TME n° 3

Ryan Lahfa

22 décembre 2019

Table des matières

Table des figures

Exercice 3

Q1. Ici, on n'emploiera pas la Symbolic Toolbox afin de calculer les dérivées, sinon cela rendrait l'intérêt de l'algorithme limité. On se contentera de faire des différencies finies d'ordre 1.

```
% Q1.
function df = approx_d(f, a, stepsize)
    df = (f(a + stepsize) - f(a)) / stepsize;
end

function [iters, x] = newton(f, x0, tol, eps)
    oldx = x0;
    x = oldx - f(oldx)/approx_d(f, oldx, eps);
    iters = [x0 x];
```

```
    while abs(oldx - x) > tol
        oldx = x;
        x = oldx - f(oldx)/approx_d(f, oldx, eps);
        iters = [iters x];
    end
end
```

Q2. On utilisera ce code pour tracer:

```
% Q2. D'après WA, on attend x ~0.865474.
x_ = 0.865474;
test_f = @(x)(x^3 - cos(x));

[iters, x] = newton(test_f, 0, 0.01, 0.01);
fprintf("Newton solution: %f\n", x);

error = abs(iters - x_);
plot(0:(length(iters) - 1), error);
legend('x_k');
ylabel('Erreur en valeur absolue');
xlabel('Itération');
title('Itérations de la méthode de Newton appliqué au problème x^3 = cos(x)');
```

Et on obtient le résultat dans la figure 1. Il semblerait que le point en 0 soit un peu faux, mais on peut l'ignorer. De toute façon, on constate qu'on commence loin de la solution, puis qu'on converge géométriquement vers la solution.

Exercice 4

Q1. On a besoin d'une fonction de calcul d'inverse modulaire, donc on en écrit une. Cette fois-ci, on utilisera directement `diff` de la Symbolic Toolbox, puisqu'on travaille dans des corps finis, cela semble mieux d'éviter une méthode par des différences finies qui peuvent entraîner des problèmes numériques. Mais il est certainement possible d'introduire un opérateur de différence finie avec une précision intéressante.

```
% Q1.
function x = newton_padique(f, p, a, itmax)
    g = diff(f);
    i = mulinv(g(a), p);
    x = a;
    r = p;
    for j=1:itmax
        x = x + r*mod(-f(x)*i/r, p);
        r = p*r;
        disp(x);
    end
```

```
end

function y = mulinv(x,p)
    if ~isprime(p)
        disp('The field order is not a prime number');
        return
    elseif x>=p
        disp('All or some of the numbers do not belong to the field');
        return
    elseif x == 0
        disp('0 does not have a multiplicative inverse');
        return
    end
    k = 0;
    m=mod(k*p+1,x);
    while m
        k=k+sign(m);
        m=mod(k*p+1,x);
    end
    y=(k*p+1)./x;
end
```

Q2. On constate les itérées suivants:

```
x_1 = 10, f(x_1) = 0 mod 49
x_2 = 108, f(x_2) = 0 mod 343
x_3 = 2166, f(x_3) = 0 mod 2401
x_4 = 4567, f(x_4) = 0 mod 16807
x_5 = 38181, f(x_5) = 0 mod 117649
x_6 = 155830, f(x_6) = 0 mod 823543
x_7 = 1802916, f(x_7) = 0 mod 5764801
x_8 = 24862120, f(x_8) = 0 mod 40353607
x_9 = 266983762, f(x_9) = 0 mod 282475249
x_10 = 1961835256, f(x_10) = 0 mod 1977326743
x_11 = 5916488742, f(x_11) = 0 mod 13841287201
x_12 = 19757775943, f(x_12) = 0 mod 96889010407
x_13 = 116646786350, f(x_13) = 0 mod 678223072849
x_14 = 116646786350, f(x_14) = 0 mod 4747561509943
x_15 = 9611769806236, f(x_15) = 0 mod 33232930569601
x_16 = 42844700375837, f(x_16) = 0 mod 232630513987207
x_17 = 275475214363044, f(x_17) = 0 mod 1628413597910449
x_18 = 6789129606004840, f(x_18) = 0 mod 11398895185373143
x_19 = 75182500718243698, f(x_19) = 0 mod 79792266297612001
x_20 = 154974767015855699, f(x_20) = 0 mod 558545864083284007
x_21 = 1830612359265707720, f(x_21) = 0 mod 3909821048582988049
x_22 = 9650254456431683818, f(x_22) = 0 mod 27368747340080916343
x_23 = 17386273849691718l876, f(x_23) = 0 mod 191581231380566414401
```

```
x_24 = 1323350126780315668282, f(x_24) = 0 mod 1341068619663964900807
x_25 = 5346555985772210370703, f(x_25) = 0 mod 9387480337647754305649
x_26 = 52283957674010981898948, f(x_26) = 0 mod 65712362363534280139543
x_27 = 380845769491682382596663, f(x_27) = 0 mod 459986536544739960976801
x_28 = 3140764988760122148457469, f(x_28) = 0 mod 3219905755813179726837607
x_29 = 12800482256199661328970290, f(x_29) = 0 mod 22539340290692258087863249
x_30 = 102957843418968693680423286, f(x_30) = 0 mod 157775382034845806615042743
```

