Project 4
Non-linear systems of equations / Newton-Raphson method

Group 2 - Team 8630
Responsible : MAGON Eloi
Secretary : DECOU Nathan
Coders : DELPEUCH Sébastien RIDACKER Vincent STAN Sophie

Abstract: The idea of this project is to implement nonlinear system solving using the Newton-Raphson method and then apply it to the resolution of concrete problems such as Lagrange points or the elctrostatic equilibrum.

1 Implementation of the Newton Raphson method

Nathan Decou

The first part of the project consists in implementing a method to solve a non-linear system. For this purpose the Newton-Raphson method has been implemented. This method is iterative, it allows to realize successive approximations of the equation $f(x) = 0$. And generalizing about finding a root of a function $f : \mathbb{R}^n \to \mathbb{R}^m$. Let U be a vector, $f(U)$ the application of f to this vector, H the Jacobian matrix of f and a V vector such that $f(U + V) = 0$ then using Taylor-Lagrange decomposition of f we can easily deduce the relationship

$$H(U) \times V = -f(U) \tag{1}$$

This relationship will be the basis of the Newton-Raphson algorithm.

1.1 Implementation of the generic method

To start a basic Newton-Raphson method has been implemented. It does not look at each iteration whether it converges to the solution or not.

1.1.1 Theoretical elements of the Newton-Raphson method

First, let's look at the theoretical operation of the method. It consists in introducing a sequence (x_n) of successive approximation of the equation $f(x) = 0$. The algorithm starts at a x_0 ideally close to the solution. Starting at x_0, the new term x_1 is chosen as follows : the trangent to $\mathcal{C}_f$ is drawn as x_0, this tangent intersects the abscissa axis in x_1. This process is then repeated by calculating x_2 in the same way etc.The algorithm stops when we find a $f(x_n)$ lower than the requested precision or when the maximum iteration number is reached. Mathematically this translates into the following recurrence formula. For any n between 0 and N where N is the maximum iteration number we set and as long as $x_{n+1} > \epsilon$ with ϵ the requested precision

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{2}$$

Level of complexity of the algorithm depends on the input function. Indeed as long as we haven't done N iteration loops and that $f(x_n) > \epsilon$, then the calculation of x_{n+1} is done etc. So, the complexity can be easily bounded by the maximum number of iterations but also by the maximum number of evaluation of f. Finally the complexity is bounded by $N \times O(f)$.

A generalization of Newton-Raphson to n dimension can easily be done. The difficulty is in obtaining the derivative of f at each iteration. This is not made not by a simple derivative but by using the Jacobian of the function at one point of application. Moreover, where the calculation of $\frac{f(x_n)}{f'(x_n)}$ was performed (assumed constant time), we must now use the equation 1. This changes the complexity of the algorithm, in fact, every iteration it is necessary to call upon *scipy.optimize.least_squares*, according to the documentation this function has the same complexity as an SVD decomposition seen in the previous project, i.e. for a $m \times n$ matrix, $O(m^2 n + n^3)$. At . sum the complexity of the Newton-Raphson method in dimension n rises to $O(N \times (m^2 n + n^3) \times O(f))$.

1.1.2 Test set-up for the generic method

Now that we have the methods and the theoretical elements, the tests can be implemented. These tests are intended to highlight the proper functioning of the algorithm. For the implementation we will define two errors. First the forward error. x is the known root of f toward wich x_n is supposed to converge $\delta_f = ||x - x_n||$. The latter porvides us with information on the distance between the solution root and the root that the algorithm is calculating at each iteration. We also define the backward error $\delta_b = ||f(x_n)||$. This one represents the norm of $f(x_n)$ where x_n is the solution given by the algorithm at each iteration. Since we want x_n to be the root of f, this error should converge toward 0.

The graph 1 is thus plotted, the generic Newton-Raphson method is used on the function $f : x \mapsto x^2 - 9$. First some generalities. The backward error (green curve) actually converges to 0. Moreover the forward error (red curve) shows that the algorithm succeeds in finding the solution with a precision of 10^{-7}.

