (-128) до 2^7-1 (127)
uint8()	Преобразование в формат однобайтовых целых чисел без знака	от 0 до 2^8-1
int16()	Преобразование в формат двухбайтовых целых чисел со знаком	от -2^{15} до $2^{15}-1$
uint16()	Преобразование в формат двухбайтовых целых чисел без знака	от 0 до $2^{16}-1$
int32()	Преобразование в формат четырехбайтовых целых чисел со знаком	от -2^{31} до $2^{31}-1$
uint32()	Преобразование в формат четырехбайтовых целых чисел без знака	от 0 до $2^{32}-1$
int64()	Преобразование в формат восьмибайтовых целых чисел со знаком	от -2^{63} до $2^{63}-1$
uint64()	Преобразование в формат восьмибайтовых целых чисел без знака	от 0 до $2^{64}-1$
double()	Преобразование числового аргумента в формат вещественного числа с удвоенной точностью	
single()	Преобразование числового аргумента в формат вещественного числа с одинарной точностью	

Для хранения матриц, большая часть элементов которых равна нулю, используют разреженные матрицы (массивы), которые описываются командой *sparse(A)*.

Структуры представляют собой особый тип данных и состоят из полей. Структуры могут содержать разнородные данные, относящиеся к некоторому именованному объекту. Поле структуры может содержать другую вложенную структуру или массив структур. Это позволяет создавать вложенные структуры и даже многомерные массивы структур.

Одним из способов формирования структуры является использование функции *struct()*, которая заполняет указанный элемент массива структур. Аргументами этой функции являются пары, содержащие имя поля и соответствующее значение. Удобнее всего начинать заполнение с последнего элемента, чтобы задать массив нужного размера.

>> str(n)=struct('Name1','Value1','Name2','Value2','Name3','Value3',…);

где *n* – номер последнего элемента структуры; *str* – название структуры; *Name1*, *Name2*, *Name3* – имена полей структуры; *Value1*, *Value2*, *Value3* – значения полей структуры для n-го элемента.

Теперь можно заполнять пустые поля в двух предшествующих структурах, используя индексирование элементов массива и составные имена соответствующих полей:

>> str(i).Name=' Value';

Здесь *i* – номер заполняемого элемента структуры; *Name* – имя поля структуры; *Value* – значение поля.

С помощью функции *fieldnames*() можно получить информацию об именах полей указанной структуры:

>> fieldnames(str)

Однако для вывода содержимого полей приходится набирать имя каждой структуры – элемент соответствующего массива.

>> str(i)

Второй способ задания структуры, который требует существенно меньших затрат по набору исходных данных, использует групповое задание значений каждого поля.

>> str=struct('Name',{'Value1','Value2'},...

В существующем массиве структур легко добавить новое поле путем присвоения значения в любом элементе массива:

>>str (i).Newname= 'Newvalue';

Для удаления ненужного поля из всех записей используется функция *rmfield*():

>> str=rmfield(str,'Name');

Для возврата содержимого поля структуры используется функция *getfield*():

>> getfield(str(i),'Name')

В массиве типа Cell Array каждый элемент может быть представлен значением любого типа, независимо от типа соседних элементов. Каждый такой элемент называют ячейкой. Элементы массива ячеек задаются поочередной индексацией, причем индексы (i,j) указываются в фигурных скобках:

>>q{i,j}=A;

Термин handle обозначает системный способ идентификации некоторых программных компонент. Иногда его именуют дескриптором, но более точным переводом является указатель. Указатель на функцию обеспечивает возможность доступа ко всей информации, необходимой для вычисления значения этой функции.

Создается такой указатель одним из двух следующих способов:

>> f_h=@fun1;

>>F_H=str2func('fun1');

В результате и той, и другой операции создается указатель f_h который «смотрит» на функцию с именем fun1. Указатель может быть передан в качестве аргумента другой функции:

>> y=integral(a,b,f_h);

Для того чтобы функция *integral*() могла воспользоваться значением функции fun1(x), она использует системную процедуру *feval*():

>> z=feval (f_h,x);

Логические массивы появляются в результате различных проверок, выполняемых с помощью стандартных функций. Элементы логического массива занимают в памяти по одному байту, в которых может находиться либо 1 (логическая истина), либо 0 (логическая ложь).

Табл. 9. Стандартные функции определения типа данных

Функция	**Действие**
isletter(q)	Используется для выделения букв в символьной строке или массиве.
isspace(q)	Проверяет, является ли q пробелом.
isprime(q)	Проверяет, какие элементы массива A являются простыми числами. К спорному моменту относится значение *>> isprime(1)=0.*
isreal(q)	Проверяет, являются ли элементы массива q вещественными или комплексными числами. Если число задано так, что его мнимая часть равна 0, то оно все равно будет считаться комплексным.
isnumeric(q)	Проверяет, являются ли ее аргументы числовыми данными.
ischar(q)	Проверяет, являются ли ее аргументы символьными данными.
islogical(q)	Проверяет, является ли ее аргументы данными логического типа.
iscell(q)	Позволяет узнать, является ли аргумент массивом ячеек.
iscellstr(q)	Позволяет узнать представлена ли каждая ячейка массива строкой.
isstruct(q)	Проверяет, является ли ее аргумент структурой.
issparse(A)	Позволяет проверить, является ли матрица A разреженной.
isempty(A)	Проверяет, является ли ее аргумент пустым массивом.
isfinite(A)	Определяют, какие из элементов вещественного массива A принадлежат к допустимому диапазону числовых значений.
isinf(A)	Определяют, какие из элементов вещественного массива A принадлежат к бесконечным значениям.
isnan(A)	Определяют, какие из элементов вещественного массива A принадлежат к неопределенным значениям.
issorted(v)	Позволяет проверить упорядочены ли по возрастанию компоненты числового или символьного вектора v.

Проверка на принадлежность к тому или иному типу данных осуществляется с помощью функции isa:

>>A=isa(name, ‘class’)

Функция isa возвращает логическую истину (A=1), если объект с указанным именем name принадлежит заданному классу class, в противном случае она возвращает логическую ложь (A=0). Список значений аргумента ‘class’ приведен в табл. 10:

Табл. 10. Значения аргумента ‘class’

Класс	**Функция isa проверяет, является ли ее первый аргумент:**
char	символьным массивом
numeric	числовым массивом (целочисленным или вещественным)
logical	логическим массивом
int8	массивом однобайтовых целых чисел со знаком
uint8	массивом однобайтовых целых чисел без знака
int16	массивом двухбайтовых целых чисел со знаком
uint16	массивом двухбайтовых целых чисел без знака
int32	массивом четырехбайтовых целых чисел со знаком
uint32	массивом четырехбайтовых целых чисел без знака
int64	массивом восьмибайтовых целых чисел со знаком
uint64	массивом восьмибайтовых целых чисел без знака
single	массивом вещественных чисел с одинарной точностью
double	массивом вещественных чисел с удвоенной точностью
cell	массивом ячеек
struct	массивом структур
function_handle	массивом указателей на функции
float	массивом вещественных чисел с любой точностью
integer	массивом целых чисел со знаком или без него

Задание 4

Вариант 1

1. Задать символьный массив вида:

Robert

Bill

Poul

2. Определить длину, размер и количество измерений символьного массива.

3. Задать числа класса: int8, single. Можно ли выполнить над ними арифметические операции и функции (например, корень)?

4. Создать разреженную матрицу и сравнить объем памяти, который она занимает, с обычной матрицей такого же размера.

5. Создать структуру из 3 элементов со следующими полями:

Фамилия	Имя	Возраст

6. Вывести имена полей указанной структуры; вывести второй элемент структуры.

7. Добавить к структуре новое поле Отчество.

8. Удалить из структуры поле: Возраст.

9. Показать значение поля Имя для 2-го элемента структуры.

10. Создать массив ячеек вида:

$$3 \quad Hello \quad \begin{matrix} 1 & 3 \\ 5 & 0 \end{matrix} \quad 2+i$$

11. Какие элементы этого массива ячеек являются числовыми массивами, комплексными числами, строками, простыми числами.

