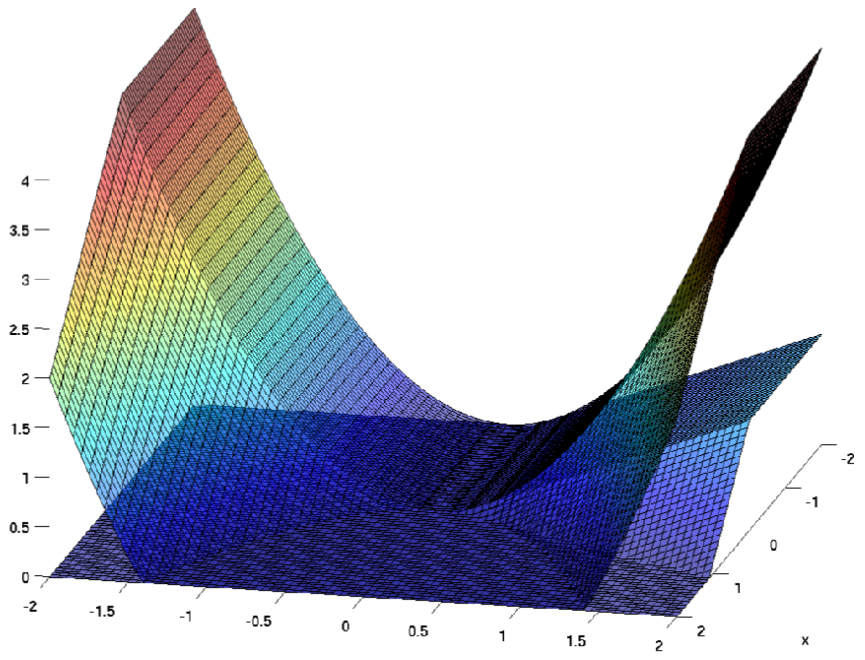
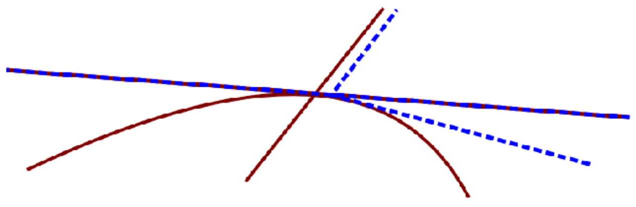
Approximate resolution of a non-linear equation





Made by: Rémy DUTTO, Victor DELOR, Hugo BONNETAIN

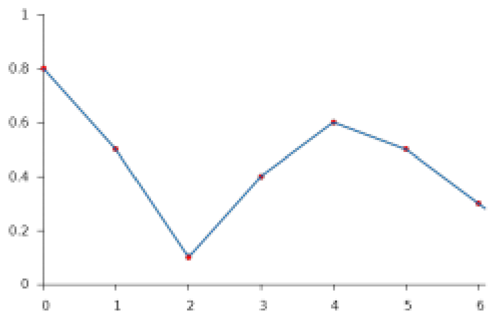
Supervisor: Lionel FILLATRE

School year 2019-2020

Introduction:

In many engineering fields, it is frequently necessary to solve some non-linear equations such as $F(x) = 0$ where F: $\mathbb{R}^n \rightarrow \mathbb{R}^m$ is a non-linear function. This project will be focus on the type of functions which could be approximate by piecewise linear continuous functions.

The following graph shows a piecewise linear function. This kind of functions is a step between linear function and non-linear function. The goal of this project is to find some solutions of the equation system $F(x) = 0$ using this approximation of the F function.



In a first step, we studied the "Representation and Analysis of Piecewise Linear Functions in Abs-Normal Form" this document helped us to understand the theoretical part. In another step, we programmed all the methods in python.

We proceeded, following the schema below (figure 1), which explains the logic and the methods applied. We took a function, from that we transformed the function in piecewise function, to create a computational graph after. Nevertheless, sometime it is possible to directly get the computational graph with the function. This computational allows us to calculate the abs normal form. Finally, when the abs normal form is defined we can apply an algorithm: Gauss Seidel or Newton to solve our system.

