Некоммерческое акционерное общество

АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

Кафедра Автоматизации и управления

ПРОГРАММНЫЕ СРЕДСТВА СИСТЕМ АВТОМАТИЗАЦИИ

Методические указания к выполнению лабораторных работ
для студентов специальности 5В070200 – Автоматизация и управление

Алматы 2019

СОСТАВИТЕЛИ: Ибраева Л.К., Абжанова Л.К., Ильясов А.З. Программные средства систем автоматизации. Методические указания к выполнению лабораторных работ для специальности 5В070200 – Автоматизация и управление. – Алматы: АУЭС, 2019 – 58 с.

Методические указания предназначены для освоения инструментариев системы программирования и моделирования MATLAB и содержат описания к 9 лабораторным работам.

Ил.-17, табл.- 20, библиогр.- 10 наим.

Рецензент:

Печатается по плану издания некоммерческого акционерного общества «Алматинский университет энергетики и связи» на 2019 г.

Содержание

1 Лабораторная работа №1. Знакомство с системой. Обычные вычисления

Цель работы: изучить в среде MATLAB представление данных, простые числовые выражения и математические формулы.

1.1 Задание на лабораторную работу

В процессе выполнения лабораторной работы студент должен:

- изучить назначение и компоненты пользовательского интерфейса системы MATLAB;
- изучить объекты MATLAB;
- изучить правила представления данных;
- подготовить отчет по работе.

1.2 Интерфейс пользователя

Система MATLAB имеет многооконный пользовательский интерфейс, в котором размещен ряд средств прямого доступа к различным компонентам системы (рисунок 1.1).

Основную часть окна системы занимает командное окно – *Command Window*, в котором расположена строка ввода, начинающаяся специальными символами «>>» (символ вводится автоматически). На этой строке записываются команды для выполнения системой.

В левой части окна расположено окно истории команд – *Command History*, в котором отображаются вводимые пользователем команды. При необходимости эти команды можно снова выполнить, сделав двойной щелчок мыши по нужной команде в окне истории команд.

Окно рабочей области MATLAB - *Workspace* показано на рисунок 1.2 и содержит список переменных (именованные массивы), накопленных в памяти в процессе работы, расширение списка переменных при обращении к функциям, выполнении М-файлов и загрузке сохраненных переменных.

Как и все окна рабочего стола системы MATLAB, окно рабочей области, сопровождается контекстным меню, которое включает следующие опции:

- *Open Selection*… (Открыть выделенное…);
- *GraphSelection* (Построить график);
- *SelectAll* (Выделить все);
- *Import Data* (Импорт данных);
- *Save Selection A*s… (Сохранить выделенное как…);
- *SaveWorkspaceAs*… (Сохранить рабочую область как…);
- *Delete Selection* (Удалить выделенное);
- *Delete Workspace* (Удалить рабочую область).

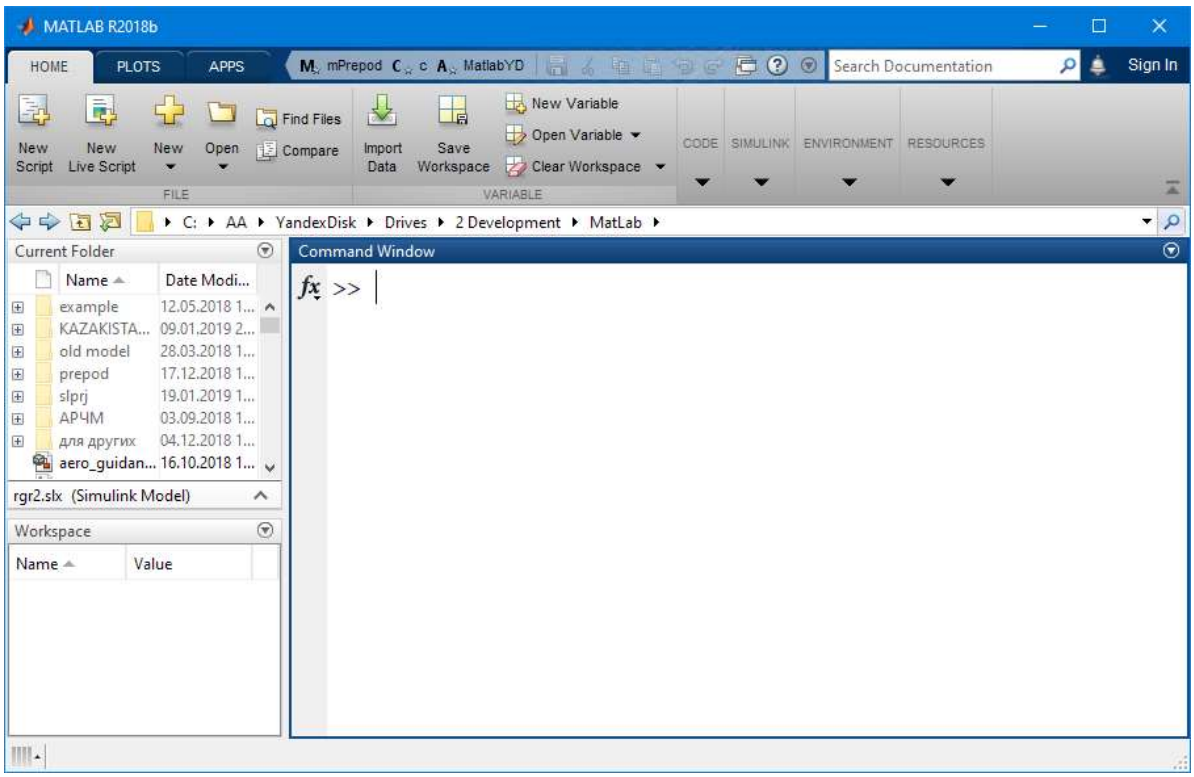


Рисунок 1.1 - Окно MATLAB, которое содержит *Command Window*, *Workspace* и текущее окно *Current Folder* рабочей области

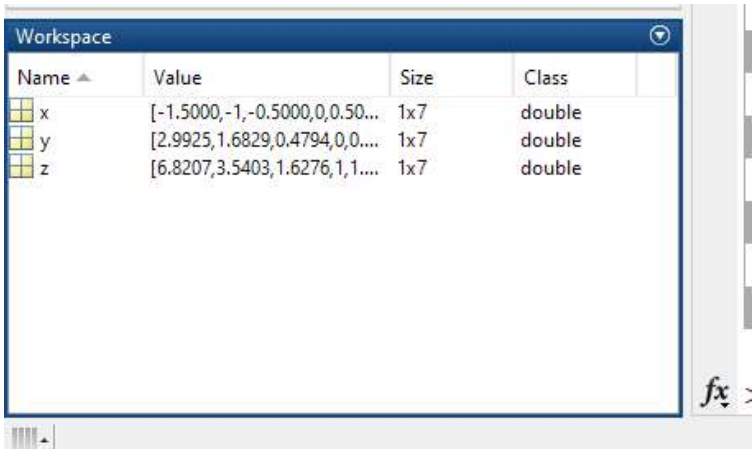


Рисунок 1.2 – Окно рабочей области *Workspace*

MATLAB содержит мощную подсистему справки *MATLAB Help*:

- меню *Help – Help Window* или соответствующая кнопка на панели инструментов, а также команда *helpwin*;

- для вывода окна справки по отдельным разделам используется *helpwin topic*;

- *helpops* используется для вывода списка операторов MATLAB.

Другая полезная команда MATLAB – команда *look for*. Ее можно использовать для поиска команды по ключевому слову.

Управление рабочей областью. Содержимое рабочей области сохраняется между выполнением отдельных команд. Таким образом, результаты одной

задачи могут повлиять на следующую. Чтобы избежать такой возможности, рекомендуется в начале каждого нового независимого вычисления давать четкую команду на очистку рабочей области.

Все переменные, созданные в сеансе MATLAB, хранятся в рабочей области MATLAB. Чтобы при выходе из MATLABэто рабочее пространство не было уничтожено, используйте *File – Save Workspace As* ... - всех переменные в рабочей области сохранятся в текущем каталоге в файле с расширением*.mat*. Эти*.mat* файлы являются двоичными файлами, что означает, что вы не можете редактировать или читать их. Вы можете перезагрузить файл в любое время позже, используя команду: *File-Import Data*. Этот метод сохранения сохраняет все переменные в рабочей области. Можно выбрать любое количество переменных.

Команда *dir* дает вам список всех файлов, которые находятся в текущем каталоге.

1.3 Представление данных в MATLAB

Центральным понятием всех математических систем является *математическое выражение*. Математические выражения строятся на основе чисел, констант, переменных, операторов, функций и разных спецзнаков.

Число – простейший объект языка MATLAB, представляющий количественные данные. Числа используются в общепринятом представлении о них. Представление чисел в MATLAB соответствует правилам языков программирования.

Константа – это предварительно определенное числовое или символьное значение, представленное уникальным именем. В MATLAB существуют некоторые стандартные константы (таблица 1.1).

Таблица 1.1 – Предопределенные постоянные величины

Имя переменной	Пояснение к переменной
pi	Число π=3,14159…
i,j	Мнимая единица, $\sqrt{-1}$.
inf	Бесконечность, ∞
NaN	Не число (not a number)

Символьной константой считается любая последовательность символов, заключенных в апострофы, например, '*Hello*!'.

Переменная – это объект с именем, который хранит некоторые данные. В зависимости от этих данных переменные могут быть числовыми или символьными, векторными или матричными.

Имя переменной должно начинаться с буквы, может содержать буквы, цифры и символ подчеркивания. Имя не должно совпадать с именами других переменных, функций и процедур системы. Прописные и строчные буквы в MATLAB различаются.

Для уничтожения определения переменной используется специальная команда:

-*clear*– уничтожает все переменные;

- *clear x*– уничтожает переменную *x*;

- *clear x,y*– уничтожает переменные *x* и *y*.

Оператор – это специальное обозначение для определенной операции над данными – *операндами*. Например, арифметическими операторами являются знаки суммы (+), вычитания (–), умножения (*), деления (/), возведения в степень (^). Операторы используются совместно с операндами. Например, в выражении 2+3 знак «+» является оператором сложения, а числа 2 и 3 – операндами.

Для задания переменной некоторого значения используется оператор присваивания:

Имя_переменной = Значение;

Для вывода значения переменной нужно в командной строке ввести ее имя и нажать клавишу *Enter*.Переменная должна иметь значение перед тем, как она будет использована. После ввода переменной она хранится в окне *Workspace.* Двойным щелчком в этом окне на имя переменой, можно просмотреть информацию о ней (размерность, тип).

Если не присвоить выражение переменной, то ответ передается в переменную с именем *ans* (answer), которую можно в дальнейшем использовать. Знак « ; » подавляет вывод результата на экран(не обязателен для ввода).

Пример ввода команды и вывод результата с присвоением переменной и без присваивания (>> - приглашение системы, после этих символов набирается команда):

```
>> x=25*625                    >> 25*625
x =                             ans =
     15625                           15625
```

Если необходимо разместить несколько выражений на одной строке, можно использовать знаки (,), (;).

Символ (%) используется для комментариев.

Команда *who* выводит все имена используемых переменных.

Команда *whos* выводит более полную информацию о переменных.

При вычислении значений арифметических выражений нужно набрать в командной строке это выражение и нажать клавишу *Enter*. Перед тем как вычислять значение математического выражения, необходимо определить значение каждой входящей в него переменной. Вычисляемое выражение может содержать любое количество переменных, операторов и функций.

При арифметических вычислениях в MATLAB соблюдается порядок, принятый в математике. Для изменения порядка действий используются круглые скобки.

Отличительной особенностью MATLAB является то, что при правильном вводе любого элемента, этот элемент автоматически сохраняется в памяти до следующего изменения в поле *Workspace*.

Команды форматирования. По умолчанию, MATLAB представляет числа с четырьмя десятичными знаками. Это так называемый *short* формат. Если необходима большая точность, необходимо использовать команду *format*. Формат *long* отражает 16 знаков после запятой. Формат *format bank* округляет число до двух десятичных знаков.

Большие числа отображают, используя экспоненциальное представление. Формат *shorte* отображает число в экспоненциальной форме с четырьмя десятичными знаками и экспонентой. Формат *longe* отображает число в экспоненциальной форме с 16 десятичными знаками и экспонентой.

В примере 1.1 приведена работа с форматами.

Пример 1.1.

```
>> format shorte
>> 5/3
ans =
  1.6667e+00

>> format longe
>> 5/3
ans =
   1.666666666666667e+00

>> format short
>> 5/3
ans =
   1.6667

>> format bank
>> 5/3
ans =
        1.67
```

Пример 1.2.

Вычислить значение выражения:

$$z = \frac{x^2 + y}{3 - |sinx|} + 2,$$

при x=25, y=3,6.

Порядок ввода в *CommandWindow*:

```
>>x=25;
>>y=3.6;
>>z=(x^2+y)/(3-abs(sin(x)))+2
```

z =

221.2040

В результате получим *z*=221.2040.

1.4 Порядок выполнения лабораторной работы

1.4.1 Запустите MATLAB. Изучите компоненты пользовательского интерфейса.

1.4.2 Настройте окна с помощью пункта меню *Layout-Default*. Настройка окон MATLAB также может быть выполнена с помощью кнопки в виде «стрелки» (в правом верхнем углу).

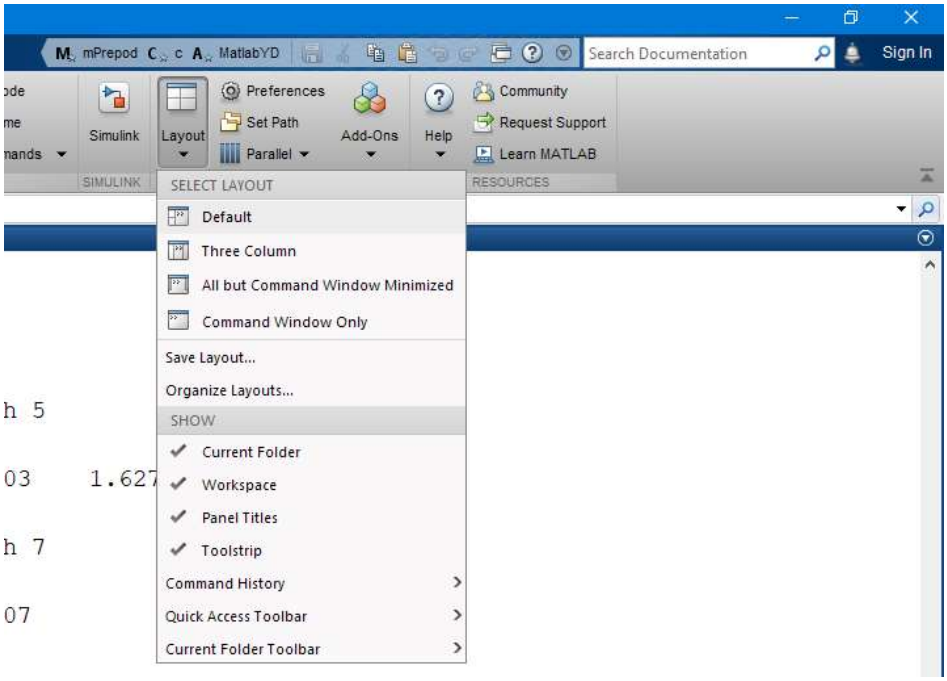


Рисунок 1.5 - Настройка окон MATLAB с помощью Layout

1.4.3 Выполните настройки (шрифт, цвет и т.д.) с помощью кнопки «*Preferences*».

1.4.4 Измените рабочую (текущую) папку с помощью команды:

>>cd 'путь к папке'

(например*>>cd 'c:\Users')* или с помощью окна *Current Folde*r.

1.4.5 Выполните операции с числами:

- введите два числа;
- сложите эти числа;
- получите произведение полученного результата на любое другое число;
- запишите результат в переменную *a*;
- введите несколько арифметических выражений, изменяя порядок выполнения арифметических операций;
- для заданных значений аргументов вычислите значения выражений (по варианту); выполните это же задание, размещая в конце выражения знак «;»;

- измените значения некоторых аргументов и вычислите значение функции, не набирая заново выражение.

1.4.6 Просмотрите список используемых переменных.

1.4.7 Получите полную информацию о переменных.

1.4.8 Выполните действия по управлению переменной: изменить ее значение; удалить из рабочей области и т.д.

1.4.9 Сохраните сессию, закройте MATLAB.

1.4.10 Загрузите MATLAB, загрузите сохраненную сессию.

1.4.11 Вычислите выражение, а также значение функции *f(x)* на отрезке *[a,b]* с шагом *h* (по варианту – таблица 1.2).

