Лабораторная работа №2

Программирование в системе MATLAB

Цель работы: Знакомство с основами программирования в MATLAB, основными управляющими конструкциями в MATLAB. Получение навыков организации последовательности, ветвления и цикла в MATLAB.

Наряду с работой в командной строке, существует еще один способ выполнения команд – написание программ. Программа – это предписание ЭВМ на языке программирования, позволяющее решить требуемую задачу. В системе MATLAB программы записываются в файл с расширением «.m». Так, если в командной строке обычно реализуют последовательную структуру записи команд, то с помощью m-файлов возможна реализация всех управляющих структур структурного программирования*.

М-файлы представляют собой обычные текстовые файлы и для их написания можно использовать любой текстовый редактор.MATLAB имеет встроенный текстовый редактор.

Каждый язык программирования обладает своим синтаксисом – набором правил написания программ и построения конструкций языка. Особенности синтаксиса и семантики (значения) языка программирования MATLAB рассмотрим ниже.

Основы работы с m-файлами

Для создания нового m-файла программы (в системе MATLABпрограммы-сценарии называют «скриптами».) в редакторе MATLAB необходимо нажать кнопку «NewScript» в верхнем левом углу. После нажатия данной кнопки появится окно текстового редактора (рисунок 1).

** Теорема о структурном программировании: Любая программа, заданная в виде блок-схемы, может быть представлена с помощью трех управляющих структур:последовательность, ветвление и цикл.*

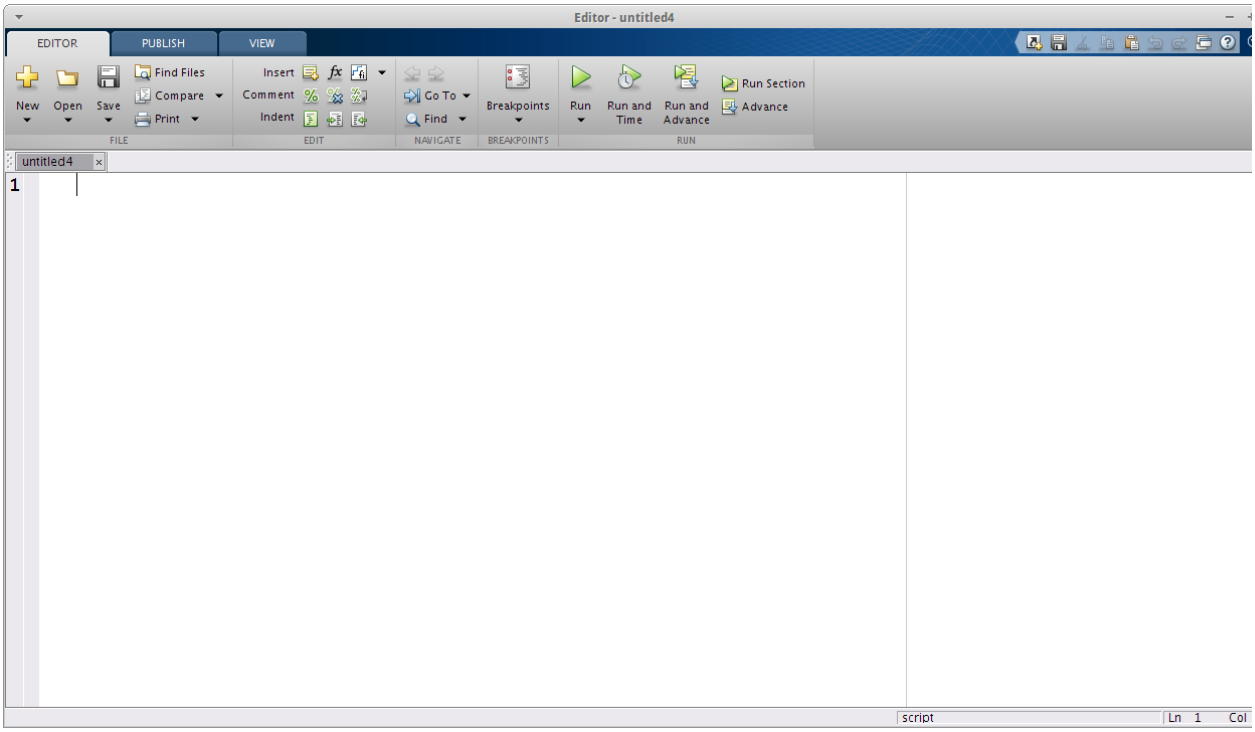


Рисунок 1. Редактор кода MATLAB

Создадим небольшую программу:

```
disp('HelloWorld!')
```

Теперь необходимо сохранить данныйскрипт, для чего необходимо нажать на кнопку «Save» в левом верхнем углу. MATLAB предложит сохранить файл в текущей директории («CurrentFolder»). Дадим скрипту имя «Example1.m» и сохраним его. Желательно, чтобы скрипт был сохранен в текущей директории, так MATLAB будет искать скрипты для запуска в папках, которые перечислены во внутренней переменной MATLAB `path`. Также в этот список входит и текущая директория, отображаемая в среде MATLAB справа в специальной плавающей области. По умолчанию программа будет сохранена в текущий каталог. Запустить программу на выполнение можно, введя ее имя (имя m-файла, в который она была сохранена) в командной строке и нажав «Enter», или же нажав кнопку «Run» в редакторе кода. В обоих случаях в командном окне отобразится следующий вывод:

```
>>Example1
HelloWorld!
```

Рассмотрим другой пример. Имеется программа:

```
x = 0:0.02:2*pi;
a = 0.3;
b = 0.7;
```

```
y = a * sin(x) + b * cos(x);
plot(x, y)
```

Сохраним в файл Example2.m и запустим на выполнение. В результате MATLAB посчитает и выведет на экран график функции $y = a \cdot \sin(x) + b \cdot \cos(x)$.Следует отметить, что после выполнения программы в окне «Workspace» появились переменные `x`, `a`, `b` и `y`. Для пояснениярассмотрим понятие рабочей области.

Базовая рабочая область в MATLAB

Рабочая область системы MATLAB — это область памяти, в которой размещены переменные системы. Переменные, которые вводятся из командной строки или которые создаются в результате выполнения скриптов, вызываемых из командной строки, хранятся в baseworkspace – базовой рабочей области. Все переменные в рабочей области существуют в ней с момента их объявления при работе с данной рабочей областью и до явного их удаления с помощью команды `clear` или до конца действия данной рабочей области (например, для базовой рабочей области - это закрытие MATLAB).

При запуске скрипта не создается новой рабочей области. Программа работает с рабочей областью, из которой она была вызвана. При вызове скрипта из командного окна, работа ведется с базовой рабочей областью, поэтому программе доступны все переменные, созданные до вызова скрипта. Так же если скрипт создаст новые переменные, они останутся доступными и после его завершения.

Управляющие структуры в MATLAB

Как отмечалось ранее, согласно теореме о структурном программировании, любая программа, заданная в виде блок-схемы, может быть представлена с помощью трех управляющих структур: последовательность, ветвление и цикл. Последовательность – однократное выполнение операций в том порядке, в котором они записаны в тексте программы. С организацией ветвления и цикла в MATLAB познакомимся ниже.