A test on the Newton-Raphson method generalized to the n dimension was performed with the following dimension 2 function $f : (x, y) \mapsto (x^2 - y - 5, x \times y - 12)$ in the figure 2. Analogous to dimension 1 the δ_b curve (green curve) converges towards 0 indicating that the algorithm converges to the solution $f(x, y) = (0, 0)$. And the δ_f curve (red curve) converges also towards 0 indicating that the algorithm is converging to the true solution.

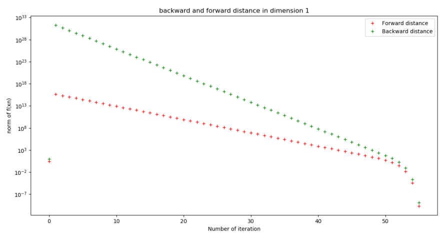

Figure 1: Plotting δ_b and δ_f for the function $x \mapsto x^2 - 9$ using the Newton-Raphson method with a required accuracy of 10^{-7}

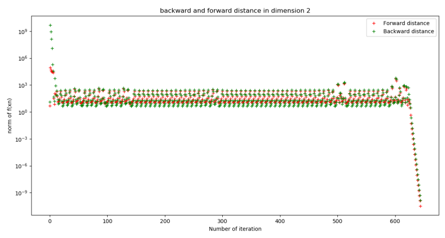

Figure 2: Plotting δ_b and δ_f for the function $(x, y) \mapsto (x^2 - y - 5, x \times y - 12)$ using the generic Newton-Raphson method with a required accuracy of 10^{-9}

In conclusion, it has been shown that Newton-Raphson's method is functional to solve a non-linear system, the latter functions both in dimension 1 and in dimension n and has a speed of quadratic convergence.
Sébastien Delpeuch

1.2 Implementation of the method with backtracking

The generic implementation of the Newton-Raphson method is certainly functional but has a defect. Indeed when the calculation of x_{n+1} is performed, the result can get away from the solution and therefore slow down the algorithm.The idea of this section is to present a solution to this problem. The figures 4 and 3 quickly summarize the operation of newton-raphson with (left) and without (right) backtracing. The right graph show that if a point found by the algorithm moves it away from the solution, then it is not chosen.

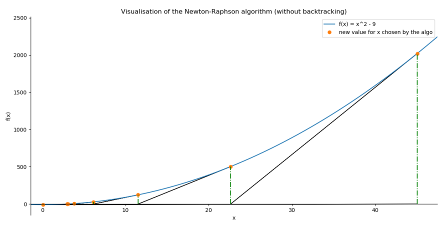

Figure 3: Theorical summarize of Newton-Raphson method without backtracking

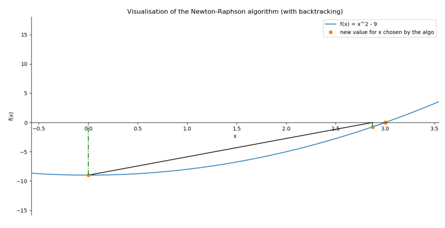

Figure 4: Theorical summarize of Newton-Raphson method with backtracking

1.3 Theoretical elements of the backtracking method

To solve this problem, implementing an algorithm with backtracking is effected. In other words, if the element calculated from the x_{n+1} point away from the solution then it is not chosen and another point is chosen such that $f(x_{n+1}) < f(x_n)$. This makes it possible to be sure that at any moment the algorithm converges towards the solution.

From a complexity point of view, it always depends on the complexity of the function. However, the number of times the function is evaluated (in the case of the generic method the number of evaluation was the number of iteration of the method) varies depending on the number of times backtracking is called.

