Mas314 Matlab Programming I

1. Flow Control

As with other computer languages, Matlab has several commands for flow control including:

If; **Switch and case; For; While; Continue; Break**.

We next give a brief introduction on how to use these commands for programming.

- **If** The `if` statement evaluates a logical expression and executes a group of statements when the expression is *true.* The optional `elseif` and `else` keywords provide for the execution alternate groups of statements. An `end` keyword, which matches `if`, terminates the last group of statements. *No braces or brackets are involved in statements.* Examples:
  (i) `if A==B, disp('A and B are equal'); end`
  (ii)

```
if A > B,
  disp('greater');
else
  disp('lessorequal');
end
```

  (iii)

```
if A > B,
  disp('greater');
elseif A < B
  disp('less');
elseif A == B
  disp('equal');
else
  disp('Unexpectedsituation');
end
```

- **Switch and case** The `switch` statement executes groups of statements based on the value of a value of a variable or expression. **Note**: Unlike C language, `switch` in `Matlab` does not fall through. If the first `case` statement is *true,* the other `case` statements do not execute. So, `break` statements are not required.

Generally, `switch` statement takes the form:

```
SWITCH switch_expr
  CASE case_expr,
    statement, ..., statement
  CASE case_expr1,
    statement, ..., statement
    ⋮
  OTHERWISE,
    statement, ..., statement
END
```

- **For** The `for loop` repeats a group of statements a fixed, predetermined number of times. We also need an `end` to match.

```
sum = 0;
for n = 1 : 100
    sum = sum + n;
end
sum
```

  The `for loop` can also be nested.

- **While** The **while loop** repeats a group of statements an indefinite number of times under control of a logical condition. A matching `end` is also needed.

```
sum = 0; n = 1;
while n <= 100,
    sum = sum + n;
    n = n + 1;
end
sum
```

- **Continue** The `continue` statement passes control to the next iteration of the `for loop` or `while loop` in which it appears, skipping any remaining statements in the body of the loop.
- **Break** The `break` statement lets you exit early from a `for loop` or `while loop`. In nested loops, `breaks` exits from the innermost loop only. We will use `break` statement in the code of *Bisection Method*.