Организация ветвления в MATLAB

Ветвление используют в случае, когда необходимо выполнить действие или последовательность действий в зависимости от некоторого условия (условий). Язык

MATLAB предоставляет следующие операторывветвления (или иначе – структуры выбора):

- **if** <выражение> <инструкции> **end**– оператор с единственным выбором;
- **if** <выражение_1> <инструкции_1>**else**<инструкции_2>**end** – оператор с двойным выбором
- **if** <выражение_1> <инструкции_1> **elseif** <выражение_2> <инструкции_2> **else** <инструкции_3> **end** – оператор с тройным и более выбором;
- **switch** <переменная> **case** <значение переменной> <инструкции_1> otherwise <инструкции_2> **end** – оператор с множественным выбором.

Блок-схемы алгоритмов и примеры использования операторов ветвления представлены в таблице 1.

Условный оператор if

Оператор if<выражение_1><инструкции_1> **elseif** <выражение_2> <инструкции_2> **else** <инструкции_3> **end.** В операторе вычисляется некоторое выражение_1, стоящее после ключевого слова **if**. Если значение выражения_1 истинно (true, 1) - выполняется соответствующая группа инструкций, а затем продолжается выполнение программы в строке после end. Если выражение_1, стоящее после ключевого слова **if, ложно** (false, 0), будет вычислено выражение_2 после ключевого слова elseif. Если оно окажется истинным, будут выполнены соответствующие инструкции_2, если и оно окажется ложным – будут выполнены инструкции_3, стоящие после ключевого слова else. Оператор else не содержит вычисляемого выражения. Инструкции, связанные с ним, выполняются, если предшествующий оператор if (и, возможно, elseif) ложны. Обязательными элементами условного оператора **if** являются только операторы if и end. Любые из операторов elseif и else могут отсутствовать. Так же оператор elseif может многократно использоваться внутри оператора условия if.

Условный оператор `switch` **(оператор переключения)**

Оператор **`switch`** `<переменная>` **`case`** `<значение переменной> <инструкции_1> otherwise <инструкции_2>` **`end`** работает, сравнивая значение переменной или вычисленного выражения, стоящего после ключевого слова `switch`, со значениями групп `case`.

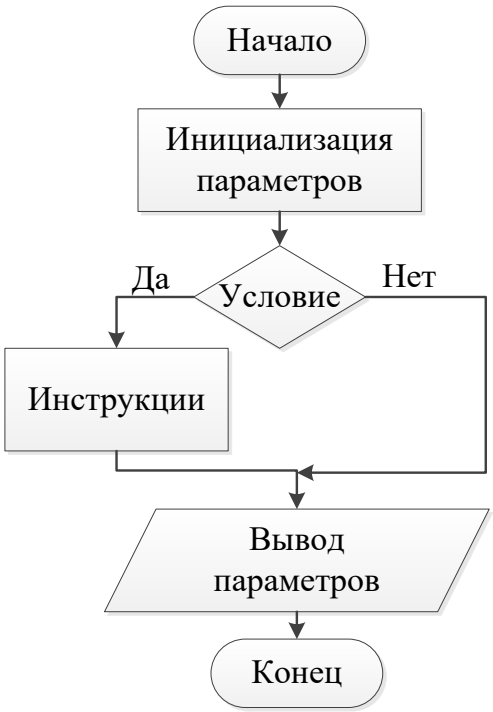
Оператор переключения включает:

- Заголовок `switch`, за которым следует вычисляемое выражение или переменная (только скаляр или строка)
- Произвольное количество групп `case`; заголовок группы состоит из слова `case`, за которым в той же строке следует возможное значение выражения или переменной, указанной в `switch`. Последующие строки содержат инструкции, которые выполняются, если значение выражения или переменной совпадает со значением, указанным после ключевого слова `case`. Выполнение продолжается до тех пор, пока не встретится следующий оператор `case`или оператор `otherwise`. На этом выполнение блока `switch` завершается.
- Группа `otherwise`. Заголовок включает только слово `otherwise`, начиная со следующей строки размещаются инструкции, которые выполняются, если значение выражения оказалось не обработанным ни одной из групп `case`. Выполнение завершается оператором `end`.
- Оператор `end` является последним в блоке переключателя.

Таблица 1. Организация ветвления в MATLAB

Синтаксис	Блок-схема	Пример
`if``<выражение><инструкции>`**`end`**		

```
if<выражение>
<инструкции>
end
```

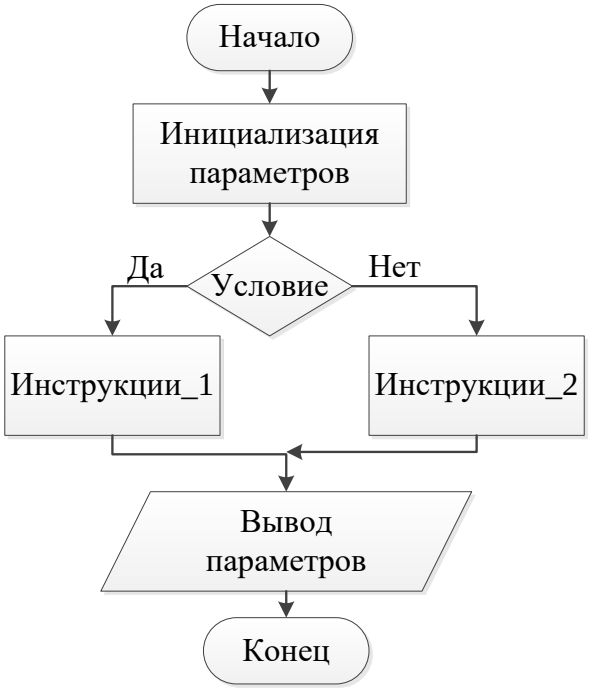



```
% Сравнение двух чисел

a = 1;
b = 3;
ifa>b
   disp('абольшеb');
end
if a < b
   disp('a меньше b');
end
if a == b
    disp('a равно b');
end
```

if<выражение_1><инструкции_1>**else**<инструкции_2>**end**

```
if<выражение>
<инструкции_1>
else
<инструкции_2>
end
```