Explaining graph of our reasoning

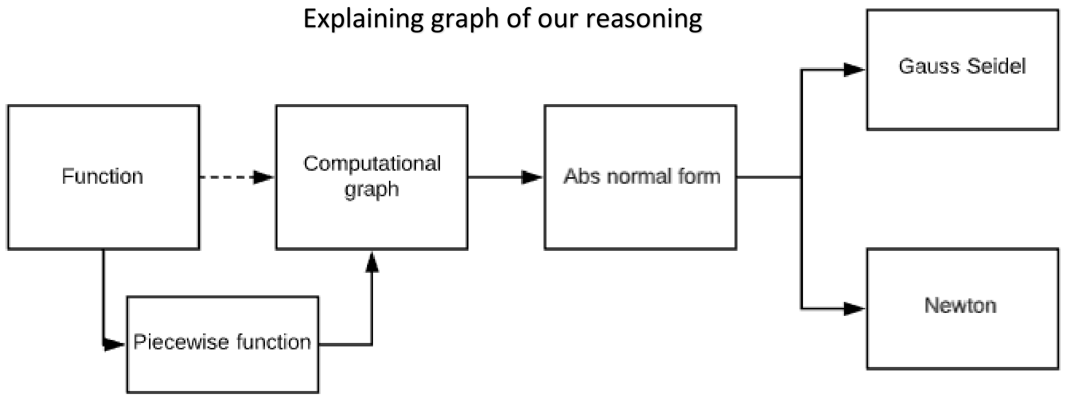


Figure 1

Summary:

I) Approximation of the function by a piecewise function

Every piecewise functions could be written with two different methods:

- Firstly, every one-dimensional piecewise linear functions could be written with the min-max representation.

$$f(x) = \max_{j \in J} \min_{i \in S_j} g_i(x)$$

- Secondly, for a set of points given (x_i , y_i): $i = \{0, \dots, n\}$, where $x_0 < x_1 < \cdots < x_n$, two outer slopes s_0 and s_{n+1} and n inner slopes $s_i = \frac{y_i + y_{i-1}}{x_i + x_{i-1}}$

We got the following formula:

$$y = \frac{1}{2}[\, y_0 + s_0(x - x_0) + \sum_{i=0}^{n}(s_{i+1} - s_i)|x - x_i| + y_n + s_{n+1}(x - x_n)] \qquad (1)$$

$$y = \frac{1}{2}[\, y_0 + s_0(x - x_0) + y_n + s_{n+1}(x - x_n) + \sum_{i=0}^{n}(s_{i+1} - s_i)|x - x_i|]$$

In the sum, the subtraction $s_{i+1} - s_i$ allows to reload the new slope, in every step.

We have the abs function in the sum because we want to have a break of the function when x become taller than x_i.

It is possible to link these two methods with the following formulas:

$$\max(x, y, z) \equiv \max(\max(x, y), z)$$

$$\max(x, y) \equiv 0.5 * (x + y + |x - y|)$$

$$\min(x, y) \equiv 0.5 * (x + y - |x - y|)$$

For example, $f(x) = \max(a, \min(x, b)) = 0.5 * [a + |x - a| - |x - b| + b\,]$

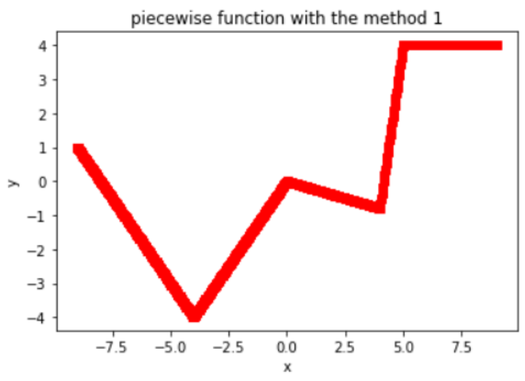
In the rest of our project, we mostly used the second method.

To code this method we decided to choose these initial parameters.

Example 1:

```
#Main
xi=[-4,-2,0,4,5]
yi=[-4,-2,0,-1,4]
s0n=[-1,0]
```

Figure 2



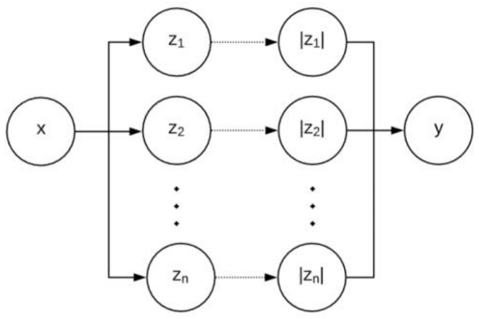
After coding this method, we obtained the opposite figure (figure 2), an example of a piecewise function with the parameters above.

II) Computational graph

To develop the abs-normal form, we need to decompose this function on a computational graph.

This is a graph where nodes correspond to basic functions; this creates new variables needed for the result.

