Cardiff School of Computer Science and Informatics

Coursework Assessment Pro-forma

Module Code:	CM2208
Module Title:	Scientific Computing
Lecturer:	Prof David Marshall, Prof. Paul Rosin
Assessment Title:	Resit Coursework
Assessment Number:	Resit
Date Set:	Monday 19th July 2021
Submission date and Time:	Monday 9th August 2021 at 9:30am

This coursework is worth 100% of the total marks available of the Resit 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`
Source code	Compulsory	Two or more Matlab files packaged as a *single* zip file **per question**. Clearly delimit the sub-part solutions in Question 1 also.	`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

---

Questions

Question 1: 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.

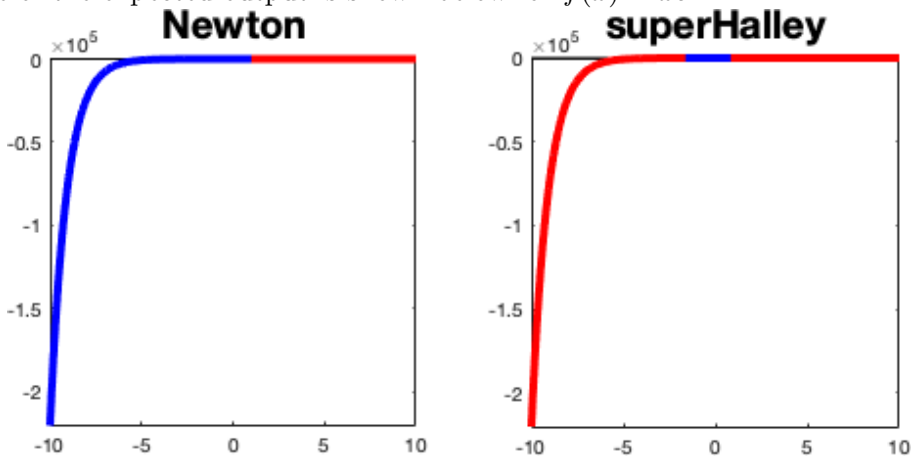
Note (applying to the following tasks):

1. This assignment does **not** involve writing a GUI (e.g. using AppDesigner).
2. Use the code for `Newton.m` that is provided on Learning Central as the base versions for your solutions.
3. Set the maximum allowed number of iterations to 100.
4. Set the error tolerance to 1×10^{-5}.

Task 1 (5 Marks): Write a modified version of `Newton.m` to implement an extended version of Halley's method called `superHalley.m`, that further improves the convergence rate of Newton's method by incorporating the second order derivative, replacing the update with the following update equation:

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

Task 2 (15 Marks): Write a program `visualiseConvergence1.m` that uses `Newton.m` and `superHalley.m` or modified versions of them. It should plot any 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 $f(x) = xe^{-x}$.



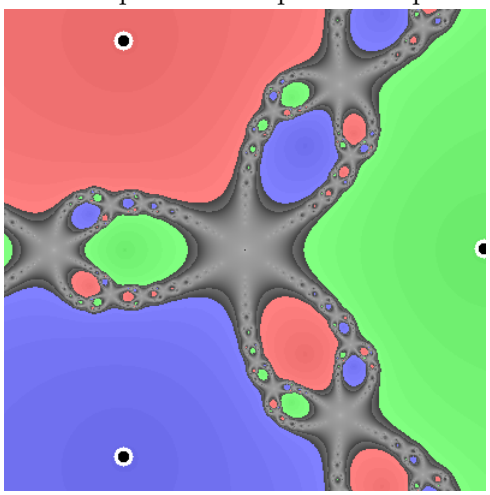
Note:

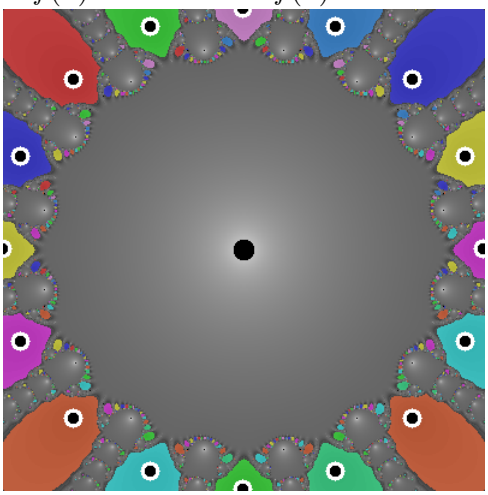
1. Set the range of x to $[-10, 10]$.
2. Sample the curve at 1000 points.

Task 3 (30 Marks): Write a program `visualiseConvergence2.m` that uses `Newton.m` or a modified version of it. 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:

- Assign a unique colour to each of the roots detected and allocate this colour to each pixel that converges to this root. If the root finding does not converge then set the colour to mid gray. Create an image for this element of the visualisation.
- Create another image for the second element of the visualisation. At each pixel that the root finding converges at, set the image value to the number of iterations required to find the root. For those pixels as which the root finding does not converge assign the pixel value to be the absolute difference of the estimated solutions at the last two iterations of Newton's method, $\epsilon = p_n - p_{n-1}$, and perform log mapping, i.e. $\log(\epsilon)$. Then rescale this image to a fixed dynamic range $[0, 1]$.
- The final output should be the result of blending the two images with equal weight (i.e. average). Also, highlight in the image (e.g. using a circle as shown below) the solutions to the equation.

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





Note:

1. Set the range of a and b to $[-1, 1]$.
2. Set the sampling rate of a and b such that the image dimensions are 400×400.
3. When checking for duplication of roots use a high tolerance threshold of 1×10^{-3} to avoid proliferation of roots.
4. Set the maximum allowed number of iterations to 10 (not 100).

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.

Question 2 (50 Marks): Implement an M**-fold wah-wah** audio effect in MATLAB. Using the Wah-wah code discussed in lectures as starting point:

- *http://www.cs.cf.ac.uk/Dave/Multimedia/Lecture_Examples/ Digital_Audio_FX/wah_wah.m*
- See also Week 3 Lab sheet and the solutions for this lab in Learning Central: *http://users.cs.cf.ac.uk/Dave.Marshall/CM2208/Labs/labwk3.mlx*
- You could also base your solution on code developed for the Graphic Equaliser coursework.

Implement an M**-fold wah-wah** effect which creates M delayed bandpass filters with different centre frequencies and modulate each centre frequency as with the single wah-wah.

The submitted solution should include:

- A basic M-fold wah-wah that can be controlled with a variety of parameters:
  - It should allow for a variable number of M
  - The Wah-Wah modulation rate
  - The positioning of filter bandpass centre frequencies and bandwidths
- Some example MATLAB code to show the code working
  - Include all example audio media with your submission.

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% – superHalley's method (Q1, Task 1)
2. 15% – Visualise convergence in 1D (Q1, Task 2)
3. 30% – Visualise convergence in complex plane (Q1, Task 3)
4. 50% – M-fold Wah-wah method (Q2)

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 Learning Central early September 2021