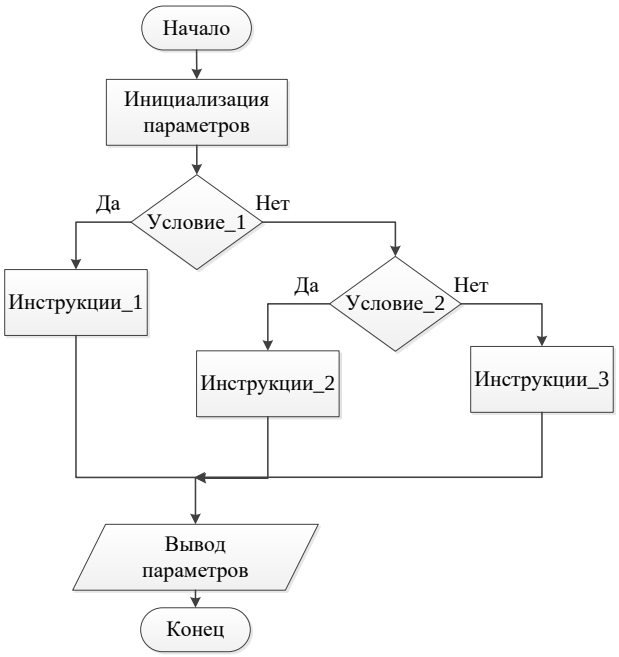



```
% Опеделение равенства
двух чисел

a = 1;
b = 3;
if a == b
   disp('a равно b');
else
   disp('a неравно b');
end
```

if<выражение_1><инструкции_1>**elseif**<выражение_2><инструкции_2>**else**<инструкции_3>**end**

```
if<выражение_1>
<инструкции_1>
elseif<выражение_2>
<инструкции_2>
else
<инструкции_2>
end
```

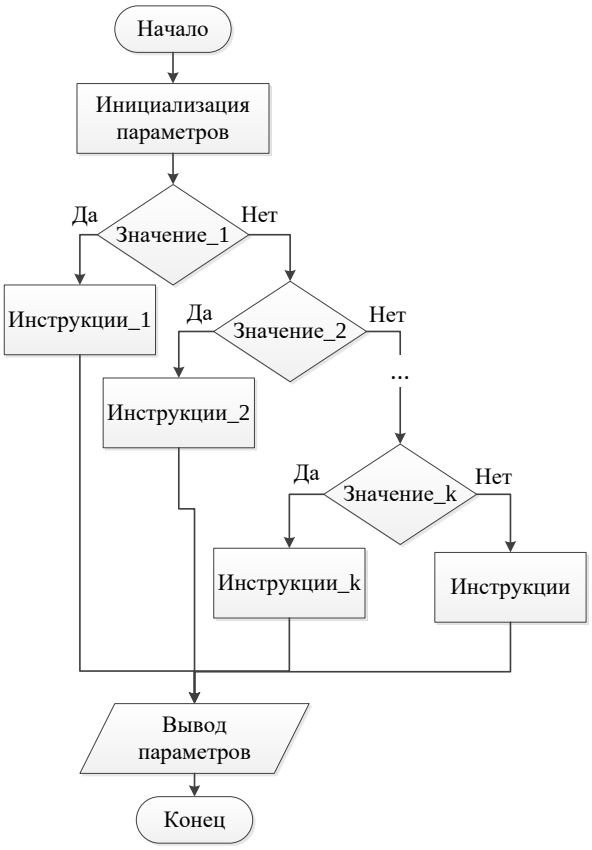



```
% Сравнение двух чисел

a = 1;
b = 3;
if a > b
  disp('a больше b');
elseif a < b
  disp('a меньше b');
else
  disp('a равно b');
end
```

switch..case..end

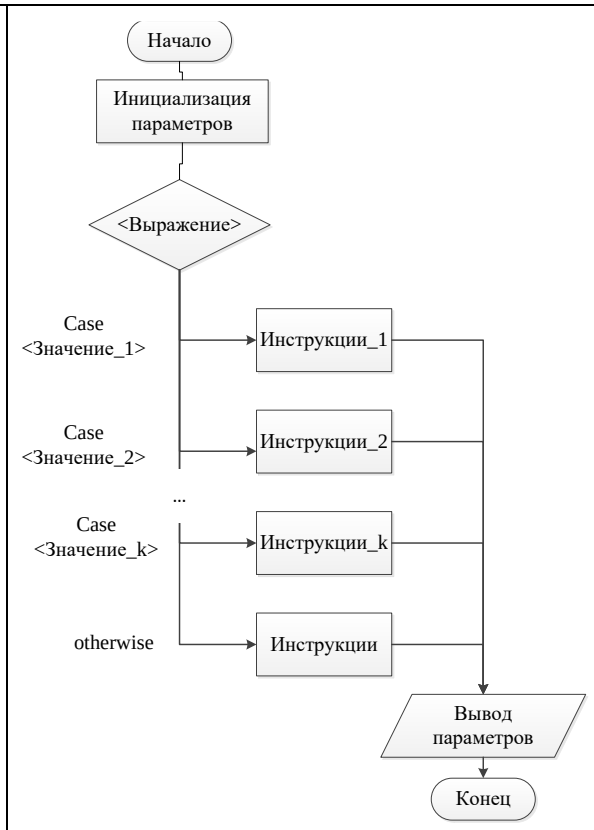
```
switch<выражение>
% скаляр или строка
case<значение_1>
<инструкции_1>
case<значение_2>
<инструкции_2>
. . . . .
case<значение_k>
<инструкции_k>
otherwise
<инструкции>
end
```



ИЛИ

```
% Арифметические операции

a = 1;
b = 3;
operation=input('Введите
арифметическую
операцию:','s');
switch operation
      case '+'
        c=a+b;
      case '-'
        c=a-b;
      case '*'
        c=a*b;
      case '/'
        c=a/b;
      otherwise
        disp('Введена
неопределенная
арифметическая операция')
end
```



Организация циклов в MATLAB

В язык MATLAB включены два оператора цикла: `for` и `while`. Оператор цикла `for` используют в том случае, когда заранее известно количество повторений. Оператор цикла `while` применяется, когда необходимо выполнить повторение, если некоторое условие истинно.

Оператор цикла `for` (оператор цикла с определенным числом операций)

Оператор цикла `for` выполняет инструкцию или группу инструкций предопределенное число раз. После ключевого слова `for` указывается индекс –счетчик цикла и диапазон его значений. Для положительных индексов выполнение цикла завершается, когда значение индекса превышает `<конечное значение>;` для отрицательных приращений выполнение завершается, когда индекс становится меньше чем `<конечное значение>.`

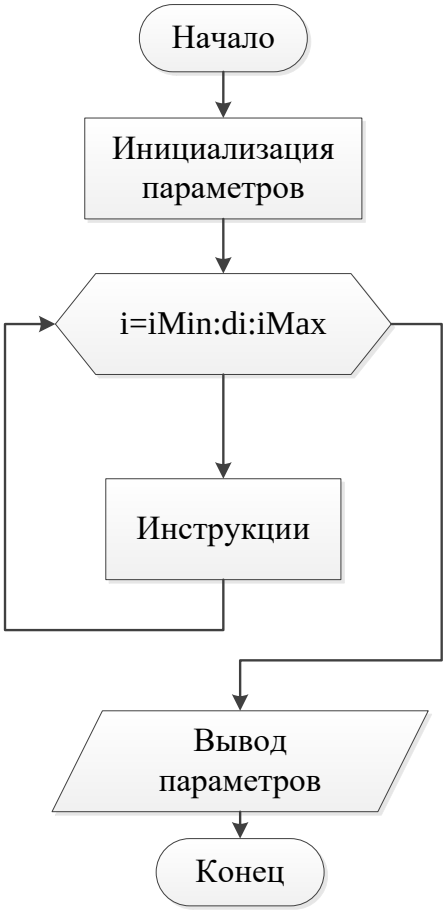


Рисунок 2. Блок-схема алгоритма цикла for..end

Оператор цикла `for..end` имеет следующий синтаксис:

```
for индекс = начальное_значение : шаг : конечное_значение
  операторы_тела_цикла;
end
```

В определении индекса шаг может быть явно не задан, тогда по умолчанию принимается значение шага, равное 1.

