Cardiff School of Computer Science and Informatics

Coursework Assessment Pro-forma

Module Code:	CM2208
Module Title:	Scientific Computing
Lecturer:	Prof. Paul Rosin
Assessment Title:	Solving one variable non-linear equations
Assessment Number:	2
Date Set:	Monday 22nd March 2021 (week 8)
Submission date and Time:	Monday 3rd May 2021 at 9:30am (week 11)
Return Date:	Monday 24th May 2021

This coursework is worth 35% of the total marks available for this module. If coursework is submitted late (and where there are no extenuating circumstances):

1. If the assessment is submitted no later than 24 hours after the deadline, the mark for the assessment will be capped at the minimum pass mark;
2. If the assessment is submitted more than 24 hours after the deadline, a mark of 0 will be given for the assessment.

Your submission must include the official Coursework Submission Cover sheet, which can be found here:
`https://docs.cs.cf.ac.uk/downloads/coursework/Coversheet.pdf`

Submission Instructions

Description		**Type**	**Name**
Cover sheet	Compulsory	One pdf file	`[Student number].pdf`
Report	Compulsory	One pdf file	`Report_[Student number].pdf`
Source code	Compulsory	One or more Matlab files packaged as a single zip file	`Code_[Student number].zip`

Any code submitted will be run on a system equivalent to those available in the School laboratory and must be submitted as stipulated in the instructions above.

Any deviation from the submission instructions above (including the number and types of files submitted) will result in a mark of zero for the assessment.

Staff reserve the right to invite students to a meeting to discuss coursework submissions

---

Assignment

Modify the code `Newton.m` which is provided on Learning Central and has been covered in class during the sessions on solving one variable non-linear equations.

Note (applying to the following tasks):

1. This assignment does **not** involve writing a GUI (e.g. using AppDesigner).

2. Your code needs to be able to handle functions that are not just polynomials (as demonstrated in some of the examples below).
3. Use the code for `Newton.m` and `Bisection.m` that are provided on Learning Central as the base versions for your solutions.
4. Set the maximum allowed number of iterations to 100.
5. Set the error tolerance to 1×10^{-5}.

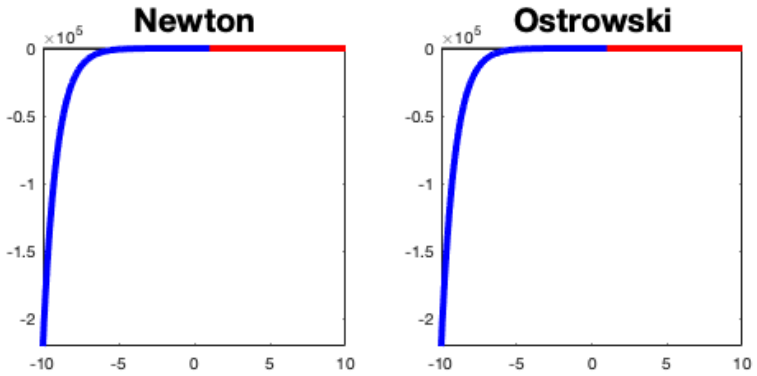
Task 1 (5% weight): `Newton.m` uses this update equation:

$$q_{n-1} = p_n = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})}$$

which we will denote the result of by q_{n-1}. Write a modified version of `Newton.m` to implement Ostrowski's method called `Ostrowski.m`, which improves the convergence rate of Newton's method by replacing the update with the following update equation:

$$p_n = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})} \cdot \frac{f(p_{n-1}) - f(q_{n-1})}{f(p_{n-1}) - 2f(q_{n-1})}.$$

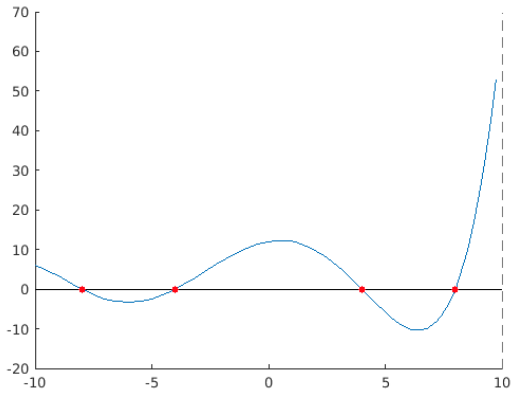
Task 2 (15% weight): Write a program `visualiseConvergence1.m` that uses `Newton.m` and `Ostrowski.m` or modified versions of them. It should plot the given non-linear function $f(x_i)$ colour coded such that plotting $f(x_i)$ blue indicates that initialising the root finding with $p_0 = x_i$ led to the root finding algorithm converging, whereas red indicates that $p_0 = x_i$ led to the root finding algorithm not converging within the maximum allowed number of iterations. An example of the expected output is shown below for $f(x) = xe^{-x}$.