Таблица 1.2 - Варианты заданий

№	Выражение	Функция	Интервал [a,b] и шаг h
1.	$\frac{\left(12\frac{1}{6}-6\frac{1}{27}-5{,}25\right)13{,}5+0{,}111}{0{,}02}$	$f(x)=\frac{x^2}{1+0.25\sqrt{x}}$	[1,1..3,1], h =0.2
2.	$\frac{\left(1\frac{1}{12}+2\frac{5}{32}+\frac{1}{24}\right):9{,}6+2{,}13}{0{,}00004}$	$f(x)=\frac{x^2+3}{2x+24}$	[1..5], h =0.5
3.	$\frac{\left(6{,}6-3\frac{3}{14}\right)5\frac{5}{6}}{(21-1.25):2{,}5}$	$f(x)=\frac{x^3}{3x^2+\sqrt{x}}$	[0..3], h =0.1
4.	$\frac{2.625-\frac{2}{3}\cdot 2\frac{5}{14}}{\left(3\frac{1}{12}+4.375\right):19\frac{8}{9}}$	$f(x)=\frac{x^3+3}{x+0.25\sqrt{x}}$	[0.5..5], h =0.2
5.	$\frac{0{,}134+0{,}05}{18\frac{1}{6}-1\frac{11}{14}-\frac{2}{15}\cdot 2\frac{6}{7}}$	$f(x)=\frac{x^2}{x-0.5x+\sqrt{x}}$	[1..6], h =0.3
6.	$\frac{\left(58\frac{4}{15}-56\frac{7}{24}\right):0{,}8+2\frac{1}{9}\cdot 0{,}225}{8{,}75\cdot 0{,}6}$	$f(x)=\frac{x^2+2x}{x^3-10x}$	[0..10], h =0.8
7.	$\frac{\left(\frac{0.216}{0.15}+0.56\right):0{,}5}{\left(7.7:24.75+\frac{2}{15}\right)4.5}$	$f(x)=\frac{4}{x^3-0.25x}$	[1..8], h =0.6
8.	$\frac{1\frac{4}{11}\cdot 0.22:0.3-0.96}{\left(0.2-\frac{3}{40}\right)1{,}6}$	$f(x)=\frac{10x^3+x}{5x+x^2}$	[0..6], h =0.4
9.	$\frac{\left(\frac{3}{5}+0{,}425-0{,}005\right):0{,}12}{30{,}5+\frac{1}{6}+3\frac{1}{3}}$	$f(x)=\frac{x^2}{5x+0.25\sqrt{x}}$	[0..10], h =0.7
10.	$\frac{3\frac{1}{3}+2{,}5}{2{,}5-1\frac{1}{3}}\div\left(\frac{0{,}05}{\frac{1}{7}-0{,}125}+5{,}7\right)$	$f(x)=\frac{10}{x+x^3}$	[1..8], h =0.5

№	Выражение	Функция	Интервал [a,b] и шаг h
11.	$\frac{0{,}725+0{,}42}{0{,}128-6{,}25-(0{,}0345/0{,}12)}\cdot 0{,}25$	$f(x)=\frac{x^3}{7+5\sqrt{x}}$	[0..10], h =0.5
12.	$\frac{\left(4{,}5\cdot 1\frac{2}{3}-6{,}75\right)\cdot 0{,}6}{\left(3{,}333\cdot 0{,}3+0{,}222\cdot\frac{4}{9}\right)\cdot 2\frac{2}{3}}$	$f(x)=\frac{x}{5x+0.25}$	[1..9], h =0.4
13.	$\frac{\left(5\frac{4}{45}-4\frac{1}{6}\right):5\frac{8}{15}}{\left(4\frac{2}{3}+0{,}75\right)\cdot 3\frac{9}{13}}$	$f(x)=\frac{x^2}{7+0.5x}$	[0..5], h =0.1
14.	$\frac{\left(3\frac{4}{17}-4\frac{1}{6}\right):5\frac{2}{3}}{\left(2\frac{2}{3}+2{,}5\right)\cdot 3\frac{1}{5}}+\frac{2}{5}$	$f(x)=\frac{x^3+x}{2x+0.5}$	[0..6], h =0.2
15.	$\frac{\left(12\frac{3}{8}+45\frac{1}{24}\right)+5\frac{2}{7}\cdot 0{,}5}{0{,}75\cdot 0{,}6}$	$f(x)=\frac{x^2}{x-0.5x^2}$	[1..10], h =0.3
16.	$\frac{\left(4\frac{4}{7}-10\frac{7}{33}\right):\frac{2}{17}+2\frac{1}{9}\cdot 0{,}225}{0{,}6\frac{3}{8}}$	$f(x)=\frac{10+x}{x^2+4}$	[1.1..2], h =0.05
17.	$\frac{(68{,}023-66{,}028):6\frac{1}{9}+\frac{7}{40}\cdot 4{,}5}{0{,}042+0{,}086}$	$f(x)=\frac{x}{x^3+0.47\sqrt{x}}$	[0.5..10], h =0.7
18.	$\frac{(1{,}88+2{,}127)\cdot 0{,}01875}{0{,}625-\frac{13}{18}/3{,}13}+8{,}29$	$f(x)=\frac{x^2}{2x}$	[-3..3], h =0.2
19.	$\frac{\frac{3}{0{,}4}-0{,}009:(0{,}15:2{,}5)}{0{,}32\cdot 6+0{,}033-(5{,}3-3{,}88)}$	$f(x)=\frac{x^2+x}{0.25\sqrt{x}}$	[-5..5], h =0.5
20.	$\frac{(34{,}06-33{,}81)\cdot 4}{6{,}84/(28{,}57-25{,}15)}+1{,}33/\frac{4}{21}$	$f(x)=\frac{10x^2}{x+5}$	[-5..5], h =0.4

1.5 Требования к отчету

Отчет по работе должен содержать:
- вариант задания;
- скриншоты работы с пользовательским интерфейсом с комментариями;
- скриншоты выполнения вычислений;

1.6 Контрольные вопросы

1.6.1 Объясните назначение системы MATLAB.
1.6.2 Что такое окно Current Folder?
1.6.3 Для чего используется Command History?
1.6.4 Назначение окна Workspace?

1.6.5 Как можно отразить на экране отдельные элементы пользовательского интерфейса?

1.6.6 Как можно ввести в командное окно переменную?

1.6.7 Перечислите правила именования переменных.

1.6.8 Назовите зарезервированные переменные MATLAB.

1.6.9 Объясните правила выполнения арифметических операций.

1.6.10 Как можно подавить вывод на экран результата.

2 Лабораторная работа №2 Работа с матрицами и векторами

Цель работы: изучить работу с матрицами, а также использование математических функций в инженерных расчетах.

2.1 Задание на лабораторную работу

В процессе выполнения лабораторной работы студент должен:

- изучить математические функции MATLAB;
- изучить различные операции с векторами, матрицами, комплексными переменными;
- научиться применять изученные функции к решению инженерных задач.

2.2 Операции с массивами. Математические функции MATLAB

Структура хранения всех данных в MATLAB представляет собой *матрицу*. Матричная переменная в MATLAB может иметь любое количество строк и столбцов. *Скалярная* переменная тоже хранится как матрица с одной строкой и одним столбцом. *Матричная* переменная может быть любой переменной, то есть может быть *скаляром*, *вектором* или *матрицей*.

Существует два основных типа векторов в MATLAB: *вектор-строка* и *вектор-столбец*. Вектор-строка хранит свои числа «горизонтально», а вектор-столбец – «вертикально». Эти массивы обычно заключаются в квадратные скобки.

Создание векторов и матриц. Вектор – одномерный массив чисел. Для отделения элементов. Вектор-строка создается посредством заключения множества элементов в квадратные скобки с использованием пробела или запятой между элементами. Все индексы вектора нумеруются, начиная с 1.

Матрица – двумерный массив чисел. В MATLAB матрица создается посредством ввода каждой строки как последовательности чисел, отделенных друг от друга запятой или пробелом; конец строки обозначается точкой с запятой.

MATLAB можно использовать как простой «карманный калькулятор» для матриц: можно быстро и легко умножать, складывать или вычитать их. Это очень удобный инструмент для проверки матричных вычислений.

MATLAB дает возможность создавать некоторые матрицы автоматически, без необходимости печатания каждого элемента:

- *zeros(m,n)* – создает матрицу размерности *mxn* с нулевыми элементами;
- *ones(m,n*) – создает матрицу размерности *mxn* с элементами, равными единице;
- *eye(m,n)* – создает матрицу размерности *mxn,* элементы главной диагонали равны единице;
- *rand(m,n)* – создает матрицу размерности *mxn*, элементы которой являются случайными чслами между 0 и 1.

Математические функции. MATLAB предлагает множество предопределенных математических функций для инженерных вычислений. С помощью *help elfun* и *help specfun* сможно просмотреть список встроенных элементарных и специальных функций MATLAB. Ко всем элементарным функциям можно получить доступ по их именам (таблица 2.1).

Таблица 2.1 – Список наиболее часто используемых функций, где переменные могут быть числами, векторами и матрицами

cos(x)	Косинус	abs(x)	абсолютная величина
sin(x)	Синус	sign(x)	сигнум функция
tan(x)	Тангенс	max(x)	максимальное значение
acos(x)	арк-косинус	min(x)	минимальное значение
asin(x)	арк-синус	ceil(x)	округление на повышение
atan(x)	арк-тангенс	floor(x)	округление на понижение
exp(x)	экспонента в степени x	round(x)	округлене к ближайшему целому
sqrt(x)	корень квадратный	rem(x)	остаток от деления
log(x)	натуральный логарифм	angle(x)	фазовый угол
log10(x)	десятичный логарифм	conj(x)	комплексно-сопряженное число

Арифметические операции с матрицами. В MATLAB все арифметические операции: +, −, * и ˆ могут быть применены к матрицам. Также существуют арифметические операции, которые позволяют проводить вычисления *поэлементно*. Список таких операций приведен в таблице 2.2.

Таблица 2.2 – Поэлементные операции с массивами

.*	По-элементное умножение
./	По-элементное деление
.^	По-элементное возведение в степень

Эти операции выполняются следующим образом. Если A и B – две матрицы одинакового размера с элементами $A = [a_{ij}]$ и $B = [b_{ij}]$, тогда команда >> *C=A.*B* формирует матрицу C of того же размера с элементами $c_{ij} = a_{ij} * b_{ij}$.

Чтобы возвести скалярное число в степень, используется знак ^, например, *10^2*. Если же эту операцию надо применить к каждому элементу матрицы, надо использовать *«.^»*.

Действия с матрицами. Основные операторы с матрицами выполняются в MATLAB также, как матричные вычисления в математике (сложение, вычитание, транспонирование, умножение матрицы на число, произведение двух матриц, определение степени матрицы) с учетом правил математики на размерности матриц, к которым эти операции применяются.

Другие операторы:

- инверсия матрицы *inv(A)* - вычисляет матрицу, обратную к матрице *A*;
- *det (A)* – вычисляет определитель матрицы;
- особенностью MATLAB является наличие двух неизвестных в математике операций деления матриц: деление слева направо (‘/’) справа налево (‘\’).

Операция *B/A* эквивалентна команде *B*inv(A); A\B* – команде *inv(A)*B*. То есть, для матрицы знак «слэш» означает инверсию: если это обратный слэш - *«\»* , тогда инверсия относится к первой матрице, обычный слэш «/» - инверсия относится к второй матрице.

В инженерных приложениях возникает необходимость решения матрично-векторных уравнений. Эти задачи легко решаются, так как изначально система MATLAB специально была ориентирована на решение матричных уравнений.

Например, если задано матричное уравнение *Ax=b* (система линейных алгебраических уравнений), где *A* – матрица системы, *b* – вектор правых частей системы, *x* – вектор неизвестных переменных, то для решения этой системы достаточно в командном окне MATLAB ввести команду *x=inv(A)*b*. Компоненты матрицы *A* и вектора правых частей *b*, конечно, должны быть введены в командное окно MATLAB.

Операции с векторами и матрицами как с массивами данных (обработка результатов измерений). Предположим мы имеем некоторую зависимость *y(x)*, которая задана своими числовыми значениями. Ее можно представить как матрицу с двумя строками – значениями *x* и значениями *y*, назовем эту матрицу *xydata*.

Основные инструменты обработки данных:

- *size(xydata)* – определяет число строк и столбцов в матрице *xydata*; обращение: *[n,p]= size(xydata),* где *n*- число строк, *p* – число столбцов;
- *max(v)* – возвращает значение максимального элемента вектора *v*;
- *min(v)* - возвращает значение максимального элемента вектора *v*;
- *sort(v)* – формирует вектор, элементы которого отсортированы в порядке возрастания их значений;
- *sum(v)* – вычисляет сумму элементов вектора *v*;
- *prod (v)* - вычисляет произведение всех элементов вектора *v*;

- *diff (v)* - создает вектор, элементы которого являются разностью между соседними элементами вектора *v*; размер полученного вектора меньше, чем размер вектора *v*;

- *cumsum(v)* - создает вектор, каждый элемент которого является суммой всех предыдущих элементов вектора *v;*

- *cumprod(v)* - создает вектор, каждый элемент которого является произведением всех предыдущих элементов вектора *v;*

- *mean(v)* – определяет среднее значение элементов вектора *v;*

- *std(v)* – определяет стандартное отклонение элементов вектора *v.*

Эти же самые функции *sum*, *max*, *min, sort, prod, diff, cumsum*, *cummprod, mean, std* могут быть применены к матрицам. При этом эти операции выполняются не по отношению к строкам матрицы, а по отношению к каждому из столбцов матрицы (за исключением функции *size*).

Функции комплексного аргумента. Почти все элементарные функции можно применить к комплексным значениям аргументов и получить комплексное значение результата. Благодаря этой возможности функция *sqrt* вычисляет квадратный корень отрицательного числа, функция *abs* – модуль комплексного числа.

Также в MATLAB есть несколько дополнительных функций, которые применяются только к комплексным аргументам:

- *real(z)* – определяет действительную часть комплексного числа *z*;

- *imag(y)* - определяет мнимую часть комплексного числа *z*;

- *angle(z)* - вычисляет значение аргумента комплексного числа (в радианах от – π до + π);

- *conj(z)* - определяет число, комплексно сопряженное к *z*.

Эти функции дают возможность выполнять вычисления с действительными числами, результат которых является комплексным числом (например, найти комплексные корни квадратного уравнения).

Табулирование функции. Табулирование функции – вычисление всех ее значений при каждом значении аргумента в указанном диапазоне. Для задания диапазона чисел необходимо написать имя переменной, поставить знак присваивания, а затем через двоеточие – начальное значение, шаг и конечное значение:

Имя_переменной=Начальное_значение:Шаг:Конечное_значение;

Если шаг равен 1, то его можно не указывать, а задать только начальное и конечное значения. После ввода диапазона значений аргумента функции задается сама функция.

Так как аргумент имеет несколько значений, то выполнение операций умножения, деления и возведения в степень должно выполняться поэлементно над каждым из значений. Для этого используются операции «с точкой»: «**.***»,« **./** »,« **.^**» соответственно.

Пример 2.1. Вычислить значения функций:

$y = 2 \cdot x \cdot sin(x);\ z = 3x^2 + cos(x)$

для $x \in [-1{,}5; 1{,}5]$ с шагом 0,5.

Порядок ввода в *Command Window*:
>>x=-1.5:0.5:1.5
*>>y=2*x.*sin(x)*
*>>z=3*x.^2+cos(x)*

2.3 Порядок выполнения лабораторной работы

2.3.1 Запустите MATLAB.

2.3.2 Выполните операции с векторами:

- введите любой вектор *v*. Напечатайте *n= size(v)* - это пример функции MATLAB. Объясните, что определяет эта функция;
- введите вектор-столбец с тремя строками и одним столбцом;
- введите два вектора с тремя элементами; примените к этим векторам следующие операции: *+, - ; x′ ; *;*
- транспонируйте вектор-столбец в вектор-строку и наоборот - операция «'»;
- сформируйте вектор-столбец, элементы которого первые 10 чисел, используя команду (1:10)'.

2.3.3 В MATLAB активно используется знак «:»:

- введите h=10:2:20 (без квадратных скобок!). просмотрите результат;
- создайте этот же вектор другим способом;
- создайте вектор с использованием знака «:», но с отрицательным приращением;
- создайте вектор с использованием знака «:», но без приращения.

2.3.4 Выполните операции с матрицами:

- введите две 5x5 матрицы *A* и *B* в командное окно;
- найдите матрицу, обратную к матрице *A*;
- найдите произведение полученной матрицы и матрицы *B*;
- найдите произведение матрицы *A* и инверсии матрицы *B*, назовите переменную результата *C*;
- найдите инверсию матрицы *C*, запишите результат в переменную *ans*, примените инверсию к этой переменной, сравните результаты двух последних действий;
- вычислите вторую степень матрицы *A*, найдите произведение матрицы A на самое себя, сравните результаты двух последних операций;
- вычислите отрицательную вторую степень матрицы *A*. Найдите произведение обратной матрицы на самое себя. Сравните результат;.
- вычислите *sin(A*); объясните результат;
- найдите произведение каждого элемента матрицы *A* на каждый элемент матрицы *B*;
- создайте матрицу A размерности 2x2 и матрицу B размерности 2x3;
- напечатайте *C=[A B] и D=[A,B]; type E=[A B;B A]*. Объясните результат;
- напечатайте *E=[A;B]*. Почему эта операция не может быть выполнена?

- перепишите первый столбец матрицы *A* в переменную *A2: A2 =A(1:2,1)*

- эту операцию можно выполнить командой *A2 = A(:,1).* Выполните команду *A(2,:).* Поясните результаты.

2.3.5 Введите две квадратные матрицы в окно MATLAB. Вычислите:

- произведение *A*B*;

- произведение *B*A*;

- сумму *A+B*;

- произведение *3*A*;

- введите *A=2*. Что изменится?

2.3.6 Создайте матрицу размерности 5x5, элементы которой – случайные числа.

2.3.7 Создайте диагональную матицу размерности 4x4.

2.3.8 Создайте матрицу размерности 10x10 элементы которой – случайные числа между 0 и 10. Все элементы в первой строке и первом столбце замените на 1.

2.3.9 Создайте матрицу *A*, транспонируйте ее в матрицу *B*.

2.3.10 Приведите примеры использования функций *zeros (m,n), ones(m,n), eye(m,n) and rand(m,n).*

2.3.11Решение систем линейных алгебраических уравнений:

- сформируйте систему линейных алгебраических уравнений;

- введите матрицу системы и вектор правых частей в командное окно;

- проверьте, имеет ли система решение (детерминант должен быть отличен от нуля);

- решите систему тремя способами: используя операцию деления вектора на матрицу; используя обратную матрицу A^{-1}; используя процедуру *inv* ; сравните результаты.

2.3.12 Необходимо вычислить значения функции *y = a*exp(-k*x)*sin(x)* для значений аргумента *x* от *0* до *10* с шагом *1* (*a* и *k* выбираются студентов).

2.3.13 Введите вектор *v* в командное окно. Выполните следующие операции:

- найдите размер вектора;

- определите максимальный и минимальный элементы вектора;

- найдите сумму и произведение элементов вектора.

2.3.14 Введите матрицу 7x7 в командное окно. Примените операции предыдущего пункта к этой матрице.

2.3.15 Введите в командное окно комплексные числа *u* и *z;* выполните следующие действия:

- используя функцию «*disp*» отразите на экране сумму комплексных чисел *u* и *z*;

- отразите действительную, мнимую части, а также аргумент этих комплексных чисел; the real, imaginary parts and arguments of these complex numbers;

- сформируйте два квадратных уравнения с положительным и отрицательным дискриминантом; решите эти уравнения в командном окне;

- найдите комплексно-сопряженные числа к числам *u* и *z*.

2.4 Отчет по лабораторной работе

Отчет по лабораторной работе должен содержать скриншоты всех выполненных заданий с комментариями.

2.5 Контрольные вопросы

2.5.1 Как вводятся комплексные переменные в командное окно MATLAB?

2.5.2 Как можно определить минимум, максимум и среднее значение массива данных?

2.5.3 Объясните отличие между обычными арифметическими операциями и операциями с точкой.

2.5.4 Как решается система линейных алгебраических уравнений в MATLAB?

2.5.5 Как находится обратная матрица?

2.5.6 Как можно найти определитель матрицы?

2.5.7 Как применяются элементарные функции к векторам и матрицам?

2.5.8 Как можно определить действительную, мнимую части и аргумент комплексного числа?

2.5.9 Какие функции обработки данных вы знаете? Какие отличия в применении этих функций к векторам и матрицам?

2.5.10 Объясните операции деления матриц (левое и правое деление).

3Лабораторная работа №3ВведениевграфикуMATLAB

Цель работы: изучить правила создания2-хи 3-мернойграфики вMATLAB.

3.1 Задание на лабораторную работу

В процессе выполнения лабораторной работы студент должен:

- научиться создавать двухмерную графику;
- приобрести навыки оформления графиков;
- научиться создавать трехмерную графику и оформлять ее.

3.2 Знакомство с графикойMATLAB

Графика в MATLAB имеет следующие особенности:

- высокий уровень, то есть не требует детального знания графической подсистемы;
- объектно-ориентированная, то есть каждый объект на фигуре имеет свойства, которые могут быть изменены;
- доступ к графике возможен как через просмотр объектов, так и с помощью встроенных функций.