Example 1: We take back the first example, and then expressed this function with a computational graph. We create new variables $z_i = x - x_i$. With that, we calculate its abs values to calculate y. The following schema sums up this method:



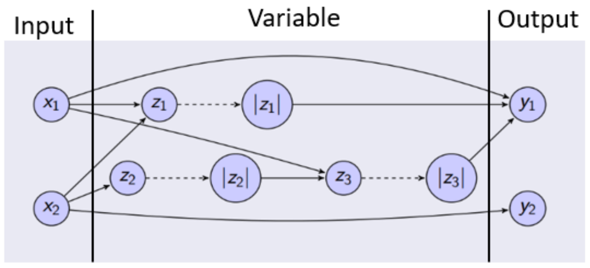
$$z_1 = x - x_1$$
$$z_2 = x - x_2$$
$$\vdots$$
$$z_n = x - x_n$$

Computational graph of the function 1

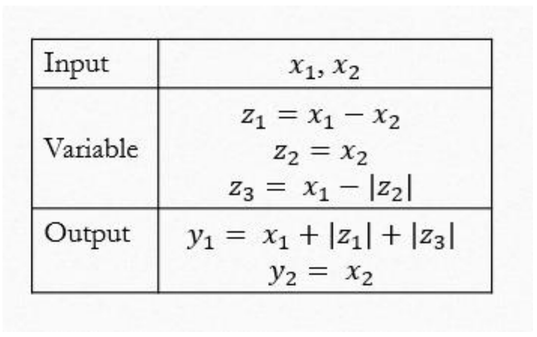
Example 2: $F: \mathbb{R}^2 \to \mathbb{R}^2$ in this case we do not know if F is piecewise or not, but we have seen that we are able to treat this case with the abs normal form.

$$(x_1, x_2) \mapsto \begin{bmatrix} x_1 + |x_1 - x_2| + |x_1 - |x_2|| \\ x_2 \end{bmatrix}$$

As we did in the example above, we decompose the function F with this computational graph. We made a link between the input variables and the output variables.



Computational graph of the function 2



Input	x_1, x_2
Variable	$z_1 = x_1 - x_2$ $z_2 = x_2$ $z_3 = x_1 - \|z_2\|$
Output	$y_1 = x_1 + \|z_1\| + \|z_3\|$ $y_2 = x_2$

With these graphs, we can create the entire matrix needed for the abs normal form. We can see in this graph that all dotted lines correspond to the absolute function and all other nodes correspond to the sum. Then we need to study the abs normal form.

III) Abs normal form method

In Scholtes books, “Springer”, the using of the min/max method says that all piecewise continuous linear functions can be written on the abs normal form.

In this part, we will make a link between the equation (1) and the abs normal form of a function.

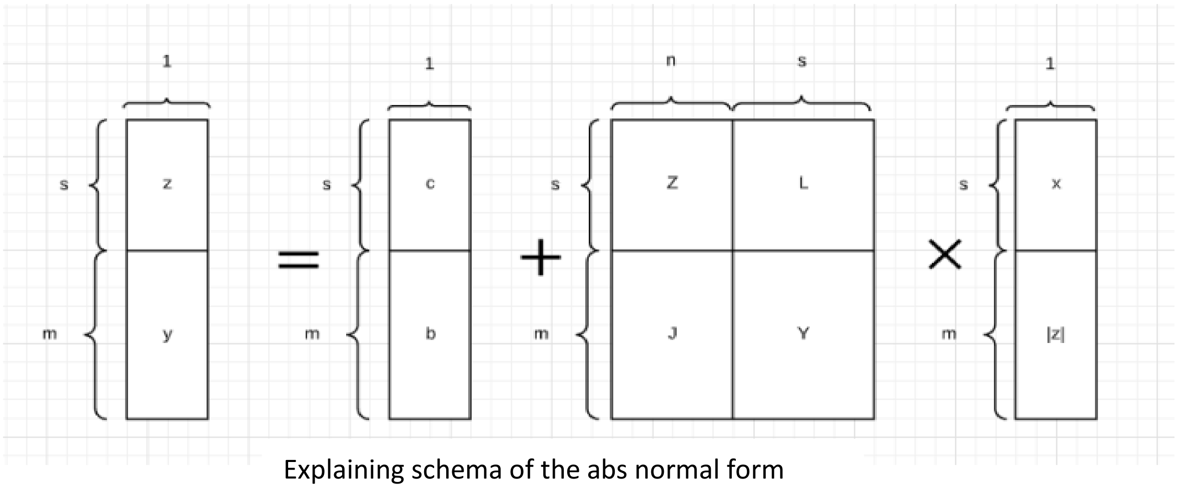
Definition: For Z $\in \mathbb{R}^{s*n}$, L $\in \mathbb{R}^{s*s}$, J $\in \mathbb{R}^{m*n}$, Y $\in \mathbb{R}^{m*s}$ matrixes, where L is a strictly lower triangular form and the vectors c $\in \mathbb{R}^{s}$, b $\in \mathbb{R}^{m}$, the system:

$$\begin{bmatrix} z \\ y \end{bmatrix} = \begin{bmatrix} c \\ b \end{bmatrix} + \begin{bmatrix} Z & L \\ J & Y \end{bmatrix} \begin{bmatrix} x \\ |z| \end{bmatrix}$$

This system is called abs-normal form. The modulus operation |z| has to be understood component wise here. An abs-normal form is called simply switched if L = 0.