1.4 Tests and comparison of the two methods

After implementing the algorithm a check of the theoretical elements is imperative. The interest is double, it is necessary to verify that the algorithm can produce a correct solution and check the theoretical gain. The definitions of the forward error δ_f and the bakward error δ_b are reused. The graphs 5 and 6 are then tracings. The curves representing δ_f (right green curve) and δ_b (left green curve) for the solution with backtracking may lead to the same conclusion as before, i.e. the two converge towards 0 (under 10^{-9}) which assures us that the algorithm found correct the solution. Now let's realize the comparison between δ_f for backtracking (right green curve) and for the normal algorithm (right red curve) it is remarkable that the algorithm with bracktracking finds the solution with an accuracy of 10^{-10} in about 600 fewer iterations of the algorithm without backtracking. This proves what has been theoretically advanced, the algorithm with backtracking comes close to the solution much faster and avoids the remarkable oscillations on the curve on the right in red. Analogously, let's look at δ_b for the backtracking (left green curve) and for the normal algorithm (left red curve). This leads to the same conclusions as the forward error.

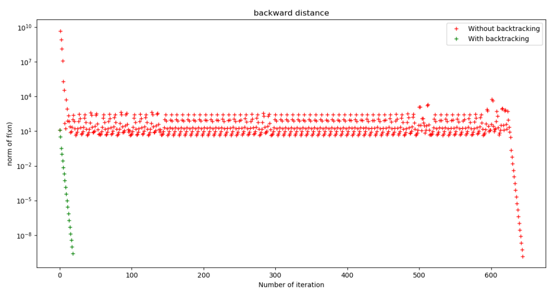

Figure 5: Plotting δ_b for the function $(x, y) \mapsto (x^2 - y - 5, x \times y - 12)$ using the method with and without backtracking with a required accuracy of 10^{-9}

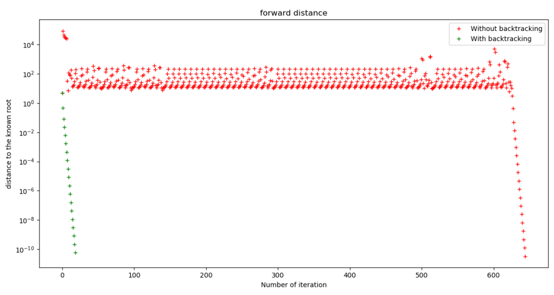

Figure 6: Plotting δ_f for the function $(x, y) \mapsto (x^2 - y - 5, x \times y - 12)$ using the method with and without backtracking with a required accuracy of 10^{-9}

1.5 Limitations of the Method

Before moving on to the application of the Newton-Raphson method some limitations will be discussed. If the algorithm is used to find the root of the function $x \in \mathbb{R} \mapsto x^2 + 1$ taking a $x_0 = 0$ or $x_0 = 1$, the suite does not converge, this is highlighted in the figure 7. Indeed the backward error, using the algorithm with (left) or without (right) backtracking does not converge to 0. This means that the algorithm does not converge to the solution. It is easy to see this with backtracking, when the algorithm reaches a backward error of 10^0 it is no longer able to get closer to 0. This is due to the fact that the tangents to the curve representing the $x \mapsto x^2 + 1$ function in 0 and in 1 intersect the x-axis in 1 and in 0 respectively. If we take 0 or 1 as the starting point, the method oscillates between these two points and therefore does not converge.

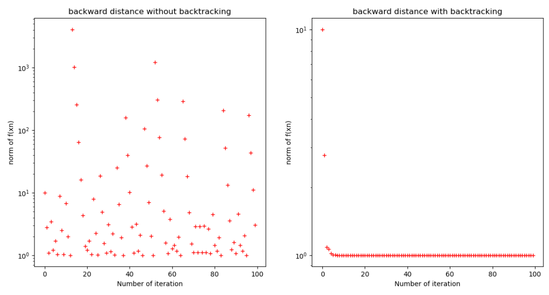

Figure 7: Highlighting Non-Convergence on the $x \mapsto x^2 + 1$ function

Eloi Magon

2 Computation of the Lagrangia points

Lagrangia points are space's positions near two large bodies in orbit. At these locations, the centripetal forces of the two bodies balance out each other in a way that a smaller body would stay in the orbital move of the two bodies. In this part, the previous methods are applied to set a calculus system of the Lagrangia points.

