Project No.8

Non-linear systems of equations

Newton-Raphson method

Group No.1 - Team No.1

Responsible : sbalafrej
Secretary : melmoumni
Coders : gdouezangrard, westupin, jmarzin

Abstract : The purpose of this project was to evaluate the Newton-Raphson method in order to find the roots of non-linear systems of equations. The first part shows the algorithm used to implement this method, and suggests a solution to enhance it through backtracking. The following parts are dedicated to test this method in various problems.

1 Newton-Raphson Method

1.1 Simple Newton-Raphson Algorithm

Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be a function that is differentiable along both dimentions. f and its derivatives are defined as following :

$$f : \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix} \to \begin{pmatrix} f_1(x_1, ..., X_N) \\ \vdots \\ f_N(x_1, ..., x_N) \end{pmatrix}, \; H : \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix} \to \left(\frac{\partial f_i(x_1,...,x_N)}{\partial x_j} \right)_{i \in [1,N], j \in [1,N]},$$

where H is the Jacobian matrix of f.

The Newton-Raphson method tries to solve the equation $f(U) = 0$. It assumes that given a current position U, the best direction leading to a root of the function is the direction given by the slope of f at U.

Let U be an initial position chosen arbitrary. The algorithm looks for a point V such that :

$$f(U + V) = 0$$

And

$$f(U + V) = f(U) + H(U).V \tag{1}$$

It is easy to notice that :

$$f(U) + H(U).V = 0 \tag{2}$$

Then, U is replaced by $U + V$, and the process is repeated until either U converges or a maximum number of iterations is reached. The Taylor approximation in Equation 1 ensures that we are looking for the root in the direction of the slope. However, no guarantee is given that the point reached at the end of this direction is minimizing the function f.

This is why the Newton-Raphson method has an unfortunate tendency to diverge if the initial guess is not sufficiently close to the root.

1.2 Backtracking

A global method is a one that converges to a solution from almost any starting point. In this section an algorithm called backtracking is developped, it combines the rapid superlinear speed of convergence of Newton's method, with a globally convergent strategy that will guarantee some progress towards the solution at each iteration. In other words, at each step of the algorithm, the next point given by Equation 2 is only chosen if it minimizes the function. Figure 1 on the following page shows the full algorithm and Figure 2 on the next page compares the behaviour of these algorithms.

The complexity of the algorithm is $\Theta(N.c)$ - c is the complexity of the function solve.

1.3 Tests

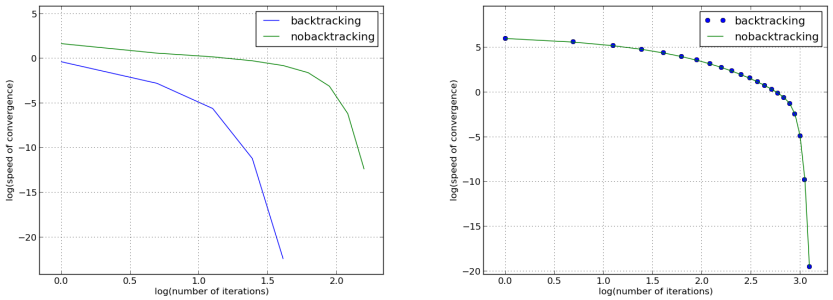
One way to easily test this method is to check if the point returned U is a root of f. This can be done by computing $f(U)$. In this purpose, it is interesting to test the convergence of f towards 0. Figure 2 on the following page show the speed of the convergence. The conclusion is that the speed of the convergence is superlinear (second order) . At worse, the backtracking convergence is the same as a classical Newton-Raphson.

The method has also been tested with the function $f_a : x \to x^2 - a$, solving the equation $f_a(x) = 0$, to check the validity of the returned roots depending on the cases :

1. two roots,
2. one root,
3. or complex roots : in this case, the algorithm ends.

Data: f : a function, H : the jacobian matrix of f, U_0 : a starting position, N : maximal number of iterations, ε : precision.
Result: U : a root of f
$U \leftarrow U_0$
tmp $\leftarrow 1$
for i *from* 1 *to* N **do**
  $V \leftarrow \text{solve}(f(U) + H(U).V = 0)$
  while $\|f(U + step * V)\| > \|f(U)\|$ **do**
    step $\leftarrow$ step $* 2/3$
  end
  $U \leftarrow U +$ step $* V$
  if $\|f(U) < \varepsilon\|$ **then**
    return U
  end
end
return U

Figure 1 – Backtracking algorithm



(a) From $(0.25, 0)$ (b) From $(1025, 0)$

Figure 2 – Comparison between Newton-Raphson with and without Backtracking, considering $f : x \to x^3 - 1$

2 Computation of the Lagrangian points

2.1 Introduction

Given a set of forces which apply to an object in the plane, our goal is to compute the equilibrium positions of this object. We will restrain ourselves to three kind of interactions : elastic, centrifugal and gravitational forces. We will need the previously written Newton-Raphson method, which will be applied to the resultant force. As a result, we will obtain the roots of this three-dimensional function, corresponding to the equilibrium positions.

