Universitat Autònoma de Barcelona

Optimitzation

Master's Degree in Modelling for Science and Engineering

Unconstrained optimization
Newton's method

Daniel Bedmar Romero
(NIU:1565494)

14 December 2024

1 Newton's method implementation

The Newton method is an iterative approach used to find the minimum of a given function $f(x)$. Starting from an initial guess, the subsequent values of x are calculated using the formula:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \tag{1}$$

where $f'(x)$ represents the derivative of f evaluated at the point x.

This process is repeated until consecutive values fall within a predefined neighborhood, determined by an error threshold, or when the derivative of the function at the point is sufficiently close to zero. To prevent indefinite iterations, a maximum number of iterations is typically specified.

Therefore, the implementation requires as inputs; an initial guess, the maximum number of iterations and tolerances for the error thresholds. The algorithm proceeds as follows:

- Initially, the iteration counter and a matrix to store the subsequent guesses and their values are initialized. The matrix is pre initialized with a size equal to `MAXit` $\times$ 2 to avoid memory reallocation during runtime.
- Next, a loop iteratively updates the value of x, appends the value of x and its new corresponding value to the matrix, and checks if any of the stopping criteria are met.
- Once the loop is finished, the matrix is truncated to match the number of iterations performed, freeing unused memory.
- Finally two plots are generated, a plot of the function and the different approximations made and a plot showing the convergence of the method by plotting the distance of consecutive points.

The extension to n dimensions is relatively straightforward. In essence, the n-dimensional equivalent of Equation 1 involves utilizing the gradient and the inverse of the Hessian matrix so the subsequent values of x are calculated using the formula:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \end{bmatrix} - H^{-1}\nabla f(x_k, y_k) \tag{2}$$

Errors are calculated by taking the norm of the difference vector or the gradient. Finally, the matrices are resized to release unused memory, and the error evolution is visualized.

2 Univariate root finding

To fully understand how the Newton's method converges in different scenarios I've chosen the function $2x^3 - 3x^2 + 3$, which has a solution at $x = -0.80644$ and a local minimum at $x = 1$ and then selected three different points:

The implementation in MATLAB can be found in the following github repository [1].

x = -0.5

This point is near the solution and it is not located in the convex valley of the local minimum. With this point I intend to see how this method work in a simple and easy scenario.

As expected, applying Newton's method to the function $f(x) = 2x^3 - 3x^2 + 3$ demonstrates fast convergence (Figure 1). The method converged in just 4 iterations, showcasing the characteristic quadratic convergence of Newton's method when the initial guess is sufficiently close to the root.

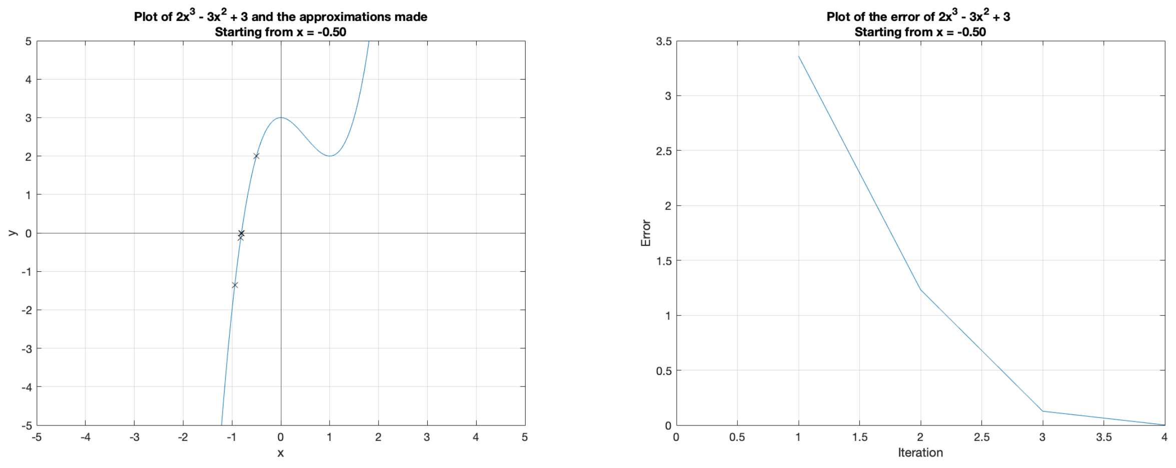

Figure 1: Behaviour of Newton's method choosing a value that is close to the minimum and there is no local minimum in between

$\mathbf{x = 0.5}$

This point is also near the solution but is inside the convex valley of the local minimum. With this point I intend to understand how the method behaves when it is near a delicate zone as the local minimum.

