Notes:

- Using symbolic math operations like those in Matlabs symbolic math toolbox or the package sympy in python should be avoided for numerical work especially when they are iterated as they slow things down a lot. In Matlab functions for use iterating algorithms should be defined either using something like `function [y1,...,yN] = myfun(x1,...,xM) ... end` or `myfun = @(x1,...,xM)...;`.
- The Matlab command `error(...)` should also be avoided in this course as it crashes out of the script. Some peoples script for Q1 did not run because of this.
- Note that there are no marks for turning your script into a general piece of root finding software your time is better spent producing a script that the grader can run by just pressing run than producing a complicated function that takes general inputs.

1 Question 1:

The Matlab script I used to solve this question is `ModelSolHwk1Q1.m`*. This can be run by loading it into Matlab and hitting run.*

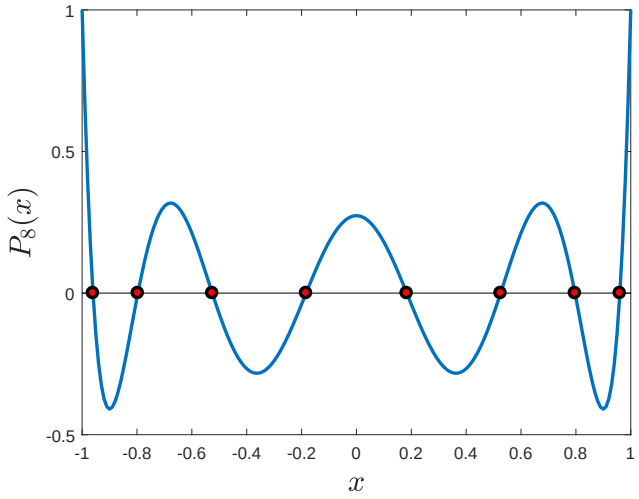
The goal is to find the roots of the polynomial

$$P_8(x) = \frac{1}{128}(6435x^8 - 12012x^6 + 6930x^4 - 1260x^2 + 35). \tag{1.1}$$

Which has the derivative

$$P_8'(x) = \frac{1}{128}(51380x^7 - 72072x^5 + 27720x^3 - 2520x). \tag{1.2}$$

$P_8(x)$ is plotted below and the roots are indicated.



We see that there are eight roots. There are multiple approaches one can take to finding these in the example code we divide the region $-1 < x < 1$ into $n = 20$ parts and the preform a hybrid Bisection/Newton's method root finding on any region where the sign of $P_8(x)$ is different on each end of the region. For this to work n needs to be small enough so that each region has at most one root.
The hybrid method takes three steps using the bisection method then uses the result as a starting point for the Newton's method. The roots found using this method are shown in the figure above and are,

$$-0.9603 \quad -0.7967 \quad -0.5255 \quad -0.1834 \quad 0.1834 \quad 0.5255 \quad 0.7967 \quad 0.9603. \tag{1.3}$$

In some version of the homework the coefficient of the x^8 term is different $6435 \to 6425$ in this case we find the roots

$$-0.9638 \quad -0.7942 \quad -0.5257 \quad -0.1834 \quad 0.1834 \quad 0.5257 \quad 0.7942 \quad 0.9638. \tag{1.4}$$

2 Question 2

The Matlab script I used to solve this question is `ModelSolHwk1Q2.m`*. This can be run by loading it into Matlab and hitting run.*

Solving the two equations stated in the question is equivalent to solving the vector equation

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} f_1(x,y) \\ f_2(x,y) \end{pmatrix} = \begin{pmatrix} x^2 - 2x - y + \frac{1}{2} \\ x^2 + 4y^2 - 4 \end{pmatrix} = 0 \tag{2.1}$$

where $\mathbf{x}$ is the vector

$$\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix}. \tag{2.2}$$

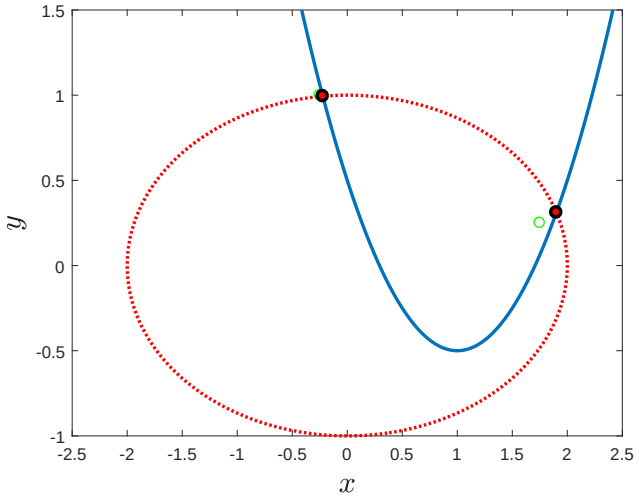
In Newton's method the $n+1$'th approximation $\mathbf{x}_{n+1}$ to the root can be written in terms of the n'th approximation $\mathbf{x}_n$,

$$\mathbf{x}_{n+1} = \mathbf{x}_n - [\mathbf{J}(\mathbf{x}_n)]^{-1}\,\mathbf{f}(\mathbf{x}_n) \tag{2.3}$$

where $\mathbf{J}(\mathbf{x}_n)$ is the Jacobian matrix

$$\mathbf{J}(\mathbf{x}_n) = \begin{pmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{pmatrix} = \begin{pmatrix} 2x - 2 & -1 \\ 2x & 8y \end{pmatrix}. \tag{2.4}$$

The curves $f_1(x,y) = 0$ (solid blue line) and $f_2(x,y) = 0$ (dotted red line) are shown in the graph below



from which we can see there are two roots (filled circles in the graph) and we can make first estimates for each of the roots. Root one $(x,y) \approx (1.75, 0.25)$ and $(x,y) \approx (-0.25, 1)$ which are shown as empty green circles on the graph. Running the algorithm finds the roots

$$\text{Root 1: } (x,y) \approx (1.9007,\ 0.3112) \tag{2.5}$$

$$\text{Root 2: } (x,y) \approx (-0.2222,\ 0.9938) \tag{2.6}$$

Marking Schedule

Marks were awarded for the following

- 2 marks for the code running efficiently. I had to be able to run the code and get the results you got. One mark for each question. One mark of these marks was lost when the code was really slow because symbolic functions were used in the iterating steps.
- 1 mark for the write up. Ideally this would look like sections one and two of this document, it should briefly explain what you've done why you've done it and give your results.

Question 1:

- 1 mark for code that correctly implemented the bisection algorithm.
- 1 mark for have a sensible criteria for changing over between the bisection method and the Newton's method. This meant either the bisection method should have a large tolerance than the Newton's method or equivalently (like in the solutions) only a small number of bisection steps should be taken before moving on to Newton's method.
- 1 mark for code that correctly implemented Newton's method.
- 1 mark for code that found all 8 roots on the specified interval.
- 1 mark for having the correct results (either coming from the code or in the write-up preferably both).

Question 2:

- 1 mark a graph that can help you guess the two solutions.
- 2 marks for code that correctly implements a multivarible Newton's method.
- 1 mark for the correct solutions (both of them).