MATLAB Marina: Logic Expressions

Student Learning Objectives

After completing this module, one should:

1. Be able to form logic expressions using logical and relational operators for evaluating conditions.
2. Be able to use the result of logic expressions for indexing.

Terms

Boolean, logical, logical operator, relational operator, logic expression, compound logic expression

MATLAB Functions, Keywords, and Operators

==, <, <=, >, >=, ~, ~=, &, &&, |, ||, true, 1, false, 0, logical, any, all, isempty, isnumeric, islogical

Logical Values

Logical (Boolean) variables can take on one of two values: true (1) or false (0). Variables can take on a logical value either through direct assignment or as the result of a logic expression. MATLAB converts logical true to 1 and logical false to 0 for storage in the workspace.

```
>> a = true
a =
     logical 1
>> b = false
b =
     logical 0
```

Figure 1. Assigning Variables Logical Values

Logic Expressions

Logic expressions are used to determine if a condition is true or false. Logic expressions typically consist of one or more variables, logical and relational operators, and constants (sometimes mathematical operations are also used). Logic expressions are evaluated from left to right adhering to operator precedence rules.

```
>> v = 5.0;
>> w = 7.0;
>> res_log = v > w
res_log =
     logical 0
```

Figure 2. Sample Logic Expression

Relational Operators

Commonly used MATLAB logical and relational operators for comparisons are given Table 1.

Symbol	Operation
&	Element by element logical AND
&&	Short circuit logical AND
|	Element by element logical OR
||	Short circuit logical OR
~	Element by element logical NOT
==	Equal to
~=	Not equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to

Table 1. MATLAB Logical and Relational Operators

MATLAB precedence for logical and relational operators is:

1. Parentheses ()
2. Logical negation ~
3. Less than (<), less than or equal to (<=), greater than (>), greater than or equal to (>=), equal to (==), not equal to (~=)
4. Element by element AND &
5. Element by element OR |
6. Short circuit AND &&
7. Short circuit OR ||

See MATLAB's help for precedence involving both logical and arithmetic operations.

Compound Logic Expressions

Logical operators such as AND and OR can be used to combine multiple logic expressions for complex conditions. Generally, element by element AND and OR should be used for comparisons on arrays and short circuit AND and OR should be used for comparisons on scalars.

```
>> v = 5.0;
>> w = 7.0;
>> res_log = v > w || v > 0
res_log =
     logical 1
```

Figure 3a. Compound Logic Expression using Short Circuit OR

The result in Figure 3a is true since the statement is checking if either `v > w` or if `v > 0`. In this example, `v > w` is false but `v > 0` is true so the compound comparison is true.

```
>> v = [3.5, -1.0, 0.0, 5.3, 7.4];
>> res_log = v >= 0 & v < 6
res_log =
     1x5 logical array
     1   0   1   1   0
```

Figure 3b. Compound Logic Expression using Element by Element AND

Note in the Figure 3b, the result is an array of logicals since an element by element comparison on an array was performed. The result has true values in the array where the elements satisfy both `v >= 0` and `v < 6`.

Logic Expressions Uses

Logic expressions are typically used for two things: selectively executing blocks of statements and indexing a portion of an array. Logic expressions used with `if-else` and `while` statements (covered later) for selectively executing blocks of code should result in a scalar logical result. Logic expressions that will be used for indexing an array should result in an array of logicals of the same size as the array to index.

Logic expressions used for selectively executing blocks of code as part of an `if-else` or `while` statement do not typically need the result saved in a variable whereas logic expressions used for indexing typically need the result saved to a variable that will be used for the indexing.

MATLAB has some additional functions that are useful for determining the type of data and for comparisons on arrays. The MATLAB `isempty` function returns true of the argument is non empty; this is more efficient than checking if the length or size of an array is zero. MATLAB functions such as `islogical` and `isnumeric`, return true if the variable contains the appropriate data type. The MATLAB functions `any` and `all` return a scalar true if any value in an array is true or all values in an array are true.

Comparison Considerations

Checking for equality using real numbers often does not work well as rarely will a real number be exactly equal to the desired value; 4.0001 is not equal to 4.0. Use either less than or equal to or greater than or equal to rather than equal to or perform a compound comparison to determine if a real number is “close” to a desired value. The MATLAB code of Figure 4 illustrates determining if a number is within 1% of a desired value.

```
>> des = 5.0;
>> tol = 0.01*des;
>> val = 4.97;
>> res_log = (val >= des-tol) && (val <= des+tol)
res_log =
     logical 1
```

Figure 4. Determining if a Number is Close to a Desired Value

The value is checked that it is within the tolerance from the desired value. The parentheses in the logic expression are not needed as plus and minus have higher precedence than the logic and relational operators but are used here for readability.

Recall that MATLAB is a weakly typed language; MATLAB variables of any data type can be used in logic expressions. Nonempty values other than zero are treated as true if used in a logic expression and values of zero are treated as false: `logical(1.5)` is true, `logical(-5)` is true, and `logical(0.0)` is false.

Element by element AND & and OR | should be used for element by element comparisons involving arrays. Short circuit AND && and OR || are generally only used with scalar data and as part of the condition for `if-else` statements.

Remember that the single equals (=) is for assignment and the double equals (==) is for comparisons in logic expressions.

Last modified Friday, September 18, 2020