We can observe (Figure 2) that initially the method exhibit instability by large jumps and oscitations in the convergence. This behaviour arises from the small derivative values near the local minimum, which cause Newton's formula to produce large updates. This ultimately results in an overshoot in the 5th iteration, which then enables the posterior stabilization that progressively converges to the root, even though in an oscillating way and with significantly more iterations than the last example.

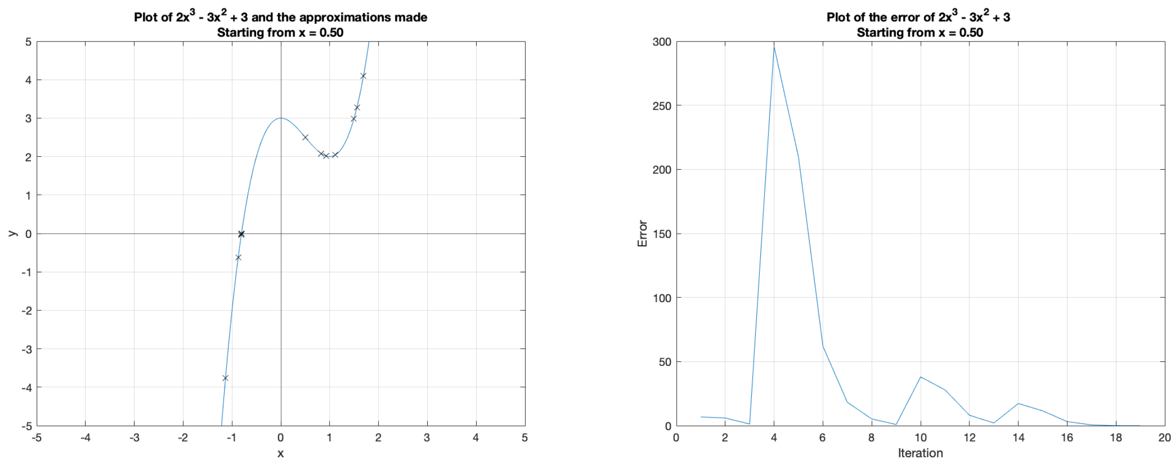

Figure 2: Behaviour of Newton's method choosing a value that is closer to a local minimum that to the absolute minimum

x = 100

I selected this point far from both the local minimum and the solution to analyse whether the presence of a local minimum affects Newton's method convergence. The local minimum has a smaller magnitude compared to the overall scale of the function. This experiment aims to determine if Newton's method can bypass the local minimum efficiently and still converge to the correct root.

The results, shown in Figure 3, highlight that despite the local minimum being between the starting point and the solution, the trajectory of Newton's method indicates smooth progress toward the root without any apparent oscillations or instabilities, as the error decreases exponentially, with no irregularities, demonstrating Newton's method's characteristic quadratic convergence. This is because the global behaviour of the function dominates over the magnitude of the local minimum.

Even thought 22 iterations are needed to meet the convergence criteria, by iteration 6, the error is close to zero, confirming efficient convergence.

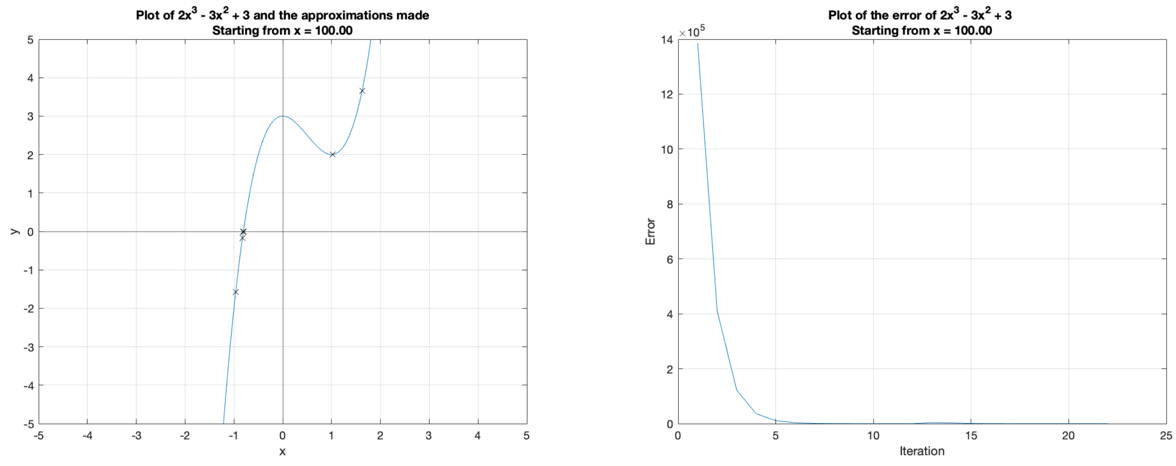

Figure 3: Behaviour of Newton's method choosing a value that is far from both local and absolute minimums

