MATH.2720 Introduction to Programming with MATLAB
Logical Variables and Programming Structures, Part 2

Loops

Sometimes you need to execute the same or similar commands a number of times. Loops are good for that purpose. **In case of emergency, you can terminate execution of a runaway program by hitting the Ctrl and c keys simultaneously.**

1. `for` loops

Create and run a script file with the following commands to see an example of how a `for` loop works. This script asks for a positive integer n as input and sums the integers from 1 to n. (How could you do this without using a `for` loop?)

```
n = input('Enter a positive integer ');

err = 0;

if n ~= abs(round(n)) % What is the purpose of this if-else structure?

    err = 1;

else

intsum = 0;

for i=1:n

    intsum = intsum + i;

end % This ends the for loop

end % This ends the if-else structure

if err

disp('You must enter a positive integer.')

else

fprintf('The sum of the integers from 1 to %i is %i\n',n,intsum)

 % The %i format is for integers

end
```

The variables in a for loop can change in increments other than 1. For example, if you just want to sum the even integers from 2 to n you can replace the `for` loop in the previous example with

```
for i=2:2:n

    intsum = intsum + i;

end
```

You can also put a `for` loop inside another `for` loop. Below is an example of a script file that uses nested for loops to generate a square matrix A with entries $A_{ij} = i + j$.

```
n = input('Enter a positive integer: ');

A = zeros(n);

for i = 1:n

    for j = 1:n

        A(i,j)=i+j;

    end

end

disp(A)
```

2. `while` **loops**

Create and run a script file with the following commands to see an example of how a `while` loop works. This script asks for a positive number a as input and estimates $\sqrt{a}$ using Newton's Method. Newton's Method applied to the function $f(x) = x^2 - a$ produces a sequence of numbers $x_1, x_2, x_3, \ldots$ that converges to a root of f. Starting with an initial estimate x_1, successive estimates are calculated using the formula $x_{n+1} = \dfrac{x_n}{2} + \dfrac{a}{2x_n}$.

```
a = input('Enter a positive number: ');

err = 0;

tolerance = 1.e-3;

if a<=0

    err=1;

else

    current_est = a/2;

    new_est = current_est/2 + a/(2*current_est);

    while abs(new_est - current_est) > tolerance

        current_est = new_est;

        new_est = current_est/2 + a/(2*current_est);

    end % ends while loop

end % ends if structure

if err

    disp('You must enter a positive number')

else

    fprintf('The square root of %g is approximately %7.3f \n',a,new_est)

end
```

Practice Problems.

1. Create a script file using a `for` loop that asks the user to input an odd positive integer n and calculates the sum $1 + 3 + 5 + \cdots + n$.

2. (Gilat, Chapter 6, problem 10) Fibonacci numbers are the numbers in a sequence in which the first two elements are 0 and 1, and the value of each subsequent element is the sum of the previous two elements: 0, 1, 1, 2, 3, 5, 8, 13, ...
   Create a script file that uses a `for` loop to generate an array named `Fib` containing the first 20 Fibonacci numbers.

3. Approximate the value of the sum $\sum_{n=1}^{\infty} \frac{1}{n^2}$ by computing a partial sum $\sum_{n=1}^{N} \frac{1}{n^2}$. Use a `while` loop that terminates when the difference between two successive approximations is less than 10^{-10} (1.e-10 in MATLAB notation).

   Compare the value you obtain with the number $\pi^2/6$.