12. Задать матрицу A, проверить являются ли ее элементы числами, буквами. Проверить также является ли матрица A пустой или разреженной.

$$A = \begin{pmatrix} 2 & 1 & 3 \\ 5 & 10 & 2 \\ 1 & 4 & 3 \end{pmatrix}$$

Вариант 2

1. Задать символьный массив вида:

Elena

Maria

Gloria

2. Определить длину, размер и количество измерений символьного массива.

3. Задать числа класса: int16, single. Можно ли выполнить над ними арифметические операции и функции (например, корень)?

4. Создать разреженную матрицу и сравнить объем памяти, который она занимает, с обычной матрицей, того же размера.

5. Создать структуру из 3 элементов со следующими полями:

Фирма	Вид деятельности	Процентная ставка

6. Вывести имена полей указанной структуры; вывести второй элемент структуры.

7. Добавить к структуре новое поле Адрес.

8. Удалить из структуры поле Процентная ставка.

9. Показать значение поля Фирма для 3-го элемента структуры.

10. Создать массив ячеек вида:

$$Street \quad 17 \quad 17+6i \quad \begin{matrix} 1 & 3 \\ 2 & 5 \end{matrix}$$

11. Какие элементы этого массива ячеек являются числовыми массивами, комплексными числами, строками, простыми числами.

12. Задать матрицу A, проверить являются ли ее элементы числами, буквами. Проверить также является ли матрица A пустой или разреженной.

$$A = \begin{pmatrix} 1 & 10 & 1 \\ 1 & 2 & 3 \\ 1 & 3 & 6 \end{pmatrix}$$

Вариант 3

1. Задать символьный массив вида:

Europe

Asia

Africa

2. Определить длину, размер и количество измерений символьного массива.

3. Задать числа класса: int32, single. Можно ли выполнить над ними арифметические операции и функции (например, корень)?

4. Создать разреженную матрицу и сравнить объем памяти, который она занимает, с обычной матрицей, того же размера.

5. Создать структуру из 3 элементов со следующими полями:

Фамилия	Марка машины	Номер машины

6. Вывести имена полей указанной структуры; вывести второй элемент структуры.

7. Добавить к структуре новое поле Цвет.

8. Удалить из структуры поле Номер машины.

9. Показать значение поля Цвет для 1-го элемента структуры.

10. Создать массив ячеек вида:

$$10i \quad \begin{matrix} 6.05 \\ 3.56 \end{matrix} \quad 100 \quad Apple$$

11. Какие элементы этого массива ячеек являются числовыми массивами, комплексными числами, строками, простыми числами.

12. Задать матрицу A, проверить являются ли ее элементы числами, буквами. Проверить также является ли матрица A пустой или разреженной.

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 4 & 5 & 10 \\ 16 & 25 & 81 \end{pmatrix}$$

Вариант 4

1. Задать символьный массив вида:

Germany

England

France

2. Определить длину, размер и количество измерений символьного массива.

3. Задать числа класса: uint16, single. Можно ли выполнить над ними арифметические операции и функции (например, корень)?

4. Создать разреженную матрицу и сравнить объем памяти, который она занимает, с обычной матрицей, того же размера.

5. Создать структуру из 3 элементов, со следующими полями:

Фамилия	Предмет	Оценка

6. Вывести имена полей указанной структуры; вывести второй элемент структуры.

7. Добавить к структуре новое поле Экзамен/зачет.

8. Удалить из структуры поле Оценка.

9. Показать значение: поля Предмет для 2-го элемента структуры.

10. Создать массив ячеек вида:

$$\begin{matrix} 3 & 15 \\ 2 & 7 \end{matrix} \quad 1-2i \quad \textit{University} \quad 16.7$$

11. Какие элементы этого массива ячеек являются числовыми массивами, комплексными числами, строками, простыми числами.

12. Задать матрицу А, проверить являются ли ее элементы числами, буквами. Проверить также является ли матрица А пустой или разреженной.

$$A = \begin{pmatrix} 1 & 5 & 25 \\ 1 & 7 & 49 \\ 1 & 8 & 10 \end{pmatrix}$$

Тема 5. Программирование на языке MatLab

Скрипт - файл является просто записью серии команд без входных и выходных параметров. Главной особенностью такого файла является то, что он работает только с переменными Workspace. Отсутствие локальных переменных в скриптах затрудняет создание цепочек программ из-за необходимости согласованного использования общих переменных. Файл сохраняется с расширением .m.

MatLab располагает редактором М-файлов, который вызывается по команде New→Script. Строки, набираемые в окне редактора, автоматически нумеруются. Для запуска фрагмента необходимо выполнить команду Run в меню Editor или нажать клавишу <F5>. Результаты работы появятся в командном окне. Для запоминания программы, набранной в окне редактора кода, можно воспользоваться командой Save As. Чтобы выполнить повторно какой-либо скрипт, надо в командном окне набрать имя файла. Вторым видом М-файла является функция (New→Function), первой строкой которых является заголовок, использующий оператор *function*. В отличие от скриптов, файлы-функции могут получать исходные данные в виде списка входных параметров и возвращать результаты своей работы также в виде списка выходных параметров. Если некоторые переменные Workspace объявлены глобальными (оператор *global*) и такое же объявление *global* с указанием имен общих переменных присутствует в теле функции, то функция имеет доступ к указанным переменным.

Одной из самых важных особенностей функций является аппарат локальных переменных. Все переменные в теле функции, которые не являются глобальными, входными и выходными, являются локальными, и не могут использоваться в командном окне и скриптах не могут.

При написании программ-функций требуется, чтобы имя М-файла совпадало с именем функции. В любом М-файле можно описать несколько функций. Однако только первую из них можно будет вызвать извне, все остальные будут считаться внутренними и будут доступны только в рамках данного М-файла. Их называют подфункциями.

Комментарии в MatLab начинаются с символа %. Они могут располагаться с начала строки или находиться правее любого оператора. Однако при оформлении М-файлов начальные комментарии выполняют особую роль. Первая группа строк с подряд идущими комментариями до пустой строки образует текст, выдаваемый в командном окне по команде help с именем М-файла.

Входной язык MatLab насчитывает 9 операторов, использующих 14 служебных слов.

Табл. 11. Операторы MatLab

№	Формат оператора	Пояснение
1	*var=expr*	Оператор присваивания, вычисляет значения выражения expr и заносит результаты вычисления в переменную var.
2	*if* условие_1 операторы_1 [*elseif* условие_2 операторы_2 *elseif* условие_3 операторы_3 …….. *else* операторы] *end*	Условный оператор. Если справедливо условие_1, то выполняется группа операторы_1; если справедливо условие_2, то выполняется группа операторы_2;… Если все указанные условия оказываются ложными, то выполняются операторы, расположенные между else и end.
3	*switch expr* *case val1* операторы_1 *case val2* операторы_2 ……. [*othervise* операторы] *end*	Переключатель по значению выражения expr. Если оно совпадает с величиной val1, то выполняется группа операторы_1, если оно совпадает с величиной val2, то выполняется группа операторы_2. Если значение expr не совпадает ни с одной из перечисленных величин, то выполняются операторы, расположенные между othervise и end.
4	*for var=e1:[e2:]e3* операторы *end*	Цикл типа арифметической прогрессии, в котором переменная var при каждом повторении тела цикла изменяется от начального значения e1 с шагом e2 до конечного значения e3.
5	*while* условие операторы *end*	Цикл с предусловием, повторяющийся до тех пор, пока истинно указанное условие.
6	*try* операторы_1 *catch* операторы_2 *end*	Попытка выполнить группу операторы_1. При условии, что в результате их выполнения возникает исключительная ситуация, управление передается группе операторы_2 (обработка сбойных ситуаций). Если ошибка не возникла, то группа операторы_2 не выполняется.
7	*break*	Досрочный выход из управляющих конструкций типа for, while, switch, try-catch.
8	*function f1* *function f2(x1,x2,…)* *function y=f3(x1,x2,…)* *function [y1,y2,…]=f4(x1,x2,…)*	Заголовок функции. (x1,x2,…) – входные параметры; (y1,y2,…) – выходные параметры.
9	*return*	Досрочный выход из тела функции.