В качестве примера рассмотрим программу, которая рассчитывает значение факториала числа `n`, используяоператор цикла `for`.

```
% Вычисление факториала
% n – целое положительное число, факториал которого необходимо
% найти
n = 10;
res = 1;
for i = 1 : n
  res = res * i;
end
```

Оператор цикла `while`

(оператор цикла с неопределенным числом операций)

Оператора цикла `while` работает таким образом, что инструкции, расположенные в теле оператора, начинают выполняться и повторно выполняться только в том случае, если некоторое условие истинно. Как только условие становится ложным, происходит выход из оператора. Таким образом, оператор цикла `while` можно использовать, когда заранее неизвестно число повторений.

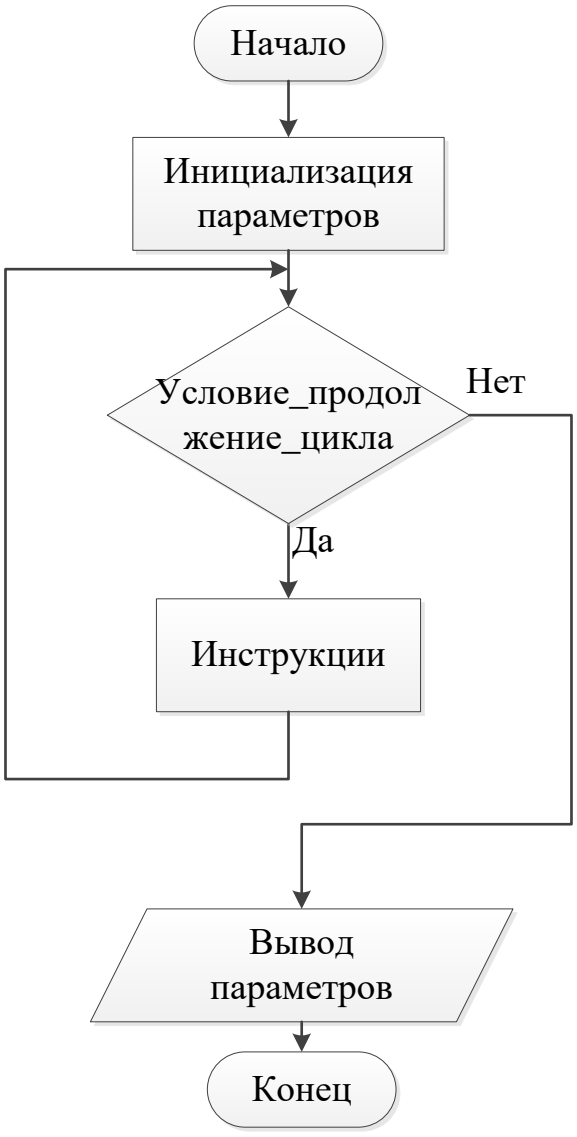


Рисунок 3. Блок-схема алгоритма цикла while..end

Оператор `while..end` имеет следующий синтаксис:

```
while условие_продолжение_цикла
    операторы_тела_цикла;
end
```

Сначала проверяется `условие_продолжение_цикла`, если оно ложно, то цикл заканчивается и выполняются операторы за циклом, т.е. стоящие после ключевого слова `end`. Если `условие_продолжение_цикла` оказалось истинным, то выполняется тело цикла, и потом снова проверяется `условие_продолжение_цикла`, и цикл будет продолжаться пока оно истинно.

Для примера вычислим наибольший общий делитель двух чисел с помощью цикла цикла `while`.

```
% вычисление наибольшего общего делителя чисел a и b
a = 78;
b = 66;
while a ~= b
   if a > b
        a = a - b;
   else
        b = b - a;
   end
end
% сохраним результат в переменнуюNODи не ставим «;»
NOD = a
```

Для примера работы данной программы вызовем ее из консоли MATLAB:

```
>> myNOD
NOD =
6
```

Следует отметить, что как было показано в примере выше, все описанные операторы ветвления и цикла можно вкладывать друг в друга.

Операторыуправления потоками

`break`, `continue`, `pause`

В циклах foriwhileчасто используют операторы, влияющие на их выполнение.

Для досрочного прерывания цикла в MATLABиспользуютоператор `break`. Для перехода к следующей итерации цикла без завершения текущей используют оператор `continue`.Оператор `pause` используют для приостановки выполнения программы (паузы), после нажатия любой клавиши выполнение продолжается.

```
for i = 1:10
  if i==3
    continue
  end
  if i==7
     break
  end
end
```

```
>>Exemple
i =
     1
i =
     2
i =
     4
i =
5
i =
     6
i =
     7
```

Как видно из примера программы `Exemple.m`, на каждой итерации цикла `for`в консоль выводится значение счетчика цикла `i`. Если идет 3-я итерация цикла, т.е. `i == 3`, значение счетчика в консоль не выводится, т.е. все операции, стоящие после команды `continue`, опускаются и цикл выполняет следующую итерацию. Если значение счетчика цикла `i == 7`, в консоль будет выведено его значение, но выполнение цикла прервется командой `break`.

Задания на лабораторную работу№2

В соответствии с номером варианта выбрать кусочно заданную функцию и реализовать программу в MATLAB, которая будет строить ее график (табл. 3).

Для вычисления интеграла воспользоваться методом прямоугольников. Шаг интегрирования задан в варианте. Суммирование ряда выполнять до тех пор, пока модуль очередного члена не будет меньше заранее заданного в варианте числа ε. При выполнении работы не использовать встроенные функции MATLAB для численного интегрирования и расчета факториала. При написании программы оптимизировать вычисления суммы ряда, используя для расчета текущего члена ряда ранее посчитанное значение предыдущего члена.

Текст программы сопроводить комментариями (см. приложение).

Составить блок-схему алгоритма программы.