2.2 Forces

Before explaining and analysing the method, we need to express both the general form of the forces and their Jacobian matrices. Each force will be parameterized with an integer - the intensity - and their origin.

We will start with the elastic force, which represent a spring action on an object. In a one dimensional space, this force is expressed by the following formula :

$$\vec{f_c} = -k(l - l_0)\vec{x},$$

with l the spring length and l_0 its free length.

In the plane $(O, \vec{x}, \vec{y})$, this force can be reduced using orthogonals projections to the following form :

$$f_e : \begin{pmatrix} x \\ y \end{pmatrix} \to \begin{pmatrix} \frac{-k(x-x_0)}{\sqrt{(x-x_0)^2+(y-y_0)^2}} \\ \frac{-k(y-y_0)}{\sqrt{(x-x_0)^2+(y-y_0)^2}} \end{pmatrix},$$

where $\begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$ is the origin coordinate and k the intensity.

We can now express the f_e Jacobian matrix :

$$H_{f_e} : \begin{pmatrix} x \\ y \end{pmatrix} \to \begin{pmatrix} \frac{(y_0-y)^2}{((x_0-x)^2+(y_0-y)^2)^{\frac{3}{2}}} & \frac{(x_0-x)(y_0-y)}{((x_0-x)^2+(y_0-y)^2)^{\frac{3}{2}}} \\ -\frac{(x_0-x)(y_0-y)}{((x_0-x)^2+(y_0-y)^2)^{\frac{3}{2}}} & \frac{(x_0-x)^2}{((x_0-x)^2+(y_0-y)^2)^{\frac{3}{2}}} \end{pmatrix}$$

Since the force expressions have already been given in the subject, we will only provide the forms of the Jacobian matrices for the centrifugal and gravitational forces :

$$H_{f_c} : \begin{pmatrix} x \\ y \end{pmatrix} \to \begin{pmatrix} k & 0 \\ 0 & k \end{pmatrix}$$

$$H_{f_g} : \begin{pmatrix} x \\ y \end{pmatrix} \to \begin{pmatrix} \frac{k(2x_0^2-y_0^2-4x_0x+2x^2+2y_0y-y^2)}{((x_0-x)^2+(y_0-y)^2)^{\frac{5}{2}}} & \frac{3k(x_0-x)(y_0-y)}{((x_0-x)^2+(y_0-y)^2)^{\frac{5}{2}}} \\ \frac{3k(x_0-x)(y_0-y)}{((x_0-x)^2+(y_0-y)^2)^{\frac{5}{2}}} & -\frac{c(x_0^2-2y_0^2-2x_0x+x^2+4y_0y-2y^2)}{((x_0-x)^2+(y_0-y)^2)^{\frac{5}{2}}} \end{pmatrix}$$

Implementation The implementation use a functionnal programming approach, which means that we use a kind of functions which will build and return a function as expression. This allow us to be more versatile by storing and using the function itself rather than using a general function, and to give it as a parameter to the Newton-Raphson method for example.

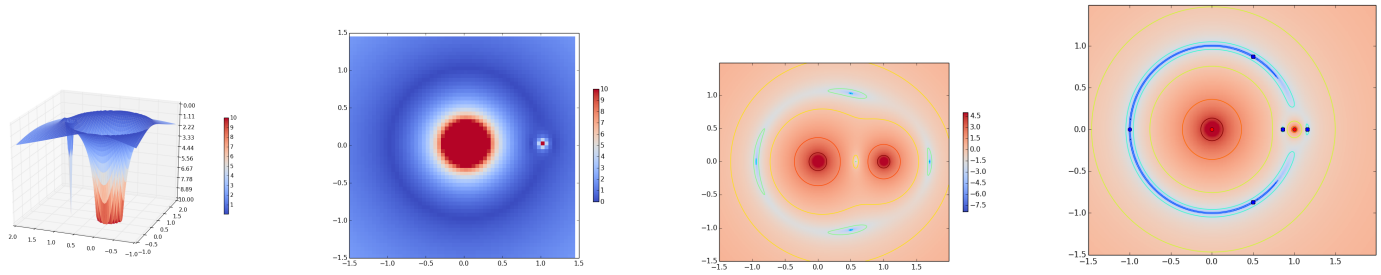
2.3 Results

2.3.1 First approach

We consider the following case :

— Two gravitational forces with coefficients 1 (resp. 0.01) and originating from $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$ (resp. $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$).

