Matlab/Freemat/Octave/Scilab: Loops: For and While

If a piece of code needs to be executed repeatedly then a loop structure can be used. In most programming languages, using loops is the means to getting a computer to do any significant *work*. However, in Matlab/Freemat/Octave/Scilab we must view things slightly differently. Matlab/Freemat/Octave/Scilab is focussed on matrix-vector operations[1]; to used these rather than loop structures – when there is a choice - would lead to more efficient code.

There are two loop structures in Matlab/Freemat/Octave/Scilab the for loop and the while loop. The for loop should be used in cases when the number of loops is *known* on entry to the loop, if it isn't then the while loop should be used.

The for Loop

In this example j takes the values 1,2,3,4,5 and j*j is output for each values of j.

```
 --> for j=1:5
j*j
end
ans =
1
ans =
4
ans =
9
ans =
16
ans =
25
```

In the above example the number step in j from one execution of the loop to the next is one; this is the default value. We can have steps of alternative size. For example in the next piece of code the step size is 2 (placed between the 1 and the 5). In this code j takes the values 1, 3, 5.

```
 --> for j=1:2:5
j*j
end
ans =
1
ans =
9
ans =
25
```

---

[1] Matlab/Freemat: Array Operations - Matrix and Vector Arithmetic

The step size in a loop does not have to be a whole number. In the following example the step size is 0.5.

```
--> for x=1:.5:3
x*x
end
ans =
1
ans =
2.2500
ans =
4
ans =
6.2500
ans =
9
```

The step size in a loop does not have to be positive. In the following example the step size is -1.

```
--> for j=4:-1:2
j*j
end
ans =
16
ans =
9
ans =
4
```

The while loop

The while loop continues while a condition is met. For example, in the following loop, x is continued to halve until x is no longer greater than 1 and the loop terminates. The while loop must include a logical statement.

```
--> x=10
x =
10
--> while (x>1)
x=x/2
end
x =
5
x =
2.5000
x =
1.2500
x =
0.6250
```