Для ввода числовой и символьной информации используется функция input:

>> x=input('приглашение ')

В ответ на приглашение, выданное программой, пользователь может набрать требуемое значение или выражение, величина которого будет подсчитана с учетом текущего состояния переменных рабочего пространства и возвращена в качестве значения функции.

Второй формат обращения к функции input имеет вид:

>> x=input('приглашение ','s')

В этом случае текст, набираемый пользователем, рассматривается как строка символов, которая и возвращается в качестве значения функции.

Для вывода значения выражения, в частном случае которого может быть имя любой переменной, достаточно не набрать после него символ «;», подавляющий выдачу результата. В других случаях можно прибегнуть к функции disp, которая отличается от автоматического вывода только тем, что не выводит имя отображаемого значения.

>> disp(a)

В языке MatLab входные и выходные аргументы функций четко разделены. Все входные параметры задаются в круглых скобках после имени функции. Все выходные параметры объявляются как массив результатов и в операторах присваивания обычно записываются в квадратных скобках.

MatLab упаковывает все входные параметры в массив ячеек с именем *varargin* и запоминает количество передаваемых функции аргументов в глобальной переменной *nargin*. Функция, возвращающая переменное число значений, обычно начинается с заголовка:

function [varargout]=name_of_function(arguments)

Это означает, что результаты работы функции должны присваиваться компонентам массива ячеек *varargout*. К таким образом определенной функции можно обращаться с разным количеством выходных параметров:

>> [y1,y2,y3]= name_of_function(arguments)

>> [z1,z2]= name_of_function(arguments)

>> w1= name_of_function(arguments)

В первом случае переменной *y1* будет присвоено значение ячейки *varargout*{1}, переменной *y2* – значение ячейки *varargout*{2}, переменной *y3* – значение ячейки *varargout*{3}. Во втором случае будут использованы значения первых двух ячеек, в третьем случае – единственное значение первой ячейки массива *varargout*. Чтобы функция вычисляла только требуемое количество результатов, ей сообщают в глобальной переменной *nargout* фактическое число запрашиваемых данных.

Функция *eval*() работает с текстовыми переменными для вычисления и реализации текстовых строк. Она использует интерпретатор MatLab для вычисления и выполнения выражения, содержащегося в текстовой строке string.

>> t=eval('string')

Для выполнения вычислений, представленных строкой *expression*, в заданной рабочей области *ws* служит функция:

>>q= evalin(ws,expression)

Переменная *ws* может иметь два значения: *'base'* — для основной рабочей области и *'caller'* — для рабочей области вызванной функции.

>>feval (@name_of_function,x1,x2,...)

Функция *feval*() позволяет передавать в вычисляемую функцию список ее аргументов. При этом вычисляемая функция задается только своим именем.

Задание 5

Вариант 1

1. Создать функцию, которая определяет знак многочлена $ax^3 - bx^2 - cx + d$ в точке, значение которой задается с клавиатуры. Параметры многочлена передаются в функцию в качестве входных параметров.

2. Написать функцию, которая вычисляет для данного n:

$$\sum_{k=0}^{n} \frac{(-1)^k (k+1)}{k!}$$

Вариант 2

1. Создать функцию, которая определяет, принадлежит ли число, заданное с клавиатуры, массиву чисел, который передается в функцию как параметр.

2. Пусть $y_0=0$; $y_k = \frac{y_{k-1}+1}{y_{k-1}+2}$, k=1,2,… Дано действительное ε>0, найти первый член y_n , для которого выполняется $y_n - y_{n-1} < \varepsilon$.

Вариант 3

1. Создать функцию, которая определяет, содержит ли строка, введенная с клавиатуры, пробелы. Вывести результат в виде строки.

2. Пусть $y_0=0$; $y_k = \frac{y_{k-1}+3}{y_{k-1}+5}$, k=1,2,… Дано действительное ε>0, найти первый член y_n , для которого выполняется $y_n - y_{n-1} < \varepsilon$.

Вариант 4

1. Создать функцию, которая определяет, какими числами являются корни квадратного уравнения $ax^2 + bx + c$: действительными или комплексными. Параметры многочлена передаются в функцию в качестве входных параметров.

2. Написать функцию, которая вычисляет для данного n:

$$\sum_{k=0}^{n} \frac{(-1)^{k+1} k^2}{(k+1)}$$

Тема 6. Обработка символьных данных

Символьные данные могут быть представлены как в виде отдельных строк (вектор-строка), так и виде массивов строк (матрица) и массивов ячеек. Создание переменных происходит при формировании символьных значений с помощью оператора присваивания.

На каждый символ во внутреннем представлении отводится по 2 байта в кодировке ASCII Unicode. Функция *char(X)* — преобразует массив X положительных целых чисел (числовых кодов от 0 до 65 535) в массив символов системы MatLab. Для выполнения обратной операции можно воспользоваться функцией *double*().

Если в качестве аргумента функции *char*() выступают строки разной длины, то функция возвращает символьный массив, дополняя более короткие аргументы пробелами справа.

Для доступа к любому фрагменту строки можно использовать пару индексов, разделенных двоеточием:

>> s(i:j)

Чтобы из массива строк извлечь символ, находящийся в i-ой строке, j-ом столбце, используется команда:

>> s(i,j)

Пробелы в символьных данных играют роль разделителя слов. Иногда их приписывают в начале или в конце строки для того, чтобы произвести соответствующее выравнивание по левой границе или по длине. Для формирования строки, содержащей заданное количество пробелов (n), можно использовать функцию:

>> blanks(n)

Концевые пробелы можно удалить с помощью функции:

>> deblank(str)

Если в строке s присутствуют начальные и конечные пробелы, то, не изменяя длину строки, с помощью функции *strjust()* можно ее значимое содержимое разместить по центру, прижать к левой или правой границе:

>> strjust(s,'center')

>> strjust(s,'left')

>> strjust(s,'right')

Значение длины строки определяется с помощью функции *length*():

>> length(s)

Массивы строк размером m×n, заполненные одним и тем же символом (например,$), создаются с помощью функции *repmat*():

>> repmat('$',m,n)

Для горизонтального объединения массивов символов *s1, s2, s3* и т. д. используется функция:

>> strcat(sl,s2,s3,...)

Причем пробелы в конце каждого ряда отбрасываются, и возвращает объединенную строку (ряд) результирующего массива символов, пробелы

добавляются заново после анализа строк в полученном массиве. Все входные массивы должны иметь одинаковое число строк (в частном случае должны быть представлены в виде одной строки символов), но если один из входных аргументов — не массив символов, а строковый массив ячеек, то любой из других входных аргументов может быть скаляром или любым массивом той же размерности и того же размера. Если входной массив состоит только из символов, то выходной массив также будет являться массивом символов. Если любой из входных массивов является строковым массивом ячеек, то функция *strcat*() возвращает строковый массив ячеек, сформированный из объединенных соответствующих элементов массивов s1, s2, s3. при этом любой из элементов может быть скаляром и т. д.

Строки можно объединять вертикально, используя функцию:

>> strvcat(sl,s2,s3,...)

При этом строки подписываются одна под другой, создавая столбец в результирующем массиве. Вертикальная конкатенация массива ячеек, состоящего из строк, преобразует свой единственный операнд в символьный массив.

Так как символьные данные представлены в памяти компьютера числовыми кодами своих символов, их можно сравнивать. Для сравнения символьных данных MatLab использует четыре функции:

>> strcmp(s1,s2) – возвращает логическую единицу, если две сравниваемые строки *s1* и *s2* идентичны, и логический ноль в противном случае;

>> strcmpi(s1,s2) – выполняет аналогичное сравнение, игнорируя разницу между кодами больших и малых букв;

>> strncmp(s1,s2,n) – возвращает логическую единицу, если две сравниваемые строки *sl* и *s2* содержат *n* первых идентичных символов, и логический ноль в противном случае. Аргументы *sl* и *s2* могут быть также строковыми массивами ячеек;

>> strncmpi(s1,s2,n) – сочетает возможности второй и третьей функций.