3 Bivariate minimum finding

In order to study the implementation of Newton's method for finding minimums in a bivariate function I've decided to use the Rosenbrock function defined as:

$$f(x, y) = 100(x^2 - y)^2 + (1 - x)^2 \tag{3}$$

I've chosen this function because it has several local minimums but just one global minimum in $x = 1$,$y = 1$ and they are all situated in very pronounced valleys, which will allow us to observe if we are able to find the true global minimums and how the method behaves with complex and steep surfaces.

The implementation in MATLAB can be found in the following github repository [1].

x = -1, y = 1

I selected this point as it is in a relative minimum to observe whether Newton's method can escape this local feature and converge to the global minimum.

Based on the results (Figure 4, Newton's method successfully overcomes the local minimum and converges rapidly toward the global minimum, as expected.

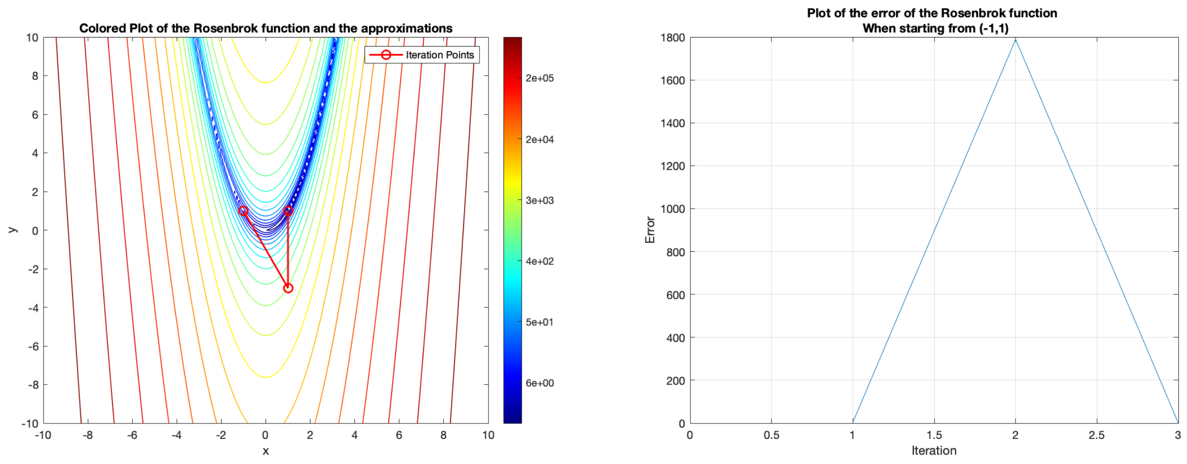

Figure 4: Behaviour of Newton's method when it is started in a relative minimum

x = 9, y = -8

I selected this starting point at a high value of the Rosenbrock function to observe the convergence behaviour of Newton's method.

Initially, the method converges to a local minimum due to the gradient and Hessian. However, it successfully escapes this local minimum again and continues to converge rapidly toward the global minimum, demonstrating its capability to navigate complex landscapes. this can be observed in Figure 5.