Двумерная графика. Главной функцией, отображающей график функции на экране, является следующая функция:

plot(x_1, y_1, s_1, x_2, y_2,s_2…),

гдеx_1, y_1–массивы значений аргументов и соответствующих функций первой кривой;

x_2, y_2 - массивы значений аргументов и соответствующих функций первой кривой и т.д.;

s_1, s_2, …- содержат три специальных символа, которые определяют тип линии графика, цвет линии и т.д. Эти переменные символьные, указывать их необязательно.

В команде построения графика можно установить для каждого графика следующие дополнительные параметры (таблица 3.1).Параметры можно комбинировать, в этом случае вначале указывается тип линии, а затем – тип маркера.

Таблица 3.1 –Свойства линии

Цвет		**Тип маркера**		**Тип линии**	
y	Желтый	.	Точка	-	Сплошная
m	Розовый	o	Кружок	:	Пунктирная
c	Голубой	x	Крестик	-.	Штрих-пунктирная
r	Красный	+	Плюс	--	Штриховая
g	Зеленый	*	Звездочка		
b	Синий	s	Квадрат		
w	Белый	d	Ромб		
k	Черный	v	Треугольник		

Пример 3.1.

```
>> x = -pi : .01 : pi;
>> y = sin(x);
>> z = cos(x);
>> plot(x, y, 'r-.', x, z, 'm:')
```

Графики в MATLAB всегда размещаются в отдельных окнах, которые называются *figure*.

Если вы сразу же построите другой график в окне фигуры, старый график будет удален из графического окна.

Чтобы отразить сетку координатных линий на графике, после вызова функции *plot* напишите *grid.* Сетка имеет целочисленное приращение. Заголовок графика отображается с помощью процедуры *title ('текст').*

Обозначения на горизонтальной оси – с помощью функции *xlabel (‘x’)* и по вертикальной оси -*ylabel(‘y’)*.

Такая форма графиков полностью отвечает требованиям инженерной графики.

Специальные графики:

- если при построении графика аргументы функции не определены, тогда система использует номер элемента вектора в качестве аргумента;

- часто используется представление в виде столбчатой диаграммы – функция *bar;*

- для построения графика функции с дискретными аргументами используется процедура *stem*;

- другая полезная для инженеров функция – построение гистограммы заданного вектора - *hist (y,x),*

где *y*–вектор гистограммы, которую мы строим;
x–вектор, задающий интервалы первого вектора.

Эта функция подсчитывает количество элементов вектора *y*, значения которых попадают в диапазон, заданный вектором x, и строит гистограмму подсчитанных чисел элементов вектора *y* как функцию, заданную диапазонами векторов *x*;

- процедура *subplot (m,n,p)* позволяет построить в одном графическом окне, но в разных зонах несколько графиков; здесь *m* указывает, на сколько частей делится графическое окно по вертикали, *n* указывает, на сколько частей делится графическое окно по горизонтали; *p* -число подокон, в котором нужно построить график (пример 3.2);

- *gtext (x,y,‘text’)* - размещает текст в поле графика, начало текста помещается в точку с координатами *x* и *y*; (здесь также можно воспользоваться меню);

-*figure* рисунок создает новое графическое окно, не удаляя прежнее (пример 3.3);

- *hold on* - используется для отображения нескольких графиков в одном окне (пример 3.4);

- *hold off*- отключает режим, установленный предыдущей командой.

- *функции в полярной системе координат*. Эти функции строятся аналогично графикам функций в декартовой системе. Для построения такого графика используется команда *polar* (пример 3.5).

Пример3.2.

```
>>x = -pi : .01: pi;
>> y = sin(x);
>> z = cos(x);
>> subplot(211); plot(x, y, ‘c’)
>>subplot(212); plot(x, z, ‘g’)
```

Пример 3.3.

```
>>x = -pi : .01: pi;
>> y = sin(x);
>> z = cos(x);
>> plot(x, y)
>> figure
>> plot(x, z, 'r')
```

Пример 3.4.

```
>>x = -pi : .01: pi;
>> y = sin(x);
>> z = cos(x);
>> plot(x, y)
>> hold on
>>plot (x, z)
```

Пример 3.5.

```
>>phi = 0 : .01: 15;
>> r = phi;
>>polar (phi, r, 'g')
```

Трехмерная графика. Для построения сложных поверхностей, представленных функцией двух переменных *z=f(x,y).*Некоторые из используемых функций приведены в таблице 3.2.О других графических функциях можно узнать в справочной системе MATLAB.

Таблица 3.2 – Функции для построения поверхностей

Функция	**Назначение**
mesh, surf	Строит поверхности
meshc, surfc	Строит поверхности и проекции
surfl	Строит поверхностии подчеркивает контуры
plot3	Трехмерная линия

Примеры построения поверхностей приведены ниже.

Пример3.6.

```
>>[X,Y]=meshgrid(-5:0.1:5);
>>Z=X.*sin(X+Y);
>>meshc(X,Y,Z)
```

В первой строке задается расположение сетки будущей поверхности с интервалом изменения *x* и *y* от -5 до 5 с шагом 0,1. Если диапазоны *x* и *y*

различны, функции передаются в двух диапазонах. Вторая строка задает выражение для вычисления *z*-значений в узлах сетки. Третья команда строит график поверхности.

Пример 3.7.

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

Вы можете изменить функцию *z(x,y)* и получить новый график. С помощью контекстного меню можно выполнить любую команду, в том числе не связанную с графикой.

3.3 Порядок выполнения лабораторной работы

3.3.1 Запустите MATLAB.

3.3.2 Постройте гистограмму случайных величин на интервале [-2,9;2,9] сшагом 0,1.

3.3.3 Задайте максимальную и минимальную температуру окружающей среды в векторах «*maxtemp*» и «*mintemp*» за первые десять дней (вектор «дата»). Выполните следующие действия:

- пос тройте график максимальной температуры;
- постройте график второй фигуры;
- постройте оба графика на одной фигуре;
- добавьте заголовок графика (используйте контекстное меню);
- добавьте надписи осей;
- на графике максимальной температуры измените стиль линии, цвет, свойства маркеров и т.д..
- выполните действия предыдущего пункта для графика минимальной температуры;
- добавьте *легенду* на график.

3.3.4 Выполните пример 3.6. Повторите этот пример для другой поверхности *z(x,y)* по вашему выбору. Используя меню фигуры, выполните некоторые изменения на графике (тип линии, цвет и т.д.).

3.3.5 Создайте трехмерную графику, используя функцию «*peaks*»:

- закройте текущее окно фигуры;
-наберите *sampleplot = peaks(25);* создается матрица *sampleplot,* элементы которой являются значениями функции *peaks;*
- наберите *surf (sampleplot)* для создания графика 3-D поверхности значений этой функции. Ось z координаты направлена вертикально;
- измените цвет фона графика: *Edit-Figure Properties:*
- щелкните *Style*, выберите нужный цвет;
- выберите*Title* и установите его;
- щелкните на построенную поверхность, выберите *Edit-Current Object Property - Color* и измените цвет.

3.3.6 Вы можете просмотреть график под разным углом:

- выберите*Tools-Rotate 3D (символ руки в панели инструментов)* и с помощью мыши можно посмотреть под разными углами;

- щелкните в любом месте рисунка и удерживайте курсор;

- немного переместите курсор. Таким образом вы можете контролировать точку зрения.

3.3.7 Постройте 2D график функции, заданной в интервале *[a,b]* с шагом *h* (по варианту -таблица 3.5.1). Отформатируйте графики

3.3.8 Постройте 3D график функции при заданных значениях аргументов *x* и *y* (по варианту - таблица 3.5.2). Отформатируйте графики.

3.3.9 Построить графики функций в полярной системе координат и функций, заданных в параметрическом виде (таблица 3.5.3).

Таблица 3.5.1 - Варианты заданий для построения двумерного графика

№	функция	a	b	h
1	$y = \|tg(x)\| + 0.1$	-2π	2π	$\pi/20$
2	$y = sin^2(x/3)$	-2π	2π	$\pi/20$
3	$y = sin(x) + cos^2(2x)$	-2π	2π	$\pi/20$
4	$z = 1+sin(x)$	-2π	2π	$\pi/20$
5	$z = 5cos(x)$	-2π	2π	$\pi/20$
6	$z = 0.025exp(-1.2x)$	-5	5	1
7	$z=0.015x^3$	-5	5	1
8	$z = 0.05x^2$	1	10	1
9	$z = 0.01x^3$	-10	10	1
10	$z = exp(x+3)/5000 - 1$	-8	8	1
11	$z = 0.00025e^3-x - 0.6$	-8	8	1
12	$z = sin(x) + sin(2x)$	$-\pi$	π	$\pi/8$
13	$z = sin^2(x) + cos(x)$	$-\pi$	π	$\pi/8$
14	$y = \|sin(x)\|exp(x/2)$	$-\pi$	π	$\pi/8$
15	$z = 5x - x^{1.5}+exp(x)$	0	5	0.5
16	$Y=1-2cos(x)$	$-\pi$	π	$\pi/8$
17	$Y=xsin(x)$	-2π	2π	$\pi/20$
18	$Y=(x-2)/(x+2)$	-5	5	1
19	$Y=x^2+1/x$	-8	8	1
20	$Y=10/(x^2+1)$	-10	10	1

Таблица 3.5.2 - Варианты заданий для построения трехмерного графика

№	Function	№	Function
1	$z=sin(x)cos(y)$	1	$z=sin(x)cos(y)$
2	$z=sin(x)cos(y)$	2	$z=sin(x)cos(y)$
3	$z=sin(x/2)cos(y)$	3	$z=sin(x/2)cos(y)$
4	$z=sin(2x)cos(y)$	4	$z=sin(2x)cos(y)$
5	$z = sin(x)cos(y/2)$	5	$z = sin(x)cos(y/2)$
6	$z = sin(x/2)cos(2y)$	6	$z = sin(x/2)cos(2y)$
7	$z = sin(2x)cos(2y)$	7	$z = sin(2x)cos(2y)$
8	$z = (1+sin(x)/x)(sin(y)/y)$	8	$z = (1+sin(x)/x)(sin(y)/y)$
9	$z = (sin(x)/x)cos(y)$	9	$z = (sin(x)/x)cos(y)$
10	$z = (sin(x)/x)\|cos(y)\|$	10	$z = (sin(x)/x)\|cos(y)\|$

Таблица 3.5.3 - Варианты заданий для построения графиков функций в полярной системе координат и функций, заданных в параметрическом виде

№	Функции в полярных координатах	№	Функции в параметрических координатах
1	$r=\varphi/2$	11	$X=t^3\ y=t^2$
2	$r=e^{\varphi}$	12	$X=10cos(t)\ y=sin(t)$
3	$r=\pi/\varphi$	13	$X=10cos^3t\ \ y=10sin^3t$
4	$r=2cos(\varphi)$	14	$X=a(cost+tsint)\ y=a(sint-tcost)$
5	$r=1/sin(\varphi)$	15	$X=at/(1+t^3)\ y=at^2/(1+t^3)$
6	$r=sec^2\varphi/2$	16	$X=2t+2-t\ \ y=2t-2-t$
7	$r=10sin(3\varphi)$	17	$X=2cos^2t\ y=2sin^2t$
8	$r=1/sin(\varphi)$	18	$X=t-t^2\ y=t^2-t^3$
9	$r=a(1+cos(\varphi))$	19	$X=a(2cost-cos(2t))\ y=a(2sint-sin(2t))$
10	$r^2=a^2cos(2\varphi)$	20	$X=a/sqrt(1+t^2)\ y=at/(sqrt(1+t^2)$

3.4 Отчет по лабораторной работе

Отчет по работе содержит скриншоты всех выполненных заданий с пояснениями.

3.5 Контрольные вопросы

3.5.1 Как можно построить график функции в MATLAB?

3.5.2 Как можно установить заголовки надписи осей графика?

3.5.3Как можно установить цвет, шрифт, размер шрифта на графике?

3.5.4 Как устанавливается интервал изменения аргумента функции для построения графика?

3.5.5 Как можно изменить свойства осей?

3.5.6 Как можно построить несколько графиков на одной фигуре?

3.5.7 В чем отличие столбчатой диаграммы и гистограммы?

3.5.8 Что необходимо для построения трехмерного графика??

3.5.9 В чем отличие двумерного и трехмерного графиков?

3.5.10 Как можно поворачивать трехмерный график?

4Лабораторная работа №4. Основы программирования в MATLAB

Цель работы: изучить типы программных файлов в MATLAB и освоить работу с ними.

4.1 Задание на лабораторную работу

В процессе выполнения лабораторной работы студент должен:

- изучить виды *m*-файлов;
- понять разницу между файл-функциями и скрипт-файлами;
- получить навыки создания *m*-файлов;
- изучить операторы отношения и логические операторы.

4.2 Типы М-файлов

Ранее командное окно MATLAB использовалось как «научный калькулятор». Однако, MATLAB также является мощным языком программирования, а также интерактивной вычислительной средой. В MATLAB можно написать последовательность команд в файл и выполнять этот файл как единое целое. В MATLAB возможно создание двух видов программных файлов: скрипты и файл-функции. Эти файлы имеют расширением *.m* и называются *m*-файлами.

Скрипт-файлы это – программные файлы, содержащие последовательность команд, которые необходимо выполнить вместе. Скрипты не имеют входных переменных и не возвращают какие-нибудь выходные. Они работают с данными в командном окне.

Все скрипты являются *m*-файлами. Но не все *m*-файлы являются скриптами.

Функция – группа выражений, которые вместе выполняют задачу. В MATLAB функции определяются в отдельном файле. Это тоже программные файлы с расширением **.m*. Можно использовать редактор MATLAB или любой другой для создания своего *.m*-файла. *M*-файл функции всегда начинается с предложения *function* и имени *.m*-файла, которое должно совпадать с именем функции.

Синтаксис функции имеет вид:

function $[out_1,out_2, ..., out_N] = myfun(in_1,in_2,in_3, ..., in_N)$,
где *myfun* – имя функции.

Первая строка функции должна начинаться с ключевого слова *function*. Она задает имя функции и перечень переменных. Сразу после этой строки можно поместить строки *комментариев* (пояснения к коду и алгоритму программы, которые начинаются и заканчиваются символом %). Эти строки появляются при наборе команды *helpmyfun.*

Для функции есть понятие локального и глобального (или основного) рабочего пространства, которое доступно из командной строки MATLAB . Функция может иметь более одной входной переменной (*in*) и возвращать более одной выходной переменной (*out*).

Пример:
```
function f=myfunction(x)
f=log(x)/sin(x)
```

В рассмотренной функции с именем *myfunction* входным аргументом является *x*, а результат вычислений – это выходная величина *f.* Файл функцию следует сохранять под тем же именем, которое указано в первой строке файла. Созданную функцию можно вызвать в командной строке или в другой функции. Для приведенного примера при вызове функции результат отражается в переменной *f*:

```
>>myfunction(2)
f =
0.7623
```

4.3. Логические выражения

Операции отношения. Операторы отношения могут работать и со скалярными и векторными данными. Операторы отношения для массивов выполняются поэлементным сравнением двух массивов и возвращают логический массив того же размера с элементами равными 1 (*true*), когда отношение верное и равными 0 (*false*) в противном случае.

Для задания условий используются следующие операторы отношений (таблица 4.1).

Таблица 4.1 - Операции отношения

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

Обратите внимание, что оператор равенства содержит два знака =, а не один.

Кроме этих перечисленных операций отношения MATLAB имеет следующие команды (функции):

- *eq(a, b)* – проверяет *a* и *b* на равенство;
- *ge(a, b)* – проверяет *a* больше или равно *b;*
- *gt(a, b)*– проверяет является ли *a* больше *b*;
- *le(a, b)* – проверяет *a* меньше или равно *b*;
- *lt(a, b)* – проверяет является ли *a* меньше *b*;
- *ne(a, b)* – проверяет является ли *a* не равно *b*;
- *isequal* – проверяет массивы на равенство;
- *isequaln* – проверяет массивы на равенство, рассматривая значения *NaN* как равные.

Логические операторы. В MATLAB логическое выражение имеет два возможных значения, которые не являются «true» или «false», но числовые, т. е. 1, если выражение истинно и 0, если оно ложно

MATLAB использует три логических оператора: *AND (&) - И, OR(|) - ИЛИ, NOT(~)- НЕ* (таблица 4.2).

Как и в случае с операциями отношения, логические операции можно применять к векторам и матрицам.

Таблица 4.2 - Логические соотношения

Оператор	Условие	Способ 1	Способ 2
Логическое И	x<3 и y=4	and (x<3, y==4)	(x<3) & (y==4)
Логическое ИЛИ	x=1 или x=2	or (x==1, x==2)	(x==1) | (x==2)
Отрицание НЕ	a≠1.9	not (a==1.9)	~(a==1.9)

Как и в случае с определенными операциями, логические операции могут применяться к векторам и матрицам.

4.4 Порядок выполнения лабораторной работы

4.4.1 Создайте *скрипт-файл* для выполнения следующих операций:

- найти произведение матриц *A* и *B*;
- найти произведение матрицы *A* на вектор *v*;
- найти матрицу *C,* обратную к матрице *B*;
- найти определитель матрицы *A*.

4.4.2 Создать *файл-функцию* для преобразования температуры, заданной в градусах Фаренгейта в градусы Цельсия, по следующей формуле:

$$t_{Cel} = (t_{Fahr} - 32) \cdot 5/9;$$

- используя эту функцию, найдите температуру кипения воды по Цельсию, если она равна 212°по Фаренгейту;
- примените эту функцию к любым векторам;
- примените эту функцию к любым матрицам;
- объясните результаты.

4.4.3 Создайте файл-функцию с несколькими входными переменными - для нахождения объема куба с размерами: длина *len*, ширина*br* и высота*dep* в качестве входных переменных;

4.4.4 Измените предыдущую функцию таким образом, чтобы она вычисляла и объем, и площадь поверхности куба; добавьте комментарии; выполните вычисления и получите результаты для нескольких значений входных переменных.

4.4.5 Разработайте блок-схему алгоритма для вычисления корней квадратного уравнения. Создайте файл-функцию для реализации этого алгоритма. С помощью этой функции вычислите корни выбранного самостоятельно квадратного уравнения. В тексте функции предусмотрите проверку дискриминанта уравнения.

4.4.6 Создайте файл-функцию для вычисления:
- объема сферы заданного радиуса;
- поверхности сферы;
- сохраните функцию;
- вычислите объемы и поверхности сфер для нескольких значений радиусов;

4.4.7 В командном окне введите скалярные величины *q, w, e*. Напечатайте:

- w< e; q==e; (e>0) | (q<0); result = ~(q<0); (e>0) | (q<0).

Объясните результаты.

4.4.8 Введите вектора *x* и *y* размерности 3. Напечатайте:
- z= (x<y);
- z = ~(x<y);

Объясните результаты.

4.5 Отчет по лабораторной работе

Отчет содержит листинги программ, скриншоты результатов работы программ.

Студент должен продемонстрировать работу программы преподавателю.

