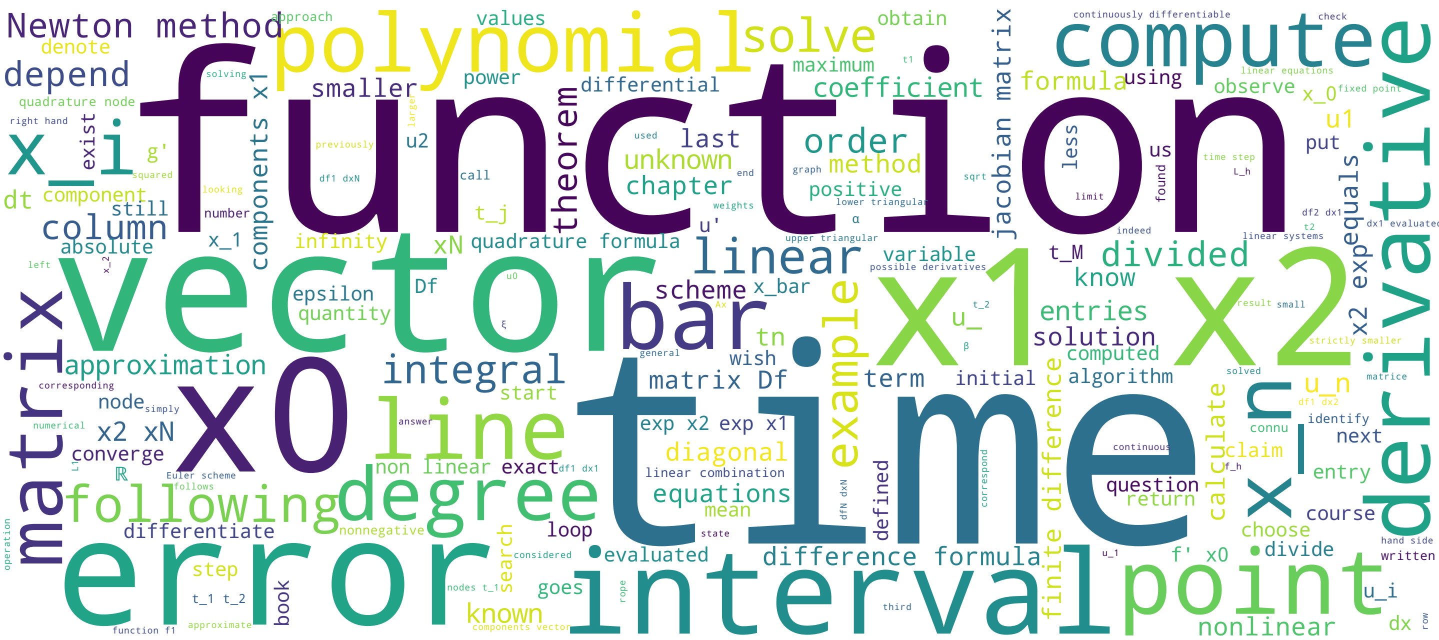
Chapitre 8 : Systèmes d'équations non linéaires

Introduction à l'analyse numérique

Prof. Marco Picasso



Video

EPFL

Chap 8 - Systèmes d'équations non linéaires

$\vec{x} \in \mathbb{R}^N$ tq $\vec{f}(\vec{x}) = \vec{0}$ $\quad \vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix}$ $\quad \vec{f} : \mathbb{R}^N \to \mathbb{R}^N$, $\vec{x} \to \vec{f}(\vec{x})$ $\quad \vec{f}(\vec{x}) = \begin{pmatrix} f_1(x_1\, x_2 \cdots x_N) \\ f_2(x_1, x_2 \cdots x_N) \\ \vdots \\ f_N(x_1\, x_2 \cdots x_N) \end{pmatrix}$