Simply switched means that in the first equation of matrixes, we have $z = f(x)$ and z does not depend on |z|.

The previous definition can be simply explained with the next diagram:



Explaining schema of the abs normal form

By simple calculus, and developing the matrix we got:

$$y = b + Jx + Y|z|$$

$$z = c + Zx + L|z|$$

We notice: $|z| = \Sigma_z z$

Then with calculus, we got:

$$z \ = (I - L\Sigma_z)^{-1}C + (I - L\Sigma_z)^{-1}Zx$$

$$|z| \ = \Sigma_z(I - L\Sigma_z)^{-1}C + \Sigma_z(I - L\Sigma_z)^{-1}Zx$$

Finally:

$$y = b + Jx + Y[\Sigma_z(I - L\Sigma_z)^{-1}C + \Sigma_z(I - L\Sigma_z)^{-1}Zx]$$

$$y = b + Y\Sigma_z(I - L\Sigma_z)^{-1}C + \ [J + \Sigma_z(I - L\Sigma_z)^{-1}Z]\ x$$

With the knowledge of Σ_z we cave have y a linear function by x.

Application in our examples:

<u>Example 1:</u> f: $\mathbb{R} \ \rightarrow \ \mathbb{R}$

For the example 1, we have $z_i \ = \ \mathrm{x} - x_i = \mathrm{f(x)}$ then we can conclude that it is a simply switched.

Then we get the following abs normal form:

$$\left(\begin{array}{c} x - x_1 \\ \vdots \\ x - x_n \\ \hline y \end{array}\right) = \left(\begin{array}{c} -x_1 \\ \vdots \\ -x_n \\ \hline y_0 - s_0 x_0 - s_{n+1} x_n \end{array}\right) + \left(\begin{array}{c|ccc} 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & & 0 \\ \hline s_0 - s_{n+1} & s_0 - s_1 & s_{i+1} - s_i & s_{n+1} - s_n \end{array}\right) \left(\begin{array}{c} x \\ \hline |x - x_1| \\ \vdots \\ |x - x_n| \end{array}\right)$$

We have programed this with the first method and obtained the Figure 3. It is possible to see that the classical function traced in red is superposed with the abs normal function traced in blue. This curb confirmed the veracity of our code.

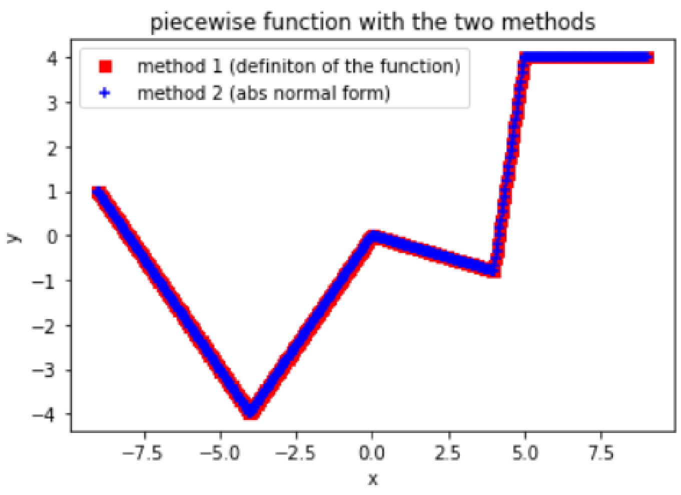


Figure 3

Example 2: $g\colon \mathbb{R}^2 \rightarrow \mathbb{R}^2$

In this case, our function is not simply switched because we have

$$z_3 = x_1 - |z_2| = f(z_2)$$

Thanks to the result, we got $\mathrm{L} \neq 0$. Using the computational graph, we obtained the following matrix.

$$\left[\begin{array}{c} z_1 \\ z_2 \\ z_3 \\ \hline y_1 \\ y_2 \end{array}\right] = \left[\begin{array}{c} 0 \\ 0 \\ 0 \\ \hline 0 \\ 0 \end{array}\right] + \left[\begin{array}{cc|ccc} 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 \\ \hline 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \end{array}\right] \left[\begin{array}{c} x_1 \\ x_2 \\ \hline |z_1| \\ |z_2| \\ |z_3| \end{array}\right]$$

The results obtained by computing, with the two methods:

- Method 1 : the function is simply printed
- Method 2 : the function is printed with the abs normal form

These graphs displayed below also depend on the different variables y_1, y_2 .