4.7 Контрольные вопросы

4.7.1 Дайте определение m-файла.

4.7.2 Дайте определение скрипт-файла.

4.7.3 Дайте определение файл-функции.

4.7.4 Как запускается файл-функция?

4.7.5 Объясните разницу между скриптом и функцией.

4.7.6 Какое расширение имеют m-файлы?

4.7.7 Как можно отредактировать m-файл?

4.7.8 Как можно разместить комментарии в m-файле?

4.7.9 Как можно при использовании m-файла просмотреть комментарии к нему?

4.7.10 Дайте определение алгоритма задачи. Что такое блок-схема алгоритма?

5 Лабораторная работа № 5. Циклические операторы в среде программирования MATLAB

Цель работы: освоить методику построения алгоритмов ветвления и циклической структуры в MATLAB.

5.1 Задание на лабораторную работу

В процессе выполнения лабораторной работы студент должен:

- освоить циклический оператор *for;*
- освоить алгоритмы использования операторов ветвления *if;*
- изучить оператор выбора *while*;
- изучить оператор переключения *switch*;
- понять различие в применении вышеперечисленных операторов;
- выполнить индивидуальные задания и подготовить отчет по работе.

5.2 Операторы управления

Важным аспектом в программировании являются *операторы управления*. Это операторы, дающие выбор и направление вычислительным процессам.

Разновидность управляющей конструкции, предназначенная для организации многократного исполнения набора инструкций, называется *циклом*. Последовательность инструкций, предназначенная для многократного исполнения, называется *телом цикла*. Единичное выполнение тела цикла называется *итерацией*. Выражение, определяющее, будет в очередной раз выполняться итерация или цикл завершится, называется *условием выхода* или *условием окончания цикла* (либо *условием продолжения* в зависимости от того, как интерпретируется его истинность — как признак необходимости завершения или продолжения цикла).

К таким операторам относятся условные операторы *if, while, switch* и *for*. Нужно отметить, что в программной среде MATLAB нет безусловного оператора «метка» (существующего в других языках программирования), что затрудняет переход к последующему или предшествующему оператору.

Начало всех циклов с операторами *while, switch* и *for* заканчиваются оператором *end*.

5.2.1 Циклический оператор *for*

Этот цикл предназначен для выполнения определенного числа повторяющихся действий. Его структура выглядит следующим образом:

for x =начальное значение :шаг : конечное значение
командыMATLAB
end

Счетчик может принимать не только целые значения, но и вещественные.

А также, цикл *for* можно применять для вычисления накопления суммы (произведения). Для этого, в начале цикла переменной цикла, в которой накапливается сумма (произведение), присваивается ноль (если произведение, то присваивается единица), затем в цикле очередной *i*-тый элемент складывается (перемножается) с предыдущим.

Циклы *for* могут быть вложены друг в друга, при этом переменные вложенных циклов должны быть разными.

5.2.2 Оператор условного перехода *if*.

Оператор условия *if* дает возможность поэтапного выполнения команд, переход к которым задается в виде условий. Существует несколько алгоритмов написания программы с оператором *if*.

MATLAB предоставляет следующие типы выражений о принятии решений:

1. Состоит из оператора *if* и логического выражения, за которым следует один или несколько операторов. Завершается оператором *end*:

if <условие>
<операторы>
... end

Если *выражение* имеет значение *true*, то блок операторов внутри оператора *if* будет исполнен. Если *выражение* имеет значение *false*, то будет выполнен первый оператор после команды *end*.

2. За оператором *if* может следовать необязательный оператор *else;* операторы, следующие за ним выполняются, если выражение ложно:

if<условие>
<операторы)>
else
<операторы>

end

3. За оператором *if* может следовать один (или несколько) необязательных *else if* и else; полезно для проверки различных условий:

if < условие 1>
<операторы)>
elseif < условие 2>
<операторы>
elseif <условие 3>
< операторы>
else
< операторы>
End

4. Вложенные операторы if - можно использовать один оператор *if* или *else if* внутри другого оператора *if* или *else if*.

if< условие 1>
if< условие 2>
<операторы>
end
end

5.2.3 Цикл *while*.

Цикл *while* используется тогда, когда неизвестно количество необходимых итераций в теле цикла. Он выполняется до тех пор, пока выполняется условие цикла:

while условие цикла
команды MATLAB
end

Цикл также называется *циклом с предусловием*. Для задания условий используются следующие операторы отношений (таблица 4.1, лабораторная работа 4).

5.2.4 Оператор переключения *switch.*

Оператор переключения имеет следующую структуру:

Switch <выражение, скаляр или строка символов>
case <значение 1>

```
 <оператор1>
case <значение 2>
 <оператор2>
...
otherwise
 <операторы>
end
```

Оператор *switch* осуществляет разветвление в зависимости от значения в строках *case*, а вычисление этого значения осуществляется в строке *switch*. Фиксированных значений может быть много, в этом случае перед каждым значением пишем слово *case*. В случае, если значения, вычисляемые в строке *switch*, не совпадают ни с одним фиксированным значением, выполняются операторы, следующие за словом *otherwise*.

5.3 Примеры использования операторов управления

5.3.1 Пример программы с циклическим оператором *for.*

Задание 1. Вывести график семейства кривых *z*, которое задано функцией, зависящей от переменной *y*:

$$Z = 5 \cdot (y - y^2) .$$

Переменная *y* изменяется в интервале [1, 10] с шагом 1.

Текст программы:

```
>> y=[1:1:10];

 Z = 5 · (y − y^2) ;
  hold on
   plot(y,z)
end
```

Результат запуска программы приведен на рисунке 5.1.

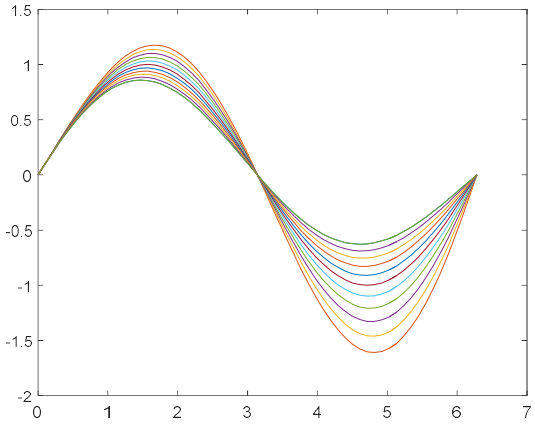


Рисунок 5.1 – Графики заданного семейства кривых

Задание 2. Вычислить сумму $s = \sum_{k=1}^{10} \frac{1}{k!}$

Тест файл-функции:

```
function summa
S=0;
for k=1:10
    S=S+1/factorial(k);
end
S
```

После написания программы необходимо запустить процесс (кнопка *Run*), после чего в командном окне появится ответ:

```
S=
   1.7183
```

5.3.2 Пример использования оператора условного перехода *if*.

Задание. Рассчитайте значения функции *f(x)* в зависимости от значения переменной *x* по следующей формуле:

$$f(x) = \begin{cases} \dfrac{x}{(5x-1)}, \text{при} \quad x < -4 \\ x^2 - x, \qquad \text{при} \quad x \geq -4 \end{cases}$$

Текст программы:

```
function f = myfunction ( x )
if x<-4
   f1=x/(5*x-1)
```

```
else f2=x^2-x
end
```

После сохранения программы, необходимо в командном окне набрать название *m*-файла и в скобках задать значение переменной (запуск программы выполнен для двух значений переменной *x*):

```
>> myfunction (-1)
f2 =
    2

>> myfunction (-5)
f1 =
   0.1923
```

5.3.3. Пример использования циклического оператора *while.*

Задание. Вычислить сумму ряда (разложение в ряд функции *sin x*) с точностью 10^{-10} (то есть в сумме учитываются только те члены ряда, которые не превышают10^{-10}):

$$S(x) = \sum_{k=0}^{\infty}(-1)^k \frac{x^{2k+1}}{(2k+1)!}.$$

Текст программы:

```
function y=mysinus(x)
s=0; k=0;
while abs(x.^(2*k+1)/factorial(2*k+1)) > 1.e-10
s=s+(-1)^k*(x.^(2*k+1)/factorial(2*k+1));
k=k+1;
end
s
```

Напомним, что знак «;» после оператора подавляет вывод результата на экран. Чтобы не выводить промежуточные расчеты, необходимо использовать этот знак.

Вызов программы:

```
>> mysinus(5)
```

В итоге выйдет ответ:

```
s =
   0.8415
```

5.3.4 Пример использования оператора ветвления *switch.*

Задание. Ввести значение натурального числа *n*. Сравнить это значение с тремя фиксированными значениями переменной *x*: -1; 0; 1. В случае, если значение *n* будет равно одному из значений *x*, будут выполняться операторы *case,* соответствующие этим значениям. В случае, если значение *n* не совпадет с этими значениями, будут выполняться операторы после ключевого слова *otherwise*.

```
n = input ('Enter a number: ');
switch n
   case -1
            disp('negative one')
   case 0
      disp('zero')
   case 1
      disp('positive one')
   otherwise
      disp('other value')
end
```

5.4 Порядок выполнения лабораторной работы

5.4.1 Изучите операторы цикла

5.4.2 Создайте блок-схему алгоритма сравнения:

- двух элементов *a* и *b,* используя оператор «*if… end*»;
- двух элементов *a* и *b,* используя оператор «*if...else...end*»;
- создайте скрипт-файл для реализации этих алгоритмов.

Необходимо в скрипте разместить комментарии.

5.4.3 Создайте блок-схему алгоритма последовательного сравнения двух чисел с несколькими другими, используя оператор *«if...elseif...elseif… ...else...end».* Создайте файл-функцию для решения этой задачи, снаразместите комментариями.

5.4.4 Выполните примеры п.5.4.

5.4.3 По своему варианту составьте файл-функции для расчета суммы ряда с использованием различных операторов: *for* (таблица 5.1)*; if* (таблица 5.2); *while* (таблица 5.3); *switch* (таблица 5.4). Для каждого оператора создается отдельный *m*-файл).

Таблица 5.1 – Варианты для применения оператора *for*

№ п/п	Функция
1	$Z = 4 \cdot (1 - y^2)$, где $y_0 = 1.5$, $\Delta y = 0.5$, $y_k = 4$
2	$Z = (x + 1)^3 - 2$, где $x_0 = 0.5$, $\Delta x = 0.5$, $x_k = 4$
3	$Z = 1 + 2 \cdot x - x^2$, где $x_0 = 0$, $\Delta x = 0.5$, $x_k = 4.5$

№ п/п	Функция
4	$Z = \sqrt{\lvert x+1\rvert}$, где $x_0 = -4$, $\Delta x = 1$, $x_k = 4$
5	$Z = \sqrt{4 \cdot x^2 + 1}$, где $x_0 = 0$, $\Delta x = 0.2$, $x_k = 2.4$
6	$Z = e^{x^2} - 1$, где $x_0 = -2$, $\Delta x = 0.2$, $x_k = 0$
7	$Z = 2 \cdot \sqrt{x+1} \cdot \cos(x)$, где $x_0 = 0$, $\Delta x = 0.25$, $x_k = 3.5$
8	$Z = \ln(x) + 5 \cdot x$, где $x_0 = 0.5$, $\Delta x = 0.5$, $x_k = 6$
9	$Z = 6 \cdot (1 - y^4)$, где $y_0 = 1$, $\Delta y = 0.5$, $y_k = 4.5$
10	$Z = (x+1)^3 - 4 \cdot x$, где $x_0 = 0$, $\Delta x = 0.5$, $x_k = 4$
11	$Z = 4 + 2 \cdot x - 3 \cdot x^2$, где $x_0 = 0$, $\Delta x = 0.5$, $x_k = 4.5$
12	$Z = \sqrt{\lvert 2 \cdot x + 1\rvert}$, где $x_0 = -4$, $\Delta x = 1$, $x_k = 4$
13	$Z = \sqrt{\frac{(4 \cdot x^2 + 3}{x}}$, где $x_0 = 0$, $\Delta x = 0.2$, $x_k = 2.4$
14	$Z = 2 \cdot x \cdot e^{x^2} - 1$, где $x_0 = -2$, $\Delta x = 0.2$, $x_k = 0$
15	$Z = 2 \cdot \sqrt{x+1} \cdot \sin(2x)$, где $x_0 = 0$, $\Delta x = 0.25$, $x_k = 3.5$
16	$Z = \ln(2x) + 5 \cdot x$, где $x_0 = 0.5$, $\Delta x = 0.5$, $x_k = 6$
17	$Z = \frac{x}{\sqrt[3]{x^2 - 4}}$, где $x_0 = 0$, $\Delta x = 0.2$, $y_k = 3$
18	$Z = x(x-1)^2(x-2)^3$, где $x_0 = 0$, $\Delta y = 0.25$, $y_k = 4$
19	$Z = 2e^{x^2 - 4x}$, где $x_0 = 1$, $\Delta x = 0.5$, $x_k = 5$
20	$Z = 2x^3 + 3x^2 - 12x + 5$, где $x_0 = 0$, $\Delta x = 0.4$, $y_k = 4$

Таблица 5.2 – Варианты для применения операторов *if*.

№ п/п	Функция	№ п/п	Функция
1	$f(x) = \begin{cases} 4 \cdot \sin(x) - 1, & \text{при } x < 1 \\ ln(x) + 5, & \text{при } x \geq 1 \end{cases}$	7	$f(x) = \begin{cases} \sqrt{4 \cdot x^4 + 3}, & \text{при } x < 2 \\ e^{2 \cdot x} - 0.5, & \text{при } x \geq 2 \end{cases}$
2	$f(x) = \begin{cases} 25 - x, & \text{при } x < 0 \\ \cos^2(x) - x, & \text{при } x \geq 0 \end{cases}$	8	$f(x) = \begin{cases} \frac{(3 \cdot x^3 + 5)}{(3 \cdot x + 1)}, & \text{при } x < -1 \\ ln\lvert x + 4\rvert, & \text{при } x \geq -1 \end{cases}$
3	$f(x) = \begin{cases} \sqrt{4 \cdot x^2 + 1}, & \text{при } x < 2 \\ e^x - 1, & \text{при } x \geq 0 \end{cases}$	9	$f(x) = \begin{cases} \frac{2.5 \cdot x}{(6 \cdot x + 4)}, & \text{при } x < -2 \\ \frac{(x^2 - 2)}{5}, & \text{при } x \geq -2 \end{cases}$
4	$f(x) = \begin{cases} \frac{(x^2 + 5)}{(x + 1)}, & \text{при } x < -1 \\ ln\lvert x + 2\rvert, & \text{при } x \geq -1 \end{cases}$	10	$f(x) = \begin{cases} -2x + 6, & \text{при } x < 10 \\ \cos^2(3x) - 2x, & \text{при } x \geq 10 \end{cases}$

№ п/п	Функция	№ п/п	Функция
5	$f(x)=\begin{cases}\dfrac{x}{(3x+2)}, \text{при } x<-2\\ x^2-2, \quad \text{при } x\geq -2\end{cases}$	11	$f(x)=\begin{cases}\dfrac{\sqrt{(4\cdot x^2+3}}{8}, \quad \text{при } x<3\\ 3\cdot x\cdot e^{5\cdot x}+1, \text{при } x\geq 3\end{cases}$
6	$f(x)=\begin{cases}-x+5, \quad \text{при } x<10\\ \cos^2(x)+2x, \text{ при } x\geq 10\end{cases}$	12	$f(x)=\begin{cases}3-5\cdot x\cdot\cos(x), \quad \text{при } x<5\\ \dfrac{(x^2+2)}{6\cdot x}, \quad \text{при } x\geq 5\end{cases}$
13	$f(x)=\begin{cases}\sqrt{4\cdot x^2+3}, \quad \text{при } x<3\\ e^x+1, \quad \text{при } x\geq 3\end{cases}$	17	$f(x)=\begin{cases}7\cdot\cos(x)+2, \quad \text{при } x<1\\ 2+(x-1)^3, \quad \text{при } x\geq 1\end{cases}$
14	$f(x)=\begin{cases}1-\cos(x), \text{при } x<5\\ x^2+2, \quad \text{при } x\geq 5\end{cases}$	18	$f(x)=\begin{cases}\sin\left(x-\dfrac{\pi}{4}\right), \quad \text{при } x<0\\ 2+x-x^2, \quad \text{при } x\geq 0\end{cases}$
15	$f(x)=\begin{cases}6\cdot cos(x)+1, \text{при } x<3\\ ln(2x)+15, \quad \text{при } x\geq 3\end{cases}$	19	$f(x)=\begin{cases}\dfrac{(2x-3)}{(3x+2)}, \quad \text{при } x<-1\\ ln\lvert x+6\rvert+5, \text{ при } x\geq -1\end{cases}$
16	$f(x)=\begin{cases}\dfrac{30-8\cdot x}{1.5}, \quad \text{при } x<0\\ \cos^2(x)-6\cdot x, \text{при } x\geq 0\end{cases}$	20	$f(x)=\begin{cases}2x^2-x^4, \quad \text{при } x<2\\ \mathrm{tg}^2(x), \quad \text{при } x\geq 2\end{cases}$

Таблица 5.3 – Варианты для применения оператора *while*

№ п/п	Ряд	Точность и значения переменной *x*
1	$\sum_{n=0}^{\infty}\frac{x^n}{n}$	x=[0..10] Точность 10^{-10}
2	$\sum_{n=0}^{\infty}(-1)^{n-1}\frac{x^n}{n}$	X=[-5..5] Точность 10^{-10}
3	$\sum_{n=0}^{\infty}\frac{x^{2n-1}}{2n-1}$	X=[-2..10] Точность 10^{-10}
4	$\sum_{n=0}^{\infty}(-1)^{n-1}\frac{x^{2n-1}}{2n-1}$	X=[-7..7] Точность 10^{-10}
5	$\sum_{n=0}^{\infty}(n+1)x^n$	X=[0..15] Точность 10^{-10}
6	$\sum_{n=0}^{\infty}(-1)^{n-1}(2n-1)x^{2n-2}$	X=[5..20] Точность 10^{-10}

№ п/п	Ряд	Точность и значения переменной x
7	$\sum_{n=0}^{\infty} n(n+1)x^{(n-1)}$	X=[0..10] Точность 10^{-10}
8	$\sum_{n=0}^{\infty} \frac{n}{x^n}$	X=[1..15] Точность 10^{-10}
9	$\sum_{n=0}^{\infty} \frac{x^{4n-3}}{(4n-3)}$	X=[-6..5] Точность 10^{-10}
10	$\sum_{n=0}^{\infty} \frac{(-1)^{n-1}}{(2n-1)3^{n-1}}$	X=[5..25] Точность 10^{-9}
11	$\sum_{n=0}^{\infty} \frac{2n-1}{2^n}$	X=[0..15] Точность 10^{-8}
12	$\sum_{n=0}^{\infty} \frac{(1+n)}{3^n}$	X=[2..20] Точность 10^{-9}
13	$\sum_{n=0}^{\infty} \frac{(-1)^n - n}{3^n}$	X=[-10..10] Точность 10^{-8}
14	$\sum_{n=1}^{\infty} \frac{(1+n)}{3^n}$	X=[-15..1] Точность 10^{-9}
15	$\sum_{n=1}^{\infty} \frac{n}{(2n+1)5^n}$	X=[0..50] Точность 10^{-8}
16	$\sum_{n=1}^{\infty} \frac{1}{n(n+1)}$	X=[50..100] Точность 10^{-9}
17	$\sum_{n=1}^{\infty} ({}^{2k+1}\sqrt{x} - {}^{2k-1}\sqrt{x}$	X=[1..15] Точность 10^{-10}
18	$\sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n}$	X=[0..10] Точность 10^{-8}
19	$\sum_{n=1}^{\infty} \frac{1}{n!\,2^n}$	X=[-5..5] Точность 10^{-9}
20	$\sum_{n=1}^{\infty} \frac{(-1)^n}{n!}$	X=[-1..15] Точность 10^{-8}