2.1 Theoretical elements and Lagrange point equations

Let us define the different forces in place before getting into the Lagrangia points calculus. An analogy with the Sun-Earth system is studied here, therefore the forces are modeled by $\mathbb{R}^2 \to \mathbb{R}^2$ functions. According to the Newton's third law of motion, the equilibrium points are points for which the sum of the forces is equal to zero. Then, a list of the forces is made :

- Sun's attraction, noted f_1 : Gravitationnal force originating from (0,0) and of parameter $k = 1$;
- Earth's attraction, noted f_2 : Gravitationnal force originating from (1, 0) and of parameter $k = 0.01$;
- Inertial force, noted f_3 : A centrifugal force (*i.e.* two elastical forces without taking account of gravity) originating from Earth's barycenter and of parameter $k = 1$;

The equations proposed are used to implement these forces. For each force a function with the coefficient k and the origin (x_0, y_0) for arguments is defined. A function for the Jacobian matrix is also needed, that's why the partial derivatives are calculated for both the centrifugal and gravitationnal forces.

$$\mathcal{J}_c = \begin{pmatrix} k & 0 \\ 0 & k \end{pmatrix} \qquad \mathcal{J}_g = \begin{pmatrix} \dfrac{2k(x-x_0)^2 - k(y-y_0)^2}{((x-x_0)^2+(y-y_0)^2)^{\frac{5}{2}}} & \dfrac{k(x-x_0)(y-y_0)}{((x-x_0)^2+(y-y_0)^2)^{\frac{5}{2}}} \\ \dfrac{k(x-x_0)(y-y_0)}{((x-x_0)^2+(y-y_0)^2)^{\frac{5}{2}}} & \dfrac{2k(y-y_0)^2 - k(x-x_0)^2}{((x-x_0)^2+(y-y_0)^2)^{\frac{5}{2}}} \end{pmatrix}$$

Thanks to these functions the system $F = f_1 + f_2 + f_3$ is defined, then it can be solved with the Newton-Raphson algorithm.

The complexity of the algorithm is the complexity of Newton-Raphson, namely a linear complexity, because the function in entry is of a constant complexity.

2.2 Solving the equation system by the Newton-Raphson method

In this section the results of the implementation are presented. First of all, criteria need to be defined to monitor the correctness of the resolution. With this in mind, we use again the backward error δ_b defined to control the algorithm convergence. The resuls are displayed in the figure 8.

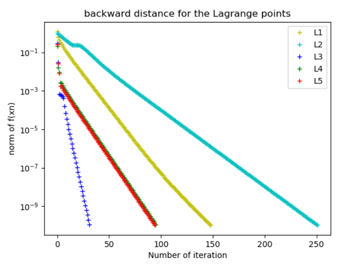

Figure 8: Relative error δ_b for the 5 resolution algorithms using Newton Raphson with backtracking with a requested acuracy of 10^{-9}

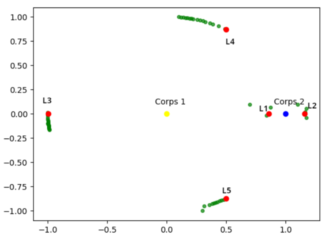

Figure 9: 5 Lagrangia points computed by the algorithm

The five curves on the figure 8 clearly converge towards 0. This implies that the five algorithms find a point where the forces cancelled each other. Now, it's necessary to verify that all these points are distinct from each other.

To do so, a visual monitoring has been realised : for each iteration of the Newton-Raphson algorithm, the location (x, y) obtained is placed on a graph. The results are gathered in Figure 9, and can be compared with a real represention here .

A visual analysis can now be realised. Firstly, the results on the figure 9 ensure that the five points found by the algorithm are distincts from each other, thereby there are five points where the sum of the force is cancelled. The evolution of the algorithm can be seen in green on the Figure 9, and the point it considers as a root, in red. The theorical location showed in the figure and the real representation of the figure 9 clearly merge, thus confirmating the correctness of our implementation.

