Crashcourse in Matlab - Part IV: Control flow

So far we have used Matlab as a big graphing calculator. Propper programing starts once you control the flow, i.e., your code has different brances and you determine dynamically which branches get executed and how often. You can find a full list of Matlab control statements here, we will only discuss the most basic ones.

Branching with if-statements

The if, elseif and else statements allow you to execute different brances of the code depending on value of certain variables. A simple use of if statements is to check that imput parameters are physically reasonable. For example most physical paramters have to be non-negative. So you could add the following code below the section where the user specifis the paramters.

```
kappa = 4; % [W/m/K] typical silicate thermal conductivity

if kappa >= 0
    fprintf('Congratulations you put in o.k. values for kappa.\n')
else
    error('Kappa must be non-negative.')
end
```

```
Congratulations you put in o.k. values for kappa.
```

The if-statement is followed by some condition. If the condition is numerical it is typically a bigger or smaller comparison. The statement following 'else' is what is executed if the condition is not met. The fprintf() function allows you to siplay text to the command window. The error() function will display the error-message and then terminate the program.

It is possible to check for multiple conditions at the same time. For example, the thermal condutivity must be non-negative but also the value should not be unreasonably high.

```
kappa = 15; % [W/m/K] typical silicate thermal conductivity

if (kappa >= 0) && (kappa < 10)
    fprintf('Congratulations you put in o.k. values for kappa.\n')
elseif (kappa >= 0) && (kappa >= 10)
    fprintf('At least kappa is not negative, but are you sure that kappa = %3.2f [W/m/K
    fprintf('Press any button to continue the program\n')
    pause()    % Program is stopped - you need to go to command window and press a butto
else
    error('Kappa must be non-negative.')
end
```

```
At least kappa is not negative, but are you sure that kappa = 15.00 [W/m/K] is reasonable?
Press any button to continue the program
```

Here we have concatinated two tests with the logical AND, i.e., both conditions must be true. Note that the '&' must be doubled up in the if-statement - I am not sure why. There is also a logical OR operation, which

also needs to be doubled up in an if-statement. See here for a full list of logical operations. There is also an 'elseif' statement that opens a third branch where you get a warning but the code is not terminated. Here the pause() function is used to make sure the user wants to proceed. Also the fprintf() function outputs the user defined value of kappa, so it is easier to decide if to proceed.

Sometimes it makes sense for conditions to be based on strings. For example, In our code we will have discretications in cartesian, cylindrical and maybe even sperical coordinates. So depending on the coordinate system we the code that comptes the discretization has to be changed. This is one of the few cases where the use of strings is useful, so we discuss some string manipulation as well. The main difficulty with strings as conditions is that the word 'cartesian' and 'cylindrical' are not the same length.

```
coord1 = 'cylindrical';
coord2 = 'cartesian';
fprintf(['The word' ...
    ' %s has %d letters.\n'],coord1,length(coord1))
```

```
The word cylindrical has 11 letters.
```

```
fprintf('The word %s has %d letters.\n',coord2,length(coord2))
```

```
The word cartesian has 9 letters.
```

Matlab things of these strings as two arrays with different sizes and hence cannot compare them. The function strcmp() overcomes this issue and is hence very useful.

```
coord = 'cartesian';

if strcmp(coord,'cartesian')
    disp('Going easy today, cartesian is for beginners.')
elseif strcmp(coord,'cylindrical')
    disp('Not bad, looks like you are getting into it.')
elseif strcmp(coord,'spherical')
    disp('Now you are really trying to impress somebody!')
else
    error('Unknown coordinate system.')
end
```

```
Going easy today, cartesian is for beginners.
```

Repeating commands with for-loops

The for-loop allows you to repeat a command a specified number of times. The main us for for-loops in our code will be the timestepping.

```
for i = 1:10
    fprintf('i = %d\n',i)
end
```

```
for i = 0:2:10
    fprintf('i = %d\n',i)
end
```

```
ivec = [-2 0 3.5 5 10];
for i = ivec
     fprintf('i = %3.2f\n',i)
end
```

Repeating comands with while-loops

While-loops allow you to repeat a command until a condition is met. The main danger with while-loops is that the condition is never met and the loop continues for ever. Hence the most important thing is to know how to kill a code stuck in a forever-loop with a Ctrl+C key combination. The second most important thing is to always set a max number of iterations for a while-loop, so it does not become a forever loop.

```
imax = 15; i = 0;
a = 10; tol = 1e-3;
while (sqrt(a) > 1+tol) && (i < imax)
    i = i+1;
    a = sqrt(a);
    fprintf('i = %d: sqrt(a)^i = %3.4e. \n',i,a)
end
fprintf('\nsqrt(10)^%d = %3.6e < tol = %3.6e.\n',i+1,sqrt(a),1+tol)
```

While reating this example I had to kill Matlab multiple times :-).