TOPIC 12 :

Handling Selection Condition Problems

In a program, sequences of a commands sometimes must be conditionally selected based on a relational test. In programming languages, this logic is provided by some variation of an if-end and if-else-end structure. MATLAB provides if-end structure and its variants, if-elseif-end to control the logical flow of a program.

A. Using If-end and If-Elseif-End command

1. If-end

The simplest structure is

```
if expression
   statements
end
```

The statements between the if and end statements are evaluated if all elements in expression are true (nonzero).

MATLAB evaluates the expression and, if the evaluation yields a logical true or nonzero result, executes one or more MATLAB commands denoted here as statements.

Example:

```
if ((attendance >= 0.90) & (grade_average >= 60))
  pass = 1;
end;
```

2. if - else - end

When using elseif and/or else within an if statement, the general form of the statement is:

```
if expression1
        commands
elseif expression2
        commands
elseif expression3
```

commands
else
commands
end

If '*expression1*' is true then commands after *if* are executed. Optionally *elseif* and *else* statements can be used. Expressions are examined until *else* or *end* is reached or one of expressions (in *if* or *elseif* statements) is true. Corresponding piece of code is executed. Nothing executes if expressions are never true and *else* statement (with corresponding commands) is not supplied.

When you are nesting ifs, each if must be paired with a matching end.

B. Using switch-case command

Switch among several cases based on expression

Syntax
switch *switch_expr*
case *case_expr*
statement,...,statement
case *{case_expr1,case_expr2,case_expr3,...}*
statement,...,statement
...
otherwise
statement,...,statement
end

The switch statement syntax is a means of conditionally executing code. In particular, switch executes one set of statements selected from an arbitrary number of alternatives. Each alternative is called a case, and consists of :

- The case statement
- One or more case expressions
- One or more statements

In its basic syntax, switch executes the statements associated with the first case where **switch_expr == case_expr.**

When the case expression is a cell array (as in the second case above), the case_expr matches if any of the elements of the cell array matches the switch expression. If no case expression matches the switch expression, then control passes to the otherwise case (if it exists). After the case is executed, program execution resumes with the statement after the end.

The *switch_expr* can be a scalar or a string. A scalar *switch_expr* matches a *case_expr* if *switch_expr==case_expr*.

A string *switch_expr* matches a *case_expr* if *strcmp(switch_expr,case_expr*) returns 1 (true).

Examples

```
method = 'Bilinear';
switch lower(method)
  case {'linear','bilinear'}
    disp('Method is linear')
  case 'cubic'
    disp('Method is cubic')
  case 'nearest'
    disp('Method is nearest')
  otherwise
    disp('Unknown method.')
end
```

This return:
Method is linear

****