Note:

1. Set the range of x to $[-10, 10]$.
2. Sample the curve at 1000 points.

Task 3 (20% weight): Write a modified version of `Newton.m` called `NewtonMulti.m` to extract multiple roots using the method of **deflation**. Deflation operates by solving for the first root r_1 of $f_1(x) = f(x)$ and then modifying the function to become $f_2(x) = f_1(x)/(x - r_1)$. This process is repeated to extract the set of roots r_i from $f_i(x) = f_{i-1}(x)/(x - r_{i-1})$. After extracting r_i from $f_i(x)$ this solution should be refined by updating the initialisation to $p_0 = r_i$ and rerunning Newton's method on the modified function. (Note that at each iteration the function becomes progressively more complicated.) Plot the function and the positions of the roots as shown in the example shown below where $f(x) = \frac{x^5}{1024} + \frac{3x^4}{256} - \frac{5x^3}{64} - \frac{15x^2}{16} + x + 12$.



Note:

1. Plot output in the range of x in $[-10, 10]$.
2. Look for a maximum of 10 roots. If less than 10 roots can be found then (automatically) stop the root finding process.

Task 4 (10% weight): Write a modified version of `Bisection.m` called `BisectionInitialise.m` that improves the initialisation of the Bisection Method. Given the initial range $[x_{min}, x_{max}]$ repeatedly subdivide this range to find a range that is suitable for the Bisection Method; i.e. $f(x_{min})$ and $f(x_{max})$ should have opposite signs. To be efficient first test if $f(x_{min})$ and $f(x_{max})$ have suitable signs, and if not then test the midpoint $f(\frac{x_{min}+x_{max}}{2})$ against $f(x_{max})$. Repeat this process for each of the subintervals until a suitable initialisation is found or a pre-specified number has been considered.

Note:

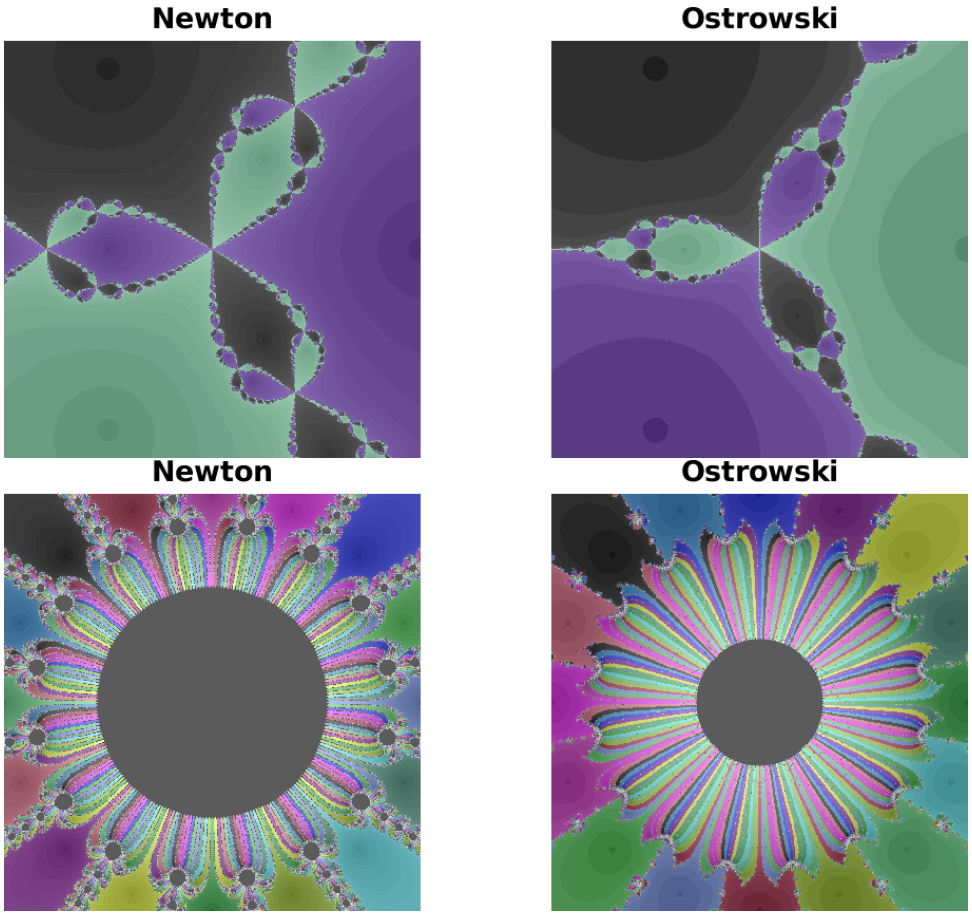
1. Stop subdividing the intervals once they have been reduced to $\frac{1}{2^{20}}$th of the original range.