Для поиска вхождений одной строки в другую существуют две функции:

>> findstr(str1,str2) – обеспечивает поиск начальных индексов более короткой строки внутри более длинной и возвращает вектор этих индексов. Индексы указывают положение первого символа более короткой строки в более длинной строке.

>> strfind(str1,str2) – ищет строку *str2* в строке *str1*.

Другой вариант поиска в массиве строк или массиве ячеек может быть осуществлен с помощью функции *strmatch*():

>> strmatch(str,STRS,'exact') – возвращает только индексы строк символов массива *STRS*, точно совпадающих со строкой символов *str*;

>> strmatch(str,STRS) – просматривает массив символов или строковый массив ячеек *STRS* по строкам, находит строки символов, начинающиеся со строки *str*, и возвращает соответствующие индексы строк.

Для поиска в строке *str1* всех вхождений заданной цепочки символов *str2* и замены каждой из них новым значением *str3* предназначена функция:

>> strrep(str1,str2,str3)

Если *str2* не найдена, то возвращается строка *str1*.

Лексема – это цепочка символов, завершающаяся тем или иным разделителем. Функция *strtok*() выделяет лексемы:

>>[t,r]=strtok(s,delim)

В строку *t* заносится найденная лексема, в строку *r* – оставшаяся часть строки *s*. Второй необязательный аргумент *delim* функции *strtok()* позволяет задать нестандартный набор символов разделителей. По умолчанию разделителем является «белый пробел».

Функция *lower (str)* — возвращает строку символов *str*, в которой символы верхнего регистра переводятся в нижний регистр, а все остальные символы остаются без изменений, а функция *upper (str)* — возвращает строку символов *str*, в которой все символы нижнего регистра переводятся в верхний регистр, а все остальные символы остаются без изменений.

Существует ряд функций для преобразования символов и чисел (табл. 12)

Табл. 12. Конвертирующие функции

Функция	**Действие**
int2str(X)	Округляет элементы массива *X* до целых чисел и возвращает массив символов, содержащих символьные представления округленных целых чисел. Аргумент *X* может быть скаляром, вектором или матрицей.
mat2str(A)	Преобразует матрицу *A* в единую строку; если элемент матрицы не скаляр, то он заменяется на [ ], при этом учитываются 15 знаков после десятичной точки.
mat2str(A,n)	Преобразует матрицу *A* в строку, используя точность до *n* цифр после десятичной точки. Функция *eval(str)* осуществляет обратное преобразование.
num2str(A)	Выполняет преобразование массива *A* в строку символов *str* с точностью до четырех десятичных разрядов и экспоненциальным представлением, если требуется. Обычно используется при выводе графиков.
num2str(A,precision)	Выполняет преобразование массива *A* в строку символов *str* с максимальной точностью, определенной аргументом *precision*. Аргумент *precision* определяет число разрядов в выходной строке.
num2str(A,format)	Выполняет преобразование массива чисел *A*, используя заданный формат *format*. По умолчанию принимается формат, который использует четыре разряда после десятичной точки для чисел с фиксированной или плавающей точкой.

Функция	Действие
str2double(s)	Выполняет преобразование численной строки *s*, которая представлена в ASCII-символах, в число с двойной точностью. При этом + и -могут быть только в начале строки.
str2num(s)	Выполняет преобразование численного массива символов — матрицы или строки *s*, который представлен в ASCII-символах, в матрицу (массив размерности 2).

Существует также ряд строковых функций для преобразования систем счисления:

Табл. 13. Преобразование систем счисления

Функция	Преобразует	
	из	**в**
dec2bin()	десятичного числа	двоичную строку
dec2hex()	десятичного числа	шестнадцатеричную строку
dec2base()	десятичного числа	строковое представление числа в системе счисления с заданным основанием
bin2dec()	двоичной строки	десятичное число
hex2dec()	шестнадцатеричной строки	десятичное число
base2dec()	строкового представления числа в системе счисления с заданным основанием	десятичное число
hex2num()	шестнадцатеричной строки	десятичное число с удвоенной точностью

Задание 6

Вариант 1

1. Создать массив ячеек str_cell:

'123' '*abcd*'

'5678' '*efg*'

2. Преобразовать массив кодов в массив символов: A=32:52.
3. Показать пятый символ из строки str= ‘It is life’.
4. Создать строку str1= ‘ East ‘, вычислить ее длину, содержимое выровнять по центру, по левой и правой стороне:
5. Удалить из строки str1 концевые пробелы. Вычислить длину строки.
6. Создать массив 3×4 заполненный символами +.

7. Объединить строки вертикально и горизонтально:

s1= ‘Happy’ s2= ‘New’ s3= ‘Year’

8. Сравнить строки всеми возможными способами:

st1=‘example’ и st2=‘EXAMple’

9. Преобразовать строку st2 к верхнему и нижнему регистрам.

10. Входит ли строка s= ‘oo’ в строку str = ‘boom’.

11. Заменить в строке str=‘London is the capital of Great Britain’ символ ‘o’ на ‘*’.

12. Выделить в строке str лексемы.

13. Преобразовать матрицу A=randn(3) в строку с точностью пять цифр после запятой.

Вариант 2

1. Создать массив ячеек str_cell:

'123456' '*ab*'

'78' '*cddefg*'

2. Преобразовать массив кодов в массив символов: A=53:64.

3. Показать пятый символ из строки str= ‘be happy’.

4. Создать строку str1= ‘ West ‘, вычислить ее длину и содержимое выровнять по центру, по левой и правой стороне:

5. Удалить из строки str1 концевые пробелы. Вычислить длину строки.

6. Создать массив 3×4 заполненный символами *.

7. Объединить строки вертикально и горизонтально:

s1= ‘Merry’ s2= ‘Christ’ s3= ‘mas’

8. Сравнить строки всеми возможными способами:

st1=‘examination’ и st2=‘exaMINation’

9. Преобразовать строку st2 к верхнему и нижнему регистрам.

10. Входит ли строка s= ‘oo’ в строку str= ‘2006’.

11. Заменить в строке str=‘London is the capital of Great Britain’ символ ‘a’ на ‘@’.

12. Выделить в строке str лексемы.

13. Преобразовать матрицу A=rand(2) в строку с точностью пять цифр после запятой.

Вариант 3

1. Создать массив ячеек str_cell:

'*zyxwv*' '1234'

'*klm*' '321'

2. Преобразовать массив кодов в массив символов: A=65:86;.

3. Показать пятый символ из строки str= ‘Are you ready’.

4. Создать строку str1= ‘ North ‘, вычислить ее длину и содержимое выровнять по центру, по левой и правой стороне:

5. Удалить из строки str1 концевые пробелы. Вычислить длину строки.

6. Создать массив 3×4 заполненный символами #.

7. Объединить строки вертикально и горизонтально:

s1= ‘Happy’ s2= ‘Birth’ s3= ‘day’

8. Сравнить строки всеми возможными способами:

st1=‘corporation’ и st2=‘CORPOratioN’

9. Преобразовать строку st2 к верхнему и нижнему регистрам.

10. Входит ли строка s= ‘oo’ в строку str= ‘loop’

11. Заменить в строке str=‘London is the capital of Great Britain’ символ ‘n’ на ‘$’.

12. Выделить в строке str лексемы.

13. Преобразовать матрицу $A=\begin{pmatrix} \pi & 0 \\ \pi/2 & 2\pi/3 \end{pmatrix}$ в строку с точностью пять цифр после запятой.

Вариант 4

1. Создать массив ячеек str_cell:

'*kkllmm*' '*aaa*'

'678' '12345'

2. Преобразовать массив кодов в массив символов: A=87:108.

3. Показать пятый символ из строки str= ‘Do not worry’.

4. Создать строку str1= ‘ South ‘, вычислить ее длину и содержимое выровнять по центру, по левой и правой стороне:

5. Удалить из строки str1 концевые пробелы. Вычислить длину строки.

