Conditional Control — if, switch

This group of control statements enables you to select at run-time which block of code is executed. To make this selection based on whether a condition is true or false, use the `if` statement (which may include `else` or `elseif`). To select from a number of possible options depending on the value of an expression, use the `switch` and `case` statements (which may include `otherwise`).

if, else, and elseif

*if* evaluates a logical expression and executes a group of statements based on the value of the expression. In its simplest form, its syntax is

```matlab
if logical_expression
    statements
end
```

If the logical expression is `true` (that is, if it evaluates to logical `1`), MATLAB executes all the statements between the `if` and `end` lines. It resumes execution at the line following the `end` statement. If the condition is `false` (evaluates to logical `0`), MATLAB skips all the statements between the `if` and `end` lines, and resumes execution at the line following the `end` statement.

For example,

```matlab
if rem(a, 2) == 0
    disp('a is even')
    b = a/2;
end
```

You can nest any number of `if` statements.

If the logical expression evaluates to a nonscalar value, all the elements of the argument must be nonzero. For example, assume `X` is a matrix. Then the statement

```matlab
if X
    statements
end
```

is equivalent to

```matlab
if all(X(:))
    statements
end
```

The `else` and `elseif` statements further conditionalize the `if` statement:

- The `else` statement has no logical condition. The statements associated with it execute if the preceding `if` (and possibly `elseif` condition) evaluates to logical `0` (false).
- The `elseif` statement has a logical condition that it evaluates if the preceding `if` (and possibly `elseif` condition) is `false`. The statements associated with it execute if its logical condition evaluates to logical `1` (true). You can have multiple `elseif` statements within an `if` block.
if n < 0 % If n negative, display error message.
    disp(’Input must be positive’);
elseif rem(n,2) == 0 % If n positive and even, divide by 2
    A = n/2;
else
    A = (n+1)/2; % If n positive and odd, increment
end

if Statements and Empty Arrays. An if condition that reduces to an empty array represents a false condition. That is,

if A
    S1
else
    S0
end

executes statement S0 when A is an empty array.

switch, case, and otherwise

switch executes certain statements based on the value of a variable or expression. Its basic form is

switch expression (scalar or string)
    case value1
        statements % Executes if expression is value1
    case value2
        statements % Executes if expression is value2
    .
    .
    otherwise
        statements % Executes if expression does not % match any case
end

This block consists of

- The word switch followed by an expression to evaluate.
- Any number of case groups. These groups consist of the word case followed by a possible value for the expression, all on a single line. Subsequent lines contain the statements to execute for the given value of the expression. These can be any valid MATLAB statement including another switch block. Execution of a case group ends when MATLAB encounters the next case statement or the otherwise statement. Only the first matching case is executed.
- An optional otherwise group. This consists of the word otherwise, followed by the statements to execute if the expression’s value is not handled by any of the preceding case groups. Execution of the otherwise group ends at the end statement.
- An end statement.

switch works by comparing the input expression to each case value. For
numeric expressions, a case statement is true if \((\text{value}==\text{expression})\). For string expressions, a case statement is true if \(\text{strcmp(value,expression)}\).

The code below shows a simple example of the `switch` statement. It checks the variable `input_num` for certain values. If `input_num` is \(-1\), 0, or 1, the case statements display the value as text. If `input_num` is none of these values, execution drops to the otherwise statement and the code displays the text 'other value'.

```
switch input_num
    case -1
        disp('negative one');
    case 0
        disp('zero');
    case 1
        disp('positive one');
    otherwise
        disp('other value.');
end
```

**Note** For C programmers, unlike the C language `switch` construct, the MATLAB `switch` does not "fall through." That is, if the first `case` statement is true, other `case` statements do not execute. Therefore, `break` statements are not used.

`switch` can handle multiple conditions in a single `case` statement by enclosing the case expression in a cell array.

```
switch var
    case 1
        disp('1')
    case {2,3,4}
        disp('2 or 3 or 4')
    case 5
        disp('5')
    otherwise
        disp('something else')
end
```