Таблица 5.4 – Варианты для применения оператора *switch*

п/п	Функция	п/п	Функция
	$f(x)=\begin{cases}4\cdot\sin(x)-1, \text{при } x=1\\ ln(x)+5, \quad \text{при } x=2\end{cases}$		$f(x)=\begin{cases}\sqrt{4\cdot x^4+3}, \quad \text{при } x=2\\ e^{2\cdot x}-0.5, \quad \text{при } x=3\end{cases}$
	$f(x)=\begin{cases}25-x, \quad \text{при } x=0\\ \cos^2(x)-x, \quad \text{при } x=5\end{cases}$		$f(x)=\begin{cases}\dfrac{(3\cdot x^3+5)}{(3\cdot x+1)}, \quad \text{при } x=-1\\ ln\lvert x+4\rvert, \quad \text{при } x=1\end{cases}$
	$f(x)=\begin{cases}\sqrt{4\cdot x^2+1}, \quad \text{при } x=2\\ e^x-1, \quad \text{при } x=0\end{cases}$		$f(x)=\begin{cases}\dfrac{2.5\cdot x}{(6\cdot x+4)}, \quad \text{при } x=-2\\ \dfrac{(x^2-2)}{5}, \quad \text{при } x=2\end{cases}$
	$f(x)=\begin{cases}\dfrac{(x^2+5)}{(x+1)}, \quad \text{при } x=-1\\ ln\lvert x+2\rvert, \quad \text{при } x=1\end{cases}$	0	$f(x)=\begin{cases}-2x+6, \quad \text{при } x=10\\ \cos^2(3x)-2x, \text{при } x=-10\end{cases}$
	$f(x)=\begin{cases}\dfrac{x}{(3x+2)}, \text{при } x=-2\\ x^2-2, \quad \text{при } x=2\end{cases}$	1	$f(x)=\begin{cases}\dfrac{\sqrt{(4\cdot x^2+3}}{8}, \quad \text{при } x=3\\ 3\cdot x\cdot e^{5\cdot x}+1, \text{при } x=-3\end{cases}$
	$f(x)=\begin{cases}-x+5, \quad \text{при } x=-10\\ \cos^2(x)+2x, \quad \text{при } x=10\end{cases}$	2	$f(x)=\begin{cases}3-5\cdot x\cdot\cos(x), \quad \text{при } x=5\\ \dfrac{(x^2+2)}{6\cdot x}, \quad \text{при } x=-5\end{cases}$
3	$f(x)=\begin{cases}\sqrt{4\cdot x^2+3}, \quad \text{при } x=3\\ e^x+1, \quad \text{при } x=-3\end{cases}$	7	$f(x)=\begin{cases}7\cdot\cos(x)+2, \quad \text{при } x=1\\ 2+(x-1)^3, \quad \text{при } x=-1\end{cases}$
4	$f(x)=\begin{cases}1-\cos(x), \text{при } x=5\\ x^2+2, \quad \text{при } x=-5\end{cases}$	8	$f(x)=\begin{cases}\sin\left(x-\dfrac{\pi}{4}\right), \quad \text{при } x=0\\ 2+x-x^2, \quad \text{при } x=10\end{cases}$
5	$f(x)=\begin{cases}6\cdot cos(x)+1, \text{при } x=3\\ ln(2x)+15, \quad \text{при } x=-3\end{cases}$	9	$f(x)=\begin{cases}\dfrac{(2x-3)}{(3x+2)}, \quad \text{при } x=-1\\ ln\lvert x+6\rvert+5, \quad \text{при } x=1\end{cases}$
6	$f(x)=\begin{cases}\dfrac{30-8\cdot x}{1.5}, \quad \text{при } x=0\\ \cos^2(x)-6\cdot x, \text{при } x=1\end{cases}$	0	$f(x)=\begin{cases}2x^2-x^4, \quad \text{при } x=2\\ \mathrm{tg}^2(x), \quad \text{при } x=-2\end{cases}$

5.5 Отчет по лабораторной работе

Отчет содержит листинги программ, скриншоты результатов работы программ. Студент должен продемонстрировать работу программ преподавателю.

5.6 Контрольные вопросы

5.6.1 Что такое операторы управления?
5.6.2 Что такое цикл?
5.6.3 Что такое тело цикла?
5.6.4 Дайте определение итерации.
5.6.5 Каким оператором заканчиваются циклы?
5.6.6 Приведите условия окончания цикла.
5.6.7 Дайте определение циклического оператора *for*.
5.6.8 Дайте определение условного оператора *if*.
5.6.9 Дайте определение оператора с предусловием *while*.
5.6.10 Дайте определение оператора ветвления *switch*.

6 Лабораторная работа №6. Численные методы решения обыкновенных дифференциальных уравнений

Цель работы: освоение методов численного решения дифференциальных уравнений первого порядка.

6.1 Задание на лабораторную работу

В процессе выполнения лабораторной работы студент должен:

- изучить методы численного решения дифференциального уравнения первого порядка: метод Эйлера и метод Рунге-Кутта;
- создать файл-функцию для расчета численных решений заданного дифференциального уравнения методом Эйлера;
- создать файл-функцию для расчета решений заданного дифференциального уравнения методом Рунге-Кутта;
- освоить процедуру*ode45* системы MatLab;
- применить процедуру *ode4 5*для решения заданного дифференциального уравнения;
- отразить на одном графике все три полученных решения: численные решения полученные методом Эйлер, Рунге-Кутта и с помощью процедуры *ode45*;
- оценить точность решений; сделать выводы.

6.2 Методы Эйлера и Рунге-Кутта решения обыкновенных дифференциальных уравнений

Дифференциальные уравнения возникают во многих областях прикладных областях науки: при исследовании сложных процессов, при разработке систем управления, при решении различных задач автоматик и др. С их помощью описываются практически любые задачи динамики.

С помощью классических методов решения дифференциальных уравнений не всегда возможно найти искомую функцию – решение уравнения. В большинстве практических задач коэффициенты или функции в дифференциальном уравнении могут быть существенно нелинейными. Тогда общим методом получения решения нелинейных обыкновенных дифференциальных уравнений является метод *численного интегрирования*. Существует множество различных численных методов решения обыкновенных дифференциальных уравнений (ОДУ). Рассмотрим наиболее широко используемые на практике методы Эйлера и Рунге-Кутта.

Рассмотрим дифференциальное уравнение:

$$y' = f(t, y) \tag{6.1}$$

с начальным условием

$$y(t_0) = y_0 \tag{6.2}$$

Подставив (t_0,y_0) в уравнение (7.1), получим значение производной в точке t_0:

$$y'(t_0) = f(t_0, y_0).$$

При малом Δt имеет место:

$$y(t_0 + \Delta t) = y(t_1) = y_0 + \Delta y = y_0 + y'(t_0) \cdot \Delta t = y_0 + f(t_0, y_0) \cdot \Delta t.$$

Обозначив $f(t_0, y_0) = f_0$, перепишем последнее равенство в виде:

$$y_1 = y_0 + f_0 \cdot \Delta t. \tag{6.3}$$

Принимая теперь (t_1, y_1) за новую исходную точку, точно также получим:

$$y_2 = y_1 + f_1 \cdot \Delta t.$$

В общем случае будем иметь:

$$y_{i+1} = y_i + f_i \cdot \Delta t. \tag{6.4}$$

Это формула называется формулой Эйлера, а сам метод - *методом Эйлера*. Величина Δt называется *шагом интегрирования, последовательные точки* t_i называются *узлами* интегрирования.

Пользуясь этим методом, мы получаем приближенные значения решения y_i, так как производная y' на самом деле не остается постоянной на промежутке длиной Δt. Поэтому мы получаем ошибку в определении значения функции y_i, тем большую, чем больше Δt.

Метод Эйлера является простейшим методом численного интегрирования дифференциальных уравнений и систем. Его недостатки: малая точность и систематическое накопление ошибок. Этот метод является методом *первого* порядка точности.

Порядок точности численного метода не связан с порядком дифференциального уравнения. Высокий порядок у численного метода означает его скорость сходимости.

Существует группа *методов Рунге-Кутта*, отличающихся друг от друга порядком точности (количеством параметров). Метод Эйлера тоже относитя к этой группе. Одним из наиболее применяемых на практике является метод Рунге-Кутта 4-го порядка точности, так как он обеспечивает высокую точность и в то же время отличается сравнительной простотой. Поэтому в большинстве случаев он упоминается в литературе просто как «метод Рунге-Кутта» без указания его порядка.

Метод Рунге-Кутта описывается системой следующих соотношений:

$$y_{i+1} = y_i + \frac{h}{6}(k_1 + 2 \cdot k_2 + 2 \cdot k_3 + k_4),$$

где

$$k_1 = f(x_i, y_i),$$

$$k_2 = f(x_i + \frac{h}{2}, y_i + \frac{h \cdot k_1}{2}),$$

$$k_3 = f(x_i + \frac{h}{2}, y_i + \frac{h \cdot k_2}{2}),$$

$$k_4 = f(x_i + h, y_i + h \cdot k_3).$$

Решение обыкновенных дифференциальных уравнений в Matlab можно реализовать, прописав алгоритм по разным схемам. Но также в Matlab есть встроенная функция ***ode45*** (*ordinary differential equation*), выполняющая численное вычисление решения методами Рунге-Кутта 4 и 5 порядков точности.

В самом простом виде функция ***ode45***имеет следующий синтаксис:

*[t, y] =ode45('<имяфункции>',interval, y0)**,***

где *t* -координаты узлов, в которых ищется решение;

y - вектор решений*y*в заданных точках*t*;

'*<имя функции>*' –строковая переменная, являющаяся именем *m*-файла, в котором вычисляется правая часть дифференциального уравнения;

Interval — массив из двух чисел, определяющий интервал интегрирования дифференциального уравнения;

y0 – начальное условие.

Функция*ode45* реализует метод Рунге - Кутта с автоматическим выбором шага. Такие алгоритмы используют тем большее количество шагов, чем медленнее изменяется функция. Поскольку функция ode45 использует формулы более высокого порядка, обычно требуется меньше шагов интегрирования и результат достигается быстрее.

6.3 Реализациячисленного решения дифференциального уравнения в Matlab

Рассмотрим, как методы Эйлера и Рунге-Кутта для решения обыкновенных дифференциальных уравнений реализуются в Matlab.

Пусть необходимо решить на отрезке *[0,1]*методом Эйлера дифферениальное уравнение (ОДУ):

$$y' = t + y,$$

при начальном условии $y(0) = 1$.

Аналитическое решение этого уравнения при заданном начальном условииимеет вид:

$$y = 2e^t - t - 1.$$

Составим в MatLab программу численного решения заданного обыкновенного дифференциального уравнения различными методами.

Текст файл-функции правой части рассматриваемого дифференциального уравнения с комментариями приведен на рисунке 6.1.

```
function dd=func1(t,y)          %название
dd=t+y;
```

Рисунок 6.1 – Текст файл-функции func1.m

На рисунке 6.2 приведен листинг файл-функции решения ДУ различными методами и построение графиков решения этими методами: аналитическим методом, методом Эйлера, методом Рунге-Кутта, с использованием процедуры ode45.

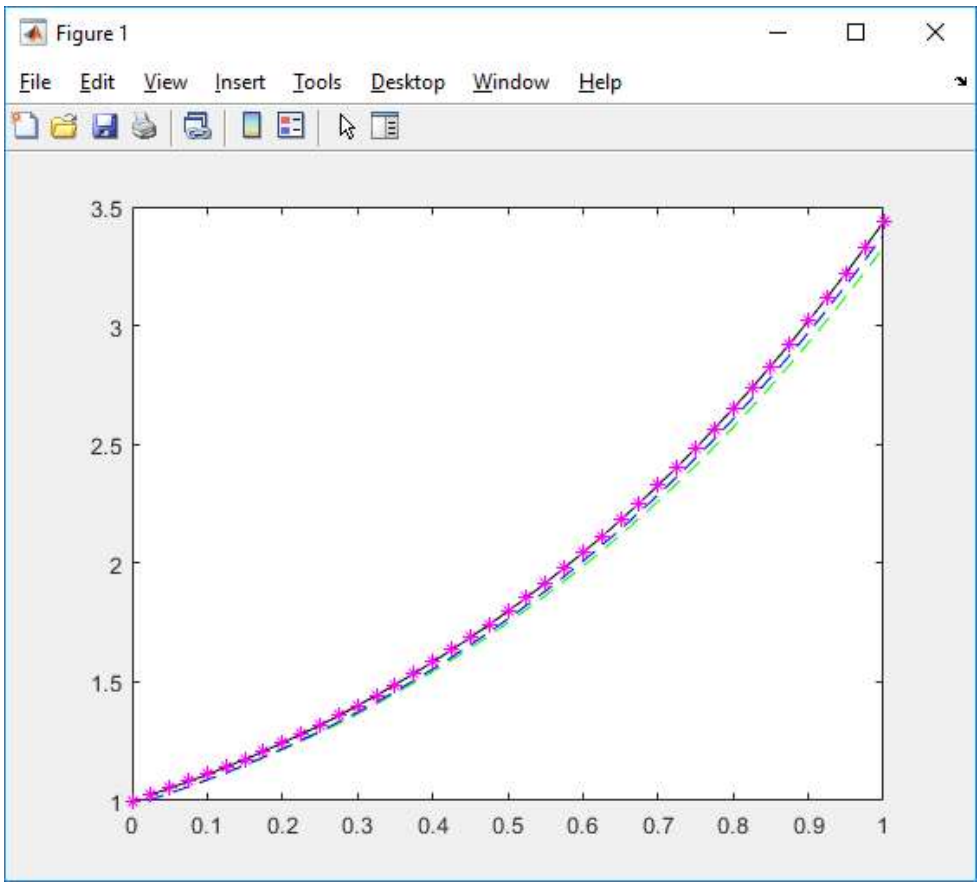
```
functiondiffur                          %заголовок файл-функции
a=0;                                    %задаем левую - a
b=1;                                    %и правую – b границы
интервала решения
m=50;                                   %задаем количество узлов
h=(b-a)/m;                              %задаем шаг интегрирования
%рассчитаем значения аналитического решения на заданном интервале
t=h:h:1;
y_Analytical=2*exp(t)-t-1;
y(1)=1; %начальное условие
%задаем подынтегральную функцию – правую часть ОДУ
F=@(t,y)(t+y);%вычисляем приближенные решения в узлах методом Эйлера
y_Euler(1)=y(1);
for i=1:m
y_Euler(i+1)=y_Euler(i)+h*F(t(i),y_Euler(i));
end
```

```
%построим на этой же фигуре график аналитического решения
plot(t,y_Analytical(1:m),'-k')
holdon
%построим график решения, полученного методом Эйлера
plot(t,y_Euler(1:m),'--g')
%вычисляем приближенные решения в узлах методом Рунге-Кутта
y_RK(1)=y(1);
for i=1:m
   k1=F(t(i),y_RK(i));
   k2=F(t(i)+0.5*h,y_RK(i)+0.5*h*k1);
   k3=F(t(i)+0.5*h,y_RK(i)+0.5*h*k2);
   k4=F(t(i)+h,y_RK(i)+h*k3);
y_RK(i+1)=y_RK(i)+(h/6)*(k1+2*k2+2*k3+k4);
end
%построим на этой же фигуре график решения методом Рунге-Кутта
hold on; plot(t,y_RK(1:m),'--b')
%для вычисления численного решения ОДУ с помощью процедурыode45
вначале
%сформируем файл-функцию function.m, описывающую правую часть
уравнения
% правая часть ДУ
[t,y_Procedure]=ode45('func1',[0,1],1);
plot(t, y_Procedure, '*m');hold off
```

Рисунок 6.2 – Текст файл-функции численного решения ОДУ

Запустим файл на выполнение. Обращение к созданной файл-функции:
>>diffur

Полученные результаты приведены на рисунке 6.2. Из рисунка видно, что решения методом Рунге-Кутта и с помощью процедуры ode45 хорошо приближены к аналитическому решению, чем решение методм Эйлера. Это связано с тем, что метод Эйлера имеет первый порядок точности, а использованный метод Рунге-Кутта – четвертый. В процедуре ode45 используется один из методов Рунге-Кутта высокого порядка точности.



6.3 - Графики решений ДУ различными методами

В приведенной программе для сравнения строится также график аналитического решения ОДУ. Но, поскольку, процедура *ode45* дает хорошую точность, при выполнении лабораторной работы построение графика аналитического решения не обязательно.

6.4 Порядок выполнения лабораторной работы

6.4.1Выполните задание п.6.3.

6.4.2 Составьте файл-функцию для решения заданного дифференциального уравнения (по варианту – таблица 6.1) методами Эйлера и Рунге-Кутта. Сохраните массивы полученных решений в векторах *ve* и *vrk*.

6.4.3 Добавьте в созданный *m*-файл строки вычисления численного решения с помощью процедуры*ode45*. Сохраните массив значений в векторе *vproc*.

6.4.4 Отразите графики всех полученных численных решений на одной фигуре.

6.4.5 Отразите вектора *ve*, *vrk* и *vproc* в одной таблице.

6.4.6 Выполните сравнительный анализ полученных результатов. Сделайте выводы.

