Thomas Lea-Redmond 1908527

Report

Task 1

To create Ostrowski.m I duplicated Newton.m and renamed it. Then I noticed the q_{n-1} in the second formula was equal to original formula so I substituted that into the second equation.

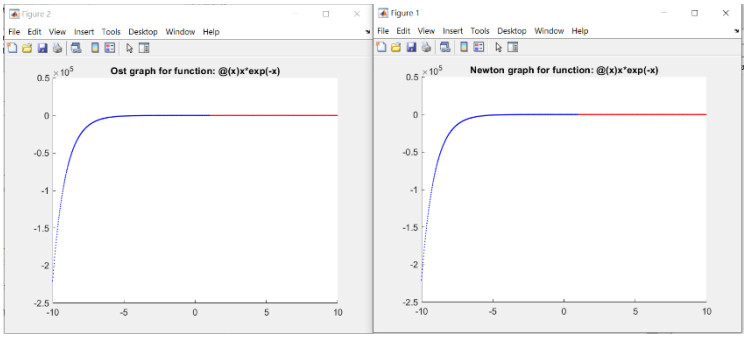
```
while i <= n0
    %Step 3:
    p = p0 - (f(p0)/dF(p0))*((f(p0) - f(p0 - f(p0)/dF(p0))) / (f(p0) - 2*f(p0 - f(p0)/dF(p0))));
    %fprintf('%3d:%16.9f\n', i, p);
    %Step 4:
    if abs(p - p0) < TOL
```

Change made to Newton.m to create Ostrowski.m

Task 2

visualiseConvergence1.m takes a single parameter, and outputs two graphs, one for the Ostrowski method and another for the Newton method indicating where the beginning those methods with a value x leads to the roots converging within an iteration limit.

The blue indicates convergence occurs within the limit, whereas the red indicates with that value of x it doesn't within the limit of 100 iterations.



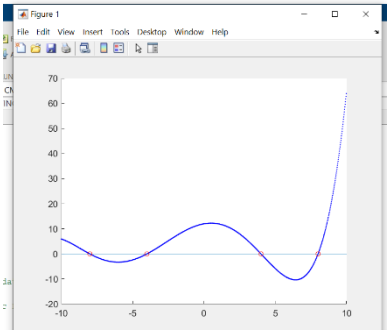
*visualiseConvergence1.m with @(x) x*exp(-x) given as the parameter.*

Task 3

NewtonMulti.m takes the same parameters as Newton.m. The function is plotted, and the roots where the line intercepts the x-axis are displayed within the bounds -10 to 10. For the given example, there is a root identified at (-12, 0) which is ignored as it lies outside those bounds.

4 roots are displayed on the graph, with red circles at their positions. A text output also states for this example.

Root @ (8, 0)
Root @ (-4.000000e+00, 0)
Root @ (4, 0)
Root @ (-8.000000e+00, 0)



*NewtonMulti.m with @(x) (x^5)/1024+(3*x^4)/256-(5*x^3)/64-(15*x^2)/16+x+12, @(x)x.*(-1.5e+1./8.0)-x.^2.*(1.5e+1./6.4e+1)+x.^3.*(3.0./6.4e+1)+x.^4.*4.8828125e-3+1.0, 10, 0.00001, 100 as the parameters*

Task 4

```
>> BisectionInitialise(@(x) (x^5)/1024+(3*x^4)/256-(5*x^3)/64-(15*x^2)/16+x+12, -10, 10)
     5     10
```

*Task4 run with parameters @(x) (x^5)/1024+(3*x^4)/256-(5*x^3)/64-(15*x^2)/16+x+12, -10, 10*

For demonstration purposes, the result is displayed in the command line. This program returns an array of two elements.

```
% Improves the initialisation of the Bisection Method.
% Given the initial range [xmin, xmax] repeatedly onedivide this range to
% find a range that is suitable
% for the Bisection Method; i.e.f(xmin) andf(xmax) should have opposite signs
% &repeat this process for each of the subintervals until a suitable initialisation is found
% or a pre-specified number has been considered [10].
function [a, b] = BisectionInitialise(f, a, b)
    % f : anymous function
    % a : lower bound
    % b : higher bound
    tic; % start function timer
    limit = 20; % set limit to iterations

    i = 1;
    while i <= limit
        Ya = f(a);
        Yb = f(b);
        if (Ya >= 0 && Yb < 0) || (Ya < 0 && Yb >= 0)
            disp([a b]);
            return; % end function
        else
            a = (a + b) / 2; % a set to midpoint of range
            i = i + 1;
        end
    end
```

Code for task 4

Thomas Lea-Redmond 1908527

Task 5

```
Command Window
>> RootFindingImproved(@(x) (x^5)/1024+(3*x^4)/256-(5*x^3)/64-(15*x^2)/16+x+12, -10, 10, 0.00001,100)
     5    10

Value Returned = 8
```

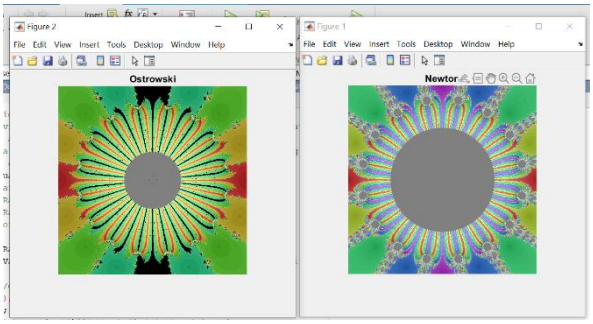
*RootFindingImproved.m run with parameters @(x) (x^5)/1024+(3*x^4)/256-(5*x^3)/64-(15*x^2)/16+x+12, -10, 10, 0.00001,100*

The value is returned by the function, but for demonstration purposes also displays the result in the command window.

Task 6

The function is given an anonymous function which it uses to create images based on the Newton.m and Ostrowski.m root finding methods. This specific example uses modified versions NewtonAndLoopNo.m and OstrowskiAndLoopNo.m which returns the p value and the iterator value in an array.

I used the Mathworks Symbolic Toolbox(Mathworks) to generate the differentiated function which was used as a parameter for the modified Newton and Ostrowski methods.



1visualiseConvergence2.m run using parameters @(x) x^16 - 1

The Images are titled with the respective methods used to create them, and colour coded to visualise the different roots identified. I used a pre-created colourmap to generate the colours to display the different roots. The colour of a specific co-ord display the root which was reached using that complex number as parameter p0.
The colour was then averaged with the log base 10 mapping of the number of loops it took to get the result, which was halved to get it in the dynamic range of 0 to 1 as the maximum number of loops was 100. Complex p0 values that did not identify a root was replaced with a flat grey.
The image is then displayed to the user. There are clear distinctions between the roots found, with colour gradients to indicate the number of iterations taken; darker => less.

Reference

Mathworks. Unknown. Symbolic Maths Toolbox. Available at: https://uk.mathworks.com/products/symbolic.html [Accessed April 2020]