Exemple: $N=2$ $\quad (x_1, x_2)$ tq $\begin{cases} 2x_1 - x_2 + e^{x_1} = 0 \\ -x_1 + 2x_2 + e^{x_2} = 0 \end{cases}$ $\quad \vec{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ $\quad \vec{f}(\vec{x}) = \begin{pmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{pmatrix} = \begin{pmatrix} 2x_1 - x_2 + e^{x_1} \\ -x_1 + 2x_2 + e^{x_2} \end{pmatrix}$

Newton ?: $\quad N=1 \quad x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)} \quad f'(x)$

Now I wish to extend Newton's method to non linear systems of equations. The problem I wish to solve is the following: I search x in R^N, where x is a vector with n components x1, x2,... xN such that the vector f of x is equal to 0. Here x is a vector with N components x1, x2,... xN and f a function of R^N to R^N which returns for x a vector f (x); the first component of f(x) is an application f1 which depends on x, that is x1, x2,... xN, the second component f2 depends on x1, x2,... xN and so on up to the last component fN which is a function of x1, x2,... xN. An example for N equal to 2: I search the couple (x1, x2) such that 2 x1 - x2 + exp(x1) = 0 and -x1 + 2 x2 + exp(x2) = 0. In this case the vector x, is of course the vector of components (x1, x2) and f(x) is f1 which depends on x1 and x2, and f2 also depends on x1 and x2. f1 is the first line of the equation, it is 2 x1 - x2 + exp(x1) the first equation, and the second line f2 is - x1 + 2 x2 + exp(x2). How to write Newton's method? Lets re-write the method for the case N=1, the equation for one unknown. We wrote x_n+1 = x_n - f(x_n) / f ' (x_n). Here f is a function depending on one variable x and its derivative is f '(x).

Notes

Summary

0m 01s

Chap 8 - Systèmes d'équations non linéaires

$\vec{x} \in \mathbb{R}^N$ tq $\vec{f}(\vec{x}) = \vec{0}$ $\quad \vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix}$ $\quad \vec{f} : \mathbb{R}^N \to \mathbb{R}^N,\ \vec{x} \to \vec{f}(\vec{x})$ $\quad \vec{f}(\vec{x}) = \begin{pmatrix} f_1(x_1, x_2 \cdots x_N) \\ f_2(x_1, x_2 \cdots x_N) \\ \vdots \\ f_N(x_1, x_2 \cdots x_N) \end{pmatrix}$

Exemple: $N=2$ (x_1, x_2) tq $\begin{cases} 2x_1 - x_2 + e^{x_1} = 0 \\ -x_1 + 2x_2 + e^{x_2} = 0 \end{cases}$ $\quad \vec{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ $\quad \vec{f}(\vec{x}) = \begin{pmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{pmatrix} = \begin{pmatrix} 2x_1 - x_2 + e^{x_1} \\ -x_1 + 2x_2 + e^{x_2} \end{pmatrix}$

Newton ?: $N=1$ $\quad x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)}$ $\quad f'(x) \longrightarrow Df(\vec{x})$ matrice $N \times N$ jacobienne

$$Df(\vec{x}) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(\vec{x}) & \frac{\partial f_1}{\partial x_2}(\vec{x}) & \cdots & \frac{\partial f_1}{\partial x_N}(\vec{x}) \\ \frac{\partial f_2}{\partial x_1}(\vec{x}) & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_N} \\ \vdots & & \ddots & \vdots \\ \frac{\partial f_N}{\partial x_1} & \frac{\partial f_N}{\partial x_2} & \cdots & \frac{\partial f_N}{\partial x_N} \end{pmatrix}$$

