Solving a system of non-linear equations using the *fsolve* function in MATLAB command window

Example 1:

Solve

Finding route near 1 of

$$3x^3 - 2x^2 + x - 7$$

In the script editor define and save FUNCTION

```
function F=basicfun(x)
F=3.*x.^3-2*x.^2+x-7;
End
```

Test function in command window

```
>> x=1
x =

    1

>> basicfun(x)

ans =

   -5
```

Set Initial guess in command window as

```
>> x0=1
```

Then type

```
>> [x,fval]=fsolve(@basicfun,x0)
```

This says by an iterative process starting with the guess ***x0*** approximate the value of ***x*** that satisfies the none-linear equations in the function *basicfun*, printing out the current *residuals* into the value fval

You will get the following

Equation solved.

fsolve completed because the vector of function values is near zero as measured by the default value of the function tolerance, and the problem appears regular as measured by the gradient.

<stopping criteria details>

x =

    1.4918

fval =

  3.8543e-009

Now let us solve the non-linear systesm

$$\begin{aligned} 3xy &+ y - z = 12 \\ x &+ yx^2 + z = 12 \\ x &- y - z = -2 \end{aligned}$$

MATLAB uses thinks as a vector so we need to write our unknowns in a column vector ***x*** with components

$x(1) \equiv x$

$x(2) \equiv y$

$x(3) \equiv z$

Strep 1: Enter an initial guess on the command page called ***x0***

```
>> x0=[1;1;1]
```

The notation [--] signifies a vector
The use of the semi-colon ; will make a column vector
On hitting the enter key you will get

```
x0 =

    1
    1
    1
```

ASIDE—
The entry using commas , instead of semi-colons ; gives a row vector

```
>> x0=[1,1,1]

x0 =

    1    1    1
```

Step 2: Enter the equations to be solved as a column vector function
--call up an m-function file and enter and SAVE the code

```
function F=ourfun(x)
F=[3*x(1)*x(2)+x(2)-x(3)-12;
   x(1)+x(2)*x(1)*x(1)+x(3)-12;
   x(1)-x(2)-x(3)+2];
end
```

Each line is a column vector component and a function

Step 3: We can check our coding by typing the following in the command window
>> ourfun(x0)

ans =

-9
-9
1

This step evaluates the vector function (each row of our equation set) with the guessed values in the column vector x0=[1;1;1]. The values shown are the right hand side values. Since these values should be zero they represent the amount by which our guess x0 fails to satisfy the equations of interest. These values are referred to as *Residuals*

Step 4: solve the equation by entering the following command window line

>> [x,fval]=fsolve(@ourfun,x0)

This says by an iterative process (see next week) starting with the guess ***x0*** approximate the vector ***x*** that satisfies the equations in the non-linear vector function *ourfun*, printing out the current *residuals* into the vector fval.

The output will read

```
>> [x,fval]=fsolve(@ourfun,x0)
Optimization terminated: first-order optimality is less than options.TolFun.

x =

   1.999999999999999   2.000000000000000   2.000000000000000


fval =

  1.0e-013 *

  -0.053290705182008
  -0.106581410364015
  -0.013322676295502

>>
```