Root Finding

Numerical methods are used to solve mathematical problems that are hard or impossible to solve analytically (by hand), by reformulating them into a large number of simpler problems that can then be solved on a computer. One such problem is finding the root(s) of complicated functions.

In the lecture we looked at bracketing and open methods. Bracketing methods are slower, but reliable. Open methods are faster, but can be unstable.

Your Task

Your task today is to write a MATLAB Function that uses the Secant method to find the root of a predefined function, from one initial guess. You can presume that the function is real and continuous. Please use the version of the Secant method that uses a perturbation:

$$x_{i+1} = x_i - \frac{f(x_i)\delta x}{f(x_i + \delta x) - f(x_i)} \qquad \varepsilon_a = \left| \frac{x_{i+1} - x_i}{x_{i+1}} \right| \times 100\%$$

where x_{i+1} is the new estimate of the root and x_i is that from the previous iteration. In the first instance, x_i = initial guess. ε_a is the approximate error at iteration $i+1$. Use a perturbation of $\delta x = 0.001$.

Your MATLAB function should stop either when the approximate error has fallen below a certain limit or a pre-defined maximum number of iterations reached. Your MATLAB function should write to screen the iteration number, estimated value of the root and approximate error each iteration. In addition, upon completion it should plot all estimated values for the root against iteration number.

Defining Functions at the Command Line

You should predefine the function that you want to determine the root(s) of at the MATLAB command line.

To define $f(x) = x^2$ you would type:

```
>> f=@(x) x^2
```

You can use `f` in the normal manner:

```
>> f(2)

ans =

     4
```

Passing a Simple Function to a MATLAB Function

Functions defined at the command line can be used by MATLAB functions. Consider the MATLAB function `evaluate` in a file called `evaluate.m`:

```
function [ ] = evaluate(f,x)
fx = f(x);
disp(['f(x) = ', num2str(fx)])
end
```

This MATLAB function takes a predefined function `f` and a value `x` and evaluates the function at `x` and then writes out the result.

If we define $f(x) = x^2$ at the command line:

```
>> f=@(x) x^2
```

and then execute the MATLAB function:

```
>> evaluate(f,2)
```

we get

```
f(x) = 4
```

On the other hand, if we define $f(x) = x^2 - 2$ at the command line:

```
>> f=@(x) (x^2) - 2
```

and then execute our MATLAB function:

```
>> evaluate(f,2)
```

we get

```
f(x) = 2
```

It should be clear from this how a function can be defined at the command line and passed to a MATLAB function and used.

The benefit of this is that you can write a general MATLAB function to perform the Secant method for an arbitrary function, and just pass the function that you are interested in to it.

More Details

The first line of your MATLAB function should look something like this:

```
function [ ] = secant(f,xi,elim,maxit)
```

where `f` is the function defined at the command line, `xi` is the initial guess, `elim` is the error limit and `maxit` the maximum number of iterations. You do not need to put any variables between the square brackets, since you will write the results to screen each iteration and plot the results upon completion.

To use your MATLAB function you would first define the function whose roots you want to determine at the MATLAB command line. For example:

```
>> f=@(x) (x^2) - 2
```

and then execute you MATLAB function:

```
>> secant(f,1,0.000001,40)
```

This would determine the root of $x^2-2 = 0$ using an initial guess of 1 and stopping when either the approximate error is less than 0.000001 % or 40 iterations have been made. Remember that your MATLAB function should write out the iteration number, estimated value for the root and error at each iteration and plot the estimated values for the root against iteration number upon completion.

Testing your MATLAB Function

Use your completed MATLAB function to find the root of the functions:

1. $f(x) = (4-x)e^{-x/2} - 2$, use an initial guess of -1
2. $f(x) = x^2$, use an initial guess of 0

Set the error limit to 0.000001 % and maximum number of iterations to 40. You may encounter problems with 2 – adjust your program accordingly.