MATLAB Examples

Flow Control and Loops

Hans-Petter Halvorsen, M.Sc.
Flow Control and Loops in MATLAB

Flow Control:
- `if-elseif-else` statement
- `switch-case-otherwise` statement

Loops:
- `for` Loop
- `while` Loop

The behavior is the same as in other programming languages. It is assumed you know about For Loops, While Loops, If-Else and Switch statements from other programming languages, so we will briefly show the syntax used in MATLAB and go through some simple examples.
If-else Statements

Given the second order algebraic equation:

\[ ax^2 + bx + c = 0 \]

The solution (roots) is as follows:

\[
x = \begin{cases} 
  \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}, & a \neq 0 \\
  \frac{-c}{b}, & a = 0, b \neq 0 \\
  \emptyset, & a = 0, b = 0, c \neq 0 \\
  \mathbb{C}, & a = 0, b = 0, c = 0 
\end{cases}
\]

where \( \emptyset \) - there is no solution, \( \mathbb{C} \) - any complex number is a solution
If-else Statements

Create a function that finds the solution for x based on different input values for $a$, $b$ and $c$, e.g.,

$$\text{function } x = \text{solveeq}(a,b,c)$$

We will do the following:

• Use if-else statements to solve the problem
• Test the function from the Command window to make sure it works as expected, e.g.,

$$\gg a=0, \ b=2, \ c=1$$
$$\gg \text{solveeq}(a,b,c)$$
You may define the function like this:

```matlab
function x = solveeq(a,b,c)
if a~=0
    x = zeros(2,1);
    x(1,1)=(-b+sqrt(b^2-4*a*c))/(2*a);
    x(2,1)=(-b-sqrt(b^2-4*a*c))/(2*a);
elseif b~=0
    x=-c/b;
elseif c~=0
    disp('No solution')
else
    disp('Any complex number is a solution')
end
```
We test the function:

```matlab
>> a=0; b=2; c=1
a =
  0
b =
  2
c =
  1
>> solveeq(a,b,c)
ans =
 -0.5000
```

```matlab
>> a=1; b=2; c=1;
>> solveeq(a,b,c)
ans =
 -1
  0
```

```matlab
>> a=0; b=0; c=1;
>> solveeq(a,b,c)
No solution
```

```matlab
>> a=1; b=1; c=2;
>> solveeq(a,b,c)
ans =
  -0.5000 + 1.3229i
  -0.5000 - 1.3229i
```

```matlab
>> a=0; b=0; c=0;
>> solveeq(a,b,c)
Any complex number is a solution
```
Switch-Case Statements

• Create a function that finds either the Area or the circumference of a circle using a Switch-Case statement

\[ A = \pi r^2 \]
\[ O = 2\pi r \]

• You can, e.g., call the function like this:

```matlab
>> r=2;
>> calc_circle(r,1) % 1 means area
>> calc_circle(r,2) % 2 means circumference
```
We can define the function like this:

```matlab
function result = calc_circle(r,x)
switch x
    case 1
        result=pi*r*r;
    case 2
        result=2*pi*r;
    otherwise
        disp('only 1 or 2 is legal values for x')
end
```

Testing the function:

```
>> r=5;, calc_circle(r,1)
ans =
  78.5398
>> r=5;, calc_circle(r,2)
ans =
  31.4159
```

Using an illegal value gives:

```
>> r=5;, calc_circle(r,3)
only 1 or 2 is legal values for x
```
Fibonacci Numbers

• In mathematics, Fibonacci numbers are the numbers in the following sequence:
  
  $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...$

• By definition, the first two Fibonacci numbers are 0 and 1, and each subsequent number is the sum of the previous two. Some sources omit the initial 0, instead beginning the sequence with two 1s.

• In mathematical terms, the sequence $f_n$ of Fibonacci numbers is defined by the relation:

  $$f_n = f_{n-1} + f_{n-2}$$

• with seed values:

  $$f_0 = 0, f_1 = 1$$
We will write a function in MATLAB that calculates the N first Fibonacci numbers, e.g.,

```matlab
>> fibonacci(N)
an =
0
1
1
2
3
5
8
13
21
34
```

We will see a For loop to solve the problem.
We define the Function:

```matlab
function f = fibonacci(N)

f=zeros(N,1);
f(1)=0;
f(2)=1;

for k=3:N
    f(k)=f(k-1)+f(k-2);
end
```

We execute the function:

```
>> fibonacci(N)
ans =
     0
     1
     1
     2
     3
     5
     8
    13
    21
    34
```
While Loops

• Create a Script or Function that creates Fibonacci Numbers up to a given number, e.g.,

```plaintext
>>> maxnumber = 2000;
>>> fibonacci(maxnumber)
```
The function can be written like this:

```matlab
function f = fibonacci2(max)
f(1)=0;
f(2)=1;
k=3;
while f < max
    f(k)=f(k-1)+f(k-2);
k=k+1;
end
```

Testing the function gives:

```matlab
>> maxnumber=200;
fibonacci2(maxnumber)
ans =
     0     1     1     2     3     5     8    13    21    34    55    89   144   233
```
For Loops

• Extend your `calc_average` function from a previous example so it can calculate the average of a vector with random elements. Use a For loop to iterate through the values in the vector and find sum in each iteration:

```
mysum = mysum + x(i);
```

• Test the function in the Command window
Previous Version of `calc_average` function:

```
function av = calc_average(a, b)

av = (a + b)/2;
```

We test the function in the Command window

```matlab
>> z=calc_average(x,y)
z = 3
```
The function can be written like this:

```matlab
function av = calc_average2(x)

mysum=0;
N=length(x);

for k=1:N
    mysum = mysum + x(k);
end

av = mysum/N;
```

Testing the function gives:

```matlab
>> x=1:5
x =
    1     2     3     4     5
>> calc_average2(x)
ans =
    3
```
Create a function where you use the “if-else” statement to find elements larger than a specific value in the task above. If this is the case, discard these values from the calculated average.

Example discarding numbers larger than 10 gives:

```
x =
   4  6  12
>> calc_average3(x)
an =
   5
```
The function can be written like this:

```matlab
function av = calc_average2(x)

mysum=0;
total=0;
N=length(x);

for k=1:N
    if x(k) < 10
        mysum = mysum + x(k);
        total=total+1;
    end
end

av = mysum/total;
```

Testing the function gives:

```
x =
    4     6    12
>> calc_average3(x)
anans =
      5
```
Hans-Petter Halvorsen, M.Sc.

University College of Southeast Norway

www.usn.no

E-mail: hans.p.halvorsen@hit.no
Blog: http://home.hit.no/~hansha/