1.5 LAB 1: the bisection and secant methods

The goal of this section is to help you gain familiarity with the fundamental tasks that can be accomplished with MATLAB: defining vectors, computing functions, and plotting. We'll then see how to implement and analyse the Bisection and Secant schemes in MATLAB.

You'll find many good MATLAB references online. I particularly recommend:

- Cleve Moler, *Numerical Computing with MATLAB*, which you can access at `http://uk.mathworks.com/moler/chapters`
- Tobin Driscoll, *Learning MATLAB*, which you can access through the NUI Galway library portal.

MATLAB is an interactive environment for mathematical and scientific computing. It the standard tool for numerical computing in industry and research.

MATLAB stands for Matrix Laboratory. It specialises in matrix and vector computations, but includes functions for graphics, numerical integration and differentiation, solving differential equations, etc.

MATLAB differs from most significantly from, say, Maple, by not having a facility for abstract computation.

1.5.1 The Basics

MATLAB an *interpretive* environment – you type a command and it will execute it immediately.

The default data-type is a matrix of double precision floating-point numbers. A scalar variable is an instance of a 1×1 matrix. To check this set,
`>> t=10` and use `>> size(t)`
to find the numbers of rows and columns of `t`.

A vector may be declared as follows:
`>> x = [1 2 3 4 5 6 7]`
This generates a vector, x, with $x_1 = 1$, $x_2 = 2$, etc. However, this could also be done with `x=1:7`

More generally, if we want to define a vector $x = (a, a+h, a+2h, \ldots, b)$, we could use `x = a:h:b;`
For example
`>> x=10:-2:0` gives $x = (10, 8, 6, 4, 2, 0)$.
If h is omitted, it is assumed to be 1.

The i^{th} element of a vector is access by typing `x(i)`. The element of in row `i` and column `j` of a matrix is given by `A(i,j)`

Most "scalar" functions return a matrix when given a matrix as an argument. For example, if x is a vector of length n, then `y = sin(x)` sets y to be a vector, also of length n, with $y_i = \sin(x_i)$.

MATLAB has most of the standard mathematical functions: `sin`, `cos`, `exp`, `log`, etc.
In each case, write the function name followed by the argument in round brackets, e.g.,
`>> exp(x)` for e^x.

The `*` operator performs matrix multiplication. For element-by-element multiplication use `.*`
For example,
`y = x.*x` sets $y_i = (x_i)^2$.
So does `y = x.^2`. Similarly, `y=1./x` set $y_i = 1/x_i$.

If you put a semicolon at the end of a line of MATLAB, the line is executed, but the output is not shown. (This is useful if you are dealing with large vectors). If no semicolon is used, the output is shown in the command window.

1.5.2 Plotting functions

Define a vector
`>> x=[0 1 2 3]` and then set `>> f = x.^2 -2`
To plot these vectors use:
`>> plot(x, f)`
If the picture isn't particularly impressive, then this might be because Matlab is actually only printing the 4 points that you defined. To make this more clear, use
`>> plot(x, f, '-o')`
This means to plot the vector `f` as a function of the vector x, placing a circle at each point, and joining adjacent points with a straight line.

Try instead: `>> x=0:0.1:3` and `f = x.^2 -2` and plot them again.

To define function in terms of *any* variable, type:
```
>> F = @(x)(x.^2 -2);
```
Now you can use this *function* as follows:
```
>> plot(x, F(x));
```
Take care to note that MATLAB is *case sensitive*.

In this last case, it might be helpful to also observe where the function cuts the x-axis. That can be done by also plotting the line joining, for example, the points $(0, 0)$, and $(3, 0)$:
```
>> plot(x,F(x), [0,3], [0,0]);
```
Tip: *Use the* `>> help` *menu to find out what the* `ezplot` *function is, and how to use it.*

1.5.3 Programming the Bisection Method

Revise the lecture notes on the *Bisection Method*.
Suppose we want to find a solution to $e^x - (2-x)^3 = 0$ in the interval $[0, 5]$ using Bisection.

