# 6. Режим программирования: организация разветвлений и циклов

## 6.1. Цель работы

Изучить средства организации разветвлений и циклов в MATLAB и овладеть навыками их использования при разработке М-файлов.

## 6.2. Краткая теоретическая справка

Для организации разветвлений и циклов в М-файлах используются операторы языка MATLAB, рассматриваемые в следующих разделах.

### 6.2.1. Операторы организации разветвлений

Имеется две основные разновидности разветвлений, реализуемые двумя операторами MATLAB:

1. Разветвление по условию выполняется с помощью оператора `if`, простейший формат которого с одним условием имеет вид:

   ```
   if <условие>
        <фрагмент>
   end
   ```

   где <*фрагмент*> — фрагмент программы.

   *Действие оператора*: если значение <*условия*> "истинно" (выполняется), то управление передается <*фрагменту*>, в противном случае управление передается части программы, следующей за `end`.

   *Условие* представляет собой логическое выражение — простое, с одной операцией отношения (см. табл. 1.7), или более сложное, включающее логические операции (см. табл. 1.8).

   Пример использования оператора `if` с простым условием:

   ```
   if i==j
      a(i,j) = 1;
   end
   ```

   и с более сложным условием:

   ```
   if (i==j)&((i+j)>50)
      a(i,j) = 10;
   end
   ```

Расширенный формат оператора `if` с одним условием имеет вид:

```
if <условие>
     <фрагмент1>
else
     <фрагмент2>
end
```

*Действие оператора*: если значение <*условия*> "истинно", то управление передается <*фрагменту1*>, если значение <*условия*> "ложно", то выполняется <*фрагмент2*>; после этого управление передается части программы, следующей за `end`.

Пример использования оператора `if` расширенного формата:

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

Формат оператора `if` с несколькими условиями имеет вид:

```
if <условие1>
     <фрагмент1>
elseif <условие2>
         <фрагмент2>
...
elseif <условиеN-1>
         <фрагментN-1>
...
else
     <фрагментN>
end
```

*Действие оператора*: если значение <*условия1*> "истинно", то управление передается <*фрагменту1*>, если значение <*условия2*> "истинно", то управление передается <*фрагменту2*> и т. д. вплоть до <*условияN-1*>; если значения всех условий "ложно", то управление передается <*фрагментуN*>; после этого оно передается части программы, следующей за `end`.

Пример использования оператора `if` с несколькими условиями:

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

```
else
   a(i,j) = 0;
end
```

2. Разветвление в зависимости от значения выражения (арифметического, символьного или логического) выполняется с помощью оператора `switch` следующего формата:

```
switch <выражение>
    case <значение1>
          <фрагмент1>
    case <значение2>
          <фрагмент2>
    ...
    otherwise
          <фрагментN>
end
```

*Действие оператора*: в зависимости от значения <*выражения*> управление передается соответствующему <*фрагменту*>; если значение выражения не равно ни одному из указанных, то управление передается <*фрагментуN*> (который может отсутствовать); после этого управление передается части программы, следующей за `end`.

Пример использования оператора `switch`:

```
x = [pi/6 pi/8 pi/16];
a = input('a = ');
b = input('b = ');
switch (a+b)
    case 0
         y = sin(x);
    case 1
         y = cos(x);
    otherwise
         y = tan(x)
end
```

## 6.2.2. Операторы организации циклов

Имеются две основные разновидности циклов, реализуемые двумя операторами MATLAB:

1. Арифметический цикл с заранее известным (фиксированным) числом повторений организуется с помощью оператора `for` одного из следующих форматов:

- с простой переменной цикла:

  **for <переменная> = <нач.значение>:[<шаг>:]<кон.значение>**

  **<тело цикла>**

  **end**

  где:

  <*переменная*> — имя простой переменной цикла;

  <*нач.значение*>, <*кон.значение*>, <*шаг*> — соответственно начальное и конечное значения переменной цикла и шаг ее изменения; если шаг равен 1, то его можно не указывать;

  <*тело цикла*> — повторяющийся фрагмент программы.

  *Действие оператора*: при изменении значений <*переменной*> от <*нач.значения*> до <*кон.значения*> с заданным <*шагом*> повторяется <*тело цикла*>, каждый раз с новым значением <*переменной*>; после этого управление передается части программы, следующей за `end`.

  Пример использования оператора `for` с простой переменной цикла (полужирным шрифтом выделены элементы, вычисляемые в цикле):

```
x = [2 3 5];
for i = 1:3
   x(i) = i^2
end
x =
     1     3     5
x =
     1     4     5
x =
     1     4     9
```

- с переменной цикла — вектором:

  Например:

  **for <переменная> = <вектор>**

  **<тело цикла>**

  **end**

  где <*вектор*> — вектор, как правило, числовой.

