Project 4
Non-linear systems of equations and the Newton-Raphson method

Group 3 - Team 3

Manager : Lucas Guédon
Secretary : Salim Bekkari
Programmers : Mohammed Boudali, Imad Boudroua, Simon Bullot

Abstract : The aim of the project is to implement the Newton-Raphson method to solve non-linear systems of equations and apply it to problems.

1 Newton-Raphson method

By Lucas Guédon

The goal of this section is to implement a solver of non-lineal systems of equations. The equations are in the form of $f(X) = 0$, where $f : R^n \to R^m$ is differentiable. To achieve this, the Newton-Raphson method was used.

1.1 Algorithm

Given a starting point U_n, the objective is to find a point supposedly closer to a root than U_n. The new point $U_{n+1} = U_n + V$ is a good approximation of the root, so $f(U_{n+1}) \approx 0$. The Taylor expansion of f around U_n allowed us to find that $f(U_{n+1}) = f(U_n) + H(U_n) \times V$, thus $H(U_n) \times V = -f(U_n)$. This linear system of equations could be solved using numpy.linalg.solve, but the latter was replaced with numpy.linalg.lstsq in order to avoid numerical problems related to singular matrices. The process is repeated until $||f(U_n)|| < \epsilon$ where ϵ is the desired precision.

1.2 Backtracking

In some cases, the previously shown algorithm diverges, backtracking avoids this. The function $\Phi(U) = U^T U$ was added to evaluate the approximations. When computing U_{n+1}, if $\Phi(U_n+V) < \Phi(U_n)$ then $U_n + V$ is an approximation worse than U_n. In this case, V is multiplied by the value *step*, with $0 < step < 1$. This process is repeated until the approximation is better than U_n.

1.3 Testing

In order to test the algorithm, two simple test cases were created. The first finds one root of the second-degree polynomial $f(x) = x^2 - 2x + 1$ using our algorithm. The expected result is 1. The result given by the algorithm is 0.998, which has a relative error of 0.002.

The second case solves a system with multiple variables and equations :

$$\begin{cases} x^2 + y = 0 \\ x + 3y + 4 = 0 \end{cases}$$

This system is represented by the function $f(x, y) = (x^2 + y, x + 3y + 4)$. The algorithm found the solution $x = 1.333, y = -1.778$.
To verify whether this is a correct solution, f was applied to x and $y : f(x, y) = (1.930 \times 10^{-12}, 4.441 \times 10^{-16})$. The result was almost $(0, 0)$ so the root was correct.

To test the efficiency of backtracking, the algorithm was applied to the function $f(x) = x^3 - 2x^2 + 1$ with a starting point of 0.01. This is a special edge case because f(0.01) is close to 1 and f'(0.01) is close to 0. During the first iteration, $V = \frac{f(0.01)}{f'(0.01)}$ which means that U_1 moved away from the solution. This can be seen in figure 1.

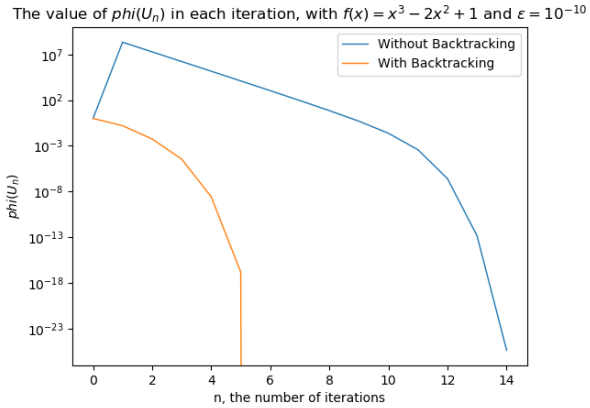

FIGURE 1 – Comparison of the value of $\Phi(U_n)$ with and without backtracking

2 Computation of the Lagrangian points

By Simon Bullot

In celestial mechanics, the Lagrangian points are points near two objects where a lighter object will remain at the same place relatively to the two other masses. The Newton-Raphson method is used to find these points.

2.1 Forces implementation

First, a function which takes two parameters will return another function representing centrifugal, elastic or gravitational force. For instance, a function g will return the centrifugal force function corresponding to the parameters k and X_0.

$$g : k, X_0 \mapsto f with f : X = (x, y) \mapsto (k(x - x_0), k(y - y_0))$$

Then another function would return Jacobian matrix for any of the forces needed.