Таблица 6.1 - Варианты заданий

№ вар-та	Дифференциальное уравнение
1	$y' + 2y = x^2,\ y(0) = 0, на\ \ интервале \quad [0,1], с\ шагом \quad h = 0.1.$
2	$y' = \sin\ x - \cos\ y\ \ y(0) = 1, на\ \ интервале \quad [0,1], с\ шагом \quad h = 0.2.$
3	$y' = x(e^y - 1),\ \ y(0) = 1, на\ \ интервале \quad [0,1], с\ шагом \quad h = 0.1.$
4	$yy'(1 + e^x) = e^x,\ \ y(0) = 1, на\ \ интервале \quad [0,2, с\ шагом \quad h = 0.2.$
5	$y' \sin x = (1 - y)\cos x\ \ y(\frac{\pi}{2}) = 0, на\ интервале\ [-2\pi, 2\pi], с\ шагом\ \ h = \frac{\pi}{10}.$
6	$(1 + x^2)y' = 2x\sqrt{1 - y^2},\ \ y(0) = 0, на\ интервале\ \ [0,1], с\ шагом\ \ h = 0.1.$
7	$y' + 2xy = 0,\ \ y(0) = 1, на\ \ интервале \quad [0,2], с\ шагом \quad h = 0.2.$
8	$y' = x^2 y^2 sn(x + y)^2,\ \ y(0) = 1, на\ \ интервале \quad [0,2], с\ шагом \quad h = 0.2.$
9	$y' - \frac{2y}{x+1} = (x + 1)^2 e^x,\ \ y(0) = 1, на\ \ интервале \quad [0,1], с\ шагом \quad h = 0.1.$
10	$y' + \sqrt{y}\ \sin(x) = 0,\ \ y(\frac{\pi}{2}) = 3, на\ \ интервале \quad [-2\pi, 1\pi], с\ шагом \quad h = \frac{\pi}{10}.$

6.5 Требования к отчету

Отчет по работе должен содержать:

- исходное дифференциальное уравнение (по варианту);
- формулы Эйлера и Рунге-Кутта;
- листинг файл-функции для решения заданного дифференциального уравнения методами Эйлера и Рунге-Кутта;
- пояснения к использованию процедуры MatLab решения дифференциального уравнения - *ode45*;
- решение заданного дифференциального уравнения с помощью процедуры *ode45*;
- таблицу с численными решениями уравнения всеми указанными методами;
- сравнительные графики решений уравнения всеми указанными методами (на одной фигуре);
- сравнение решений; вывод о степени точности.

6.6 Контрольные вопросы

6.6.1 Определение дифференциального уравнения.

6.6.2 Какое дифференциальное уравнение называется обыкновенным дифференциальным уравнением?

6.6.3 Что является решением обыкновенного дифференциального уравнения?

6.6.4 Чем определяется порядок дифференциального уравнения?

6.6.5 Для чего используется начальное условие?

6.6.6 Необходимость применения численных методов решения обыкновенных дифференциальных уравнений.

6.6.7 Объясните алгоритм метода Эйлера.

6.6.8 Как используется процедура *ode45*?

6.6.9 Как по графику можно оценить точность приближенного решения дифференциального уравнения?

6.6.10 Как определяется шаг интегрирования в методах Эйлера и Рунге-Кутта?

7 Лабораторная работа №7. Аппроксимация функций

Цель работы: освоение процедуры приближения функции методом наименьших квадратов.

7.1 Задание на лабораторную работу

В процессе выполнения лабораторной работы студент должен:

- изучить процедуру аппроксимации функций методом наименьших квадратов;
- составить файл-функцию для аппроксимации функции полиномами второго и третьего порядков;
- применить разработанную программу для функции, заданной своими значениями (по варианту); записать выражение аппроксимирующего полинома;
- выполнить аппроксимацию заданной функции с помощью процедуры *polyfit* системы MatLab; записать выражение аппроксимирующего полинома;
- сравнить результаты; сделать выводы.

7.2 Основные понятия

Термин «аппроксимировать» означает «приближённо заменить». Предположим, известны значения некоторой функции в заданных точках. Нам требуется найти промежуточные значения этой функции. Это задача называется задачей о *восстановлении* функции. Иногда, при проведении расчетов, сложные функции удобно заменять алгебраическими многочленами или другими элементарными функциями, которые достаточно просто вычисляются – такие задачи называются задачами о *приближении* функции. В любом случае нам надо построить аналитическое выражение для функции, заданной своими табличными значениями.

Аппроксимацией (приближением) функции *f(x)* называется нахождение такой функции *g(x)* - *аппроксимирующей функции*, которая была бы близка заданной. Близость исходной и аппроксимирующей функций определяется числовой мерой — *критерием аппроксимации* (близости). Критерии близости функций *f(x)* и *g(x)* могут быть различные. При заданной структуре необходимо таким образом подобрать параметры аппроксимирующей функции, чтобы

получить наименьшее значение критерия близости – наименьшее расхождение между *аппроксимируемой* (заданной) и *аппроксимирующей* (искомой) функциями. Наибольшее распространение получил среднеквадратичный критерий, равный сумме квадратов отклонений значений аппроксимирующей и заданной функций.

Для практики важен случай аппроксимации функции *полиномами*, то есть функциями вида:

$$g(t) = a_o + a_1 t + a_2 t^2 ... + a_n t^n. \tag{7.1}$$

Пусть для исходных данных t_i, $f(t_i)$, $i=1,...,N$, выбран вид аппроксимирующей зависимости с неизвестными коэффициентами в виде полинома степени n. Запишем сумму квадратов отклонений между заданными значениями $f(t_i)$ и вычисленными в тех же точках значениями аппроксимирующего полинома:

$$S = \sum_{j=1}^{N} [f(t_j) - (a_o + a_1 t_j + ... + a_n t_j^n)]^2. \tag{7.2}$$

Неизвестные параметры – коэффициенты искомого полинома будем находить из условия минимума функции S. Известно, что в точке минимума все частные производные от S по всем неизвестным a_i равны нулю:

$$\frac{\partial S}{\partial a_0} = 0, \frac{\partial S}{\partial a_1} = 0, ..., \frac{\partial S}{\partial a_n} = 0.$$

Приравнивая все производные функции (7.2) нулю и собирая коэффициенты при неизвестных a_i , получим следующую систему уравнений:

$$\begin{cases} a_0 \cdot N + a_1 \sum_{j=1}^{N} t_j + ... + a_n \sum_{j=1}^{N} t_j^n = \sum_{k=1}^{N} f(t_j), \\ ..... \\ a_0 \sum_{j=1}^{N} t_j^n + a_1 \sum_{j=1}^{N} t_j^{n+1} + ... + a_n \sum_{j=1}^{N} t_j^{2n} = \sum_{g=1}^{N} t_j^n . f(t_j) \end{cases} \tag{7.3}$$

Полученная система линейных алгебраических уравнений называется *нормальной* системой. Из ее решения находятся параметры a_j аппроксимирующей функции, обеспечивающие наилучшее возможное квадратичное приближение.

7.3 Пример решения задачи аппроксимации

Рассмотрим примеры аппроксимации функции полиномами различных степеней в среде MatLab.

Первый способ – составление программы для расчета коэффициентов системы (7.3) и решение этой системы с помощью процедур MatLab.

Второй способ – применение функции *polyfit*. Синтаксис этой функции:

p = polyfit(x, y, n),

где n – порядок аппроксимирующего полинома;

x, y – вектора аргументов и значений функции;
p - вектор коэффициентов полинома; длина этого вектора равна *(n + 1).*

Пример. Функция задана своими значениями:

t	-5	-3.5	-2	1.5	3.25	5
y	0.5	1.2	1.4	1.6	1.7	1.5

Требуется найти аппроксимирующий полином второй степени.

Решим задачу определения коэффициентов полинома второго порядка методом наименьших квадратов. В этом случае *N=6* и система нормальных уравнений имеет вид:

$$\begin{cases} a_0 \cdot 6 + a_1 \sum_{j=1}^{6} t_j + a_2 \sum_{j=1}^{6} t_j^2 = \sum_{k=1}^{6} f(t_j), \\ a_0 \sum_{j=1}^{6} t_j + a_1 \sum_{j=1}^{6} t_j^2 + a_2 \sum_{j=1}^{6} t_j^3 = \sum_{k=1}^{6} t_j \cdot f(t_j), \\ a_0 \sum_{j=1}^{6} t_j^2 + a_1 \sum_{j=1}^{6} t_j^3 + a_2 \sum_{j=1}^{6} t_j^4 = \sum_{g=1}^{6} t_j^2 . f(t_j) \end{cases}$$

Для решения этой системы необходимо вначале рассчитать значения коэффициентов матрицы системы, записать их в двумерный массив, рассчитать компоненты вектора правых частей, затем решить систему с помощью известных процедур.

Текст файл-функции для определения коэффициентов и решения нормальной системы приведен на рисунке 7.1.

```
function syst
t = [-5 -3.5 -2  1.5 3.25 5];          % исходный вектор аргументов
y = [0.5  1.2  1.4 1.6  1.7  1.51];    % исходный вектор значений функции
d(1,1)=6;                              % D – матрица системы
% расчет коэффициентов матрицы
d(1,2)= sum(t);
d(2,1)= sum(t);
d(2,2)= sum(t.^2);
d(1,3)= sum(t.^2);
d(3,1)= sum(t.^2);
d(2,3)= sum(t.^3);
d(3,2)= sum(t.^3);
d(3,3)= sum(t.^4);
% расчет вектора правой части
b(1) = sum(y);
b(2) = sum(t.*y);
b(3) = sum(t.^2.*y);
% получили систему уравнений D*a=b
%решение системы уравнений
a=inv(d)*b'
```

Рисунок 7.1 – Текст файл-функции для реализации метода наименьших квадратов

Запустим файл на выполнение и получим ответ – вектор параметров аппроксимирующего полинома:

a =

1.6470
0.0872
-0.0241

Сам полином имеет вид:

$$y = 1.6470 + 0.0872\,t - 0.0241\,t^2 .$$

Теперь применим функцию *polyfit* для аппроксимации заданной функции полиномами различных степеней. Текст файл функции приведен на рисунке 7.2.

```
x = [-5 -3.5 -2  1.5 3.25 5];       %исходный вектор аргументов
y = [0.5  1.2  1.4 1.6  1.7  1.51]; %исходный вектор значений функции
p1=polyfit(x,y,1);                  %аппроксимация полиномом 1 –ой степени
p2=polyfit(x,y,2);                  %аппроксимация полиномом 2 –ой степени
p3=polyfit(x,y,3);                  %аппроксимация полиномом 3 –ой степени
p4=polyfit(x,y,4);                  %аппроксимация полиномом 4 –ой степени
%для  представления графика  заданной дискретной функции в виде
%отдельных вертикальных линий для каждого из значений аргумента
stem(x,y);
% для отражения графика дискретной функции на той же фигуре
hold                                %
% чтобы график функции получился гладким выбираем малый шаг по %аргументу
x=-5:0.05:5;
%расчет значений полиномов для заданных значений аргумента
y1=polyval(p1,x);
y2=polyval(p2,x);
y3=polyval(p3,x);
y4=polyval(p4,x);
plot(x,y1,x,y2,x,y3,x,y4);
grid;
title('Полиномиальная аппроксимация');
xlabel('argument');
ylabel('function')
```

Рисунок 7.2 – Использование функции *polyfit* для аппроксимации заданной функции

Результат работы программы приведен на рисунке 7.3.

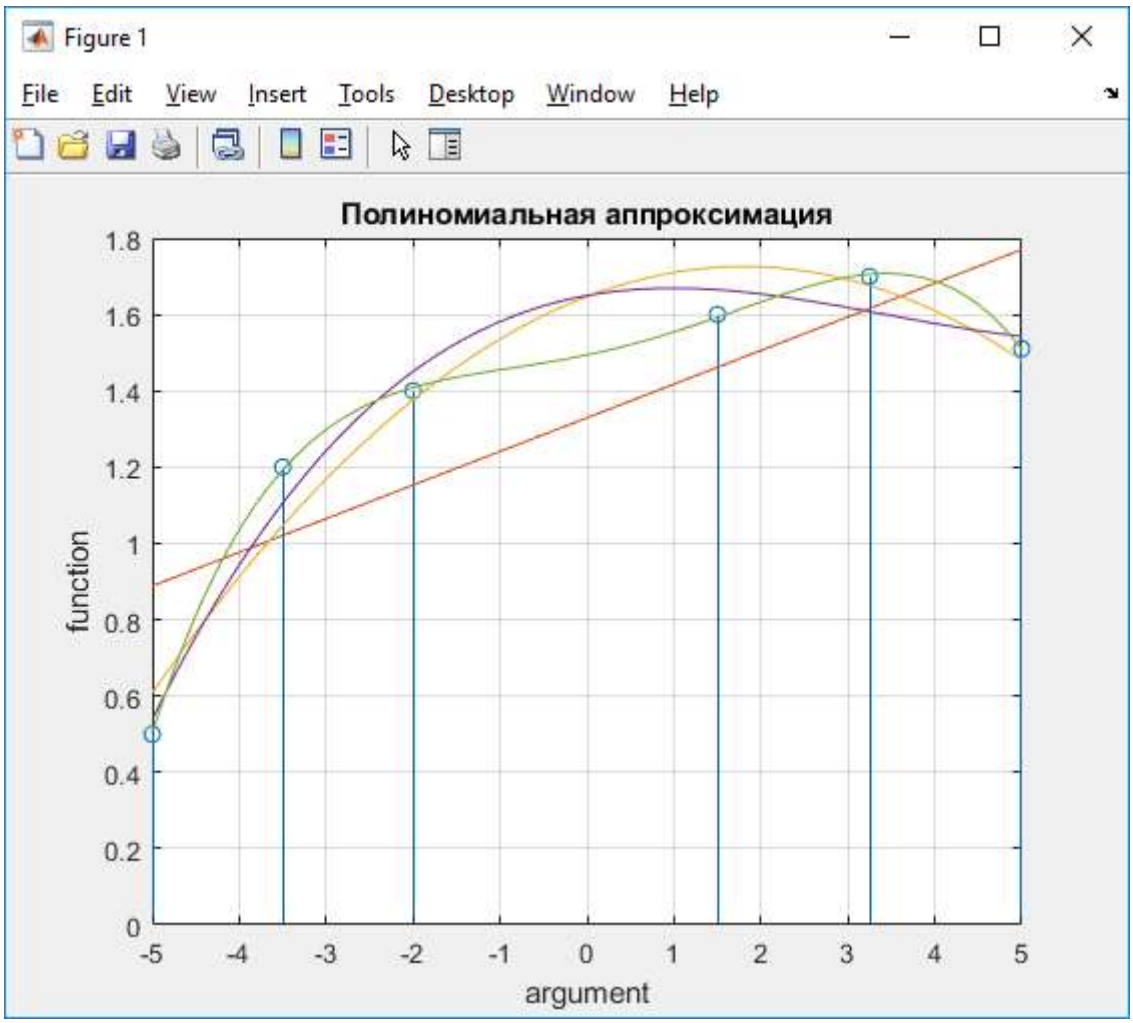


Рисунок 7.3 – Аппроксимация дискретной функции полиномами различных степеней

7.4 Порядок выполнения работы

7.4.1 По заданной функции (по варианту – таблица 7.1) сформировать массив ее значений из 10 элементов. В дальнейшем аппроксимируется полученная табличная функция, интервал аппроксимации для всех вариантов выбрать [-1,1].

7.4.2 Составьте файл-функцию для решения задачи аппроксимации заданной функции полиномом третьего порядка.

7.4.3 Используя составленную программу, найдите коэффициенты аппроксимирующего полинома; запишите окончательный вид полинома.

7.4.4 Постройте график полученного полинома и исходной функции (по варианту) на одной фигуре. Сделайте вывод о точности аппроксимации

7.4.5 Составьте файл-функцию для аппроксимации табличной функции полиномами 1,2,3,4 порядков, используя функцию *polyfit* системы MatLab.

7.4.6 Постройте графики дискретной функции и аппроксимирующих полиномов на одной фигуре.

7.4.7 Запишите выражения для полученных полиномов. Сравните коэффициенты полиномов третьей степени с коэффициентами полинома, полученного в п.7.4.3.

Таблица 7.1 - Варианты заданий

№	Функции для выполнения задания	№	Функции для выполнения задания
1	$f(t) = t - \sin t$	11	$f(t) = t\ln(t+2)^2$
2	$f(t) = t^3 + e^t$	12	$f(t) = t^3 \cdot \sin t$
3	$f(t) = \sqrt{t+1} - \cos t$	13	$f(t) = t \cdot tgt$
4	$f(t) = t^2 \cdot \cos t$	14	$f(t) = t \cdot \ln t^2$
5	$f(t) = \sin t - \frac{t}{2t+6}$	15	$f(t) = t^2 \cdot tgt$
6	$f(t) = \ln t^2 + t^3$	16	$f(t) = \sqrt{t^2} + \ln t^2$
7	$f(t) = 3 \cdot t + -\cos(t+1)$	17	$f(t) = e^t \cdot (t-2)^2$
8	$f(t) = t \cdot \sqrt{t+2}$	18	$f(t) = (t+3) \cdot \cos t$
9	$f(t) = t^2 \cdot \sin t$	19	$f(t) = t^2 \cdot \ln(t+3)$
10	$f(t) = t^2 + \sin t$	20	$f(t) = t \cdot \cos(t+3)$

7.5 Требования к отчету

Отчет по лабораторной работе должен содержать:

- вариант задания;
- табличное представление исходной функции;
- вывод системы нормальных уравнений для аппроксимации полиномом третьего порядка;
- файл-функцию реализации метода наименьших квадратов; результат работы этой файл-функции; окончательный вид аппроксимирующего полинома;
- сравнительные графики исходной функции и аппроксимирующего полинома третьего порядка;
- файл-функцию для аппроксимации заданной функции полиномами различных степеней с помощью процедуры *polyfit;* результат работы этой файл-функции; окончательный вид аппроксимирующих полиномов;
- сравнительные графики исходной, дискретной функций и аппроксимирующих полиномов;
- выводы по работе.

7.6 Контрольные вопросы

7.6.1 Дайте определение понятию «аппроксимация» функции.

7.6.2 Что такое критерий близости функций?

7.6.3 Объясните метод наименьших квадратов.

7.6.4 Дайте определение нормальной системе при решении задачи аппроксимации.

7.6.5 Каким свойством обладает матрица нормальной системы?

7.6.6 С помощью какой команды MatLab можно решить систему нормальных уравнений?

7.6.7 Объясните входные переменные процедуры *polyfit*.

7.6.8 Что является выходом процедуры *polyfit?*

7.6.9 Какие команды MatLab используются для размещения графиков нескольких функций на одной фигуре?

7.6.10 Как определить степень близости полученной аппроксимирующей функции к исходной дискретной функции?

8 Лабораторная работа №8. Основы Simulink

Цель работы: изучить основные приемы работы в пакете динамического моделирования Simulink.

8.1 Задание на лабораторную работу

В процессе выполнения лабораторной работы студент должен:

- изучит назначение и приемы работы в среде Simulink;
- изучить процедуру создания блок-диаграмм;
- создать блок-диаграммы рассматриваемых задач;
- выполнить анализ моделирования; сделать выводы.

8.2 Описание Simulink

Simulink является графическим расширением MatLab для решения задач моделирования динамических систем. В *Simulink* система представляется в виде *блок-диаграммы*.

Simulink запускается из командной строки вводом команды *simulink* или нажатием на кнопку «*Simulink Library Browser»* в верхней части командного окна MatLab. В результате на экране появится окно обозревателя библиотеки Simulink - *Library Browser window* (рисунок 8.1). Необходимые блоки для моделирования систем можно найти во вложенных папках библиотеки (они открываются нажатием на знак «+»).

Существует два главных класса элементов в: *блоки* и *линии*. Блоки используются, для создания, изменения, вывода и отражения сигналов. Линии используются для передачи сигналов из одного блока в другой.