To sum up, an application of the Newton-Raphson algorithm to the resolution of a non-linear system has been presented.

Sophie Stan

3 Electrostatic equilibrium

Anoter use of non-linear equations lies in the research of electrostatic equilibrium.
Let us consider an interval $[-1, 1]$ and two electric charges (positive or negative), one located at the position -1 and the other one at 1. In addition, let us consider N other charges each positioned at $x_1, x_2, ..., x_N$. We assume that these charges can move freely in the given interval $[-1, 1]$. To start with, let us examine the total electrostatic energy of this system:

$$E(x_1, x_2, ..., x_N) = \sum_{i=1}^{N} log|x_i + 1| + log|x_i - 1| + \frac{1}{2} \sum_{j=1\ i \neq j}^{N} log|x_i - x_j| \tag{3}$$

3.1 Computing the Jacobian matrix of the vector function

In order to determine the system's equilibrium points, it is necessary to find an extrema of the previous fonction E. This boils down to use the Newton-Raphson algorithm in order to solve the following system of non-linear equations: $\frac{\partial E(x_1, x_2, ..., x_n)}{\partial x_i} = 0$. This system is equivalent to $\frac{\partial E(x_1, x_2, ..., x_n)}{\partial x_i} = \frac{1}{x_i + 1} + \frac{1}{x_i - 1} + \sum_{j=i\ i \neq j}^{N} \frac{1}{x_i - x_j}$. Thus, given this equation, it is easy to compute the Jacobian matrix :

$$\begin{cases} \mathcal{J}_{(i,i)} = -\frac{1}{(x_i+1)^2} - \frac{1}{(x_i-1)^2} - \sum_{i=j\ i \neq j}^{N} \frac{1}{(x_i - x_j)^2} & \text{on the diagonal} \\ \mathcal{J}_{(i,j)} = \frac{1}{(x_i - x_j)^2} & \text{others coefficients} \end{cases} \tag{4}$$

3.2 Solving the equation system with the Newton-Raphson method

From now on, the Jacobian is ready to be used by the Newon-Raphson algorithm with backtraking. Nevertheless, it is essential to make sure that the solutions given by the algorithm are realistic. To that end, two functions δ_f and δ_b have been set up, and plotted with $N = 3$ (see Figures 10 and 11).

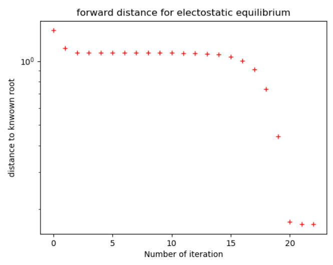

Figure 10: Forward distance δ_f plotting for the function and the Jacobian modeling the electrostatic balance problem using Newton-Raphson with backtracking with an required accuracy of 10^{-14}

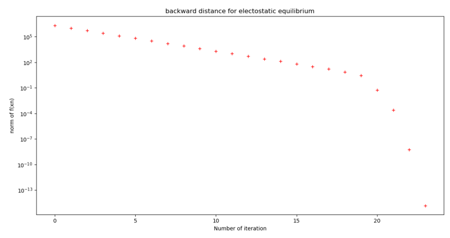

Figure 11: Backward distance δ_b plotting for the function and the Jacobian modeling the electrostatic balance problem using Newton-Raphson with backtracking with an required accuracy of 10^{-14}

The results are coherent since the backward error and the forward error (meaning the two curves), during the execution of Newton-Raphson with backtracking, converge towards 0. This means that the vector V found, is really closed to verify $f(V) = 0$. Thus, the solution found is closed to the actual solution and it confirms its validity.
Vincent Ridacker

