



## Numerical Methods Lab 3

08 March 2018

### Instructions

- Read all the instructions carefully.
- MATLAB has a help file for every function if you get stuck.
- There are also numerous sources available on the internet, Google is your friend!

#### SECTION 1

### Some Useful MATLAB Built-in Commands

Command	Result
<code>zeros(m,n)</code>	Creates an $m \times n$ matrix consisting of all zeros.
<code>ones(m,n)</code>	Creates an $m \times n$ matrix consisting of all ones.
<code>eye(m,n)</code>	Creates an $m \times n$ Identity matrix.
<code>X = diag(v,k)</code>	Creates a diagonal matrix with elements of $v$ on the $k$ th diagonal. $k = 0$ is the main diagonal.
<code>v = diag(X)</code>	Returns a vector of the main diagonal of matrix $X$ .
<code>transpose(X)</code>	Returns the transpose of matrix $X$ .
<code>trace(X)</code>	Returns the sum of the elements on the main diagonal of $X$ .
<code>inv(X)</code>	Returns the inverse of the square matrix $X$ .
<code>det(X)</code>	Returns the determinant of the square matrix $X$ .
<code>[V, D] = eig(X)</code>	Returns the eigenvalues and eigenvectors of matrix $X$ .
<code>fliplr(X)</code>	Returns $X$ with rows preserved and columns flipped in the left/right direction.
<code>flipud(X)</code>	Returns $X$ with columns preserved and rows flipped in the up/down direction.
<code>tril(X,k)</code>	Extracts the lower triangular part of the matrix $X$ , consisting of the elements on and below the $k$ th diagonal.
<code>triu(X,k)</code>	Extracts the upper triangular part of the matrix $X$ , consisting of the elements on and above the $k$ th diagonal.
<code>linspace(a,b,n)</code>	Returns a vector of $n$ equally spaced points between (and including) $a$ and $b$ .

#### SECTION 2

### Main Exercises

The due date for the lab submission is the 12th of March 2018 at 12pm.

## 2.1

---

### Exercise 1

Program the Bisection method to find the roots of equations. Your function should take as inputs, some function `f` (you will need to look up how function handles work), a tolerance value `tol`, as well as some initial bracket  $I_0 = (a_{\text{initial}}, b_{\text{initial}})$  respectively. Your function should return as output the root  $c$ . Use the stopping criteria  $\frac{|c_n - c_{n-1}|}{|c_n|} < \text{tol}$  for the convergence of the sequence  $c_n$ 's. The first line of your function must look like the following:

```
function root = bisectionSearch(f, tol, I_0)
```

## 2.2

---

### Exercise 2

Program the False position method to find the roots of equations. Your function should take as inputs, some function `f` (you will need to look up how function handles work), a tolerance value `tol`, as well as some initial bracket  $I_0 = (a_{\text{initial}}, b_{\text{initial}})$  respectively. Use a similar stopping criteria as in Exercise 1. Your function should return as output the root  $c$ . The first line of your function must look like the following:

```
function root = RegularfalsiSearch(f, tol, I_0)
```

## 2.3

---

### Exercise 3

Program the Newton root finding method for a scalar equation. Your function should take as inputs, two function handles `f` and `fprime` (you will need to look up how function handles work). It must also take in an initial guess  $x_0$  and a tolerance value `tol`. Your function should return as output the root  $x$ . Use the stopping criteria  $\frac{|x_n - x_{n-1}|}{|x_n|} < \text{tol}$ . The first line of your function must look like the following:

```
function x = Newtonmethodscaler(f, fprime, x0, tol)
```

## 2.4

---

### Exercise 4

Do plot the error given by  $\frac{|c_n - c_{n-1}|}{|c_n|}$  against the number of iterations for the bisection and the regular falsi Method. On the same graph, plot the error  $\frac{|x_n - x_{n-1}|}{|x_n|}$  against the number of iterations for the Newton's method.

What do you see?

## 2.5

---

### Exercise 5

Program the Newton root finding method for systems. Your function should take in as inputs, two function handles (anonymous functions) -  $F$  (system of equations as a column vector) and its Jacobian  $J$  (a matrix). It must also take in an initial guess  $x_0$  and an tolerance value `tol`. The output of your function must be the roots of the system as a column vector  $x$ . The first line of your function should look like:

```
function x = Newtonmethodsystem(F, J, x0, tol)
```