  *Действие оператора*: при изменении значений <*переменной*>, которой последовательно присваиваются значения элементов <*вектора*>, повторяется <*тело цикла*>, каждый раз с новым значением <*переменной*>; после этого управление передается части программы, следующей за `end`.

  Пример использования оператора `for` с переменной цикла — вектором:

```
a = [-1 0 15];
```

```
for i = a
   x = i+a
end
x =
    -2    -1    14
x =
    -1     0    15
x =
    14    15    30
```

- с переменной цикла — матрицей:

  **for <*переменная*> = <*матрица*>**

  **<*тело цикла*>**

  **end**

  где <*матрица*> — матрица, как правило, числовая.

  *Действие оператора*: при изменении значений <*переменной*>, которой последовательно присваиваются значения столбцов <*матрицы*>, повторяется <*тело цикла*>, каждый раз с новым значением <*переменной*>; после этого управление передается части программы, следующей за `end`.

  Пример использования оператора `for` с переменной цикла — матрицей:

```
a = [1 2 3;4 5 6;7 8 9];
for i = a
   x = i'
end
x =
     1     4     7
x =
     2     5     8
x =
     3     6     9
```

2. Итерационный цикл с заранее неизвестным (не фиксированным) числом повторений организуется с помощью оператора `while` следующего формата:

   **while <*условие*>**

   **<*тело цикла*>**

   **end**

   где <*условие*> — логическое выражение, в котором хотя бы одна из переменных встречается в <*теле цикла*>.

*Действие оператора*: <*тело цикла*> повторяется до тех пор, пока <*условие*> "истинно", после чего управление передается части программы, следующей за `end`.

Пример использования оператора `while` для вычисления суммы геометрической прогрессии $s = \sum_{n=0}^{\infty}(-0,5)^n$ (`s`) с точностью до $\varepsilon = 10^{-4}$ (`e`) с выводом после выхода из цикла значения суммы и погрешности ее вычисления (вектор `[s e]`):

```
n = 0; s0 = 0; e = 100;
while e>1e-4
   s = s0+(-0.5).^n;
   e = abs(s-s0);
   s0 = s;
   n = n+1;
end
[s e]
ans =
    0.6667    0.0001
```

Принудительный выход из цикла реализуется оператором:

**`break`**

после которого управление передается части программы, следующей за `end`.

## 6.3. Литература


1. Солонина А. И., Арбузов С. М. Цифровая обработка сигналов. Моделирование в MATLAB. — СПб.: БХВ-Петербург, 2008, *гл. 7*.
2. Сергиенко А. Б. Цифровая обработка сигналов. 3-е издание — СПб.: БХВ-Петербург, 2010, *Приложения 1—2*.


## 6.4. Содержание лабораторной работы

Содержание работы связано с изучением средств MATLAB для организации разветвлений и циклов при разработке script-файлов и function-файлов.

## 6.5. Задание на лабораторную работу

*Задание на лабораторную работу* включает в себя следующие пункты:

1. Организация разветвлений с одним условием.

   Создать function-файл `y1` для вычисления функции

$$y_1(x) = \begin{cases} a\sin bx, & \text{если } a \neq 0 \text{ и } b \neq 0; \\ (a+2)x + b, & \text{иначе,} \end{cases} \quad (6.1)$$

где:

аргумент $x$ — вектор фиксированных значений в диапазоне $[-4;\ 4]$ с шагом $\Delta x = 0{,}1$;

$a$, $b$ — произвольные вещественные константы (скаляры).

Вывести график функции $y_1(x)$.

Обратиться к function-файлу `y1` в режиме прямых вычислений для проверки разветвления по условию в (6.1).

Пояснить:

- какой оператор использован для организации разветвления;
- что проверяется при организации разветвления;
- какие параметры function-файла `y1` являются входными и выходными.

2. Организация разветвлений с несколькими условиями.

Создать function-файл `y2` для вычисления функции

$$y_2(x) = \begin{cases} a\sin bx, & \text{если } a \neq 0 \text{ и } b \neq 0; \\ (a+2)x + b, & \text{если } a > -2 \text{ и } b > 0; \\ (2-a)x^2 + b, & \text{иначе,} \end{cases} \quad (6.2)$$

где параметры $x$, $a$ и $b$ определены в п. 1.

Вывести график функции $y_2(x)$.

Обратиться к function-файлу `y2` в режиме прямых вычислений для проверки разветвления по условиям в (6.2).

Пояснить, какой оператор использован для организации разветвления.

3. Организация цикла с заранее известным числом повторений.

Создать function-файл `Fibonacci` для формирования ряда Фибоначчи — вектора $F$ из $M$ членов, где каждый следующий член равен сумме двух предыдущих:

$$F_i = F_{i-1} + F_{i-2}, \quad i = 3, 4, \ldots, M. \quad (6.3)$$

Задать начальные значения $F_1 = 0$ и $F_2 = 1$.