Figure 4

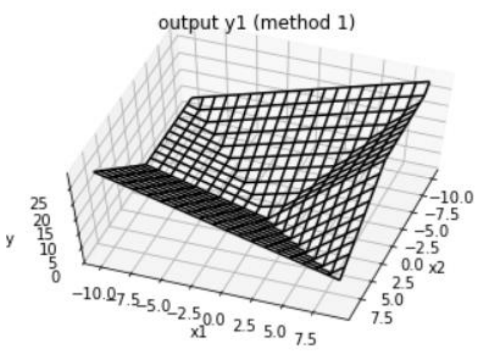


Figure 5

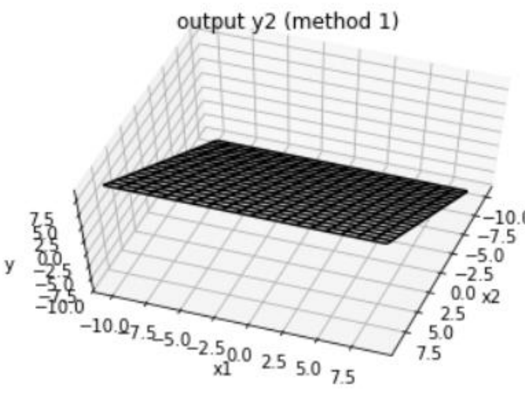


Figure 6

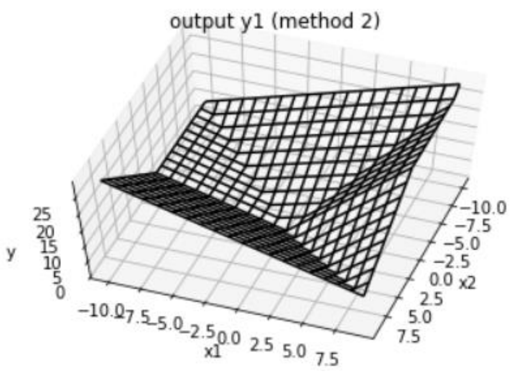
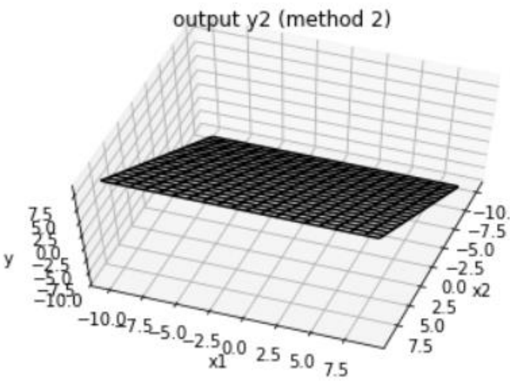


Figure 7



With both methods, we can see that the figures 4 and 6 are the same, and figure 5 and 7 too. This means we got the same results, the abs normal form is correct.

IV) Gauss Seidel method

The Gauss-Seidel method is an iterative method providing the resolution of equation in the following form: $A * x = b$ with $A \epsilon \mathbb{R}^{n*n}$, $x \epsilon \mathbb{R}^n$ and $b \epsilon \mathbb{R}^n$.

The method is indeed iterative as it generates a series of vectors x^k (with k a positive integer) thanks to the last vector x^{k-1} obtained. We stop the iteration when we consider the new vector suitable (very near with the solution). We have two different ways of using this method.

Let us consider at first the situation where the system is simply linear.

a) Linear form

As we have considered that $A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & & & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$, $b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{pmatrix}$

We know the current iterated x^k we construct our new vector x^{k+1} as:

At the step 1: $a_{11}x_1^{k+1} + a_{12}x_2^k + \cdots + a_{1n}x_n^k = b_1$

At the step 2: $a_{21}x_1^{k+1} + a_{22}x_2^{k+1} + a_{23}x_3^k + \cdots + a_{2n}x_n^k = b_2$

And so on, until we have at the step i (i a positive integer between 0 and n):

$$a_{i1}x_1^{k+1} + \cdots + a_{i,i-1}x_{i-1}^{k+1} + a_{ii}x_i^{k+1} + a_{i,i+1}x_{i+1}^{k+1} + \cdots + a_{in}x_n^k = b_i$$

Hence, if we suppose the elements on the diagonal of A are all non-null we have, for all i:

$$x_i^{k+1} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{k+1} - \sum_{j=n+1}^{n} a_{ij}x_j^k\right)$$

b) Matrix form

Classic version:

We consider $A = M - N$.

Replacing into the initial equation and considering , we have:

$$Ax = b \Leftrightarrow Mx = Nx + b \Leftrightarrow x = M^{-1}Nx + M^{-1}b$$
$$= F(x)$$

With M supposed invertible and F a linear function

Then, we split the matrix A into three different parts: $A = D - E - F$ with -E the triangular lower part of A, -F its triangular upper part and D its diagonal parts such that $M = D - E$ and $F = N$.

Consequently, it follows that: $x^{(k+1)} = (D - E)^{-1}\mathrm{F}x^{(k)} + (D - E)^{-1}\mathrm{b}$

Like that, we have our formula to compute the new vector from the current one under matrix expression.