Ex : $N=2$ $\quad Df(\vec{x}) = ($

Now, for N larger than 1, we have to deal with a function f which has first component a function f1 of N variables, so as f2 to fN. I can compute the partial derivatives df1 / dx1, df1 / dx2, df1 / dxN and df2 / dx1 and so on up to dfN / dxN. There are N times N derivatives which can be computed from f(x). I can put these N times N derivatives in what I shall denote as Df(x) which is the N times N matrix, N rows and N columns, which I call the jacobian matrix. Df in any point x is the matrix containing all the possible derivatives, the first is df1 / dx1 evaluated at (x1, x2,... xn), that is the vector x, next df1 / dx2 evaluated in x, this up to df1 / dxN evaluated in x. These are all the possible derivatives from this function f1 which depends on (x1, x2,... xn). I can do the same for the second function f2, df2 / dx1 evaluated in x df2 / dx2 up to df2 / dxN, and I can do the same thing untill the last function fN dfN / dx1, dfN / dx2, up to dfN / dxN which is the derivative of fN with respect to the last variable xN. This is a matrix N times N, when N = 2, for the example I considered here, I have f(x), here the expression : (f1 (x1, x2), f2 (x1, x2)) and I can compute four derivatives, that I place in the 2 times 2 matrix Df(x).

Notes

Summary

2m 47s

Chap 8 - Systèmes d'équations non linéaires

$\vec{x} \in \mathbb{R}^N$ tq $\vec{f}(\vec{x}) = \vec{0}$ $\quad \vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix}$ $\quad \vec{f} : \mathbb{R}^N \to \mathbb{R}^N$, $\vec{x} \to \vec{f}(\vec{x})$ $\quad \vec{f}(\vec{x}) = \begin{pmatrix} f_1(x_1, x_2 \cdots x_N) \\ f_2(x_1, x_2 \cdots x_N) \\ \vdots \\ f_N(x_1, x_2 \cdots x_N) \end{pmatrix}$

Exemple: $N=2$ (x_1, x_2) tq $\begin{cases} 2x_1 - x_2 + e^{x_1} = 0 \\ -x_1 + 2x_2 + e^{x_2} = 0 \end{cases}$ $\quad \vec{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ $\quad \vec{f}(\vec{x}) = \begin{pmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{pmatrix} = \begin{pmatrix} 2x_1 - x_2 + e^{x_1} \\ -x_1 + 2x_2 + e^{x_2} \end{pmatrix}$

Newton?: $N=1$ $\quad x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)}$ $\quad f'(x) \longrightarrow Df(\vec{x})$ matrice $N \times N$ jacobienne

$$Df(\vec{x}) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(\vec{x}) & \frac{\partial f_1}{\partial x_2}(\vec{x}) & \cdots & \frac{\partial f_1}{\partial x_N}(\vec{x}) \\ \frac{\partial f_2}{\partial x_1}(\vec{x}) & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_N} \\ \vdots & & \ddots & \vdots \\ \frac{\partial f_N}{\partial x_1} & \frac{\partial f_N}{\partial x_2} & \cdots & \frac{\partial f_N}{\partial x_N} \end{pmatrix}$$

Ex : $N=2$ $\quad Df(\vec{x}) = \begin{pmatrix} 2+e^{x_1} & -1 \\ -1 & 2+e^{x_2} \end{pmatrix}$

$N=1$ $\quad f'(x_n)(x_n - x_{n+1}) = f(x_n)$

N qque $\quad Df(\vec{x}^n)(\vec{x}^n - \vec{x}^{n+1}) = \vec{f}(\vec{x}^n)$ $\quad \vec{x}^n = \begin{pmatrix} x_1^n \\ x_2^n \\ \vdots \\ x_N^n \end{pmatrix}$

I differentiate the first line with respect to x1, I get 2 + exp(x1), by differentiating the first line with respect to x2 I get -1, and on, I differentiate the second line, I obtain -1 and 2 + exp(x2). I can therefore compute the jacobian matrix Df(x). of the function f evaluated in point x. Lets come back to Newton's method. I said for the case N equal to 1, I can write the Newton's method in a slightly different way, I can write f ' (x^n) times (x^n - x^n+1) equal to f(x^n) which is an alternative formulation. In the general case, any N, N equations with N unknowns, instead of f ' (x^n) I must use the jacobian matrix evaluated in x^n. Here the vector x^n is the vector with N components x1^n, x2^n,.. xN^n. Of course I start from x^0, which is x1^0, x2^0,... up to xN^0. Let's return to the formulation, here I have the derivative times the increment. Note here that the index corresponding to the Newton iteration, n, is a superscript. unlike previously where it was a lowerscript, the lower index being reserved for the components. of the vector. So Df(x^n) * (x^n - x^(n+1)) = f(x^n). You have here a matrix, a vector and here a vector. So a matrix N times N multiplied by a N components vector is equal to a N components vector.