— A centrifugal force with coefficient 1 at the barycenter of the two masses, i.e., at $\begin{pmatrix} \frac{0.01}{1.01} \\ 0 \end{pmatrix}$.



(a) 3D shape (b) Equilibrium (c) Lagrange points (d) Lagrange points computed

Figure 3 – Resultant force norm

We can easily plot the situation using the norm of the resultant force. So we get the Figure 3a. From above, the graph is more explicit, as shown in the Figure 3b. The situation is similar to the Earth rotation around the Sun. Centrifugal force and gravitational attraction by the Sun are applied to the Earth. The centrifugal force applied to the barycenter of the two bodies is characteristic of a two-body interaction.

2.3.2 Lagrangian points

The main issue is that the Newton-Raphson method can only compute one root while the we seem to have an infinite number of roots. The algorithm can be called by specifying a step over the whole grid. The purpose is to exhibit that in this kind of interaction, the equilibrium points are particular and well-known as the Lagrangian points.

This is not obvious with the last example that only five points correspond to an equilibrium situation. However, the datas can be a bit modified to make this phenomenon more clear and spread the values with the logarithm, as in Figure 3c.

As the Newton-Raphson algorithm follows the slope of the curve, it's called on equireparted points on the grid, distant by a fixed step noted τ. The algorithm must work in a closed domain. Moreover, at each iteration, the algorithm will output a

position corresponding to a root. Assuming that enough iterations will be performed enough iterations to find at least all the roots, we a maximal precision on a root coordinate must be determinate. The minimal distance between two distinct roots, noted ε, is another parameter of this algorithm.

Considering this algorithm, we've been able to build Figure 3d on the preceding page, where the blue points denotes the roots found.

3 Bairstow Method

The Bairstow method enables to find the roots of a polynomial P. This method can be used when the roots of the polynomial P are real. Howewer, the method is unable to find complex roots of P. The Bairstow method is a method computing the irreducible factors of degree 2 of a real polynomial. Incidentally, it avoids the computations with complex numbers. Technically, it simply consists in applying the Newton-Raphson method to a specific function.

1. In order to use the Newton-Raphson method, a 2-dimension fonction has to be defined.
   The polynomial P is defined by :
   $$P(x) = a_n \cdot z^n + a_{n-1} \cdot z^{n-1} + \ \ldots \ + a_1 \cdot z + a_0$$
   Then P is divided by the real quadratic polynomial $\boxed{x^2 + B \cdot x + C}$. Then P can be written :
   $$P(x) = (x^2 + B \cdot x + C) \cdot (b_{n-2} \cdot z^{n-2} + \ \ldots \ + b_0) + R \cdot x + S$$
   where $R \cdot x + S$ is the remainder, $\forall k \in [|0, n-2|]$, $b_k = a_{k+2} - B \cdot b_{k+1} - C \cdot b_{k+2}$ and $b_{n-1} = b_n = 0$.
   Hence, the coefficients b_k can be regarded as defined fonctions of B and C. Thus, R and S are functions of B and C and are defined by :
   $$R(B,C) = a_1 - B \cdot b_0 - C \cdot b_1 \,, \quad S(B,C) = a_0 - C \cdot b_0$$
   The solution of R(B,C) = 0 and S(B,C) = 0 yields B and C such that $x^2 + B \cdot x + C$ is a quadratic factor of P.
   After obtaining a quadratic factor, the polynomial P is deflated and the same procedure is applied to the deflated polynomial. Thus, the 2-dimension fonction needed to use the Newton-Raphson method is :
   $$f(B,C) = (R(B,C), S(B,C))$$

2. This function can be written by using the function *numpy.polydiv*. Indeed, this fonction computes simultaneously the quotient and the remainder of the division. B and C given, the polynomial P has to be divided by $x^2 + B \cdot x + C$. The remainder is $Rx + S$. Then, a 2-dimension vector representing R and S is built. It is worth noticing R and S depends on the values of B and C.

