MXB103 Worksheet 6

If statements

Core topics:

- if statements (Sections 6.1, 6.2, 6.3)
- relational operators (Section 6.1)
- IVP solvers using if statements (Section 6.5)

In the previous worksheets, we have seen how to use for loops to simply repeat a sequence of statements multiple times. Today, we will learn how to get MATLAB to *choose* what to do next, based on the value of a certain *condition*. The mechanism for achieving this is a new kind of statement: the if statement.

As always, begin by setting compact formatting.

```
>> format compact
```

6.1 The if-end structure

The simplest form of if statement is the if-end structure as follows:

```
if condition
    statements
end
```

Here, condition is an expression that evaluates to either true or false. A true expression is indicated by a value of 1. A false expression indicated by a value of 0. The following are all examples of conditions that evaluate to either true (1) or false (0)

```
>> 5 > 8
ans =
     0
>> 4 < 6
ans =
     1
>> 9*5 == 45
ans =
     1
>> pi == 22/7
ans =
     0
```

Typically, the expression for the `condition` involves one or more **relational operators**. Relational operators in MATLAB are given in the following table.

`==`	equal to
`~=`	not equal to
`<`	less than
`>`	greater than
`<=`	less than or equal to
`>=`	greater than or equal to

Note: the “equal to” operator is spelled with two equals signs, `==`. This is because a single equals sign, `=` is already used for the assignment operator.

Make sure you understand that the two commands

```
>> x = y
```

and

```
>> x == y
```

are completely different. The first *assigns* the value of `y` to `x`. The second *compares* the values of `x` and `y` for equality.

Now, let’s look at an example for the `if-end` structure.

Example 1 You are sending two parcels in the mail. The postage rates are as follows: $2.50 per kg, with a $10 surcharge for anything over 5 kg. Write a function to compute the cost of postage for a parcel of given weight.

```
function cost = postage(weight)
% Postage  Cost of posting a parcel
% cost = postage(weight) computes the cost in dollars of posting a parcel
% of weight kg.

cost = weight * 2.50;
if weight > 5
    cost = cost + 10;
end
```

Note: The `if` statement uses the condition `weight > 5`. Only if that condition is `true`, is the statement, `cost = cost + 10`, executed.

The statement `cost = cost + 10` tells MATLAB to

1. Add 10 to the current value of `cost` (which from the previous line is `weight * 2.50`)
2. Assign the result to `cost` (and so overwriting the old value)

Hence we get the correct formula for computing the cost of postage.

How much will it cost to post two parcels: one weighing 3.7 kg and the other 8.2 kg?

```
>> postage1 = postage(3.7)
>> postage2 = postage(8.2)
```

```
>> total_postage = postage1 + postage2
```

6.2 The `if-else-end` structure

An `if` statement can also have an `else` branch, which takes the form:

```
if condition
    group 1 statements
else
    group 2 statements
end
```

The `if-else-end` structure allows for choosing between two groups of statements for execution. If the `condition` expression is `true`, the group 1 statements are executed, and the group 2 statements are skipped. If the `condition` expression is `false`, the group 1 statements are skipped, and the group 2 statements are executed.

Now, let's look at an example for the `if-else-end` structure.

Example 2 A courier service offers the following deal for delivering parcels: a flat rate of $8 for anything under 4 kg, otherwise $3 per kg. Write a function to compute the cost of using the courier service for a parcel of given weight.

```
function cost = courier(weight)
% courier  Cost of delivering a parcel by courier
% cost = courier(weight) computes the cost in dollars of delivering a
% parcel of weight kg using a courier.

if weight < 4
    cost = 8;
else
    cost = 3 * weight;
end
```

Note: **Only one** of the two statements, `cost = 8` and `cost = 3 * weight`, is executed, depending on whether the condition `weight < 4` is `true` or `false`.

Use the `courier` function to determine how much will it cost to deliver our same two parcels this way.

6.3 The `if-elseif-else-end` structure

The most general `if` statement has the `if-elseif-else-end` structure, which takes the following form:

```
if condition1
    group 1 statements
elseif condition2
    group 2 statements
elseif condition3
    group 3 statements
...
elseif conditionN
    group N statements
else
    group N+1 statements
end
```

Note: **Only one** group of statements is executed: the one whose condition is the **first** (from top to bottom) to evaluate to `true`. **All other groups are skipped**. Therefore, it is important to **order the groups correctly**.

Now, let's look at an example for the `if-elseif-else-end` structure.

Example 3 The recipient of our two parcels has offered to pick them up herself. Her company charges the following rates for pickup:

< 2 kg	\$1.00 per kg
2 kg up to < 5 kg	\$2.50 per kg
5 kg up to < 10 kg	\$4.50 per kg
10 kg or over	\$6.00 per kg

Write a function to compute the cost of having a parcel of given weight picked up.

```
function cost = pickup(weight)
% pickup  Cost of having a parcel picked up
% cost = pickup(weight) computes the cost in dollars of having a parcel of
% weight kg picked up by the receiver.

if weight < 2
    cost = 1.00 * weight;
elseif weight < 5
    cost = 2.50 * weight;
elseif weight < 10
    cost = 4.50 * weight;
else
    cost = 6.00 * weight;
end
```

The four possible outcomes of this function are as follows.

