Project n°4
Non-linear systems of equations - Newton-Raphson method

Group n°4 - Team n°5

Responsible : mbounakhla
Secretary : gzahar
Programmers : rboudjeltia, lcidere, pgaulon

Abstract :

Newton-Raphson method

To begin with, the Newton-Raphson algorithm had to be programmed in order to solve all the non-linear problems given. This algorithm is based on the equation $f(U+V)=0$ in which $f(U+V)$ can be approximated by $f(U)+H(U)\times V$. In this equation, f is a differentiable function $\mathbf{R}^m \to \mathbf{R}^n$, H is the Jacobian matrix of function f, U is the vector representing the current position and V represents the next position approaching a root of function f. Subsequently the relation linking H, f, U and V could be immediately deduced : $f(U)+H(U)\times V=0$ or in other words $H(U)\times V=-f(U)$.

As numpy.linalg.lstsq is based on a SVD decomposition, its complexity is approximately $\theta(n^3)$ where n is the size of the J matrix. Therefore in the worst case, the Newton-Raphson algorithm has a complexity measured by $\theta(N\times n^3)$, so a cubic complexity.

Here is a Newton-Raphson algorithm including backtracking :

Data: a function f, it's Jacobian matrix J, a starting vector $U0$, an iteration maximum number N, a convergence measurement *epsilon*
Result: a root of the function f
$U \leftarrow U0$;
$i \leftarrow 0$;
while $(i < N)$ *and* $(||f(U)|| >= epsilon)$ **do**
  $V \leftarrow \text{Solution}(J(U) * V + f(U) = 0)$;
  while $||f(U+V)|| >= ||f(U)||$ **do**
    $V \leftarrow V/2$;
  end
  $U \leftarrow U+V$;
  $i \leftarrow i+1$;
end
return U;

Algorithm 1: Newton-Raphson algorithm with backtracking

Backtracking prevents this algorithm from overlooking solutions. If the $f(U+V)$ value goes too far, backtracking forces this value to go back. Here, the choice of going back with a half step was made.

The second While loop is the code portion including backtracking. Without it, this algorithm is the simple Newton-Raphson one. For each calculus using Newton-Raphson, two graphs were calculated, representing the $||f(U)||$ value according to the iteration number. The first one was for the algorithm without backtracking and the second one with backtracking. But both graphs were the same. We

supposed that there were not enough iterations in these calculi to tell the difference. So for these calculi, either algorithm can be used without any difference.

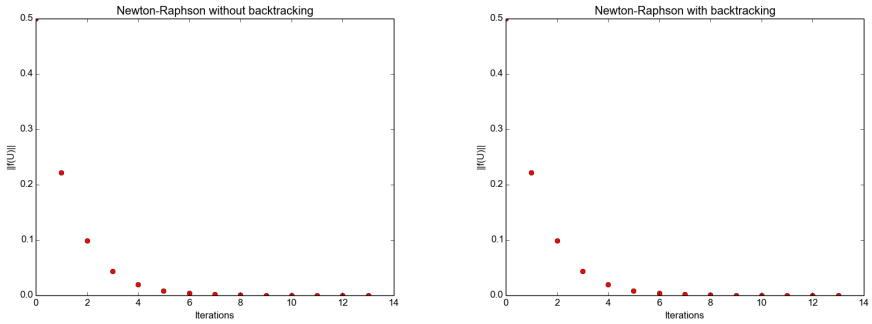

FIGURE 1 – Convergence comparison between the Newton-Raphson algorithm without backtracking and with backtracking

Computation of the Lagrangian points

The Lagrangian points are the five equilibrium points in an orbital configuration. The configuration chosen was the one including the Sun and the Earth. So there were two masses. The first one, representing the Sun, a hundred times as heavy as the second one, the Earth, and they were separated by a certain distance called radius symbolized R.

In order to solve this problem, the first step was to modelize the different forces : elastic, gravitational and centrifugal. Subsequently, to obtain the first three Lagrangian points, the Newton-Raphson method was used three times to obtain each root of each equilibrium equation. The last two points were calculated geometrically.

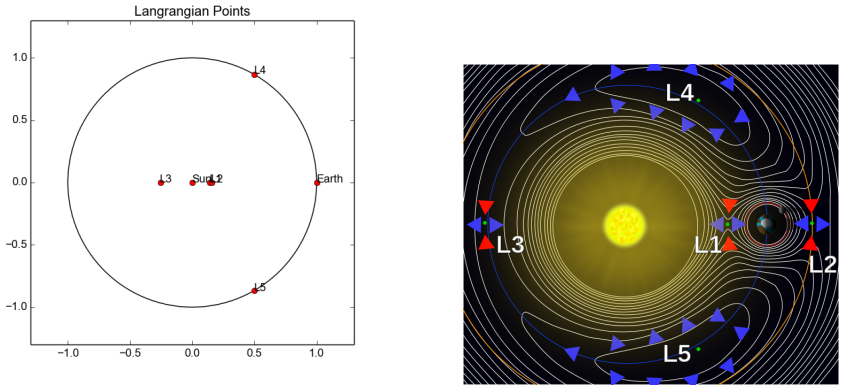

FIGURE 2 – Comparison between our solution and the solution on Wikipedia's website

In this situation it was possible to calculate the five Lagrangian points. But the solutions we found were lower than expected. The first three points should have been a little bit further from the Sun, with a distance approximately that of R. We did not find the reason for this problem. We suspected that there was a wrong equilibrium formula or a wrong derivative of this formula, or bugs in our Newton-Raphson algorithm, or even our way of using it could have been wrong. But investigating each of these possibilities did not lead to a satisfactory solution.