Пример выполнения лабораторной работы

Дана функция:

$$y(x)=\begin{cases} 1, & -0,5 \le x < 0, \\ \int\limits_{-1}^{x} 3x^2 dx, & 0 \le x < 1 \\ 1+\sum\limits_{n=0}^{\infty} \dfrac{(x-1)^n}{n!}, & 1 \le x \le 2. \end{cases}$$

Для вычисления определенного интеграла используем метод прямоугольников. Метод прямоугольников заключается в следующем: т.к. определенный интеграл функции является площадью криволинейной трапеции, ограниченной функцией, осью абсцисс и двумя прямыми, параллельными оси ординат (границы интегрирования), то приближенно можно рассчитать данную площадь, разбив ее на некоторое количество прямоугольников (рисунок 4). Тогда для метода правых прямоугольников определенный интеграл будет

рассчитываться согласно формулы:

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} f(x_i)(x_i - x_{i-1}).$$

Для метода левых прямоугольников:

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n-1} f(x_i)(x_{i+1} - x_i).$$

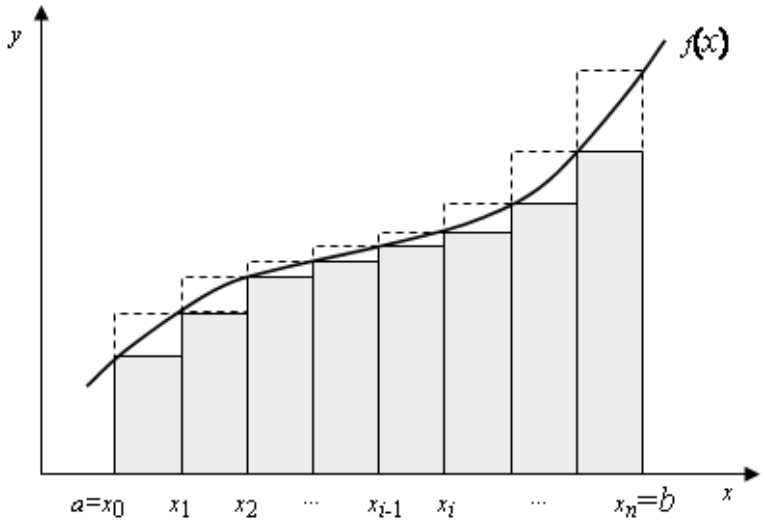


Рисунок 4. Вычисление определенного интеграла методом прямоугольников

Рисунок 5. Блок-схема алгоритма вычисления определенного интеграла методом

правых прямоугольников

Для вычисления суммы ряда использование функций возведения в степень или нахождения факториала (встроенной функции MATLAB) является достаточно ресурсозатратным. Поэтому выведем закономерности для расчета (таблица 2):

Таблица 2

n	$(x-1)^n$	$n!$
0	$(x-1)^0 = 1$	$0! = 1$
1	$(x-1)^1 = (x-1)^0 \cdot (x-1)$	$1! = 0! \cdot 1$
2	$(x-1)^2 = (x-1)^1 \cdot (x-1)$	$2! = 1! \cdot 2$
3	$(x-1)^3 = (x-1)^2 \cdot (x-1)$	$3! = 2! \cdot 3$
…		
i	$(x-1)^i = (x-1)^{i-1} \cdot (x-1)$	$i! = (i-1)! \cdot i$

Так, числитель каждого последующего члена ряда равен предыдущему члену, умноженному на $(x-1)$, а знаменатель каждого последующего члена ряда равен предыдущему члену, умноженному на номер члена ряда.

Иначе, в строгой математической формулировке, аналогичный результат можно получить, разделив выражение текущего члена ряда на предыдущего:

$$\frac{(x-1)^n}{n!} : \frac{(x-1)^{n-1}}{(n-1)!} = \frac{(x-1)^{n-1}(x-1)}{(x-1)^{n-1}} \cdot \frac{(n-1)!}{n(n-1)!} = \frac{(x-1)}{n}.$$

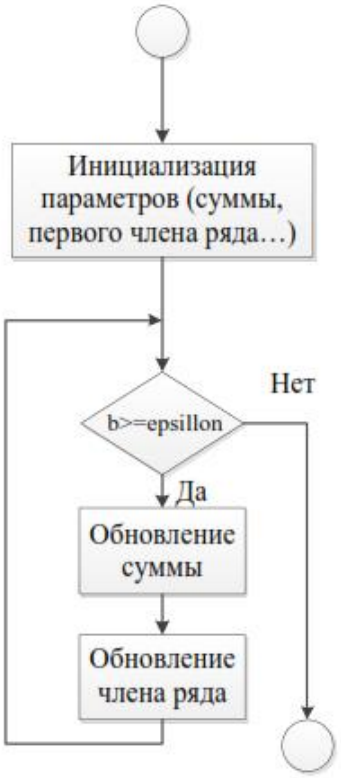


Рисунок 6. Блок-схема алгоритма вычисления суммы сходящегося ряда

В результате выполнения программы должен быть выведен график функции (рисунок 7)

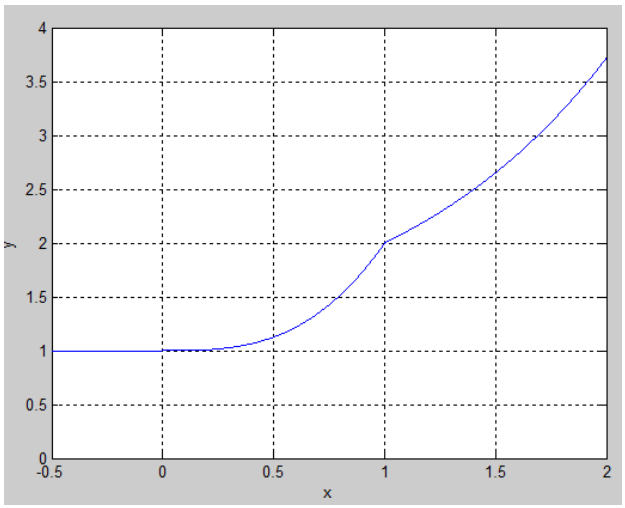


Рисунок 7. График кусочно заданной функции

Таблица 3. Варианты задания на лабораторную работу №3

Номер варианта	Функция	Шаг интегрирования dx	«Точность» ε
1	$y(x)=\begin{cases}-1, & 0\le x<2\\ -1+\int_2^x(2x-1)dx, & 2\le x<3\\ 2+\sum_{n=0}^{\infty}\frac{(x-2)^n}{n!}, & 3\le x\le 4\end{cases}$	0.001	0.0001
2	$y(x)=\begin{cases}1, & 0\le x<2\\ 1+\int_2^x(3x^2-8x)dx, & 2\le x<3\\ -1+\sum_{n=0}^{\infty}\frac{(x-3)^n}{n!}, & 3\le x\le 4\end{cases}$	0.001	0.0001
3	$y(x)=\begin{cases}0, & 0\le x<2\\ \int_2^x(4x^2-5x+1)dx, & 2\le x<3\\ 2+\sum_{n=0}^{\infty}\frac{(x-2)^n}{n!}, & 3\le x\le 4\end{cases}$	0.001	0.0001
4	$y(x)=\begin{cases}0, & -2\le x<0\\ \int_0^x e^{-(x+3)^2}dx, & 0\le x<1\\ -2+\sum_{n=0}^{\infty}\frac{(-1)^n(x-2)^{n+1}}{(n+1)!}, & 1\le x\le 3\end{cases}$	0.0005	0.0001
5	$y(x)=\begin{cases}-1, & -2\le x<0\\ \int_0^x(e^{-(x-3)^2}+x)dx, & 0\le x<1\\ x+\sum_{n=0}^{\infty}\frac{(-1)^n(x-2,2)^{n+1}}{(n+1)!}, & 1\le x\le 3\end{cases}$	0.0005	0.0001

