

Tutorial 1: Solving nonlinear equations with MATLAB

COMP5930M Scientific Computation

Today

Matlab functions

Example

Solving with Matlab

- Our own code

- Matlab `fzero()`

Programming concepts

We can use the Matlab language:

- ▶ on the interface **command line**, one statement at a time;
- ▶ as a collection of statements in **a script file**;
- ▶ as reusable blocks of code in **a function file**;
- ▶ Matlab programs (scripts or functions) have the extension **.m**, ie. **name.m**
- ▶ The Matlab library contains functions we can use on the command line or in our own **script and function files**.

Using functions

- ▶ The fundamental programming unit is the **function**.
- ▶ Matlab functions, in common with many other languages, are defined with a simple syntax:

```
function [output1,output2,...] = name(input1,input2,...)
```

- ▶ The most useful Matlab command is **help**!
 - ▶ For Matlab library functions **help name** will return the definition of the function and it's required inputs/outputs.
 - ▶ You **should** include the same information in any function you write.

Example 1

Calculate the value of $x = \sqrt{R}$
(where R is some positive real number)
without the direct use of any `sqrt` function.

- ▶ Formulate as a nonlinear problem
- ▶ Specify an appropriate initial value
- ▶ Solve computationally

Example 1 in Matlab: Function

Create a file `squareRoot1.m` in the editor:

```
function y = squareRoot1(x)

% Nonlinear function for Example 1
% Given parameter R returns y for any x

R = 2;
y = x^2 - R;

end
```

Example 1 in Matlab: A better function

Create a file `squareRoot2.m` in the editor:

```
function y = squareRoot2(x,R)

% Nonlinear function for Example 1
% Given parameter R returns y for any x

y = x^2 - R;

end
```

An implementation of Newton's Method

```
function [x,f] = myNewton( fnon, dfdx, x0, tol, maxk )

k = 0;
x = x0;
f = feval(fnon,x);

while( norm(f)>tol && k<maxk )

    d = feval(dfdx,x);

    k = k + 1;
    x = x - f/d;
    f = feval(fnon,x);

end

end
```


Example 1 in Matlab: Solution

On the command line:

```
>> x0 = 2;  
>> x = myNewton( @squareRoot1,  
                  @dSquareRootdx,x0,1e-6,10 )  
  
>> R = 2;  
>> x0 = 0.5*(R+1);  
>> x = myNewton( @(x)squareRoot2(x,R),  
                  @dSquareRootdx,x0,1e-6,10 )
```

Using the Matlab library

The *black-box solution* approach adopted by most of the available software requires you to provide only the minimum information

The Matlab `fzero()` function requires:

- ▶ The function $y = f(x)$ such that a value y is returned for any input x
- ▶ An initial point x_0 , or bracket $[x_0, x_1]$


Example 1 in Matlab: Solution

On the command line:

```
>> R = 2;  
>> x0 = 0.5*(R+1);  
>> x = fzero( @squareRoot1,x0 )
```

```
>> R = 2;  
>> x0 = 0.5*(R+1);  
>> x = fzero( @(x)squareRoot2(x,R),x0 )
```

Example 1 in Matlab: options

We can refine our use of `fzero()` 

>> `help fzero` shows what is available

- ▶ More output:

```
[x,f,flag]=fzero(@fun,x0)
```

- ▶ More detail:

```
[...]=fzero(@fun,x0,  
            optimset('Display','iter'))
```

- ▶ With a starting bracket:

```
[...]=fzero(@fun,[x0 x1],optim...)
```