2.2 Newton-Raphson method application

A third function will represent the sum of the forces by adding the desired gravitational and centrifugal forces. The same is done by another function for the matrix representing Jacobian forces sum matrix. Then the Newton-Raphson algorithm is used to find a point where forces are equal to zero. A point where sum forces are equal to zero will be an equilibrium point. In fact, due to Newton's first law, if an object does not have any velocity, and the forces acting on it are compensating themselves, then it will not move.

2.3 Obtaining the Lagrangian points

By running the Newton-Raphson algorithm with different U_0 points, different Lagrangian points can be found. For instance, if those points are used :

$$U_0 = (0.5, 0), U_1 = (1.5, 0), U_2 = (-1.5, 0)$$

The algorithm will respectively find these points :

$$L_0 = (0.85927766, 0), L_1 = (1.15775715, 0), L_2 = (-0.99754112, 0).$$

3 Electrostatic equilibrium

By Imad Boudroua and Mohammed Boudali

The purpose of this section is to show the equilibrium position of N charges placed in an electrostatic field. The movement studied is limited between -1 and 1 on the real axis.

3.1 Algorithm to compute the Jacobian matrix

The first step is to compute the derivative of total electrostatic energy function :

$$E(x_1, .., x_N) = \sum_{i=1}^{N} (\log(|x_i + 1| + \log|x_i - 1| + \frac{1}{2} \sum_{\substack{j=1 \\ j \neq i}}^{N} \log|x_i - x_j|) \tag{1}$$

After derivation, the expression found is :

$$\frac{\partial E(x_1, .., x_N)}{\partial x_i} = \frac{1}{x_i - 1} + \frac{1}{x_i + 1} + \sum_{\substack{j=1 \\ j \neq i}}^{N} \frac{1}{x_i - x_j} \tag{2}$$

Therefore, the elements of Jacobian matrix : $(\frac{\partial \nabla E_i}{\partial x_j})_{i,j}$ can be simplified to :

$$J_{i,j} = \begin{cases} -\frac{1}{(x_i-1)^2} - \frac{1}{(x_i+1)^2} - \sum_{\substack{j=1 \\ j \neq i}}^{N} \frac{1}{(x_i-x_j)^2} & \text{if } i \neq j \\ -\frac{1}{(x_i-x_j)^2} & \text{if } i = j \end{cases}$$

3.2 Application of Newton-Raphson method

After using the Newton-Raphson method to solve the equation : $\nabla E(x_1, .., x_n) = 0$, the variation of energy gradient was plotted (Figure 2) :

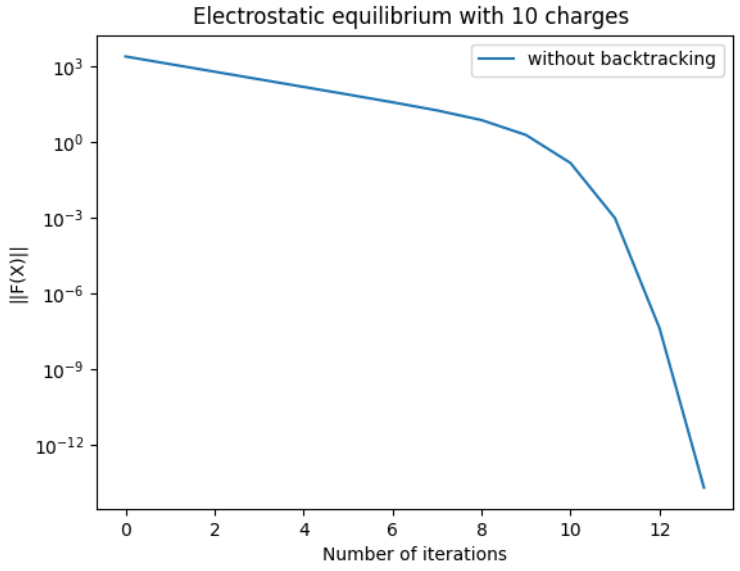

FIGURE 2 – Electrostatic equilibrium with 10 charges

In Figure 2, the curve converges to 0. This result proves that the system studied reached an electrostatic equilibrium. This property is more noticeable in Figure 3 :