6. Создать массив 3×4 заполненный символами: |.

7. Объединить строки вертикально и горизонтально:

s1= ‘Good’ s2= ‘after’ s3=’noon’

8. Сравнить строки всеми возможными способами:

st1=‘MARKET’ и st2= ‘MARket’

9. Преобразовать строку st2 к верхнему и нижнему регистрам.

10. Входит ли строка s= ‘oo’ в строку str= ‘oooooooops’.

11. Заменить в строке str=‘London is the capital of Great Britain’ символ ‘i’ на ‘&’.

12. Выделить в строке str лексемы.

13. Преобразовать матрицу $A=\log\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ в строку с точностью пять цифр после запятой.

Тема 7. Работа с файлами

Файл обычно является некоторой совокупностью данных, объединенных одним именем.

Работа с файлами в основном ограничивается 3-4 операциями: открытие файла, чтение из файла или запись в файл, закрытие файла. MatLab может работать с текстовыми или двоичными файлами.

Файла открываются с помощью функции *fopen*(), которая в случае удачного завершения возвращает целочисленный положительный идентификатор (номер файла). В случае неудачи возвращается -1. Если указано короткое имя, то система ищет файл сначала в рабочем каталоге, а потом во всех каталогах, доступных MatLab.

>>f_id=fopen('file_name','arg')

Второй необязательный аргумент функции *fopen*() определяет режим доступа к открываемому файлу. В качестве этого параметра могут выступать один, два или три символа, заключенные в одинарные кавычки.

Табл. 14. Форматы доступа к файлу

Формат вызова	**Выполняемое действие**
'r'	Открывается существующий двоичный файл для чтения
'rt'	Открывается существующий текстовый файл для чтения
'r+'	Открывается существующий двоичный файл, который можно использовать как для чтения, так и для записи (новый файл при этом не создается)
'rt+'	Открывается существующий текстовый файл, который можно использовать как для чтения, так и для записи (новый файл при этом не создается)
'w'	Открывается существующий или создается новый двоичный файл для записи. Предыдущее содержимое существующего файла пропадает
'wt'	Открывается существующий или создается новый текстовый файл для записи. Предыдущее содержимое существующего файла пропадает
'w+'	Если существующий двоичный файл был ранее открыт, то его содержимое от текущей позиции указателя и до конца усекается. Если файл с таким именем не существовал, то создается новый файл. Новый файл можно использовать для записи и чтения
'wt+'	Если существующий текстовый файл был ранее открыт, то его содержимое от текущей позиции указателя и до конца усекается. Если файл с таким именем не существовал, то создается новый файл. Новый файл можно использовать для записи и чтения
'a'	Открывается существующий двоичный файл для дозаписи или создается новый файл

Формат вызова	Выполняемое действие
‘at’	Открывается существующий текстовый файл для дозаписи или создается новый файл
‘a+’	Открывается существующий двоичный файл для чтения и дозаписи. Если файла с указанным именем нет, то он будет создан.
‘at+’	Открывается существующий текстовый файл для чтения и дозаписи. Если файла с указанным именем нет, то он будет создан.

В MatLab есть три системных файла – стандартный ввод (*stdin*), стандартный вывод (*stdout*) и файл, предназначенный для вывода сообщений об ошибках (*stderr*). За ними закреплены номера 0, 1 и 2, соответственно. Их открывать не надо, они всегда доступны.

Существует еще один формат вызова функции:

>> v=fopen ('all')

В этом случае она возвращает вектор с номерами всех файлов, открытых к данному моменту (0,1,2 не включаются)

Файл, работа с которым закончена, необходимо закрыть, с помощью функции *fclose*(). Существует два формата вызова этой функции, в первом случае закроется файл с номером *f_id*, во втором – все открытые файлы. При нормальном закрытии функция возвращает нулевое значение.

>> fclose(f_id)

>> fclose('all')

При выборке информации из текстового или двоичного файла, открытого для чтения, может наступить момент, когда прочитана последняя порция данных и в файле больше ничего нет. Контроль за тем, достиг ли указатель конца данных (*end-of-file*), обеспечивает функция *feof*():

>>k=feof(f_id);

Если возвращаемое значение равно 0, то в файле еще находятся непрочитанные данные, в противном случае будет возвращена 1.

Работа с двоичными файлами

Указатель файла смотрит на текущую порцию данных. При чтении эта информация извлекается из файла, а указатель автоматически перемещается на следующую доступную порцию, если таковая еще есть. При выводе в файл новая информация записывается, начиная с адреса, на который смотрит указатель. В двоичных файлах возможно дополнительное перемещение указателя.

Текущая позиция определяется как смещение в байтах относительно начала файла. Для открытого файла с номером *f_id* текущее положение указателя можно узнать, обратившись к функции *ftell*():

>>pos=ftell(f_id);

Для возврата указателя в начало файла используется функция *frewind*():

>>frewind (f_id);

Все остальные перемещения указателя файла реализуются с помощью функции *fseek*(), причем перемещения указателя задаются относительно одной из трех позиций:

- От начала файла на заданное число байт вперед (n>0)

>>st=fseek(f_id,n, ‘bof’) или *st=fseek(f_id,n,-1)*

- От конца файла на заданное число байт назад (n>0)

>>st=fseek(f_id,n, ‘eof’) или *st=fseek(f_id,n,1)*

- От текущей позиции указателя вперед (n>0) или назад (n<0) на заданное число байт

>>st=fseek(f_id,n, ‘cof’) или *st=fseek(f_id,n,0)*

Величина смещения может быть равна 0. Если функция возвращает 0, то операция прошла успешно, в противном случае -1.

Для чтения данных из двоичного файла используется функция *fread*(). Простейший формат вызова предполагает, что использует только номер файла, в этом случае возвращается содержимое этого файла (*A*) и количество считанных данных (*v*). При этом содержимое рассматривается вектор-столбец.

>> [A,v]=fread(f_id)

Второй формат функции может включать четыре аргумента:

>> [a v]=fread (f_id, n, ‘type’, l)

Второй входной аргумент задает количество считываемых данных. Если этот аргумент не определен, то функция *fread*() считывает данные до конца файла. Используются следующие параметры второго аргумента:

- *n* — чтение n элементов в вектор-столбец;
- *inf* — чтение элементов до конца файла и помещение их в вектор-столбец, содержащий такое же количество элементов, что и в файле;
- *[m n]* — считывает столько элементов, сколько нужно для заполнения матрицы $m \times n$. Заполнение происходит по столбцам. Если считывание достигает конца файла, не заполнив матрицу необходимого размера, то матрица дополняется нулями. Если происходит ошибка, чтение останавливается на последнем считанном значении.

Третий необязательный параметр определяет числовую точность считывания значений, он контролирует число считанных бит для каждого значения и интерпретирует эти биты как целое число, число с плавающей запятой или как символ. По умолчанию действует описатель ‘*uchar’*, обеспечивающий побайтный доступ к содержимому файла.

Если воспользоваться описателями ‘*bitN’* и ‘*ubitN’* (целое со знаком или без, длиной *N* бит), то можно рассматривать содержимое файла как целочисленные данные с точностью до бита.

Табл. 15. Форматы доступа к файлу

Описатель длины и типа элементов	**Пояснение**
‘uchar’ | *‘uint8’*	Целое число без знака, 1 байт
‘schar’ | *‘int8’*	Целое число со знаком, 1 байт
‘int16’ | *‘integer*2’*	Целое число со знаком, 2 байта
‘int32’ | *‘integer*4’*	Целое число со знаком, 4 байта
‘int64’ | *‘integer*8’*	Целое число со знаком, 8 байт
‘uint16’	Целое число без знака, 2 байта
‘uint32’	Целое число без знака, 4 байта
‘uint64’	Целое число без знака8 байт
‘single’ | *‘float32’*	Вещественное число, 4 байта
‘double’ | *‘float64’*	Вещественное число, 8 байт

Функция *fread*() включает четвертый необязательный аргумент, который определяет число байтов (*l*), которые необходимо пропустить после каждого считывания. Это может быть полезно при извлечении данных в несмежных областях из записей фиксированной длины. Если третий аргумент имеет битовый формат, такой как ‘*bitN’* или ‘*ubitN’*, то значение четвертого параметра определяется в битах.

