M-files

An M-file is a text file containing Matlab commands and saved with the extension `.m`

A **script** M-file is simply a sequence of Matlab commands; it has no input or output arguments and operates on variables in the workspace.

A **function** M-file can accept input arguments and return output arguments; its internal variables are local to the function.

To run (execute) a script, type the name of the file without the `.m` extension at the command window prompt.

To call a function, type

```
[<list of variables for output>]=<function name>(<list of input parameters>)
```

Boolean (logical) expressions

A boolean expression is an expression whose value is either "true" (1) or "false" (0). Boolean expressions are made up of comparisons, involving relational operators, that can be determined to be either true or false. Different boolean expressions can be connected by the logical operators "and", "or".

Relational operators

`<`	less than
`<=`	less than or equal to
`>`	greater than
`>=`	greater than or equal to
`==`	equal to
`~=`	not equal to

Logical operators

`&` `<boolean expression 1> & <boolean expression 2>`
is "true" if and only if `<boolean expression 1>` and `<boolean expression 2>` are both true.

`&&` (short-circuiting version of `&`)
`<boolean expression 1> && <boolean expression 2>`
returns 0 (false) if `<boolean expression 1>` is false; `<boolean expression 2>` is evaluated only if `<boolean expression 1>` has been found to be true.

`|` `<boolean expression 1> | <boolean expression 2>`
is "true" if `<boolean expression 1>` or `<boolean expression 2>` is true (or if both are true).

`||` (short-circuiting version of `|`)
`<boolean expression 1> || <boolean expression 2>`
returns 1 (true) if `<boolean expression 1>` is true; `<boolean expression 2>` is evaluated only if `<boolean expression 1>` has been found to be false.

`xor` (exclusive "or")
`xor(<boolean expression 1>,<boolean expression 2>)`
is "true" if either `<boolean expression 1>` or `<boolean expression 2>`, but not both, is true.

`~` `~<boolean expression>`
is "true" if `<boolean expression>` is false, and false if `<boolean expression>` is "true".

Conditionals (branching tests)

```
if <boolean expression>
    <statements to execute if boolean expression is true>
else
    <statements to execute if boolean expression is false>
end
```

```
if <boolean expression 1>
    <statements to execute if boolean expression 1 is true>
elseif <boolean expression 2>
    <statements to execute if boolean expression 2 is the first to be true>
elseif <boolean expression 3>
    <statements to execute if boolean expression 3 is the first to be true>
  ⋮
else
    <statements to execute if none of the above boolean expressions are true>
end
```

```
if <boolean expression>
    <statements to execute if boolean expression is true>
end
```

Under this structure, nothing is done if `<boolean expression>` is false.

```
switch <EXPRESSION>
   case <expression 1>
          <statements to execute if EXPRESSION matches expression 1>
   case <expression 2>
          <statements to execute if EXPRESSION matches expression 2>
   case <expression 3>
          <statements to execute if EXPRESSION matches expression 3>
     ⋮
   otherwise
          <statements to execute if EXPRESSION matches none of the above cases>
end
```

In a `switch` statement, `<EXPRESSION>` and `<expression 1>`, `<expression 2>` etc. can be variables, arithmetic expressions, or strings. The `switch` expression is evaluated and the statements following the first matching `case` expression are executed. If none of these cases produces a match, then the statements under `otherwise` are executed.