Task 5 (10% weight): Combine your methods developed in Tasks 1 and 4 to produce a more effective root finding algorithm that 1/ does not require careful specification of the bounding range for the root, and 2/ is efficient. Your program `RootFindingImproved.m` first runs `BisectionInitialise.m` followed by 5 iterations of `Bisection.m`, to produce a solution that is refined by `Ostrowski.m`. This program just needs to find a single root.

Task 6 (30% weight): Write a program `visualiseConvergence2.m` that uses `Newton.m` and `Ostrowski.m` or modified versions of them. Whereas the previous tasks assumed that the function was real valued, this task will use complex values for x and the values of $f(x)$. Consider $x = a + bi$, `Newton.m` can be run without change. Create a visualisation of the basins of convergence for the root finding methods using $x_0 = a + bi$ over a range of a and b values used for initialisation.

Consider 1/ the values of the roots and 2/ the number of iterations n, required to find a root. For the former, assign a unique colour to each of the roots detected and allocate this colour to each pixel that converges to this root. For the latter, perform log mapping, i.e. $\log(n)$, and rescale to the full dynamic range. Create images for these two elements of visualisation, and output the result of blending the two images with equal weight (i.e. average).

An example of the expected output is shown below for $f(x) = (x^3 - 1)(x^2 + 2)e^{-x}$ and $f(x) = x^{16} - 1$.



Note:

1. Set the range of x to $[-1, 1]$.
2. Set the sampling rate of a and b such that the image dimensions are 400×400.
3. If the root finding method does not converge then set the colour to mid gray.
4. When checking for duplication of roots use a high tolerance threshold of 1×10^{-3} to avoid proliferation of roots.
5. The colours do not need to be consistent across different root finding methods.

Please note: You are allowed to use the Geometric Processing toolbox introduced in the module and third-party libraries, as long as you clearly reference the sources in your report.

You must supply a report on your submission which provides a short written description (1–2 pages of text plus diagrams, screenshots etc.) conveying all the appropriate information to demonstrate its operation.

Learning Outcomes Assessed

1. Demonstrate an awareness of basic Scientific Computing with MATLAB
2. Demonstrate an awareness of basic Numerical Analysis Techniques with MATLAB

Criteria for assessment

Credit will be awarded against the following criteria.

1. 5% – Ostrowski's method (Task 1)
2. 15% – Visualise convergence in 1D (Task 2)
3. 20% – Extract multiple roots using Newton's method with deflation (Task 3)
4. 10% – Improved initialisation of the Bisection method (Task 4)
5. 10% – Combined method for robust and effective root finding (Task 5)
6. 30% – Visualise convergence in complex plane (Task 6)
7. 10% – Report describing the operation of your program and your extension of the basic algorithm

and the amount of credit will be awarded according to the following indicators:

1. 1st: the submission fully addresses the stated requirement for the Part, as well as meeting the *excellence indicators* below.
2. 2.1: the submission fully addresses the stated requirement for the Part, but has weaknesses in terms of the *weakness indicators* below.
3. 2.2: the submission partially addresses the stated requirement for the Part.
4. 3rd: the submission minimally addresses the stated requirement for the Part.
5. Fail: the submission does not adequately address the stated requirement for the Part.

Factor	**Weakness indicator**	**Excellence indicator**
Approach	Does not adopt a professional or defensible approach	Adopts appropriate methods with full justification for the choices made
Insight and understanding	Little or no insight and understanding	Has developed considerable insight and understanding

---

Feedback and suggestion for future learning

Feedback on your coursework will address the above criteria. Feedback and marks will be returned on Monday 24th May 2021 via email.