Notes

Summary

5m 11s

Chap 8 - Systèmes d'équations non linéaires

$\vec{x} \in \mathbb{R}^N$ tq $\vec{f}(\vec{x}) = \vec{0}$ $\quad \vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix}$ $\quad \vec{f} : \mathbb{R}^N \to \mathbb{R}^N$, $\vec{x} \to \vec{f}(\vec{x})$ $\quad \vec{f}(\vec{x}) = \begin{pmatrix} f_1(x_1, x_2 \cdots x_N) \\ f_2(x_1, x_2 \cdots x_N) \\ \vdots \\ f_N(x_1 x_2 \cdots x_N) \end{pmatrix}$

Exemple: $N=2$ (x_1, x_2) tq $\begin{cases} 2x_1 - x_2 + e^{x_1} = 0 \\ -x_1 + 2x_2 + e^{x_2} = 0 \end{cases}$ $\quad \vec{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ $\quad \vec{f}(\vec{x}) = \begin{pmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{pmatrix} = \begin{pmatrix} 2x_1 - x_2 + e^{x_1} \\ -x_1 + 2x_2 + e^{x_2} \end{pmatrix}$

Newton ?: $N=1$ $\quad x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)}$ $\quad f'(x) \longrightarrow Df(\vec{x})$ matrice $N \times N$ jacobienne

$$Df(\vec{x}) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(\vec{x}) & \frac{\partial f_1}{\partial x_2}(\vec{x}) & \cdots & \frac{\partial f_1}{\partial x_N}(\vec{x}) \\ \frac{\partial f_2}{\partial x_1}(\vec{x}) & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_N} \\ \vdots & & \ddots & \vdots \\ \frac{\partial f_N}{\partial x_1} & \frac{\partial f_N}{\partial x_2} & \cdots & \frac{\partial f_N}{\partial x_N} \end{pmatrix}$$

Ex : $N=2$ $\quad Df(\vec{x}) = \begin{pmatrix} 2+e^{x_1} & -1 \\ -1 & 2+e^{x_2} \end{pmatrix}$

$N=1$ $\quad f'(x_n)(x_n - x_{n+1}) = f(x_n)$

N qque $\quad \underbrace{Df(\vec{x}^n)}_{\text{connu}} (\underbrace{\vec{x}^n}_{\text{connu}} - \underbrace{\vec{x}^{n+1}}_{\text{inconnue}}) = \vec{f}(\vec{x}^n)$ $\quad \vec{x}^n = \begin{pmatrix} x_1^n \\ x_2^n \\ \vdots \\ x_N^n \end{pmatrix}$

Note that, since x^n is known, then the jacobian matrix Df(x^n) is computable. donc ceci est connu, x^n est connu Here x^(n+1) is the unknown, again, here f(x^n) is known since x^n is known. Therefore, a linear system of equations has to be solved, a system with N linear equations and N unknowns. At each step, Newton's method requires to solve a linear system of N equations with N unknowns, which will give us x^(n+1). The algorithm is the following: Start from a given x^0 which is an approximation of x such that f(x) = 0, the system of N non linear equations. Next for n = 0,1,2,... I can calculate the second term of the linear system which I will denote b, b is equal to f(x^n), x^n being known, I can compute f(x^n) the right hand side of the linear system, I can then calculate the matrix A of the linear system, A is Df(x^n) which can be computed as soon as x^n is known, I then solve the linear system A times y equals to b, and next I will set x^(n+1), I have that y is equal to x^n - x^(n+1), thus x^(n+1) is x^n minus y. To terminate, I need a stopping criterion, if some norm of y is smaller than a prescribed tolerance, for example 10^-8, I exit my loop and I have found a good approximation of my nonlinear system.

Notes

Summary

7m 13s