There exists another version, by “blocks”, which could be more convenient in some situations.

Block version:

As previously, except we do not consider anymore elements of the matrix but sub-matrixes called “blocks”.

With this in mind, we make the following partition: $[\![1, n]\!] = I_1 \cup I_2 \cup \ldots \cup I_p$

We decompose A and b as $A = \begin{pmatrix} a_{I_1I_1} & a_{I_1I_2} & \cdots & a_{I_1I_n} \\ a_{I_2I_1} & & & a_{I_nI_n} \\ \vdots & & \ddots & \vdots \\ a_{I_nI_1} & a_{I_nI_2} & \cdots & a_{I_nI_n} \end{pmatrix}, \quad b = \begin{pmatrix} b_{I_1} \\ b_{I_2} \\ \vdots \\ b_{I_{n-1}} \\ b_{I_n} \end{pmatrix}$

With (A_{IJ}) the sub-matrix of A with index on row in I and index on column in J, b_I the sub-vector with index in I

We suppose here the A_{IJ} are all inversible.

Hence, we get the same expression of the iteration than previously:

$$x^{(k+1)} = (D - E)^{-1}\mathrm{F}x^{(k)} + (D - E)^{-1}\mathrm{b}$$

Nevertheless, we note the definitions of D, -E and -F have changed to a writing by blocks too.

c) Convergence

As we have $x^{k+1} = M^{-1} * N * x^k + M^{-1} * b$, it is clear the iteration matrix $M^{-1}N$ is repeated (k+1) times and we can express the result above as: $x^{k+1} = (M^{-1}N)^{k+1} * x_0 + S$ with S a value which does not depend on x.

Furthermore, we have that: $\lim_{m\to\infty} A^m = 0 \Leftrightarrow \rho(A) < 1$ with $A \in M_n(\mathbb{R})$, with n > 0, $A = (a_{ij})_{i,j=1,\dots,n}$.
We recall the definition of the p-norm to prove this result below : for $1 \le p \le +\infty$, we note $\|\ \|_p$ the matrix norm calculated by the vector norm $\|\ \|_p$, i.e.

$$\| A\|_p = \sup_{\| x\|_p=1} \| Ax\|_p = \sup_{x\neq 0} \frac{\| Ax\|_p}{\| x\|_p}$$

Proof:

- Firstly, we want to show $\lim_{m\to\infty} A^m = 0 \Rightarrow \rho(A) < 1$

Suppose that $\lim_{m\to\infty} A^m = 0$. If $\rho(A) \ge 1$ then $\|A^m\|_p \ge \left(\rho(A)\right)^m$, we got $\|A^m\|_p \ge 1$. The sequence of positive numbers $\left(\|A^m\|_p\right)_{m\ge 0}$does not converge. It follows that the sequence of matrixes $(A^m)_{m\ge 0}$ does not converge too. Necessarily we got $\rho(A) < 1$

- Secondly, we want to show $\rho(A) < 1 \Rightarrow \lim_{m\to\infty} A^m = 0$

As $\rho(A) < 1$ there exists $\varepsilon > 0$ such as $\rho(A) + \varepsilon < 1$ (we just have to take $\varepsilon = \frac{(1-\rho(A))}{2}$)

We suppose there exists a matrix norm $\|\ \|$ (which depends of A and ε) such as :

$$\| A\| \le \rho(A) + \varepsilon < 1$$

As$\| A\| < 1$, the sequence of positive numbers $(A^m)_{m\ge 0}$ is converging towards the real number 0. Since $\|A^m\| \le \|A\|^m$ (by a recurrence on m), the sequence of positive number $(\|A^m\|)_{m\ge 0}$ is also converging towards the real number 0. Finally, we can conclude that $(A^m)_{m\ge 0}$ is converging towards the null matrix that means $\lim_{m\to\infty} A^m = 0$

□

Hence, we get that the iteration matrix $M^{-1}N$ and thus the Gauss-Seidel algorithm converges if and only if $\rho(\mathrm{M}^{-1}\mathrm{N}) < 1$.

In particular, we get the following results:

- If A is positive-definite and symmetric then $\rho(\mathrm{M}^{-1}\mathrm{N}) < 1$
- If A is diagonally dominant then $\rho(\mathrm{M}^{-1}\mathrm{N}) < 1$

We have seen the case when our system is linear. Let us discover how to deal with a piecewise linear system.

d) <u>In our case</u>

In our project, we are looking for a solution $\mathrm{x} \in \mathbb{R}^n$ with a piecewise linear function $\mathrm{F}: \mathbb{R}^n \Rightarrow \mathbb{R}^n$ such that $\mathrm{F(x)} = 0$.