- If `weight < 2` is `true`, MATLAB executes `cost = 1.00 * weight`.
- If `weight < 2` is `false`, MATLAB checks the next condition: `weight < 5`. If this is `true`, MATLAB executes `cost = 2.50 * weight`.
- If `weight < 2` is `false` and `weight < 5` is `false`, MATLAB checks the next condition: `weight < 10`. If this is `true`, MATLAB executes `cost = 4.50 * weight`.
- If `weight < 2` is `false` and `weight < 5` is `false` and `weight < 10` is `false`, MATLAB executes `cost = 6.00 * weight`.

Use the `pickup` function to determine how much will it cost for our two parcels this time.

6.4 Exercises

1. Write a function `maximum` that takes two numbers as inputs, and returns the larger of the two. Note: MATLAB has a built-in function `max` for doing this. But don't use that: write your own function using an `if` statement. Test your function with a few simple examples.

2. Write a function `absolute` that takes a single number as input, and returns its absolute value. Note: MATLAB has a built-in function `abs` for doing this. But don't use that: write your own function using an `if` statement. Test your function with a few simple examples.

3. Write a function `coinflip` that simulates the flipping of a coin. It should be a function taking *no* inputs, and returning either `'Heads'` or `'Tails'` with equal probability. Hint: use MATLAB's `rand` function to generate a random number between 0 and 1, and use that to decide between heads and tails. Run your function 100 times and again 1000 times (using a `for` loop) to see you get heads and tails about half the time.

6.5 Returning to our IVP solvers

One application of `if` statements is to allow a user to choose from a range of possible methods to solve a problem. For example, in worksheet 5 we wrote three functions for solving Initial Value Problems: `euler`, `taylor2` and `modeuler`. Now we can write a single function `IVPsolver` that can be used to choose between the three.

```
function [t,w,h] = IVPsolver(f, fdash, a, b, alpha, n, method)
%IVPsolver  Initial Value Problem solver
% [t,w,h] = IVPsolver(f, fdash, a, b, alpha, n, method) solves an IVP
%   using one of three methods as follows:
%   method = 'euler_method', call Euler solver
%   method = 'taylor2', call Second Order Taylor solver
%   method = 'modeuler', call Modified Euler solver
%   The user may pass any value for fdash if method = 'euler_method' or 'modeuler'

if strcmp(method,'euler_method') % call euler_method function
   disp('Using Euler''s method')
   [t, w, h] = euler_method(f, a, b, alpha, n);
elseif strcmp(method,'taylor2') % call taylor2 function
   disp('Using Second Order Taylor method')
   [t, w, h] = taylor2(f, fdash, a, b, alpha, n);
elseif strcmp(method,'modeuler') % call modeuler function
   disp('Using Modified Euler method')
   [t, w, h] = modeuler(f, a, b, alpha, n);
else % invalid method choice
    error('Invalid method! Please choose ''euler_method'', ''taylor2'', or ''modeuler''')
end
```

Note `strcmp(str1,str2)` compares two strings `str1` and `str2` and returns `1` (`true`) if the two are identical and `0` (`false`) otherwise.

Let us use the same IVP from worksheet 5:

$$\begin{aligned} \frac{dy}{dt} &= 3t - \frac{y}{t}, \quad 1 \leq t \leq 2 \\ y(1) &= 2\,. \end{aligned}$$

Now, we can compare the error in solution at $t = 2$ using 16 steps of each method, by calling `IVPsolver` in a `for` loop.

First we set up the problem (same one from worksheet 5 and lectures):

```
>> f = @(t,y) 3*t - y./t; fdash = @(t,y) 2*y./(t.^2);
>> a = 1; b = 2; alpha = 2; n = 16; h = (b-a)/n;
>> y = @(t) t.^2 + 1./t;  % the analytical solution
```

Then we call the `IVPsolver` in a `for` loop:

```
methods = {'euler_method','taylor2','modeuler'};
for k = 1:3
   [t,w,h] = IVPsolver(f, fdash, a, b, alpha, n, methods{k});
   error_at_end = abs(w(end) - y(2));
   disp('Error at t = 2');
   disp(error_at_end);
end
```

Note `methods` is defined as a cell array, which permits string elements. Cells arrays are defined and indexed using curly brackets.

Your code should return:

```
Using Euler's method
Error at t = 2
   0.0635
Using Second Order Taylor method
Error at t = 2
   7.8818e-04
Using Modified Euler method
Error at t = 2
   9.7656e-04
```