Операция записи в двоичный файл отличается только направлением перемещаемых данных и использует похожие аргументы:

>>count = fwrite (f_id, array, ‘type’, l);

При этом в заранее открытый файл с номером *f_id* записываются все элементы массива *array* (если он двумерный, то запись ведется по столбцам), с преобразованием данных в тип, который определяется третьим аргументом. Функция возвращает количество удачно записанных элементов. Четвертый входной аргумент функции *fwrite*() позволяет записывать элементы указанного массива не подряд, а со вставкой перед каждым элементом заданного количества (*l*) нулевых данных.

Работа с текстовыми файлами

Содержимым текстового файла являются строки – цепочки символов, завершающиеся парой управляющих символов.

Чтение строки из файла осуществляется с помощью функции *fgetl*(). Информация читается от текущей позиции указателя до конца строки или до исчерпания данных. Входной аргумент – номер файла, функция возвращает считанную строку, если данные исчерпаны, то она возвращает -1. Управляющие байты в состав считываемых строк не включается.

>>s=fgetl(f_id)

Функция *fgets*() считывает несколько символов из текстового файла, входным аргументом является номер файла и количество считываемых

символов (можно не указывать). Считывание будет продолжаться пока не выполниться одно из трех условий:

- прочитано указанное количество символов;
- встретился признак конца строки;
- данные в файле исчерпаны.

>>s=fgets(f_id,n)

При записи данных в файл их необходимо форматировать по двум причинам:

- в оперативной памяти каждый символ представлен двумя байтами, а в файле – одним;
- для того чтобы отделить одну строку от другой, вслед за последним символом на диск должна быть записана пара управляющих символов.

Конвертированием данных из их внутреннего представления в символьное заведуют форматные указатели, задаваемые в качестве параметров в функции *fprintf*(). Она форматирует данные, содержащиеся в действительной части матрицы *A*, под контролем строки *format* и записывает их в файл с идентификатором *f_id*. Функция *fprintf*() возвращает число записанных байтов.

>>c = fprintf(f_id, format, A)

Если опустить идентификатор *f_id* в списке аргументов функции *fprintf*(), то вывод будет осуществляться на экран, так же как при использовании стандартного вывода (*k=l*). Строка *format* определяет систему счисления, выравнивание, значащие цифры, ширину поля и другие атрибуты выходного формата. Она может содержать обычные буквы алфавита наряду со спецификаторами, знаками выравнивания и т. д.

Табл. 16. Специальные символы

Символ	**Описание**
\n	Новая строка
\t	Горизонтальная табуляция
\b	Возврат на один символ
\r	Возврат каретки
\f	Новая страница
\”	Двойные кавычки
\\	Обратный слеш
\’ или ‘’	Одиночная кавычка
%%	Процент

Для вывода числовых или символьных данных в строке формата необходимо использовать спецификаторы, перечисленные в таблице:

Табл. 17. Спецификаторы

Спецификатор	Описание
%d	Преобразование целочисленного значения в десятичное число со знаком
%c	Одиночный символ
%e	Экспоненциальное представление чисел с использованием символа «e» в нижнем регистре, например 3.1415e + 00
%E	Экспоненциальное представление чисел с использованием символа «E» в верхнем регистре, например 3.1415E + 00
%f	Преобразование числового значения в вещественное число с фиксированной запятой
%g	Преобразование числового значения в вещественное число с фиксированной или плавающей запятой в зависимости от того, какой формат числа занимает меньше места. Незначащие нули не выводятся
%G	То же самое, что и %g, но используется верхний регистр для символа «E»
%o	Преобразование целочисленного значения в восьмеричное число со знаком
%s	Преобразование в строку символов
%u	Преобразование целочисленного значения в десятичное число без знака
%x	Преобразование целочисленного значения в шестнадцатеричное число без знака с использованием символов нижнего регистра («a»...«f»)
%X	Преобразование целочисленного значения в шестнадцатеричное число без знака с использованием верхнего регистра символов («A»...«F»)

Между знаком процента и буквой в спецификатор могут быть вставлены дополнительные символы.

Табл. 17. Вспомогательные знаки

Символ	Описание	Пример
Знак «минус» (-)	Выравнивание преобразованных аргументов по левому краю	*%-5.2d*
Знак «плюс» (+)	Всегда печатать знак числа (+ или -)	*%+5.2d*
#	Для формирования префиксов (o и x) у восьмеричных и шестнадцатеричных чисел	*%#x*
Пробел	Для вывода пробела вместо знака положительного числа	*% d*
Ноль (0)	Заполнение нулями вместо пробелов	*%05.2d*
Цифры	Определяет минимальное число знаков, которые будут напечатаны	*%6f*
Цифры (после точки)	Число после точки определяет количество символов, печатаемых справа от десятичной точки	*%6.2f*

В том случае, если в файл записывается единственная строка, то форматные указатели могут быть опущены.

Выборка данных из текстового файла может быть произведена с помощью функции *fscanf*(), обращение к которой в общем виде выглядит следующим образом:

>>[A n]=fscanf (f_id, 'format', m);

Здесь *A* – принимающий массив, в элементы которого заносятся считываемые данные; *n* – количество фактически считанных данных; *f_id* – номер открытого файла; *format* – список форматных указателей, в соответствии с которыми осуществляется преобразование считываемых данных; *m* – количество запрашиваемых данных.

Обязательно нужно задавать параметры *A*, *f_id* и, по крайней мере, один форматный указатель в списке *format*. Если *m* не задано, или равно *inf*, то считываются все данные до конца файла, если задано число, то считывается вектор-столбец, если задана пара значений *[p q]*, то считывается матрица $p \times q$.

Форматные указатели для функции *fscanf*() практически совпадают с описанными ранее указателями для функции *fprintf*(). Если форматный указатель начинается с комбинации *'%*'*, то соответствующее значение, считанное из файла, в принимающий массив не передается.

Форматные указатели типа *'%s'* на выводе и вводе действуют по-разному. Строка данных может содержать пробелы. Если мы считываем единичный элемент по формату *'%s'*, то чтение прерывается на первом же пробеле, если мы запросим чтение более чем одного элемента, то «белые пробелы» игнорируются.

Табл. 18. Спецификаторы

Спецификатор	**Описание**
%s	Строка, завершающаяся «белым пробелом», преобразуется в элемент массива ячеек
%q	Строка, возможно, заключенная в двойные кавычки, преобразуется в элемент массива ячеек
%c	Символ или «белый пробел» преобразуется в символьный массив
%[s1s2…]	Строка из символов, входящих в заданный набор, преобразуется в элемент массива ячеек. Чтение прерывается при выборе символа, не входящего в заданный набор
%[^s1s2…]	Строка из символов, не входящих в заданный набор, преобразуется в элемент массива ячеек. Чтение прерывается при выборе символа, входящего в заданный набор

В форматных указателях после символа % можно задать число, которое ограничит количество преобразуемых символов текстового файла, если до этого не будет обнаружен символ-разделитель.