Номер варианта	Функция	Шаг интегрирования dx	«Точность» ε
6	$y(x)=\begin{cases} 3, & -2\le x<0 \\ \int_0^x (e^{-(x-3)^2}-3)dx, & 0\le x<1 \\ -2+\sum_{n=0}^{\infty}\frac{(-1)^n(x-2,5)^{n+1}}{(n+1)!}, & 1\le x\le 3 \end{cases}$	0.0005	0.00005
7	$y(x)=\begin{cases} 0, & -2\le x<-1 \\ \int_{-2}^x (\cos(x)+x\cdot\sin(x))dx, & -1\le x<1 \\ \sum_{n=0}^{\infty}\frac{(-1)^n x^{2n+1}}{(2n+1)!}, & 1\le x\le 2 \end{cases}$	0.0002	0.00005
8	$y(x)=\begin{cases} 1, & -2\le x<-1 \\ \int_{-2}^x (2\cos(x)+x\cdot tg(x))dx, & -1\le x<1 \\ sin(x)+\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{(2n)!}, & 1\le x\le 2 \end{cases}$	0.0002	0.00005
9	$y(x)=\begin{cases} 2, & -2\le x<-1 \\ \int_{-2}^x (\cos(x)+x\cdot\cos(x))dx, & -1\le x<1 \\ -x+\sum_{n=0}^{\infty}\frac{(-1)^n(x+\pi)^{2n+1}}{(2n+1)!}, & 1\le x\le 2 \end{cases}$	0.0002	0.00005
10	$y(x)=\begin{cases} 0, & -3\le x<-2 \\ \int_{-2}^x (\cos(x)+\frac{1}{x})dx, & -2\le x<-1 \\ \sum_{n=0}^{\infty}\frac{(x+1)^{2n+1}}{(2n+1)!}, & -1\le x\le 0 \end{cases}$	0.001	0.00005

Номер варианта	Функция	Шаг интегрирования dx	«Точность» ε
11	$y(x)=\begin{cases} -1, & -3\le x<-2 \\ \int_{-2}^{x}(x+\frac{1}{\sin}(x))dx, & -2\le x<-1 \\ -1+\sum_{n=0}^{\infty}\frac{x^{2n+1}}{(2n)!}, & -1\le x\le 0 \end{cases}$	0.001	0.00002
12	$y(x)=\begin{cases} -2, & -3\le x<-2 \\ \int_{-2}^{x}(x^{\sin(x)}+\frac{1}{x})dx, & -2\le x<-1 \\ \sin(x)+\sum_{n=0}^{\infty}\frac{(x-1)^{2n+1}}{(2n+1)!}, & -1\le x\le 0 \end{cases}$	0.001	0.00002
13	$y(x)=\begin{cases} -1, & -3\le x<-2 \\ \int_{-2}^{x}(x+\frac{1}{\sin}(x))dx, & -2\le x<-1 \\ -1+\sum_{n=0}^{\infty}\frac{x^{2n+1}}{(2n)!}, & -1\le x\le 0 \end{cases}$	0.0005	0.00002
14	$y(x)=\begin{cases} -2, & 0\le x<1 \\ 2+\int_{1}^{x}(\frac{2}{x^3}-\frac{3}{x^2}+\frac{4}{x})dx, & 1\le x<2 \\ x+\sum_{n=0}^{\infty}\frac{(x-1)^n}{n!}, & 2\le x\le 3 \end{cases}$	0.0005	0.00005
15	$y(x)=\begin{cases} -1, & 0\le x<1 \\ 2+\int_{1}^{x}(2x^3-\frac{3}{x^2}+4x)dx, & 1\le x<2 \\ x-\sum_{n=0}^{\infty}\frac{(x-2)^n}{n!}, & 2\le x\le 3 \end{cases}$	0.0005	0.00002

Номер варианта	Функция	Шаг интегрирования dx	«Точность» ε
16	$y(x)=\begin{cases}1, & -2\le x<-1\\ \int_{-2}^{x}(2\cos(x)+x\cdot tg(x))dx, & -1\le x<1\\ sin(x)+\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{(2n)!}, & 1\le x\le 2\end{cases}$	0.0002	0.00005
17	$y(x)=\begin{cases}2, & -2\le x<-1\\ \int_{-2}^{x}(\cos(x)+x\cdot\cos(x))dx, & -1\le x<1\\ -x+\sum_{n=0}^{\infty}\frac{(-1)^n(x+\pi)^{2n+1}}{(2n+1)!}, & 1\le x\le 2\end{cases}$	0.0002	0.00005
18	$y(x)=\begin{cases}0, & -3\le x<-2\\ \int_{-2}^{x}(\cos(x)+\frac{1}{x})dx, & -2\le x<-1\\ \sum_{n=0}^{\infty}\frac{(x+1)^{2n+1}}{(2n+1)!}, & -1\le x\le 0\end{cases}$	0.001	0.00005
19	$y(x)=\begin{cases}-1, & -3\le x<-2\\ \int_{-2}^{x}(x+\frac{1}{\sin}(x))dx, & -2\le x<-1\\ -1+\sum_{n=0}^{\infty}\frac{x^{2n+1}}{(2n)!}, & -1\le x\le 0\end{cases}$	0.001	0.00002
20	$y(x)=\begin{cases}-2, & -3\le x<-2\\ \int_{-2}^{x}(x^{\sin(x)}+\frac{1}{x})dx, & -2\le x<-1\\ \sin(x)+\sum_{n=0}^{\infty}\frac{(x-1)^{2n+1}}{(2n+1)!}, & -1\le x\le 0\end{cases}$	0.001	0.00002

Номер варианта	Функция	Шаг интегрирования dx	«Точность» ε
21	$y(x)=\begin{cases} 2, & -2\le x<-1 \\ \int_{-2}^{x}(\cos(x)+x\cdot\cos(x))dx, & -1\le x<1 \\ -x+\sum_{n=0}^{\infty}\frac{(-1)^n(x+\pi)^{2n+1}}{(2n+1)!}, & 1\le x\le 2 \end{cases}$	0.0005	0.00002
22	$y(x)=\begin{cases} -2, & 0\le x<1 \\ 2+\int_{1}^{x}(\frac{2}{x^3}-\frac{3}{x^2}+\frac{4}{x})dx, & 1\le x<2 \\ x+\sum_{n=0}^{\infty}\frac{(x-1)^n}{n!}, & 2\le x\le 3 \end{cases}$	0.0005	0.00005
23	$y(x)=\begin{cases} -1, & 0\le x<1 \\ 2+\int_{1}^{x}(2x^3-\frac{3}{x^2}+4x)dx, & 1\le x<2 \\ x-\sum_{n=0}^{\infty}\frac{(x-2)^n}{n!}, & 2\le x\le 3 \end{cases}$	0.0005	0.00002
24	$y(x)=\begin{cases} 1, & -2\le x<-1 \\ \int_{-2}^{x}(2\cos(x)+x\cdot tg(x))dx, & -1\le x<1 \\ sin(x)+\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{(2n)!}, & 1\le x\le 2 \end{cases}$	0.0002	0.00005
25	$y(x)=\begin{cases} 2, & -2\le x<-1 \\ \int_{-2}^{x}(\cos(x)+x\cdot\cos(x))dx, & -1\le x<1 \\ -x+\sum_{n=0}^{\infty}\frac{(-1)^n(x+\pi)^{2n+1}}{(2n+1)!}, & 1\le x\le 2 \end{cases}$	0.0002	0.00005