Блоки. В этой папке размещены общие классы блоков, доступных для использования:

- *Continuous*: линейные, непрерывные элементы (интеграторы, передаточные функции, модели пространства состояний и т.д.);
- *Discrete*: линейные, дискретные элементы систем (интеграторы, передаточные функции, модели пространства состояний и т.д.);
- *User-Defined Functions*: функции, определенные пользователем;
- *Math Operations*: математические операторы;

- *Nonlinear*: нелинейные операторы (переключатели, реле и др.);

- *Ports & Subsystems:* блоки для управления/наблюдения сигналов и создания подсистем;

- *Signal Routing*: блоки для управления сигналами;

- *Sinks*: используются для отражения выходных сигналов;

- *Sources*: используются для генерирования различных сигналов.

Блоки могут не иметь или иметь несколько входных и выходных портов.

Линии. Линии перемещают сигналы в направлении, определенном стрелкой. Сигналы всегда передаются с выходного порта одного блока на входной порт другого.

Линии никогда не могут передавать сигнал в другую линию; линии должны быть объединены с помощью другого блока, например, сумматора.

Сигналы могут быть скалярными или векторными. Линии, используемые для передачи скалярных и векторных сигналов идентичны. Тип сигнала, передаваемого линией, определяется блоками на обоих концах линии.

Поиск необходимых блоков выполняется набором имени блока в этом же окне в поле поиска «Enter search item».

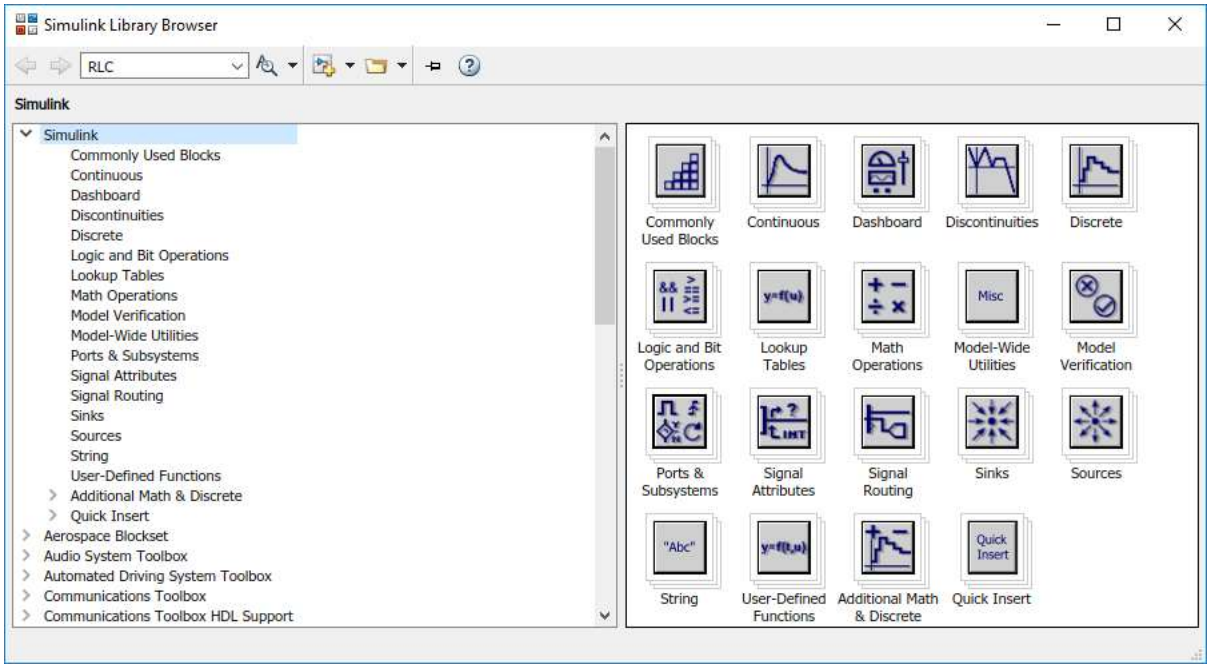


Рисунок 8.1 – Окно Simulink Library Browser

8.3 Пример создания блок-диаграммы

Создадим блок-диаграмму для решения следующего уравнения:

$$x' = \sin(t), x(0) = 0.$$

8.3.1 Откройте новое окно модели *untitled* и сохраните ее: *New-Model*.

8.3.2 Перетащите из окна *Library Browser* и разместите в окне модели следующие блоки: *Sine Wave, Integrator, Scope.*

8.3.3 Simulink позволяет настроить блоки в модели, чтобы они точнее отражали характеристики анализируемой системы. Настройка проводится двойным щелчком на блоке.

Щелкните на блок *Sine Wave*. В появившемся окне установите амплитуду, частоту и фазу синусоидального входного сигнала. Значение *Sample time* указывает временной интервал между последовательными показаниями сигнала. Установка этого значения в 0 указывает на непрерывную выборку сигнала.

Начальное условие уравнения вводится в окне настройки блока *Integrator*

8.3.4 Измените размеры блоков, выполните некоторые операции форматирования..

8.3.5 Соедините блоки. Добавьте в свободное пространство окна модели выражение уравнения.

8.2.6 Отрегулируйте время моделирования. Запустите модель.

8.3.7 Настройте параметры блока *Scope*, например: *Auto scale, Axes properties, Scope parameters*. Просмотрите результат (двойной щелчок на блоке *Scope*).

8.3.8 Выполните следующие изменения блок-диаграммы модели; после выполнения изменений запускайте моделирование и исследуйте полученные результаты:

- выведите в окно *Scope* одновременно графики входного и выходного сигналов: выберите в окне *Scope* инструмент *Parameters* и задайте параметр *Number of axes* равным 2. При этом у блока *Scope* появятся дополнительные входы, соедините с одним из входов сигнал на входе;
- чтобы вывести графики входного и выходного сигналов в одних координатных осях, используйте блок *Mux* (мультиплексор), объединяющий несколько сигналов в вектор;
- с помощью графопостроителя *XYGraph* постройте траекторию в фазовом пространстве.

8.4 Порядок выполнения лабораторной работы

8.4.1 Изучите компоненты *Library Browser* системы Simulink и операции форматирования:

- создайте диаграммы для отражения различных сигналов из группы *Sinks* в окне *Scope;*
- запустите эти простые модели; просмотрите результаты;
- выполните настройки различных блоков;
- создайте блок-диаграммы для выполнения операций с входными сигналами: умножение на число, нахождение абсолютной величины, сложение, вычитание, деление, операции отношения, логические операции, математические функции;
- выполните операции форматирования (используя контекстное меню): изменение шрифта, цвета блоков и линий, фона, цвета экрана;

- выполните изменение положения блоков и их отображение: перевернуть блок, повернуть блок, показать/скрыть тень, показать/скрыть метки портов.

8.4.2 Для освоения основных блоков Simulink, создайте блок-диаграммы вычисления значений всех нижеприведенных функций:

1) $y = (1 + 2\sin(2t))^2$;

2) $y = \sqrt{1 + 0.5\sin(2t)}$;

3) $y = t * e^{-t^2} * \cos(2\pi t)$;

4) $y = 3.5 + 0.3t - 0.06t^2 - \sqrt{e^{-2t} + t^2}$;

5) $y = \min(5t, 100 - 2t^2$;

6) $y = \sqrt{|-100 + 20t|}$.

Отразите результат в блоках: *Scope, Display, To Workspace, To File.*

8.4.3 Соберите блок-диаграмму для решения дифференциального уравнения второго порядка (рисунок 8.2):

$y'' + \sin(t) = 0, y'(0)=0$, y(0)=0.5.

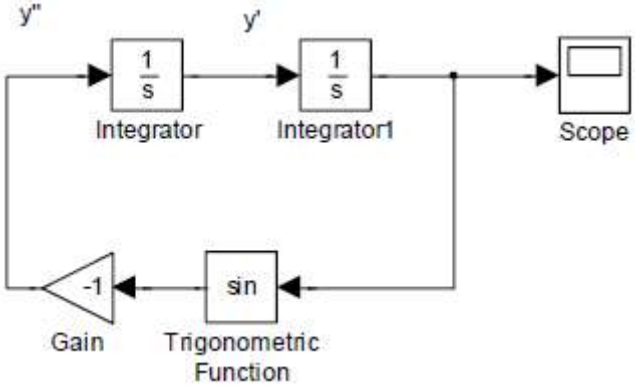


Рисунок 8.1 – Блок-диаграмма решения дифференциального уравнения

8.4.4 Добавьте к уравнению правую часть, здесь – функция e^{-t}:

$$y'' + \sin(t) = e^{-t}, y'(0)=0, y(0)=0.5.$$

Измените диаграмму на рисунке 8.2:

- добавьте блок *Sum*: на его вход подается входной сигнал (рисунок 8.3);
- выделите диаграмму полностью и создайте подсистему *Subsystem* (используйте контекстное меню); дайте название подсистеме;
- блоки *In* и *Out* необходимы для входного и выходного сигналов (раздел библиотеки *Ports&Subsystems*);
- на выходе разместите блок *Scope*;
- на входе подсистемы разместите блок *Fcn*;
- на входе блока *Fcn* разместите блок *Clock*;
- в блоке *Fcn* наберите выражение входного сигнала.
- запустите модель. Проанализируйте результаты.

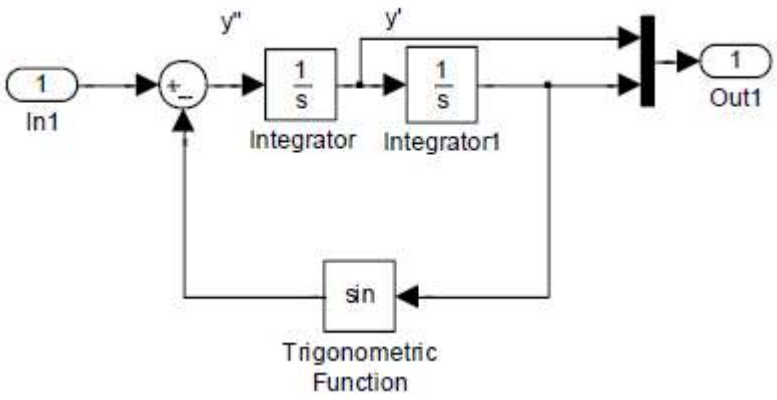


Рисунок 8.2 - Блок-диаграмма решения дифференциального уравнения с правой частью

8.4.5 Выполните задания по варианту (таблица 8.1).

8.4.6 Изучите назначение блока *Switch*.

8.4.7 Создайте блок-диаграмму для следующей задачи:

- установите время моделирования *T*;
- разместите два блока *Fcn,* в которых вычисляются два различных выражения (в качестве одного из них можете использовать выражение из п.8.4.5);
- отразите на блоке *Scope* в различные моменты времени графики различных выражений: до момента времени *T/2* – график первого выражения, после момента времени *T/2* – график второго выражения; для этого используйте блок *Switch;*
- отформатируйте диаграмму, разместите на ней комментарии;
- запустите диаграмму; результат в смотровом окне также отформатируйте.

Таблица 8.1- Варианты заданий

п/п	Уравнение
1	$(1+x^2)y'' - 2xy' = 0 \quad y = 0\, y' = 3$ при $x = 0$
2	$1 + y'^2 = 2yy'' \quad y = 1 \quad y' = 1$ при $x = 1$
3	$yy'' + y'^2 = y'^3 \quad y = 1 \; y' = 1$ при $x = 0$
4	$xy'' = y' \quad y = 0 \; y' = 0$ при $x = 0$
5	$y''y^3 = 1 \; y = 1 \quad y' = 1$ при $x = \frac{1}{2}$
6	$yy'' - 2xy' = 0 \quad y = 0 \quad y' = 3$ при $x = 0$
7	$xy''\mathrm{sqrt}(1 + y'^2) = 0 \; y = 0$ при $x = 1 \quad y = 1$ при $x = e^2$
8	$y''(1 + \ln x) + \frac{1}{x \cdot y'} = 2 + \ln x \quad y = \frac{1}{2} \; y' = 1$ при $x = 1$
9	$y'' = \frac{y'}{x \cdot \left(1 + \ln\left(\frac{y'}{x}\right)\right)} \quad y = \frac{1}{2} \quad y' = 1$ при $x = 1$
10	$y'' - y'^2 + y'(y-1) = 0 \quad y = 2 \quad y' = 2$ при $x = 0$

п/п	Уравнение
11	$3y'y'' = y + y'^3 + 1 \quad y = -2 \quad y' = 0$ при $x = 0$
12	$y^2 + y'^2 - 2yy' = 0 \quad y = 1 \; y' = 1$ при $x = 0$
13	$y'y'' + y'^2 + yy'' = 0 \quad y = 1$ при $x = 0$ $y = 0$ при $x = -1$
14	$2y' + (y'^2 - 6x)y'' = 0 \quad y = 0 \; y' = 2$ при $x = 2$
15	$y'y^2 + yy'' - y'^2 = 0 \quad y = 1 \; y' = 2$ при $x = 0$
16	$2yy'' - 3y'^2 = 4y^2 \quad y = 1 \; y' = 0$ при $x = 0$
17	$2yy'' + y^2 - y'^2 = 0 \quad y = 1 \; y' = 1$ при $x = 0$
18	$y'' = y'^2 - y \quad y = -\frac{1}{4} \; y' = \frac{1}{2}$ при $x = 1$
19	$1 + yy'' + y'^2 = 0 \quad y = 0 \; y' = 1$ при $x = 1$
20	$(1 + yy')y'' = (1 + y'^2)y' \quad y = 1 \; y' = 1$ при $x = 0$

8.4 Отчет по лабораторной работе

Отчет по лабораторной работе содержит блок-диаграммы примеров, приведенных в описании лабораторной работы, а также заданий по варианту.

8.5 Контрольные вопросы

8.5.1 Объясните назначение *Simulink*.

8.5.2 Объясните процедуру создания блок-диаграммы модели.

8.5.3 Перечислите основные компоненты библиотеки Library Browser.

8.5.4 Какие блоки используются в качестве входных сигналов модели?

8.5.5 Какие блоки используются для просмотра выходных сигналов модели?

8.5.6 Как установить время моделирования?

8.5.7 Как выполняется моделирование?

8.5.8 Для чего нужен блок *Subsystem*?

8.5.9 Приведите средства форматирования блоков и линий, окна просмотра результатов.

8.5.10 Объясните назначение и использование блоков *Fcn, Mux, Gain, Switch, In, Out*.

9 Лабораторная работа №9. Имитационное моделирование систем

Цель работы: изучить методы имитационного моделирования стохастических систем.

9.1 Задание на лабораторную работу

В процессе выполнения лабораторной работы студент должен:

- ознакомиться с основными определениями имитационного моделирования;

- изучить основные понятия и методы моделирования система массового обслуживания (СМО);

- изучить моделирующий алгоритм одноканальной СМОЖ
- составить блок-схему алгоритма задачи;
- изучить инструментарий Sim Events системы MATLAB;
- разработать блок-диаграмму модели рассматриваемого процесса в пакете Simulink с использованием инструментария Sim Events системы MatLab;
- выполнить сеансы моделировния для различных параметров рассматриваемой задачи; провести анализ полученных результатов;

9.2 Методы статистического моделирования стохастических систем

Имитационное моделирование – это распространенная разновидность аналогового моделирования, реализуемого с помощью набора математических инструментальных средств, специальных имитирующих компьютерных программ и технологий программирования. Имитационной моделью называется специальный программный комплекс, который позволяет имитировать деятельность какого-либо сложного объекта. *Имитационная модель* – это формальное описание логики функционирования исследуемой системы и взаимодействий отдельных ее элементов во времени, учитывающее наиболее существенные причинно-следственные связи, присущие системе, и обеспечивающее проведение статистических экспериментов. Имитационная модель позволяет исследовать поведение различных систем с учетом влияния случайных факторов.

Одной из систем, поведение которой можно исследовать с помощью имитационного моделирования является *система массового обслуживания* (СМО). Система массового обслуживания – это система, которая производит обслуживание поступающих в неё требований или, по-другому, система специального вида, реализующая многократное выполнение однотипных задач. Подобные системы играют важную роль во многих областях экономики, финансов, производства и быта. В качестве примеров СМО можно привести различные системы связи, погрузочно-разгрузочные комплексы, компьютерные сети, системы сбора, хранения и обработки информации, автоматизированные производственные участки и др.

Любая СМО состоит из следующих компонентов:

1. *требования*(заявки) на выполнение каких-то работ, эти требования поступают в случайные моменты времен;

2. *обслуживание* – выполнение этих работ; длительность обслуживания предполагается случайной;

3. выполнение осуществляется обслуживающими устройствами – *каналами* – устройства, которые в любой момент времени выполняют только одно требование.

Подход к изучению СМО состоит в том, чтобы сымитировать случайные моменты появления заявок и время их обслуживания в каналах, обработать и подсчитать характеристики СМО. Наиболее популярные из них: вероятность обслуживания клиента системой; пропускная способность системы; вероятность отказа клиенту в обслуживании; вероятность занятости каждого

канала и всех вместе; среднее время занятости каждого канала; вероятность занятости всех каналов; среднее количество занятых каналов; вероятность простоя каждого канала; вероятность простоя всей системы; среднее количество заявок, стоящих в очереди; среднее время ожидания заявки в очереди; среднее время обслуживания заявки; среднее время нахождения заявки в системе.

Параметрами СМО могут быть: интенсивность потока заявок, интенсивность потока обслуживания, среднее время, в течение которого заявка готова ожидать обслуживания в очереди, количество каналов обслуживания, дисциплина обслуживания и так далее.

9.3 Алгоритм работы системы массового обслуживания

Рассматривается *одноканальная* СМО с ограниченным временем ожидания *заявок* в очереди. Пусть на вход системы поступают заявки, образующие *ординарный поток однородных* случайных событий с заданным законом распределения. Эти заявки выстраиваются в очередь и обслуживаются в порядке поступления. Время ожидания является случайной величиной с заданным законом распределения. Если канал свободен, то заявка поступает на обслуживание. Система массового обслуживания рассматривается в интервале (0,T). Заявка, для которой момент обслуживания выходит за пределы интервала моделирования, получает отказ. Отказ получают также заявки, исчерпавшие предельное время ожидания и не дождавшиеся освобождения канала.

В результате моделирования можно оценить такие показатели работы СМО, как среднее время обслуживания, среднее время ожидания, вероятность обслуживания, вероятность отказа.

Примем, что в исследуемой СМО:

- интервалы между заявками подчиняются распределению *f1;*
- длительность обслуживания моделируется по закону *f2;*
- предельное время ожидания определяется по закону *f3.*

Математическая модель процесса функционирования системы выглядит следующим образом. Момент прихода *j*-ой заявки определяется из выражения:

$$T1(j) = T1(j-1) + x,$$

где *T1(j-1*) – момент прихода предыдущей заявки;

x – интервал между моментами поступления заявок, смоделированный по закону *f1*.