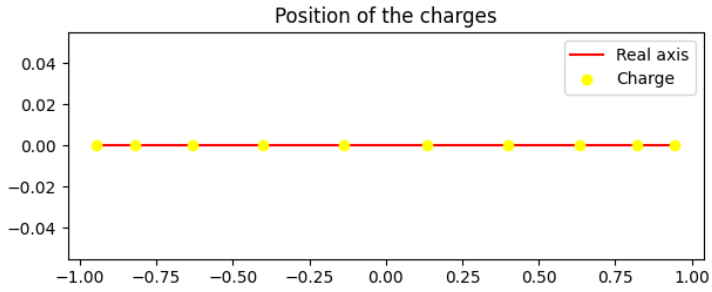

FIGURE 3 – Position of the charges on the real axis

3.3 Comparison with the Legendre polynomial derivative roots

In this section, the functions **legder**, **leg2poly** and **poly1d** were used to draw the curve of Legendre polynomial. Equilibrium positions were plotted in the same figure in order to compare the results (Figure 4) :

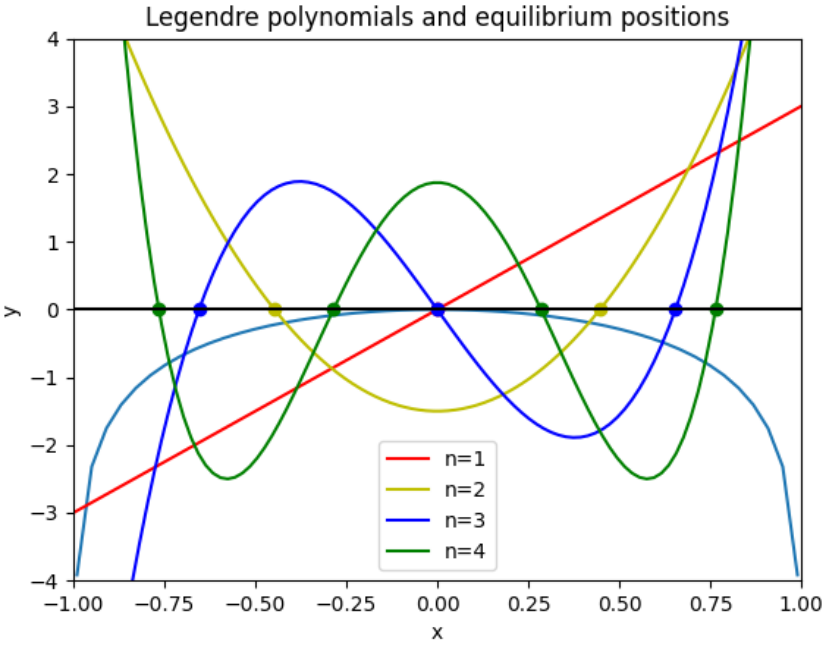

FIGURE 4 – Legendre polynomials and equilibrium positions

This figure shows that Legendre polynomial derivative roots coincide with the equilibrium positions.

3.4 The nature of the extremum

The results show that electrostatic equilibrium positions reached a critical point. To determine its nature, the curve in Figure 5 was drawn.

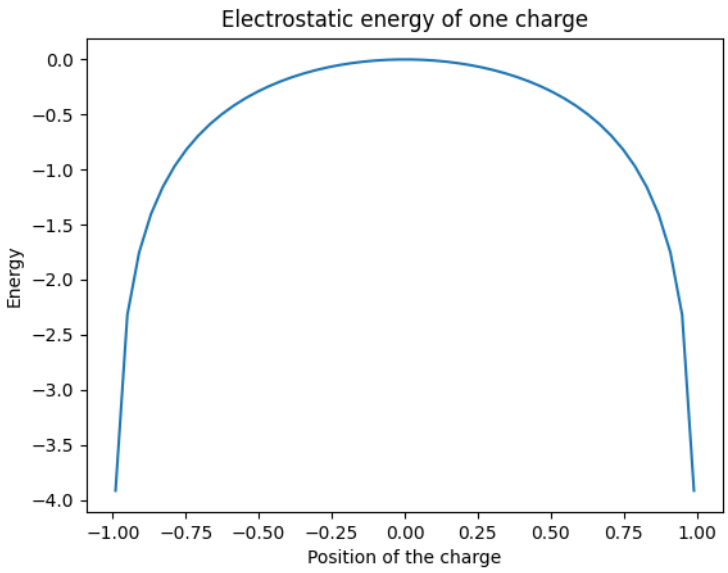

FIGURE 5 – Energy variation of one charge on [-1 ;1]

This curve is characterized by one global maximum reached in $x = 0$.

4 Conclusion

By Simon Bullot, Imad Boudroua and Lucas Guédon

The Newton-Raphson method can be a useful and powerful tool especially with the problems encountered in this project. In those cases, the Jacobian matrix of each function can be easily calculated. This enables the use of this method which works in any dimension. Moreover, the convergence of the algorithm studied is quadratic because roots are non stationary points and do not have multiplicity greater than one.