Обратиться к function-файлу `Fibonacci` в режиме прямых вычислений для вывода ряда Фибоначчи.

Пояснить:

- какой оператор использован для организации цикла;

- какие параметры function-файла `Fibonacci` являются входными и выходными.

4. Организация цикла с заранее неизвестным числом повторений.

   Создать function-файл `GeomProgression` для вычисления в цикле суммы бесконечной геометрической прогрессии:

$$S = \sum_{n=0}^{\infty} q^n \tag{6.4}$$

   с заданной точностью $\varepsilon$ и значением $q$, при котором выполняется условие абсолютной сходимости ряда.

   После выхода из цикла вычислить точное значение суммы (6.4) по формуле:

$$S_{true} = \frac{1}{1-q},$$

   и погрешность вычисления суммы:

$$\Delta S = \left| S - S_{true} \right|.$$

   Вывести значения $S$, $S_{true}$, $\varepsilon$ и $\Delta S$.

   Обратиться к function-файлу `GeomProgression` в режиме прямых вычислений для вывода требуемых значений.

   Пояснить:

   - какой оператор использован для организации цикла;
   - какие параметры function-файла `GeomProgression` являются входными и выходными;
   - как сопоставить выведенные значения.

5. Организация разветвления в зависимости от значения выражения.

   Создать script-файл `DifferentFunctions` для выполнения одного из function-файлов: `y1`, `y2`, `Fibonacci` или `GeomProgression`, в зависимости от значения переменной `variant`.

   В script-файле организовать:

   - вывод сообщения о соответствии значения переменной `variant` function-файлу;
   - ввод значения переменной `variant` с клавиатуры;
   - разветвление в зависимости от значения переменной `variant` с выводом сообщения об исполняемом function-файле;
   - вывод сообщения для непредусмотренного значения переменной `variant`, не соответствующего ни одному из function-файлов.

Обратиться к script-файлу `DifferentFunctions` в режиме прямых вычислений для проверки требуемого разветвления.

Пояснить:

- какой оператор использован для организации разветвления;
- что проверяется при организации разветвления;
- к какому типу данных может принадлежать переменная `variant`.

# 6.6. Задание на самостоятельную работу

Самостоятельное задание рекомендуется для закрепления полученных знаний и включает в себя следующие пункты:

1С. Организация разветвления по условию.

Создать function-файл для решения квадратного уравнения

$$ax^2 + bx + c = 0$$

двумя способами:

- используя известную алгебраическую формулу;
- с помощью функции вычисления корней многочлена производного порядка:

  **`x = roots(a)`**

  где `a` — вектор коэффициентов в порядке убывания степеней, а `x` — корни многочлена (вектор).

В том случае, если корни оказались комплексно сопряженными, вычислить и вывести их модуль и аргументы.

2С. Организация разветвления в зависимости от значения выражения.

Создать function-файл для вычисления значения одной из следующих функций:

$$y(x) = \begin{cases} ax - b, & (a+b) = 0,8; \\ ax^2 + b, & (a+b) = 2,5; \\ \sin bx - a, & (a+b) = -3,4; \\ \cos ax + b, & \text{иначе.} \end{cases}$$

Построить график функции $y(x)$ на выбранном интервале по оси $x$ с помощью функции `plot`.

Обратиться к function-файлу в режиме прямых вычислений, задавая значения $a$ и $b$, при которых будут выведены графики различных функций.

3С. Организация цикла с заранее известным числом повторений.

Создать function-файл для вычисления суммы конечного ряда при $0 < |x| < 1$:

$$S = \sum_{n=0}^{N-1} \frac{(-1)^n \sqrt{(n+1)x}}{n+1}.$$

4С. Организация цикла с заранее неизвестным числом повторений.

Создать function-файл для вычисления суммы бесконечного ряда с заданной точностью ε при $0 < |x| < 1$:

$$S = \sum_{n=0}^{\infty} \frac{(-1)^n \sqrt{(n+1)x}}{n+1}.$$

Определить количество циклов, требуемое для вычисления суммы при заданной точности.

# 6.7. Отчет и контрольные вопросы

Отчет составляется в редакторе Word и содержит результаты выполнения каждого пункта задания, включая листинги М-файлов (шрифт Courier New), результаты их выполнения, копируемые из окна **Command Window** (шрифт Courier New), созданные графики (копируются по команде **Edit | Copy Figure** в окне **Figure**) и ответы на поставленные вопросы (шрифт Times New Roman).

Защита лабораторной работы проводится на основании представленного отчета и контрольных вопросов из следующего списка:

1. Поясните назначение и формат оператора if.
2. Поясните назначение и формат оператора switch.
3. Поясните назначение и формат оператора for.
4. Поясните назначение и формат оператора while.
5. Как выполнить принудительный выход из цикла? Какой части программы передается управление в этом случае?