Если выполняется неравенство:

$$T1(j) \geq T,$$

то моделирование одной реализации процесса функционирования СМО закончено, и происходит обработка результатов.

Если выполняется неравенство:

$$T1(j) \geq T2(j-1),$$

где *T2(j-1)* – момент окончания обслуживания предыдущей заявки, то канал свободен, и пришедшая заявка сразу берется на обслуживание.

В противном случае канал занят, и заявка должна встать в очередь и ждать момента освобождения канала. При этом моделируется предельное время ожидания*y* по закону *f3* и определяется момент ухода заявки из системы:

$$T3(j) = T1(j) + y,$$

Если выполняется соотношение:

$$T3(j) = T2(j-1),$$

то заявка уходит необслуженной, так как предельное время ожидания истекает до того, как канал освободится. В противном случае заявка, прождав некоторое время в очереди, будет обслужена.

Фактическое время ожидания заявки:

$$TF(j) = T2(j-1) - T1(j),$$

Момент окончания обслуживания j-той заявки:

$$T2(j) = TH(j) + z,$$

где *TH(j)* – время начала обслуживания *j*- той заявки, *z* – длительность обслуживания, смоделированная по закону *f1*.

Показатели качества функционирования СМО формируются в ходе моделирования - для каждой реализации накапливается:

- сумма времен обслуживания*S1;*
- сумма времен ожидания *S2* заявок;
- общее количество пришедших заявок *J;*
- количество обслуженных заявок *S*.

Вычисляются следующие характеристики для каждой реализации:

- среднее время обслуживания:

$$S1 = \frac{S1}{J} = \sum_{j=1}^{N} TOB(j)/J,$$

где TOB(j) – длительность обслуживания *j* – той заявки;

- среднее время ожидания:

$$S2 = \frac{S2}{J} = \sum_{j=1}^{N} TF(j)/J,$$

- вероятность обслуживания: S3 = S/J;
- вероятность отказа: S4 = 1 – S3.

Аналогичным образом определяются показатели эффективности СМО за весь период моделирования (заданное число реализаций).

9.4 Инструментарий SimEvent системы MatLab

В лабораторной работе имитационное моделирование заданной системы реализуется с помощью компонент *Simulink* и *Sim Events* программной системы Matlab.

Sim Events и *Simulink* создают интегрированную среду для моделирования динамических систем, содержащих непрерывные компоненты и компоненты с дискретными событиями и дискретным временем.

При дискретно-событийном моделировании используется понятие *сущности* (*entity*). Сущности могут перемешаться через сети очередей (*queues*), серверов (*servers*) и переключателей (*switches*), управляемых дискретными событиями, в процессе моделирования.

Заявки принимаются к обслуживанию в порядке очереди. Освободившаяся линия приступает к обслуживанию той заявки, которая ранее других поступила в систему. Такую дисциплину называют “раньше поступил – раньше обслужился” (в англоязычной литературе FIFO – First In – First Out).

Событие (*event*) – это мгновенное дискретное явление, которое изменяет переменную состояния, выход и/или является причиной появления других событий. Примерами событий в модели *SimEvents* являются: перемещение сущности от одного блока к другому; завершение обслуживания сущности в сервере.

При дискретно-событийном моделировании очереди (*queues*) хранят сущности в течение некоторого интервала времени, который заранее неизвестен. Очередь пытается выпустить сущность как можно быстрее, но успех операции зависит от возможности следующего блока принять новую сущность.

Отличительными свойствами очереди являются:

- емкость (*capacity*), то есть максимальное количество сущностей, которые очередь может хранить одновременно;

- дисциплина очереди, определяющая какая из сущностей, покинет очередь первой, если она хранит несколько сущностей.

Блоки очередей находятся в разделе *Queues* библиотеки *Sim Events*.

Сервер (*server*) в дискретно событийном моделировании хранит сущности в течение некоторого промежутка времени, называемого временем обслуживания (*service time*) и затем пытается выпустить сущность. Время

обслуживания для каждой сущности вычисляется в момент ее прибытия в сервер. В отличие от этого время хранения блока в очереди принципиально заранее никогда неизвестно. Однако, если следующий блок не принимает сущность, которую уже обслужил сервер, то сервер должен хранить сущность дальше.

Отличительными свойствами сервера являются:

число сущностей, которые можно обслужить одновременно. Это число может быть конечным или бесконечным;

- характеристиками или методами вычисления времен обслуживания поступающих сущностей;

- разрешает ли сервер прибывающим сущностям занимать сервер при наличии других сущностей, хранящихся в сервере. При отсутствии этого свойства сервер с конечной емкостью, если он полон, не принимает к обработке новые прибывающие сущности.

Блоки серверов находятся в разделе *Servers* библиотеки *SimEvents*.

9.5 Порядок выполнения работы

9.5.1 Запустите Simulink.

9.5.2 Откройте библиотеку *SimEvents* - наберите в командной строке *simeventslib*.

9.5.3 В открывшемся окне выберите блок *Legacy blocks.* Появится окно библиотеки блоков *SimEvents* (рисунок 9.1).

9.5.3 Выберите блоки для всех ключевых процессов моделирования и разместите в окне модели:

- блок моделирования интервалов между заявками: *Entity Generators–Generators-Time Based Entity Generator*;

- блок моделирования длительности ожидания в очереди осуществляется с помощью блока хранения сущностей в очереди *Queues-FIFO Queue*;

- блок отсчета начало и конца времени ожидания – *Schedule Timeout*, *Cancel Timeout;*

- блок моделирования длительности обслуживания сущностей *Servers – SingleServer;*

-блок завершения обслуживания заявки: *SimEventsSinks –EntitySink*;

- блок отображения информации о ходе моделирования *SimEventsSinks-SignalScope.*

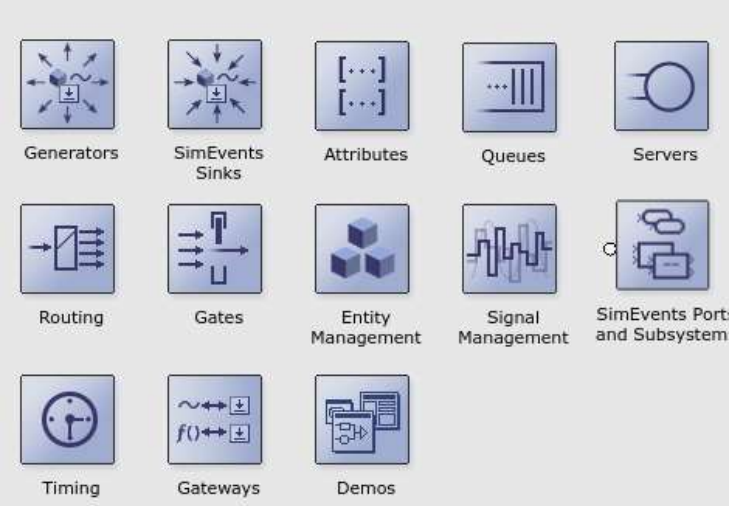


Рисунок 9.1 – Библиотека блоков *Sim Events (legacy blocks)*

9.5.3 Соедините блоки в соответствии с логикой работы системы.

9.5.4 Настройте параметры блоков (в соответствии с заданными по варианту (таблицы 9.1 и 9.2) законами распределения и их параметрами):

- в случайные моменты времени блок *Time-Based Entity Generator* генерирует события, моделирующие приход заявок. Генерирование времени между заявками можно выполнить двумя способами:

- в настройках блока указать тип распределения и его параметры (рисунок 9.2 а);

- генерирующий сигнал установить на внешний вход блока: *Inter generation time from port* (рисунок 9.2 б).

а) б)

Рисунок 9.2 – Вид блока *Time-Based Entity Generator*

а – интервал между заявками устанавливается в настройках блока,

б – интервал между заявками устанавливается на входе

- блок *FIFO Queue* сохраняет заявки, которые не могут быть немедленно обслужены. В настройках блока во вкладке *FIFO Queue* задать максимальное количество заявок в очереди. Во вкладке *Statistics* отметить параметры для вывода;

- так как рассматривается одноканальная СМО, то для моделирования обслуживания заявок использует блок *Single Server*(обслуживание не более одной заявки).

Времена обслуживания задаются через сигнальный порт *t* (*Service time from – Signal port t*) блоком *Event Based Random Number*, генерирующим случайные числа в соответствии с выбранным законом распределения и указанием необходимых параметров распределения.

При появлении ошибки в блоке *Time-Based Entity Generator* (в случае генерирования отрицательного числа) необходимо воспользоваться блоком *Switch* из раздела *Simulink-Signal Routing* (рисунок 9.3).

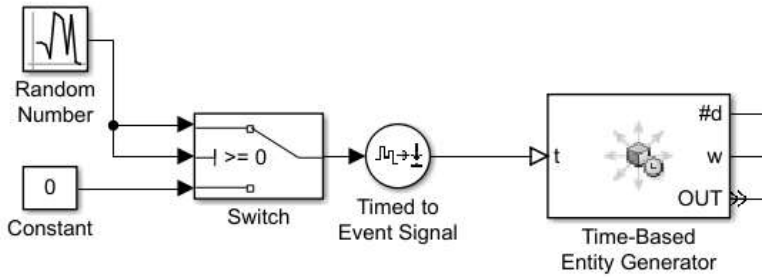


Рисунок 9.3–Вариант реализации генератора случайных чисел

- блок *Entity Sink* поглощает заявки, обработка которых завершена;

- блок *Schedule Timeout* моделирует время ожидания заявки. В паре с данным блоком необходимо предусмотреть окончание времени ожидания с помощью блока *Cancel Timeout;*

- для каждого из блоков необходимо собирать статистику: для этого нужно отметить галочкой нужные сигналы на вкладке *Statistics* в панели свойств у каждого блока.

Примеры возможных графиков приведены на рисунке 9.4.

9.5.5 Настройте параметры моделирования:

- выберите команду *Simulation-Configuration parameters-Solver*;

- в разделе *Solver options* в поле *Type* выберите *Variable-step* и в поле*Solver – Discrete*;

- в поле *Max step size* (максимальный размер шага) введите *auto*.

Время моделирования принять равным 480 минут (8 часов) – в Simulink ввести 480 (т.е. 1 секунда в модели – 1 минута реального времени).

9.5.6 Выполните сеанс моделирования.

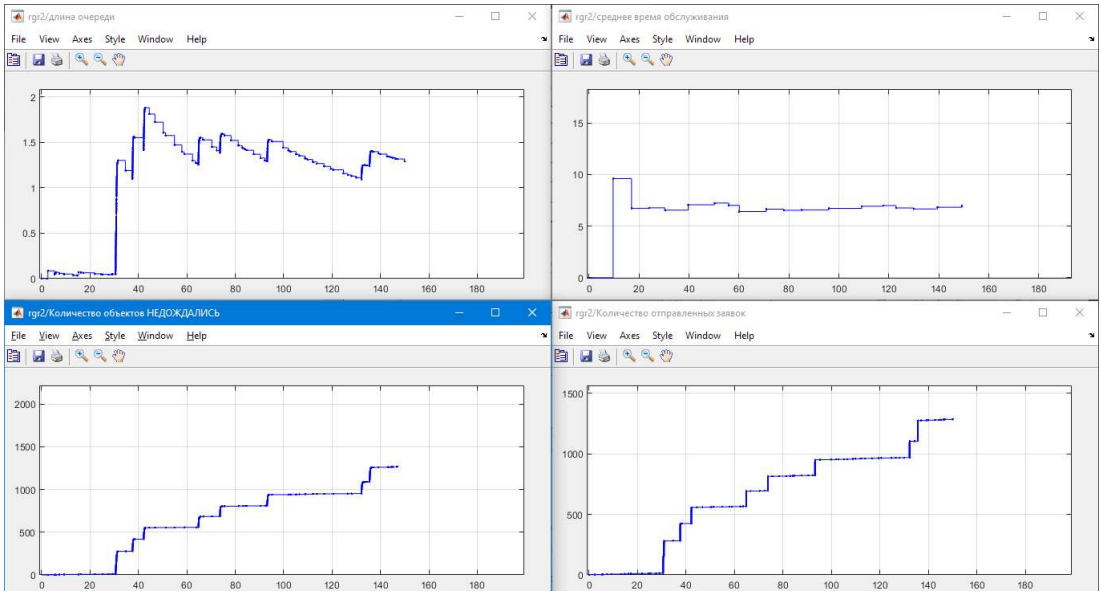


Рисунок 9.4–Графики моделирования работы одноканальной СМО

9.6 Отчет по работе

Отчет по работе должен содержать:

- блок-схему моделирующего алгоритма СМО;
- блок-диаграмму имитационной модели, которая должна быть снабжена всеми необходимыми комментариями; названия подсистем должны правильно отражать процесс;
- результаты имитационного моделирования процесса функционирования системы для всех вариантов условий завершения работы системы; комментарии к работе системы с указанием всех выбранных параметров;
- временные диаграммы процесса обслуживания.

Таблица 9.1 –Законы распределения

Вари-ант	Законы генерирования		
	Длины сообщения	Интервалов поступления сообщений от источника	Длительности записи на диск
1	нормальный	равномерный	экспоненциальный
2	равномерный	нормальный	экспоненциальный
3	экспоненциальный	равномерный	нормальный
4	нормальный	экспоненциальный	экспоненциальный
5	экспоненциальный	нормальный	равномерный
6	равномерный	экспоненциальный	нормальный
7	нормальный	экспоненциальный	равномерный
8	равномерный	нормальный	экспоненциальный
9	экспоненциальный	равномерный	нормальный
10	нормальный	равномерный	экспоненциальный
11	экспоненциальный	нормальный	равномерный
12	нормальный	равномерный	экспоненциальный

Таблица 9.2 –Параметры моделирования

№	A	B	Mx	Sx	L	T	N	f1	f2	f3
1	2	10	1.5	2.1	0.15	60	4	1	2	3
2	10	30	2.5	2.0	0.07	40	4	3	2	1
3	5	15	1.2	1.5	0.12	20	5	2	3	1
4	15	40	2.7	2.3	0.24	85	6	1	3	2
5	2	10	1.2	1.7	0.07	20	7	3	2	1
6	10	30	2.7	3.0	0.15	90	6	2	1	3
7	5	20	1.5	2.1	0.28	80	5	1	2	3
8	12	25	1.4	1.8	0.18	75	5	2	3	1
9	4	10	1.6	2.5	0.31	50	4	3	2	1
10	3	12	1.9	1.25	0.61	80	6	2	1	3
11	4	14	1.3	1.3	0.05	90	7	1	3	2
12	2	10	1.5	2.4	0.19	60	5	3	1	2
13	5	12	2.0	3.1	0.07	55	6	2	3	1
14	5	15	1.2	1.6	0.12	45	7	1	2	3
15	6	8	0.2	3.2	0.21	70	8	3	2	1
16	7	10	1.8	1.4	0.32	80	3	2	1	3
17	8	14	2.2	1.8	0.45	60	4	1	3	2
18	9	15	3.1	2.1	0.5	75	6	2	3	1
19	5	10	2.7	1.5	0.9	50	5	1	2	3
20	4	14	1.5	0.8	0.4	40	8	3	2	1

Здесь:

A, B - параметры равномерного распределения;

Mx, Sx – математическое ожидание и дисперсия нормального распределения;

L – параметр экспоненциального распределения;

T - интервал моделирования;

N - количество реализаций моделирующего алгоритма;

1 - нормальный закон распределения;

2 - равномерный закон распределения;

3 - экспоненциальный закон распределения.

9.7 Контрольные вопросы

9.7.1 Что называется случайной величиной?

9.7.2 Что определяет параметр экспоненциального распределения, параметры равномерного и нормального законов распределения? Обоснуйте выбор параметров при моделировании.

9.7.3. Что такое имитационная модель?

9.7.4. Дайте определение системы массового обслуживания.

9.7.5. В чем отличие одноканальной СМО от многоканальной?

9.7.6. При каких условиях может завершиться работа рассматриваемой системы массового обслуживания?

9.7.7 Перечислите основные показатели эффективности функционирования систем массового обслуживания.

9.7.8 Объясните назначение блока *SimEvent*.

Список литературы

1. Hunt Brian R., A Guide to Matlab : for Beginners and Experienced Users: updated for Matlab 8 and Simulink 8 / R. Hunt Brian , L. Lipsman Roland , M. Rosenberg Jonathan; All of the University of Maryland, College Park. - 3-edition. - United Kingdom : Cambridge University Press, 2014. - 317p
2. Matlab : Офиц. учебный курс Кембриджского университета / пер.с англ. - М. : Триумф, 2008. - 352с
3. Васильев А.Н., Matlab. Практический подход : Самоучитель / А.Н. Васильев. - СПб. : Наука и Техника, 2012. - 448с. - (Самоучитель)
4. Гайдук А.Р., Теория автоматического управления в примерах и задачах с решениями в MATLAB / А.Р. Гайдук, В.Е. Беляев, Т.А. Пьявченко. - 2-е изд.испр. - М. : Горячая линия-Телеком, 2011. - 464с
5. Дьяконов В.П., MATLAB и SIMULINK для радиоинженеров / В.П. Дьяконов. - М. : ДМК Пресс, 2013. - 976с: ил
6. Кетков Ю., Matlab 7: пргораммирование,численные методы / Ю. Кетков , А. Кетков , М. Шульц. - СПб. : БХВ-Петербург, 2005. - 742с
7. Кудинов Ю.И., Теория автоматического управления (с использованием MATLAB-SIMULINK) : учеб.пособие / Ю.И. Кудинов, Ф.Ф. Пащенко. - СПб. : Лань, 2016. - 256с: ил. - (Учебники для вузов.Специальная литература)
8. Мансурова М.Е., Основы программирования в Matlab : учеб.пособие / М.Е. Мансурова, К. Дуйсебекова; КазНУ им.Аль-Фараби. - Алматы : Қазақ университеті, 2010. - 149с
9. Шампайн Л.Ф., Решение обыкновенных дифференциальных уравнений с использованием Matlab : учеб.пособие / Л.Ф. Шампайн, И. Гладвел , С. Томпсон; пер.с англ.И.А.Макарова. - СПб. : Лань, 2009. - 304с. - (Учебники для вузов.Специальная литература)
10. http://matlab.exponenta.ru/simulink/book2/15.php

Основной план 2019 г., поз. 109

Лида Куандыковна Ибраева
Лауласын Косылгановна Абжанова
Азамат Замирович Ильясов

ПРОГРАММНЫЕ СРЕДСТВА СИСТЕМ АВТОМАТИЗАЦИИ

Методические указания по выполнению лабораторных работ
для студентов специальности 5В070200 – Автоматизация и управление

Редактор Л.Т. Сластихина
Специалист по стандартизации Г.И.Мухаметсариева

Подписано в печать __. __. __.
Тираж *30* экз.
Объем *3* уч.-изд. л.

Формат 60х84 1/16
Бумага типографская №1
Заказ _____. Цена ___ тг.

Копировально-множительное бюро
некоммерческого акционерного общества
«Алматинский университет энергетики и связи»
050013 Алматы, ул. Байтурсынова, 126