3. The 4 partial derivatives of the previous function are defined by :
   $$\frac{\partial R}{\partial C} = -R_1 \,, \ \ \frac{\partial S}{\partial C} = -S_1$$
   Where R_1 and S_1 are given by a second division of P by $x^2 + B \cdot x + C$:
   $$P(x) = Q(x) \cdot (x^2 + B \cdot x + C) + R \cdot x + S$$
   $$Q(x) = Q_1(x) \cdot (x^2 + B \cdot x + C) + R_1 \cdot x + S_1$$
   The last 2 partial derivatives are :
   $$\frac{\partial R}{\partial B} = BR_1 - S_1 \,, \ \ \frac{\partial S}{\partial B} = CR_1$$

4. The algorithm can be divided in the three parts : finding a quadratic factor of P by using the Newton-Raphson method, calculating its roots, dividing P. Each division of P leads to the research of a new quadratic factor and its roots. There are at least N iterations where N is the degre of P divided by two.
   In order to ascertain the proper functioning of the algorithm, a procedure generating random polynomials has been coded. An other procedure which evaluates polynomials at a given point was needed to test Bairstow method. Thanks to these two functions, the following procedure has been coded :
   In order to test this method, a random polynomial is generated. Then, Bairstow method is applied on this polynomial in order to find its roots. In the end, the polynomial is evaluated at each root and the equality in 0 is checked.

4 Electrostatic equilibrium

4.1 Jacobian Matrix

Considering the interval $[-1, 1]$ and two electrostatic charges fixed at the positions -1 and 1. We assume that there exists N charges positioned at $x_1, x_2, ..., x_N$ and that these charges can move freely in the interval $[-1, 1]$.

Findind the positions that essen the amout energy of the charges means :

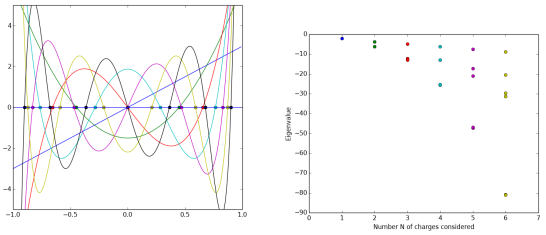
$$\nabla E(x_1, x_2, ..., x_n) = \left(\frac{\partial E(x_1,...,x_N)}{\partial x_i}\right)_{i\in[1,N]} = 0$$

Noticing that :

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

Thus, the (i, j) term of the jacobian matrix of $\nabla E(x_1, x_2, ..., x_n)$ is defined as :

$$\left(\frac{\partial^2 E}{\partial x_i \partial x_j}\right) = \begin{cases} \frac{1}{(x_i - x_j)^2} & \text{if } i \neq j \\ \frac{-1}{(x_i+1)^2} - \frac{1}{(x_i-1)^2} - \sum_{j=1,j\neq i}^{N} \frac{1}{(x_i-x_j)^2} & \text{else} \end{cases}$$



(a) Legendre polynomials and computed roots (b) Eigenvalues of the Hessian matrix

Figure 4 – Computed critical points

4.2 Equation solving

The Newton-Raphson method is applied in order to find the roots of non-linear equation system.

The equilibrium positions are exactly the roots of successive derivatives of Legendre Polynomials (as shown in Figure 4a), which are :

$$\boxed{P'_n = \frac{1}{2^n n!} \frac{d^{n+1}}{dx^{n+1}}((x^2 - 1)^n)}$$

However, it remains interesting to determine if these equilibrium positions refer to either a maximum or a minimum for the energy. The plot situation can be tried, however it will start to be unclear since the 4-dimension. Hopefully, some analytics theorems can help us to determine the type of these extremums.The Hessian matrix is needed, which corresponds to J.

Theorem 1. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a twice-differentiable function, H_f its Hessian matrix and a a critical point ; then :*

- *if $H_f(a)$ is positive definite, then a is a local minimum of f,*
- *if $H_f(a)$ is negative definite, then a is a local maximum of f,*
- *else, a is a saddle point of f.*

Considering Theorem 1, we only need to get the eigenvalues of J and determine their distribution. We show in Figure 4b that the eigenvalues are stricly negative. We can conclude experimentaly that all the equilibrium poisitions correspond to a maximum of energy.

5 Conclusion

The Newton-Raphson methods provides an approximate solution for a non-linear system of equations. This can be used to solve physician problems that are usually gifted by an approximative model of non-linear equations. This method helped, to solve computation of Lagragian points for example. However, the Newton-Raphson algorithm may not found the solution especially when it doesn't converge, in the electrostatic equilibrium for example. In that cases, To avoid this problem, backtracking is a solution.