Задание 7

Вариант 1

1. Записать в двоичный файл *myexample.m* строковый массив *A*:
‘A free market economy has no government intervention’
2. Закрыть файл *myexample*.
3. Открыть двоичный файл *myexample* и прочитать из него данные по 3 элемента через 2 элемента. Представить прочитанные данные в символьной форме в виде строки.
4. С помощью программы Блокнот записать в файл *myexample* данные в три строки.
5. Прочитать из файла *myexample* первую строку с помощью функции *fgetl*().
6. Прочитать из файла *myexample* 5 символов с помощью функции *fgets*().
7. Прочитать из файла *myexample* данные по формату *‘%s %*s %s’* в матрицу 3×3.
8. Закрыть все файлы.
9. Вывести на экран данные *a=15 b=-15* в виде:

15
–15

Вариант 2

1. Записать в двоичный файл *myexample.m* строковый массив *A*
‘Macroeconomics is the study of the economy as a whole’
2. Закрыть файл *myexample*.
3. Открыть двоичный файл *myexample* и прочитать из него данные по 3 элемента через 2 элемента. Представить прочитанные данные в символьной форме в виде строки.
4. С помощью программы Блокнот записать в файл *myexample* данные в три строки.
5. Прочитать из файла *myexample* первую строку с помощью функции *fgetl*().
6. Прочитать из файла *myexample* 5 символов с помощью функции *fgets*().
7. Прочитать из файла *myexample* данные по формату *‘%s %*s %s’* в матрицу 3×3.
8. Закрыть все файлы.
9. Вывести на экран данные *a=15 b=-15* в виде:

+15
–15

Вариант 3

1. Записать в двоичный файл *myexample.m* строковый массив *A*

‘GNP measures the total income of the economy’

2. Закрыть файл *myexample*.

3. Открыть двоичный файл *myexample* и прочитать из него данные по 3 элемента через 2 элемента. Представить прочитанные данные в символьной форме в виде строки.

4. С помощью программы Блокнот записать в файл *myexample* данные в три строки.

5. Прочитать из файла *myexample* первую строку с помощью функции *fgetl*().

6. Прочитать из файла *myexample* 5 символов с помощью функции *fgets*().

7. Прочитать из файла *myexample* данные по формату *‘%s %*s %s’* в матрицу 3×3.

8. Закрыть все файлы.

9. Вывести на экран данные *a=15 b=-15* в виде:

15
–15

Вариант 4

1. Записать в двоичный файл *myexample.m* строковый массив *A*

‘Firms and government finance R&D activities’

2. Закрыть файл *myexample*.

3. Открыть двоичный файл *myexample* и прочитать из него данные по 3 элемента через 2 элемента. Представить прочитанные данные в символьной форме в виде строки.

4. С помощью программы Блокнот записать в файл *myexample* данные в три строки.

5. Прочитать из файла *myexample* первую строку с помощью функции *fgetl*().

6. Прочитать из файла *myexample* 5 символов с помощью функции *fgets*().

7. Прочитать из файла *myexample* данные по формату *‘%s %*s %s’* в матрицу 3×3.

8. Закрыть все файлы.

9. Вывести на экран данные *a=15 b=-15* в виде:

f
–15

Тема 8. Символьные вычисления

Символьные переменные создаются при помощи функции *syms*:

>>syms x1 x2 x3

Конструирование символьных функций производится с использованием обычных арифметических операций и обозначений для встроенных математических функций.

Запись формулы для выражения в одну строку не всегда удобна, более естественный вид выражения выводит в командное окно функция *pretty*():

>>pretty(f)

Имеющиеся символьные переменные и функции позволяют образовывать новые символьные выражения.

При работе в области комплексных чисел следует помнить, что определяемые переменные являются, в общем случае, комплексными. Опция *real* означает, что переменные интерпретируются как вещественные:

>> syms a real

Символьные переменные могут являться элементами матриц и векторов. Элементы строк матриц при вводе отделяются пробелами или запятыми, а столбцов – точкой с запятой, также как для обычных матриц. В результате образуются символьные матрицы и векторы, к которым применимы матричные и поэлементные операции и встроенные функции.

>>syms a b c d

>>A=[a b; c d]

Конструирование блочных символьных матриц не отличается от числовых, также требуется следить за размерами блоков. Обращение к элементам матрицы производится при помощи индексов. Для удаления строк или столбцов используется пустой массив.

Символьные операции позволяют находить точные значения выражений или значения со сколь угодно большой точностью. Для преобразования значения числовой переменной (*A*) в символьную (*B*) служит функция *sym*().

>>B=sym(A)

При переходе от числовых к символьным выражениям по умолчанию используется запись чисел в виде рациональных дробей. Результаты действий над такими выражениями будут записаны в виде символьной переменной, причем все вычисления будут проделаны над рациональными дробями.

Вычисления с рациональными дробями позволяют получить значение символьного выражения с любой степенью точности, т.е. найти любое количество значащих цифр результата. Для вычисления символьных выражений предназначена функция *vpa*():

>> cn=vpa(c,n)

где *c* – символьное выражение; *n* – точность (количество цифр после запятой); *cn* – выходной аргумент, являющийся символьной переменной.

Для перевода символьных переменных в числовые используется функция *double*():

>>cnum=double(cn)

Визуализация символьной функции одной переменной осуществляется при помощи *ezplot*(). В качестве первого параметра указывается символьная функция, в качестве второго – вектор с границами отрезка, на котором требуется построить график функции. Если второй параметр не указан, то функция строится на отрезке [-2π, 2π].

>>ezplot(f,[lb rb])

Операции с полиномами реализуют четыре функции: *collect*(), *expand*(), *horner*() и *factor*(). Вычисление коэффициентов при степенях независимой переменной производится с использованием функции *collect*(), в качестве первого аргумента которой указывается символьная функция, а в качестве второго – переменная, при степенях которой следует найти коэффициенты.

>>pc=collect(p, 'x')

Функция *expand*() представляет полином суммой степеней без приведения подобных слагаемых. Аргументом функции может быть не только полином, но и символьное выражение, содержащее тригонометрические, экспоненциальную и логарифмическую функции.

>> pe=expand(p)

Символьные полиномы разлагаются на множители функцией *factor*(), если получающиеся множители имеют рациональные коэффициенты.

>>pf=factor(p)

Также функция *factor*() позволяет представить число в виде произведения простых чисел.

Упрощение выражений общего вида производится при помощи функции *simplify*(). Она реализует мощный алгоритм упрощения выражений, содержащих как тригонометрические, экспоненциальную и логарифмическую функции, так и специальные. Кроме того, *simplify*() способна преобразовывать выражения, содержащие символьное возведение в степень, суммирование и интегрирование.

Функция *subs*() позволяет произвести подстановку одного выражения в другое. В общем виде *subs*() вызывается с тремя входными аргументами: именем символьной функции, переменной, подлежащей замене, и выражением, которое следует подставить вместо переменной. Функция *subs*(), в частности, облегчает ввод громоздких символьных выражений, имеющих определенную структуру.

>>f=subs(f,'x',expression)

Подстановка вместо переменной ее числового значения приводит к вычислению символьной функции от значения аргумента. Число можно заменить его символьным представлением, а затем найти значение функции с произвольной точностью при помощи *vpa*().

Вычисление определителя в символьном виде производится с использованием функции *det*():

>>d=det(A)

Функция *inv*() предназначена для символьного обращения матриц:

>>a1=inv(A)

Разложение математических функций в ряд Тейлора позволяет проделать функция taylor:

>>taylor(f,x,m)

где *f* – функция; *x* – символьная переменная, по которой следует производить разложение в том случае, когда символьная функция определена от нескольких переменных; *m* – точка, в окрестности которой проводится разложение.

Функция *limit*() находит предел функции в некоторой точке, включая и плюс или минус бесконечность. Первым входным аргументом (*f*) функции является символьное выражение; вторым (*x*) – символьная переменная, а третьим (*m*) – точка, в которой вычисляется значение предела.

>>limit(f,x,m)

В качестве четвертого параметра можно указать значение '*right'* или '*left'*, и тогда можно вычислить значение одностороннего предела справа или слева, соответственно.

Производную можно вычислить с помощью функции *diff*():

>>diff(f,x,k)

где *f* – символьная запись функции; *x* – символьная переменная, по которой производится дифференцирование; *k* – порядок производной.

Для нахождения неопределенных и определенных интегралов символьной функции используется функция *int*():

>>int(f,x,a,b)

где *f* – символьная функция; *x* – символьная переменная интегрирования; *a* и *b* – необязательные параметры, которые задаются только для вычисления определенных интегралов, и задают нижний и верхний пределы интегрирования.

Для решения уравнений и систем используется функция *solve*():

>>solve(f,x)

>>solve(f1,f2,x1,x2)

где *f, f1, f2* – левая часть уравнения или системы, равная нулю; *x, x1, x2* – неизвестные.