<u>Lemma:</u> (One to One solution correspondence).
Under our general assumptions with $\det(J) \neq 0$ a point $x_* \in \mathbb{R}^n$ is a solution of $F(x) = 0$ if and only if it is a fixed point of $h_x(h_z(x))$.

- $h_z: \mathbb{R}^n \Rightarrow \mathbb{R}^s$ such as $h_z(x) = (I - L * \Sigma_x)^{-1} * (c + Zx)$
- $h_x: \mathbb{R}^s \Rightarrow \mathbb{R}^n$ such as $h_x(z) = J^{-1} * b - J^{-1} * Y * \Sigma_z * z$

These two functions correspond to an alternate form making easier the resolution of a piecewise linear problem.

<u>Proof:</u>

We recall the results issue from the abs-normal method in the part **III)**:

$$y = b + Jx + Y|z|$$

$$z = c + Zx + L|z|$$

We have the equivalences: $\mathrm{F(x)} = 0 \Leftrightarrow y = 0$

$$\Leftrightarrow \begin{cases} \mathrm{z = c + Zx + L|z|} \\ 0 = b + Jx + Y|z| \end{cases}$$

$$\Leftrightarrow \begin{cases} (\mathrm{I} - \mathrm{L}\Sigma_z)\mathrm{z} = \mathrm{c + Zx} \\ Jx = -b - Y\Sigma_z z \end{cases}$$

$$\Leftrightarrow \begin{cases} \mathrm{z} = (\mathrm{I} - \mathrm{L}\Sigma_z)^{-1}\,(\mathrm{c + Zx}) = h_z(x) \\ x = -J^{-1}\,b - J^{-1}\,\mathrm{Y}\Sigma_z z = h_x(z) \end{cases}$$

$$\Leftrightarrow x = h_x\big(h_z(x)\big)$$

□

We have seen in the Lemma above that if a vector x, i a fixed point of the composition of these functions, then it is a solution of our initial equation $\mathrm{F(x)} = 0$. This is an important result which allows us to find a solution (if it does exist) in the piecewise linear case.

Furthermore, for the rest of our project we notice:

$$x^+ = h_x\left(h_z(x)\right)$$

with x^+ the convergence points of the alternating Seidel algorithm.

Consequently, according to the One-to-one solution correspondence Lemma and as x^+ is a fixed point of $h_x(h_z(x))$, we have that x^+ is a solution of the initial equation $\mathrm{F(x)} = 0$.

Let us see now how the convergence is conditioned in this case.

<u>Condition of convergence:</u>

The document [3] provides us the following condition of convergence for the alternating Seidel algorithm:

$$\|L\|_p + \|Z * J^{-1} * Y\|_p < 1 \; \textbf{(2)}$$

Indeed, the proposition below proves this condition and plus, ensures the uniqueness of the root searched.

<u>Proposition:</u> We recall S is the Schur complement: $\mathrm{S} = \mathrm{L} - \mathrm{Z}J^{-1}\mathrm{Y}$
If in some p-norm $\|S - L\|_p + \|L\|_p < 1$
Then the iteration $x^+ = h\_x(h\_z(x))$ converges from all x_0 to the unique fixed point x_*.

Moreover, the corresponding $x_* = -J^{-1}(b + Y|z_*|)$ is the unique root of $F(x) = 0$.

Hence, as $S - L = Z * J^{-1} * Y$, the condition **(2)** implies effectively the convergence.

Example 1:

In this case, we took back the figure 3 and we add the zeros of this function

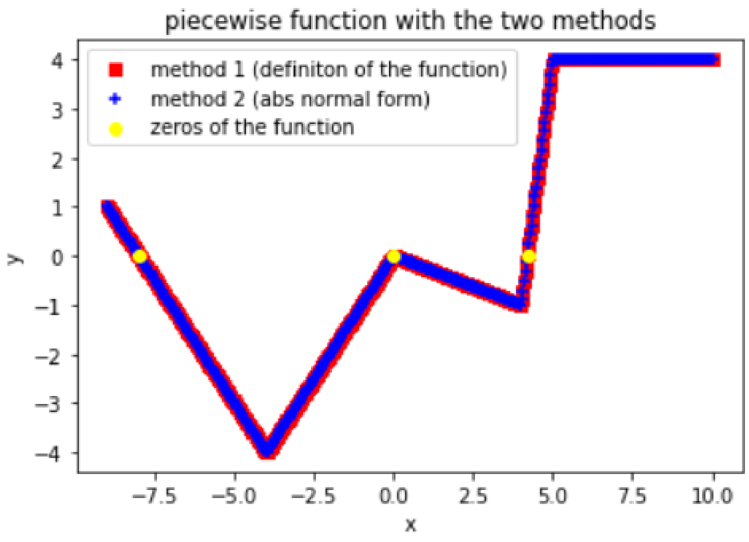


Figure 8

We can see that this function has three zeros. We supposed the Newton method acquired, we applied both methods on our problem and print the error. We took $x_0 = 4.01$ (Figure 9) and both methods converge. Nevertheless, there are not converging on the same point. We have the Newton algorithm converge towards 4.2 and the Gauss Seidel algorithm converge towards zero.
Gauss Seidel needs 32 iterates and Newton needs 28 iterates.