Номер варианта	Функция	Шаг интегрирования dx	«Точность» ε
26	$y(x)=\begin{cases}0, & -3\le x<-2\\ \int_{-2}^{x}(\cos(x)+\frac{1}{x})dx, & -2\le x<-1\\ \sum_{n=0}^{\infty}\frac{(x+1)^{2n+1}}{(2n+1)!}, & -1\le x\le 0\end{cases}$	0.001	0.00005
27	$y(x)=\begin{cases}-1, & -3\le x<-2\\ \int_{-2}^{x}(x+\frac{1}{\sin}(x))dx, & -2\le x<-1\\ -1+\sum_{n=0}^{\infty}\frac{x^{2n+1}}{(2n)!}, & -1\le x\le 0\end{cases}$	0.001	0.00002
28	$y(x)=\begin{cases}-2, & -3\le x<-2\\ \int_{-2}^{x}(x^{\sin(x)}+\frac{1}{x})dx, & -2\le x<-1\\ \sin(x)+\sum_{n=0}^{\infty}\frac{(x-1)^{2n+1}}{(2n+1)!}, & -1\le x\le 0\end{cases}$	0.001	0.00002

Контрольные вопросы

1. Как создать и запустить на выполнение программу в MATLAB?
2. Что такое рабочая область в MATLAB?
3. Для чего предназначены операторы continue и break?
4. Расскажите, как работает оператор if/elseif/else/end.
5. Какие виды циклов Вы знаете? С помощью каких конструкций они реализуются в MATLAB?
6. Какие операции отношения вы знаете?
7. Какие логические операции Вы знаете?
8. Назовите основные элементы блок-схемы алгоритма и их функции.
9. Объясните, как Вы понимаете «метод прямоугольников».
10. Что такое комментирование, как создать комментарий в MATLAB?

Требования к содержанию отчета

Отчет по лабораторной работе оформляется в любом текстовом редакторе и предоставляется в печатном виде. Отчет должен состоять из следующих разделов:

1. Титульный лист. На титульном листе необходимо указать номер и название лабораторной работы, номер варианта, ФИО и группу исполнителя, ФИО преподавателя.
2. Цель работы.
3. Задание на лабораторную работу в соответствии с номером варианта.
4. Ход работы:
   - Блок-схема алгоритма расчета значений кусочно-заданной функции;
   - Листинг программы, реализующей задание в MATLAB;
   - Результат выполнения программы.
5. Выводы по работе.

   К отчету прилагаются:
   - файл с текстом отчета;
   - m-файл с текстом программы

   Файл отчета необходимо разместить в личном кабинете

Требования к оформлению программ на языке MATLAB

Соглашение по именованию идентификаторов

Подбор идентификаторов

А. Все идентификаторы должны выбираться из соображений читаемости и максимальной семантической нагрузки.

Например:

```
dx = 0.001;            % шаг
интегрирования
Area = a*b;             % Площадь прямоугольника
```

Неудачными можно считать

идентификаторы:
```
UU qq = 0.0001;
% шаг интегрирования
kk = a*b;            % Площадь прямоугольника
```

Б. Идентификаторы рекомендуется подбирать из слов английского языка. Например:

```
% размеры изделия (ширина, высота)
Width=4; Height=3;
```

Не очень удачными можно считать идентификаторы:

```
% размеры изделия (ширина, высота)
Shirina=4; Vysota=3;
```

Написание идентификаторов

Существует два основных способа написания идентификаторов.

А. В любых идентификаторах каждое слово, входящее в идентификатор, писать, начиная с большой буквы, остальные буквы – маленькие.

Например:

```
NextX, LastX, GraphErrorMsg
```

Б. В любых идентификаторах каждое слово, входящее в идентификатор, разделять символом "_" , при этом все буквы – маленькие.

Например:

```
next_x, last_x, graph_error_msg
```

Соглашения по самодокументируемости программ

Комментарии

А. Комментарии в теле программы следует писать на русском языке и по существу так, чтобы программист, не участвовавший в разработке программы (но имеющий опыт работы на языке), мог без особого труда разобраться в логике программы, и, при необходимости, сопровождать данный программный продукт.

Б. Для указания начала комментария в MATLAB используется символ «%».

```
% программа, вычисляющая гипотенузу прямоугольного треугольника
% a, b - катеты прямоугольного треугольника
% c - вычисляемая гипотенуза

% промежуточные расчеты
x = a^2 + b^2;

% вычисляем гипотенузу
c = sqrt(x);
```

.Спецификация функций (Документирование функций)

Для каждой пользовательской функции должна быть описана в виде комментария спецификация, содержащая следующую информацию:

- назначение функции;
- описание семантики параметров-значений (параметров, передаваемых по значению); описание семантики параметров-переменных (параметров, передаваемых по ссылке);
- описание семантики возвращаемого значения;
- Если функция реализует какой-либо вычислительный метод (например: нахождение площади фигуры методом трапеций, поиск

минимума функции методом Ньютона и т.п.), рекомендуется в теле функции поместить комментарий с кратким описанием метода, либо ссылку на источник, где описан метод.

Для предоставления справки (спецификации) о программе или функции используются комментарии, расположенные между первой командой и определением функции. Эти строки будут отображены при выполнении команды `help` с именем программы или функции.

```
function [alpha, beta] = swap (alpha, beta)
% Функция меняет значения аргументов
% alpha -- 1е число
% beta -- 2е число
x = alpha;
alpha = beta;
beta = x;
end
```

Для этой функции выполнение команды

```
>> help swap
```

даст следующий результат.

```
Функция меняет значения аргументов
alpha -- 1е число
beta -- 2е число
```

Строка, следующая за строкой определения функции, называется H1 строкой помощи. Именно она будет отображаться первой при выполнении команды `help` и предоставлять обобщающую информацию о функции. Очень важно сделать эту строку как можно более описательной, т.к. она используется для поиска функции по ключевым словам функцией `lookfor`, и окном Function Browser (вызывается по «Shift+F1») для поиска функций.

Спецификация программного файла или модуля (документирование программ)

Программный файл или модуль должен начинаться со спецификации в виде комментария, содержащего следующую информацию:

- идентификация проекта, к которому принадлежит файл;
- назначение (название) и имя файла;
- версия файла; фамилия автора(ов);
- описание модуля;
- история изменений модуля.

После спецификации программного файла рекомендуется поместить комментарий с указаниями по запуску программы и работе с ней (указаниями по использованию модуля другими программистами) или ссылку на источник, который использован при составлении программы (модуля).