Задание 8

Вариант 1

1. Задать символьные переменные *a, b, c*.
2. Задать функцию $f = a^2 + \sin(b) + (1/2)\tan(c^8)$.
3. Придать выражению естественный вид с помощью функции *pretty*().
4. Задать символьную функцию $g = \dfrac{x+y}{(x-y)^2}$.

5. Объявить две вещественные переменные *n* и *m*, образовать комплексное число, считая, что *n* является действительной частью, а *m* – мнимой, и найти сопряженное к нему при помощи функции *conj*().

6. Построить график функции $y = \frac{\sin(x) + \cos(x)}{x}$ на отрезке [-4π; 4π].

7. Упростить выражение $h = (x-a)^2 + 5(x+a) + \sin^2(a)$ с помощью функций *collect*(), *expand*(), и *simplify*().

8. Разложить число 154 на простые множители.

9.Задать символьное выражение $l = \tan(x) + \frac{x^2}{\cos(x)} - \sin(x)$. Подставить $'\pi/4'$. Вычислить выражение с помощью функции *vpa*(). Перевести результат в числовой формат.

10. Задать символьную матрицу $A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$. Найти ее определитель и обратную матрицу.

11. Разложить функцию $p = e^x$ в ряд Тейлора в точке 1. Вычислить коэффициенты с точностью 3 знака после запятой.

12. Вычислить предел функции $q = \frac{5x^2 + 4x - 2}{x^2 - 3x + 1}$ в точке $x = 0$ и при $x \to \infty$.

13. Вычислить производную и неопределенный интеграл функции $\frac{\sin(x)}{\sin(x) + \cos(x)}$.

14. Решить уравнение $(x^2 + 10x + 16)^2 + (x^2 + 11x + 24)^2 = 0$.

Вариант 2

1. Задать символьные переменные *a, b, c*.

2. Задать функцию $f = b^4 + \tan(c) + a\exp(b)$.

3. Придать выражению естественный вид с помощью функции *pretty*().

4. Задать символьную функцию $g = \frac{x - y}{(x + y)^2}$.

5. Объявить две вещественные переменные *n* и *m*, образовать комплексное число, считая, что *n* является действительной частью, а *m* – мнимой, и найти сопряженное к нему при помощи функции *conj*().

6. Построить график функции $y = \frac{\sin(x) - \cos(x)}{x^2}$ на отрезке [-4π; 4π].

7. Упростить выражение $h = (x+a)^2 + 12(x+a) + \cos^2(a)$ с помощью функций *collect*(), *expand*() и *simplify*().

8. Разложить число 256 на простые множители.

9.Задать символьное выражение $l = \cos(x) + \frac{x}{\tan(x)} - x$. Подставить $'\pi/4'$. Вычислить выражение с помощью функции *vpa*(). Перевести результат в числовой формат.

10. Задать символьную матрицу $A = \begin{pmatrix} a & f & i \\ d & b & g \\ h & e & c \end{pmatrix}$. Найти ее определитель и обратную матрицу.

11. Разложить функцию $p = \sin(x)$ в ряд Тейлора в точке 1. Вычислить коэффициенты с точностью 3 знака после запятой.

12. Вычислить предел функции $q = \frac{7x^2 + x - 2}{7x^2 - 18x - 1}$ в точке $x = 0$ и при $x \to \infty$.

13. Вычислить производную и неопределенный интеграл функции $\frac{\cos(x)}{\sin(x) - \cos(x)}$.

14. Решить уравнение $(x^2 + 6x - 72)^2 + (x^2 + 15x + 36)^2 = 0$.

Вариант 3

1. Задать символьные переменные *a*, *b*, *c*.

2. Задать функцию $f = a/b + \sin(c) + 2^{a+b}$.

3. Придать выражению естественный вид с помощью функции *pretty*().

4. Задать символьную функцию $g = \frac{3x + 2y}{(x - 4y)^2}$.

5. Объявить две вещественные переменные *n* и *m*, образовать комплексное число, считая, что *n* является действительной частью, а *m* – мнимой, и найти сопряженное к нему при помощи функции *conj*().

6. Построить график функции $y = \frac{1 + \cos(x)}{x}$ на отрезке [-4π; 4π].

7. Упростить выражение $h = (x - 2a)^2 + 2(x - a) + \tan^2(a)$ с помощью функций *collect*(), *expand*() и *simplify*().

8. Разложить число 228 на простые множители.

9.Задать символьное выражение $l = \sin(x) + \frac{x^3}{\cos(x)} - \cot(x)$. Подставить $'\pi/4'$. Вычислить выражение с помощью функции *vpa*(). Перевести результат в числовой формат.

10. Задать символьную матрицу $A = \begin{pmatrix} a & d & g \\ b & e & h \\ c & f & i \end{pmatrix}$. Найти ее определитель и обратную матрицу.

11. Разложить функцию $p = \cos(x)$ в ряд Тейлора в точке 1. Вычислить коэффициенты с точностью 3 знака после запятой.

12. Вычислить предел функции $q = \frac{4x^2 + 14x + 12}{-x^2 + 13x - 6}$ в точке $x = 0$ и при $x \to \infty$.

13. Вычислить производную и неопределенный интеграл функции $\frac{\sin(x)}{\sin(x) - \cos(x)}$.

14. Решить уравнение $(x^2 + 27x - 57)^2 - (x^2 - 3x + 1)^2 = 0$.

Вариант 4

1. Задать символьные переменные *a, b, c*.

2. Задать функцию $f = \frac{a^2 + b^3}{\sin(c^5)}$.

3. Придать выражению естественный вид с помощью функции *pretty*().

4. Задать символьную функцию $g = \frac{2x - y}{(x + 2y)^2}$.

5. Объявить две вещественные переменные *n* и *m*, образовать комплексное число, считая, что *n* является действительной частью, а *m* – мнимой, и найти сопряженное к нему при помощи функции *conj*().

6. Построить график функции $y = \frac{\sin(x) + 1}{x}$ на отрезке [-4π; 4π].

7. Упростить выражение $h = 2(x - 2a) + 5(x + a)^2 + \tan(a)$ с помощью функций *collect*(), *expand*() и *simplify*().

8. Разложить число 168 на простые множители.

9.Задать символьное выражение $l = \frac{\sin^2(x) - \tan(x)}{\cos(x)} - x$. Подставить $'\pi/4'$. Вычислить выражение с помощью функции *vpa*(). Перевести результат в числовой формат.

10. Задать символьную матрицу $A = \begin{pmatrix} i & h & f \\ g & e & c \\ d & b & a \end{pmatrix}$. Найти ее определитель и обратную матрицу.

11. Разложить функцию $p = \tan(x)$ в ряд Тейлора в точке 1. Вычислить коэффициенты с точностью 3 знака после запятой.

12. Вычислить предел функции $q = \frac{3x^2 - 7x + 5}{6x^2 + x - 1}$ в точке $x = 0$ и при $x \to \infty$.

13. Вычислить производную и неопределенный интеграл функции $\frac{\cos(x)}{\sin(x) + \cos(x)}$.

14. Решить уравнение $(x^2 - 12x + 20)^2 - (x^2 + 2x - 12)^2 = 0$.

Список литературы

1. Говорухин В., Цибулин В. Компьютер в математическом исследовании. Учебный курс. – СПб.: Питер, 2001. – 624 с.
2. Кетков Ю.Л., Кетков А.Ю., Шульц М.М. MATLAB 6.x.: программирование численных методов. – СПб.: БХВ-Петербург, 2004. – 672 с.
3. Мичасова О.В. Введение в MatLab: лабораторный практикум: Учебно-методическое пособие. - Нижний Новгород: Издательство Нижегородского госуниверситета, 2008. - 58 с.
4. MatLab Documentation // http://www.mathworks.com/help/matlab/

Денис Владимирович **Капитанов,**
Ольга Владимировна **Капитанова**

Введение в MatLab

Лабораторный практикум

Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский Нижегородский государственный
университет им. Н.И. Лобачевского»
603950, Нижний Новгород, пр. Гагарина, 23