3.3 Introduction to the Legendre polynomials

Now that the system to solve has been established, it is possible to put light on the link between the extremum electrostatic problem and the Legendre polynomial. First of all, let us remind what it refers to when $E(n/2) = n/2$ if n is even and $(n-1)/2$ if n is odd : $\frac{d}{dx}[(1-x^2)\frac{d}{dx}P_n(x)] + (n+1)nP_n(x) = 0$
This polynomial can be written thanks to the Bonnet formula as following

$$\begin{cases} P_n(x) = \dfrac{1}{2^n}\sum\limits_{k=0}^{E(n/2)}(-1)^k\binom{n}{k}\binom{2n-2k}{n}x^{n-2^k} \\ P_n'(x) = \dfrac{1}{2^n}\sum\limits_{k=0}^{E(n/2)}(-1)^k\binom{n}{k}\binom{2n-2k}{n}(n-2k)x^{n-2k-1} \end{cases} \tag{5}$$

Now, let us go back to the electrostatic problem, but considering only two charges:

$$\begin{cases} \dfrac{1}{x_1+1} + \dfrac{1}{x_1-1} + \dfrac{1}{x_1-x_2} = 0 \\ \dfrac{1}{x_2+1} + \dfrac{1}{x_2-1} + \dfrac{1}{x_1-x_2} = 0 \end{cases} \Rightarrow \frac{2x_1}{x_1^2-1} + \frac{2x_2}{x_2^2-1} = 0 \tag{6}$$

By reducing, the following equation pops out: $(x_1+x_2)(x_1x_2-1) = 0$, however $|x_1| < 1$ and $|x_2| < 1$, therefore $x_1x_2 \neq 1$, hence $x_1 = -x_2$. After reinjecting this solution into the first or second equation of the system, the following equation is obtained: $5x_1^2 - 3 = 0$. Thanks to a Legendre polynomial table, it is now easy to notice that the previous equation found is also the equation $P_3'(x_1) = 0$. This leads us to think that the electrostatic system with N charges has solutions which are exactly the roots of the $N+1th$ Legendre polynomial derivative. Nevertheless, the mathematical demonstration when the number of charges is more than 2 is more complex. It is indeed necessary to solve N non-linear equations.

This intuition has already been validated, indeed when the figure 10 has been drawn, to recall the forward error is defined as $\delta_f = |||X_n - X||$ where X_n is the solution vector that the algorithm calculates and X is the known solution vector. To plot the figure 10, the X vector has been computed using the theoretical method just presented, meaning Legendre's polynomials. Thus this supports the veracity of the theory.

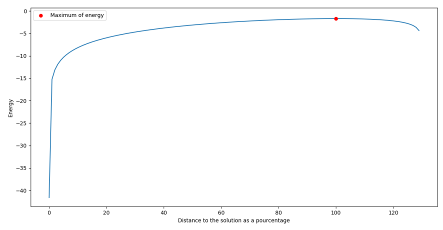

Figure 12: Energy as a function of the distance (in percentage) between the calculated solution and the real solution (found thanks to Legendre)

The Newton Raphson method, allows us to conclude that the solutions of the following system $\nabla E(x_1, x_2, ..., x_N) = 0$ are a set of N-permutations of the roots of the $N+1th$ Legendre polynomial derivative. At last, it is possible to plot the energy's system evolution when the N-vector of charges reaches for an extremum. This is plotted on the figure 12. It is easily visible that the energy is at a maximum.

In summary, the application of non-linear system resolution to electrostatic equilibrium has been performed and is conclusive. Then, the highlighting with the Legendre polynomial has been carried out which allowed to verify with more precision the quality and accuracy of the algorithms.

4 Analysis and conclusion

As a general conclusion, this project allowed us to introduce us to the resolution of non-linear equations throughout the Newton-Raphson method. First of all, a generic method was discovered and then one of its slowdown has been highlighted. This has led to an improvement in the method: Newton Raphson with backtracking. Once these two methods were mastered, the backtracking method was reused to perform two applications of the resolution of a non-linear system. First, a spatial problem: the Lagrange points have been found using the algorithms set up. Then, a more theoretical problem implying an electrostatic equilibrium during which, satisfatory and consistent results have been found. Newton-Raphson's method implemented is consistent (backtracking calculating the solution faster, linear then quadratic convergence speed, divergence cases *etc*). Nevertheless, several difficulties were raised during the transition to the n dimension, but they were solved.