Соглашения по читаемости программ

Лесенка

"Лесенка" должна отражать структурную вложенность языковых конструкций. Рекомендуется отступ не менее 2-х и не более 8-и пробелов. Принятого отступа нужно придерживаться во всем тексте программы. Правила написания некоторых конструкций:

```
if ( <условие> )
    <операторы>
else
    <операторы>
end

while ( <условие> )
    <операторы>
end

for индекс = начальное_значение : шаг : конечное_значение
    операторы_тела_цикла;
end

switch ( <выражение> )
    case <выражение>:
        <операторы>;
    ........
    otherwise
        <операторы>;
```

```
end
```

Длина строк программного текста

Длина строк программы не должна превышать ширины экрана (80 символов).

Прочие рекомендации

А. Рекомендуется операнды бинарных операций (+, = и т.п.) отделять от знака операции одним пробелом " ".

Например:

```
Sum = A + B;
```

Б. Рекомендуется при перечислении идентификаторов после запятой "," ставить один пробел " ".

Например:

```
fprintf( 'Сумма: %d; Разность: %d.', A + B, A – B );
```

В. Рекомендуется 16-ричные числа писать большими буквами.

Отладка

Отладкой называется этап разработки компьютерной программы, на котором выявляются и устраняются неточности в ее работе. Чаще всего возникают ошибки двух типов: синтаксические и лексические. Лексические ошибки связаны с неправильным написанием имени функции/переменной/скрипта. Синтаксические ошибки заключаются в неправильном использовании какой-либо конструкции MATLAB.

Отладка программы происходит с использованием особого инструмента -- отладчика (debuger). В MATLAB удобно работать с отладчиком через графический интерфейс.

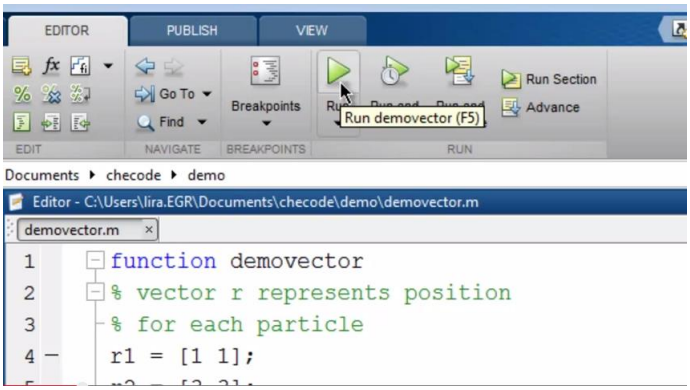


Рисунок 1. Работа с отладчиком

Чтобы начать отладку необходимо запустить программу на выполнение. Для запуска программы можно использовать кнопку «Run» во вкладке меню «Editor». При запуске программы произойдет ее автоматическое сохранение. Файл с текстом программы должен быть предварительно сохранен на диске для того, чтобы иметь возможность запускать и отлаживать программу.

Одним из удобных методов отладки является приостановка выполнения программы в тех местах, которые могут содержать ошибки выполнения. Для того, чтобы остановить выполнение программы, необходимо расставить точки останова в тех местах, которые вас интересуют. Остановив выполнение, вы можете проверять значения переменных, проверять ход выполнение функции пошагово.

Точки останова помещаются на определенную строку кода щелчком мыши по пространству справа от номера строки.

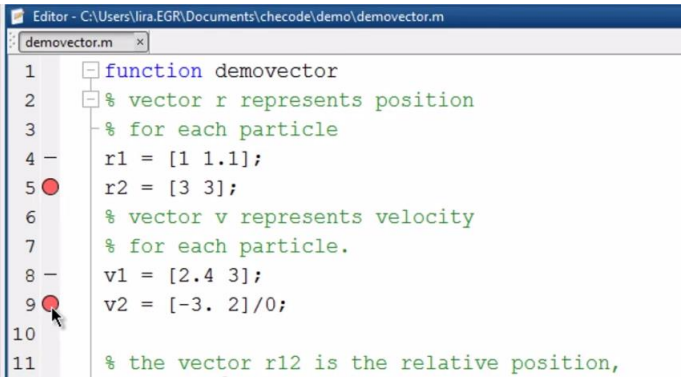


Рисунок 2. Точки останова

Теперь, при запуске программы ее выполнение прекратится на той строчке, где стоит первая точка останова. Продолжить выполнение программы можно несколькими способами:

- выполнять до следующей точки останова, если такая есть, а если нет – то отладчик продолжит свою работу до самого окончания программы;

- выполнять программу пошагово;
- выполнять до позиции курсора.

Команды управления отладкой находятся в секции «debug».

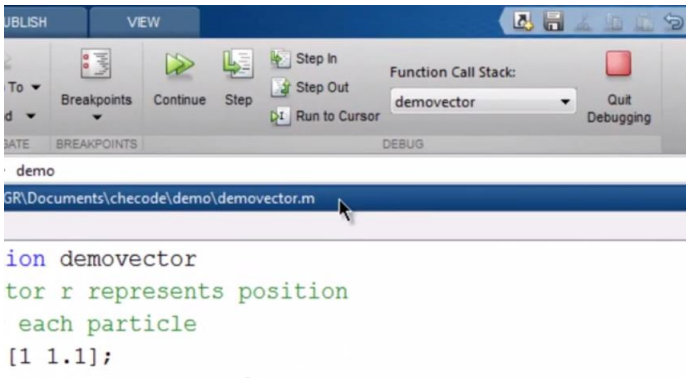


Рисунок 3. Команды отладчика

Каждая команда может быть доступна по горячим клавишам:

Таблица 1. Горячие клавиши режим отладки

Команда	Клавиша
Запуск программы (продолжить выполнение)	F5
Вычислить выделенное выражение и вывести результат в командное окно	F9
Выполнить одну команду (шаг)	F10
Шаг с заходом	F11
Шаг с выходом	F11

Шагом является выполнение инструкции(ий), находящих на текущей строке, обозримой отладчиком. Текущая строка обозначается зеленой стрелкой слева от поля редактирования программы.

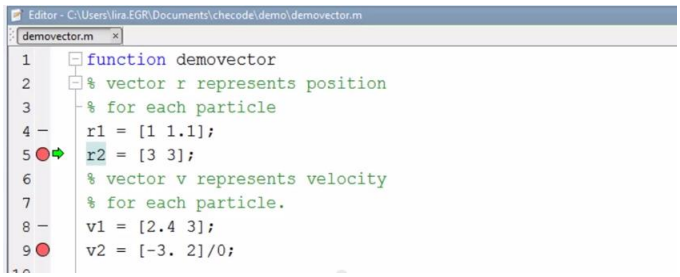


Рисунок 4. Обозримая строка отладчика

Обычный шаг (F10) подразумевает выполнение всех инструкций находящихся на данной строке кода и переход к следующей строке. Шаг с заходом (F11) будет переводить

фокус отладчика внутрь выполняемой инструкции. Это означает, что если инструкция содержит вызов какой-либо функции, то отладчик перейдет на пошаговое выполнение вызываемой функции. Шаг с выходом подразумевает выполнение всех инструкций текущей функции и возврат к вызывающему коду.

Во время отладки можно выполнять команды интерпретатора MATLAB, например, чтобы проверить значения промежуточных вычислений.

Текст ошибок, возникающих во время выполнения функции будет выводиться в командное окно.