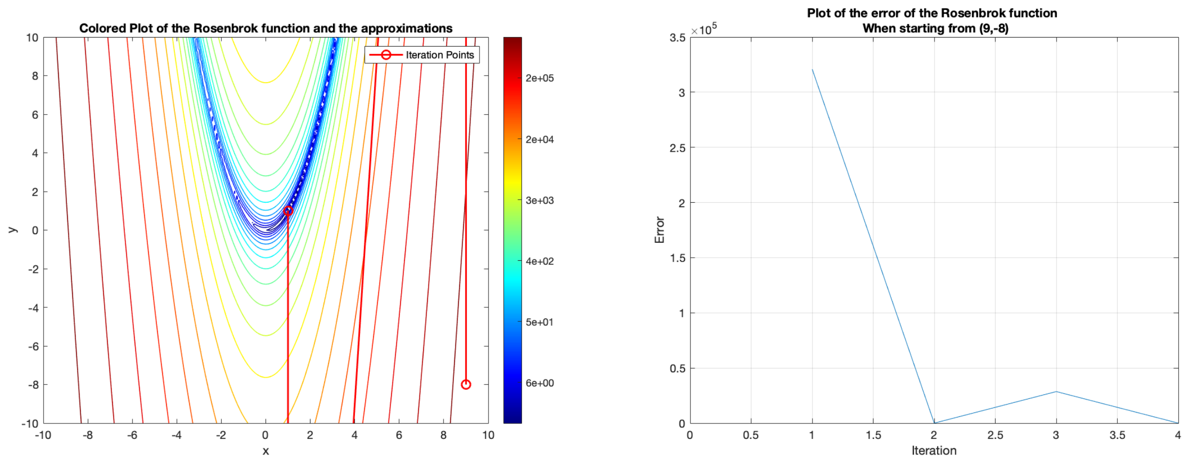

Figure 5: Behaviour of Newton's method when it is started in a point far from the minimum

x = 934, y = -835

Starting from an even far away point I wanted to confirm how the Newton's method behaves in more extreme cases.

The results in Figure 6 demonstrates the robustness of Newton's method. The algorithm quickly identifies the curvature-dominant region and converges to the global minimum in just a few iterations. The error decreases exponentially, reflecting the method's quadratic convergence.

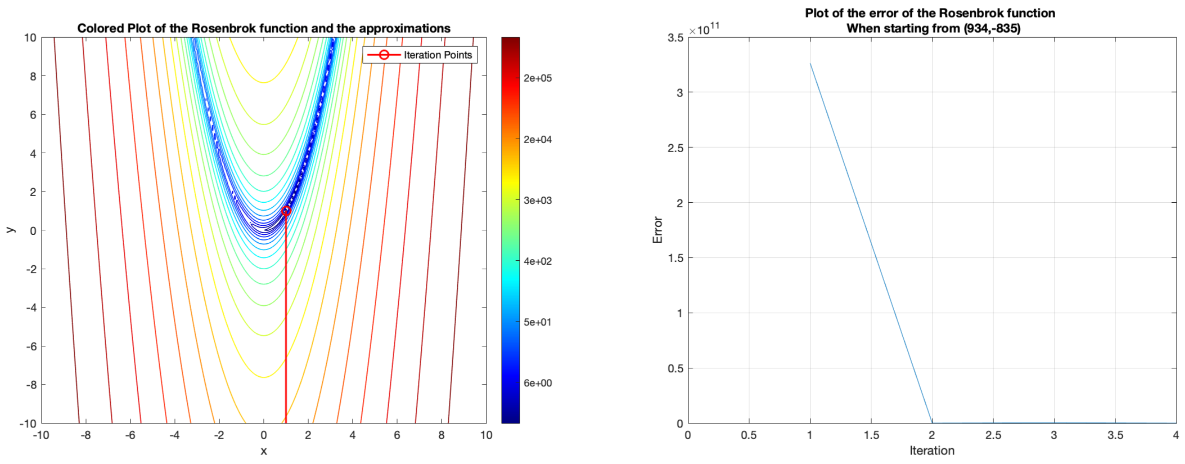

Figure 6

4 Conclusions

After these different observations, some clear conclusions can be made:

- **Strong dependency on the initial guess:** A major drawback of Newton's method is its incredible sensitivity to the initial guess. For example, when initialized near a local minimum, the method tends to converge to that local minimum before escaping and converging to the real solution or the global minimum.

  This behavior can clearly be seen in Figure 2 and Figure 4.

- **Fast convergence in well-conditioned regions:** One of the strengths of Newton's method is its quadratic convergence when the method is in a region where the curvature is well-defined. Even if the point is far from the solution, the method requires very few iterations to reach it.

  This behavior is clearly demonstrated in Figure 3 and Figure 6.

- **High computational cost:** Although using this method for 1D or 2D cases does not present a major computational challenge, applying it to higher-dimensional scenarios could become problematic. This is because the computational cost of calculating the Hessian and solving the linear system can be much higher.

In my opinion, Newton's method is a good option for low-dimensional scenarios where the function is well-behaved and where the curvature strongly governs the function. In more complex cases, where choosing an appropriate initial guess is crucial for convergence, Newton's method may not be the optimal choice.

References

[1] "Github repository with code." `https://github.com/DaniBedmar/Modeling_Master/tree/main/Optimization/Newton_Method`.