Computational Physics HW7

Ben Johnston

November 2023

1 Problem 1

This problem involved implementing Brent's 1D minimisation method on the function given by:

$$y = (x - 0.3)^2 e^x \tag{1}$$

This minimisation method involves combining both parabolic minimisation and golden section search methods to minimise the function in question. Parabolic interpolation searches for an extreme point, x_{min} by fitting a second-degree polynomial to the function y over an interval that bounds x_{min}. The equation used to iterate towards a solution here is given by:

$$x = b - \frac{1}{2}\frac{(b-a)^2[f(a) - f(c)] - (b-c)^2[f(b) - f(a)]}{(b-a)[f(a) - f(c)] - (b-c)[f(b) - f(a)]} \tag{2}$$

The golden section search algorithm is a method in which the minimum is successively bracketed down with increasing iteration number where we have the golden section given by:

$$w = \frac{3 - \sqrt{5}}{2} \approx 0.382 \tag{3}$$

Brent's method for minimisation uses parabolic approximations, but it keeps track of a bracketing interval, and reverts to golden section search under the following conditions:

1. The parabolic step falls outside the bracketing interval.
2. The parabolic step is greater than the step before last

This minimisation method was then carried out on the function given in Eq(1) - this minimised value was calculated to be $x_{min} = 0.3000000044095357$. As the analytical solution to the minimum of the solution can be calculated as $x_{min} = 0.3$ it can be said that the implementation of Brent's 1D minimisation was said to be successful. *Figure 1* shows a plot of the minimisation process as a function of iteration number:

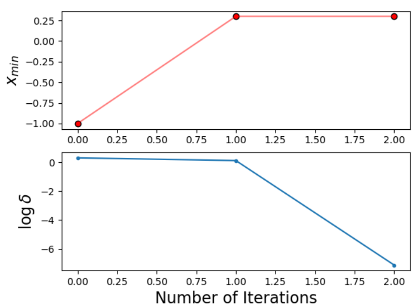

Figure 1: Convergence plot for minimisation process

After this minimisation process was implemented, the same function was minimised using SciPy's optimize.brent function. The resulting minimised value for the function was $x_{min} = 0.300000000023735$. Upon comparison of the values determined using both methods it can be said that they agree very well.

Figure 2 shows the function itself with the associated calculated minimised values for both methods plotted. Going from left to right on *Figure 2* zooms in further on the minimisation point for both methods:

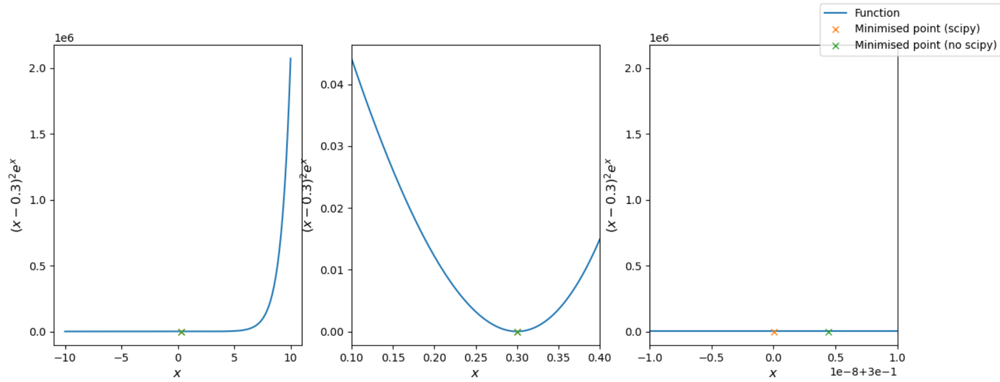

Figure 2: Function with associated minimisation points

2 Problem 2

This problem involved solving a simple likelihood problem. A 'survey' was taken from the population, asking people a simple yes or no question, namely "Do you recognize the phrase 'Be Kind, Rewind', and know what it means?". You have a hypothesis that whether people answer yes should depend on age. The standard way people analyze the results to look for a correlation in situations like this is something called logistic regression. You model the probability as the logistic function:

$$p(x) = \frac{1}{1 + \exp[-(\beta_0 + \beta_1 x)]} \tag{4}$$

Figure 3 is a plot showing values for the log-likelihood for varying values of β_0 and β_1:

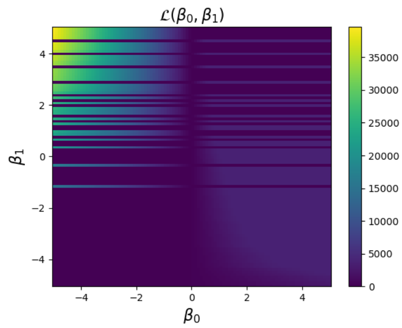

Figure 3: Log-likelihood as plotted using various values of β_0 and β_1

A script was then written to minimise the negative log likelihood using scipy.optimize.minimize to perform a least squares fit in order to find the optimal parameters for β_0 and β_1. These values were determined to be:

1. $\beta_0 = -5.62023227 \pm 0.02396866$
2. $\beta_1 = 0.10956339 \pm 0.00299568$

Using these parameters the logistic function was plotted as a function of age - this can be seen in *Figure 4* below:

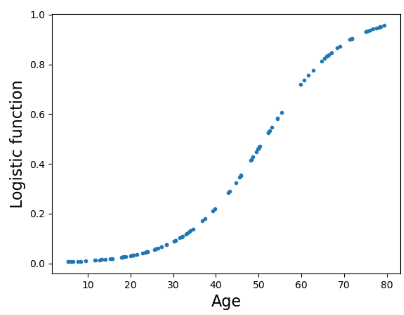

Figure 4: Logistic function as a function of age for optimised parameters

The covariance matrix of β_0 and β_1 was then determined to be:

$$C = \begin{pmatrix} 5.74496821e-04 & -7.17954457e-05 \\ -7.17954457e-05 & 8.97411154e-06 \end{pmatrix} \tag{5}$$

This has a condition number of 336225.1548420219, which is quite high.