Nevertheless, this does not mean that Newton is faster than Gauss Seidel. In some case, when we change x_0 , the two algorithms have the same speed like when we took $x_0 = 3$ (Figure 10).

Moreover, we code the convergence condition of Gauss Seidel and these conditions were not respected. In spite of, we had convergence, this means those conditions are sufficient but not necessary.

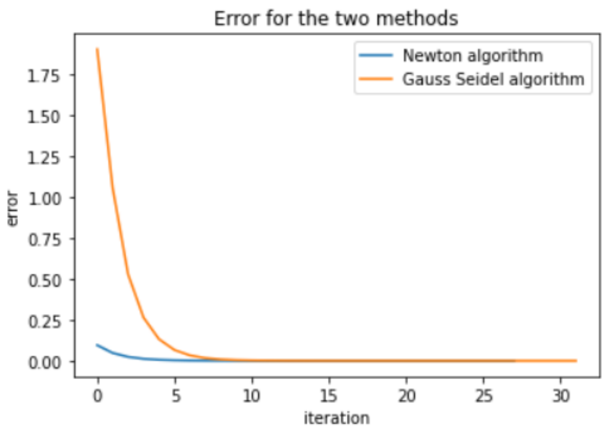


Figure 9

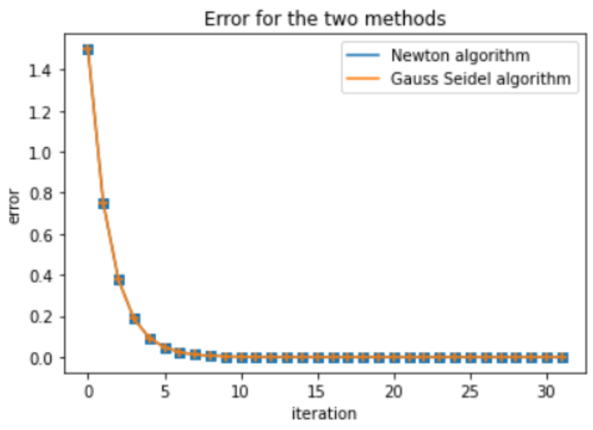


Figure 10

Conclusion

In this project, we tried to solve a non-linear system.
To reach this objective, we were interested by approaching a function by a piecewise linear function.
With this in mind, we studied the work of a sequential graph and we used it to calculate the abs normal form. To solve our system, we studied the Gauss Seidel method in the linear and piecewise linear cases, and then we compared it with the Newton method.

While we studied the theoretical part, we made tests thanks to the programming language Python.
Firstly, we were interested by functions from $\mathbb{R} \rightarrow \mathbb{R}$. According to the formula (1), we created and drawed functions for some initial points. Then we coded the abs-normal form and the algorithms of Gauss-Seidel and Newton with success.

Secondly, we applied those methods on a given function from $\mathbb{R}^2 \rightarrow \mathbb{R}^2$.
We add that we did a code object-oriented in order to have a better visibility and understanding.

The results obtained were satisfying with a well understanding of the theoretical part and good numerical results on Python. With more time, we could have automatized the sequential graph, to get the abs normal for every functions given from$\mathbb{R}^2 \rightarrow \mathbb{R}^2$ and approach the resolution of systems with functions from $\mathbb{R}^n \rightarrow \mathbb{R}^n$.

Bibliography

[1] https://en.wikipedia.org/wiki/Piecewise

[2]https://jermwatt.github.io/machine_learning_refined/notes/3_First_order_methods/3_5_Automatic.html

[3] https://hal.inria.fr/hal-01286443/document

[4] https://fr.wikipedia.org/wiki/Algorithme_du_gradient

[5] Scholtes, S.: Introduction to piecewise differentiable equations. Springer (2012)

[6] https://math.unice.fr/~jabin/CTD3-8.pdf

[7] http://www.optimization-online.org/DB_FILE/2013/12/4186.pdf

[8] https://fr.wikipedia.org/wiki/M%C3%A9thode_de_Gauss-Seidel

[9]http://www.lesmathematiques.net/phorum/file.php?4,file=5528,filename=Pages_de_analysematricielle2.pdf

[10] https://math.unice.fr/~jabin/CTD3-8.pdf

Appendix

The code:

https://colab.research.google.com/drive/1zLklfJFHAwhEGb2PlxBYFz7cMTgKt9KI#scrollTo=k5GhknTVvRPr&uniqifier=1