- Define the function `f` as:
  ```
  >> f = @(x)(exp(x) - (2-x).^3);
  ```
- Taking $x_1 = 0$ and $x_2 = 5$, do 8 iterations of the Bisection method.
- Complete the table below. You may use that the solution is (approximately)
  $\tau = 0.7261444658054950$.

k	x_k	$\lvert\tau - x_k\rvert$
1		
2		
3		
4		
5		
6		
7		
8		

Implementing the Bisection method by hand is very tedious. Here is a program that will do it for you. You don't need to type it all in; you can download it from www.maths.nuigalway.ie/MA385/lab1/Bisection.m

```
3  clear; % Erase all stored variables
4  fprintf('\n\n---------\n Using Bisection\n');
5  % The function is
6  f = @(x)(exp(x) - (2-x).^3);
7  fprintf('Solving f=0 with the function\n');
8  disp(f);
9
10
11 tau = 0.72614446580549503614; % true solution
12 fprintf('The true solution is %12.8f\n', tau);
13
14 %% Our initial guesses are x_1=0 and x_2 =2;
15 x(1)=0;
16 fprintf('%2d | %14.8e | %9.3e \n', ...
17    1, x(1), abs(tau - x(1)));
18 x(2)=5;
19 fprintf('%2d | %14.8e | %9.3e \n', ...
20    2, x(2), abs(tau - x(2)));
21 for k=2:8
22    x(k+1) = (x(k-1)+x(k))/2;
23    if ( f(x(k+1))*f(x(k-1)) < 0)
24       x(k)=x(k-1);
25    end
26    fprintf('%2d | %14.8e | %9.3e\n', ...
27       k+1, x(k+1), abs(tau - x(k+1)));
28 end
```

Read the code carefully. If there is a line you do not understand, then ask a tutor, or look up the on-line help. For example, find out what that `clear` on Line 3 does by typing `>> doc clear`

Q1. Suppose we wanted an estimate x_k for τ so that $|\tau - x_k| \leqslant 10^{-10}$.

(i) In §1.1 we saw that $|\tau - x_k| \leqslant (\frac{1}{2})^{k-1}|b - a|$. Use this to estimate how many iterations are required in theory.

(ii) Use the program above to find how many iterations are required in practice.

1.5.4 The Secant method

Recall the the Secant Method in (1.2.1).

Q2 (a) Adapt the program above to implement the secant method.

(b) Use it to find a solution to $e^x - (2-x)^3 = 0$ in the interval $[0, 5]$.

(c) How many iterations are required to ensure that the error is less than 10^{-10}?

. . . . . .

Q3 Recall from Definition 1.2.4 the *order of convergence* of a sequence $\{\varepsilon_0, \varepsilon_1, \varepsilon_2, \ldots\}$ is q if

$$\lim_{k\to\infty} \frac{\varepsilon_{k+1}}{\varepsilon_k^q} = \mu,$$

for some constant μ.

We would like to verify that $q = (1+\sqrt{5})/2 \approx 1.618$. This is difficult to do computationally because, after a relatively small number of iterations, the round-off error becomes significant. But we can still try!

Adapt the program above so that at each iteration it displays

$$\frac{|\tau - x_{k+1}|}{|\tau - x_k|}, \quad \frac{|\tau - x_{k+1}|}{|\tau - x_k|^{1.618}}, \quad \frac{|\tau - x_{k+1}|}{|\tau - x_k|^2},$$

and so deduce that the order of converges is greater than 1 (so better than bisection), less than 2, and roughly $(1+\sqrt{5})/2$.

1.5.5 To Finish

Before you leave the class upload your MATLAB code for the Q3 (only) to "**Lab 1**" in the "Assignments and Labs" section Blackboard. This file must include your name and ID number as comments. Ideally, it should incorporate your name or ID into the file name (e.g., `Lab1_Donal_Duck.m`). Include your answers to Q1 and Q2 as comments in that file.

1.5.6 Extra

The bisection method is popular because it is robust: it will always work subject to minimal constraints. However, it is slow: if the Secant works, then it converges much more quickly. How can we combine these two algorithms to get a fast, robust method? Consider the following problem:

$$\text{Solve } 1 - \frac{2}{x^2 - 2x + 2} = 0 \quad \text{on } [-10, 1].$$

You should find that the bisection method works (slowly) for this problem, but the Secant method will fail. So write a hybrid algorithm that switches between the bisection method and the secant method as appropriate.

Take care to document your code carefully, to show which algorithm is used when.

How many iterations are required?