4. Программирование в MATLAB.

4.1. Скрипты. Функции. Ввод-вывод данных

```
edit MyFile
```

Замечания по формированию М-файлов:

1) Хорошим тоном считается указывать комментарий перед кодом (%), поскольку содержимое этого комментария отображается в левой нижней панели главного окна MATLAB, а также демонстрируется при вызове команды help MyFunction.
2) Если в М-файл записывается набор команд MATLAB (скрипт), то имя этого файла может быть любым, а первыми командами обычно идут clear (очистка workspace) и clc (очистка экрана главного окна).
3) Если набор команд представляется в виде функции, то имя файла должно совпадать с именем создаваемой функции, а команды желательно заканчивать символом точки с запятой:

```
function z = MyCubeSumm(x, y)
% Вычисление суммы кубов
z = x.^3+y.^3;
```

```
help MyCubeSumm
MyCubeSumm(2, 3)
[x y] = meshgrid([0 : 10]);
surf(x, y, MyCubeSumm(x, y))
```

Если функции возвращает набор результатов, используется следующий синтаксис:

```
function [C, fn, fk] = MyCombination(n,k)
% возвращает число сочетаний из n по k, и факториалы n! и k!
if (k <= n) && (n >= 0) && (k >= 0) % запрет векторных операций!
    fn = factorial(n);
    fk = factorial(k);
    C = fn/fk/factorial(n-k);
else
    error('Ошибка в параметрах функции!')
    C=0; fn=0; fk=0;
end
```

```
[Z a b] = MyCombination(8,3)
[Z a b] = MyCombination(8,9)
```

Пример «специфической» функции с заранее неизвестным числом аргументов (используется макроопределение (специальный массив) *varargin*)

```
function MyShowLine(varargin)
% построение линии по набору произвольного числа точек [x y]
for k = 1:length(varargin)
    x(k) = varargin{k}(1); % переменный размер массива – низкая скорость
    y(k) = varargin{k}(2);
end
plot (x,y)
```

```
MyShowLine([1 1], [2 5], [5 4], [3 2], [1 1])
```

Формирование строки ввода данных:

```
x = input('Введите параметр: ');
S = input('Введите формулу : ','s'); eval(S)
```

Дополнительные возможности для корректного вывода информации из скриптов (функций):

```
disp('Текстовая строка') % и нет никакого ans=
v=5; disp(['промежуточный результат равен ' num2str(v)])
warning('Предупреждение!')
```

В одном М-файле может быть несколько функций, одна – главная, из неё вызываются другие.

Создание «анонимных» функций

```
MySqr = @(x) x.^2;
MySqr(y(:, 1))
plot(y(:, 1),MySqr(y(:, 1)),'b-o')
```

4.2. Ветвления, циклы, управляющие инструкции

Ветвления:

IF с несколькими условиями:

```
if i > j
    a(i, j) = i+j;
elseif i == j
    a(i, j) = 1;
else
    a(i, j) = i-j;
end
```

Задание: написать скрипт, запрашивающий размер квадратной матрицы и формирующий матрицу в соответствии с указанными выше условиями.

Оператор switch:

```
switch x
case 0
  y = sin(x);
case 1
  y = cos(x);
otherwise
  y = tan(x);
end
```

Циклы:

```
p = [-5 -1 4 9 20]
```

```
for k = p; exp(k), end
%
e=10; k=1;
while e>1e-6; e=e/2; k=k+1; end
k
%
break % принудительный выход из цикла
```

Задание: создать М-функцию для вычисления в цикле суммы бесконечной геометрической прогрессии $\sum_{k=0}^{\infty} q^k$ с заданной точностью ε и значением q, которые следует задать как параметры. Для параметра q проверить условие сходимости ряда. Выходные данные функции – сосчитанная сумма и число выполненных итераций.

Управляющие инструкции:

```
disp('Для продолжения нажмите любую клавишу...')
pause % приостановка программы до нажатия клавиши
%
return – выход из какого-либо места функции (скрипта)
```

4.3. Создание GUI

Для создания пользовательских приложений (программ), использующих интерактивные графические элементы управления (кнопки, списки выбора, скроллинг, области ввода-вывода, меню) лучше всего использовать конструктор GUIDE. Для создания нового GUI-приложения следует выбрать в меню *New* : *Graphical User Interface*.

```
guide
```

Для хранения GUI-приложения используются два файла с одинаковом именем и расширениями .fig и .m. М-файл служит для добавления программного кода в стиле разработки программ, управляемых событиями. Редактировать следует содержание функций, имеющих окончание имени callback, а начало имени – как в поле *Tag* редактора свойств соответствующего графического элемента.

Пример: с помощью мастера создадим чистое (blank) GUI-приложение, разместим на нём элемент Axis и элемент Push Button (pushbutton1). Изменим имя кнопки на «Построить график». Добавим к кнопке код:

```
x = -5:0.1:5;
y = sin(x)./x;
plot(x,y);
```

Далее, добавим на свободное поле окна вторую кнопку (pushbutton2), назовём её «Очистить». Добавим к кнопке код

```
cla
```

Поработаем над условиями доступа к созданной кнопке. Установим в окне свойств для pushbutton2 свойство *Enable* в Off. Добавим в конец кода для pushbutton1:

```
set(handles.pushbutton2, 'Enable', 'on');
```

а для pushbutton2:

```
set(hObject, 'Enable', 'off');
```

Сохраним приложение под уникальным именем. Для запуска приложения достаточно набрать это имя в командной строке.

В развитие темы следует разместить два элемента Checkbox на свободном поле окна программы, и назвать их «Сетка по X» и «Сетка по Y». В программый код для кнопки pushbutton1 следует внести следующие изменения:

```
x = -5:0.1:5;
y = sin(x)./x;
plot(x,y);
if get(handles.checkbox1, 'Value')
   set(gca,'XGrid', 'on')
else
   set(gca,'XGrid', 'off')
end
if get(handles.checkbox2, 'Value')
   set(gca,'YGrid', 'on')
else
   set(gca,'YGrid', 'off')
end
set(handles.pushbutton2, 'Enable', 'on');
```