Cette suite converge seulement dans les corps $\mathbb{Z}_p$ p-adiques [1], d'où, on ne s'attend pas à la voir converger sauf cas particuliers.

Exercice 5

Q1. On sait que l'itération de Newton est de la forme:

$$X_{n+1} = X_n - J_f^{-1}(X_n)f(X_n)$$

Or: $J_f = \mathrm{Mat}(df)$ (dans une base canonique).

À présent, soit $X \in \mathcal{M}_n(\mathbb{R})$, pour tout $H \in \mathcal{M}_n(\mathbb{R})$:

$$\begin{aligned} df_X(H) &= -d\mathrm{Inv}_X(H) \\ &= X^{-1}HX^{-1} \end{aligned}$$

En effet, on remarque très rapidement: $(X + H)^{-1} = (I_n + X^{-1}H)^{-1}X^{-1} = (I_n - X^{-1}H + X^{-2}H^2\varepsilon_X(H))X^{-1}$ sous hypothèse que $X+H$ et X sont inversibles.

Ce qui donne le résultat désiré, puisqu'il est linéaire en H (et la continuité est automatique en raison de la dimension finie).

Ensuite, en faisant l'approximation de Newton, on obtient:

$$X_{n+1} = X_n + X_n f(X_n)X_n = 2X_n + X_n A X_n$$

Q2. On écrit la méthode directement avec une détection de la convergence majorée par un nombre d'itération.

```
% Q2.
function [iters, B] = inv_matrix_newton(A, tol, itmax)
    B = A' / (norm(A, 'fro')^2); % A^T / Tr(A^T A) = A^T / ||A||^2_Frobenius
    iters = [iteration_error(B, A)];
    i = 0;
    while iteration_error(B, A) > tol && i < itmax
```

1. Qu'on peut très joliement interpréter comme la limite projective des $\mathbb{Z}/p^n\mathbb{Z}$!

```
        B = 2*B - B*A*B;
        iters = [iters iteration_error(B, A)];
        i = i + 1;
    end
end

function e = iteration_error(X_n, A)
    n = length(A);
    e = norm(eye(n) - X_n*A);
end
```

Notons que cette méthode est très intéressante puisqu'elle permet aussi d'avoir des pseudo-inverses dans le cas non-inversible.

Puis, on effectue le tracé des erreurs en fonction des itérations dans la figure 2 et 3.

On constate que pendant des tas d'itérations, l'erreur en norme ne change pas, mais il est probable que la norme de X_n change, puis soudainement, au bout de 70 itérations, la norme chute très rapidement vers 0 et atteint son optimum. Ce qui rend difficile l'analyse de l'efficacité de cet algorithme.

De même, à la figure 3, on observe une convergence très rapide en peu d'itérations qui ne dépend pas de l'ordre. Les résultats ont été reproduits plusieurs fois pour s'assurer de leur signification statistique.

Cependant, on voit à la figure 4 que le conditionnement joue un rôle dans la convergence, puisque que sur une matrice d'ordre 20, on obtient ni convergence mais en plus les itérations sont hautement instables. Ce n'est pas surprenant, la méthode de Newton tel qu'on l'implémente est sensible aux erreurs, puisqu'on repose sur des produits matriciels (somme + produit) et notre point de départ dépend de la norme de Frobenius qui repose sur un calcul de trace et de produit matriciel encore.

Or, la norme de Frobenius de cette matrice est en 10^{49}, ce qui fait que le point de départ se retrouve très proche de la matrice nulle.

En définitive, cette méthode dépend fortement du conditionnement de la matrice, lorsque que celui-ci est contrôlée, la méthode s'avère efficace comparée à une inversion et assez précise.

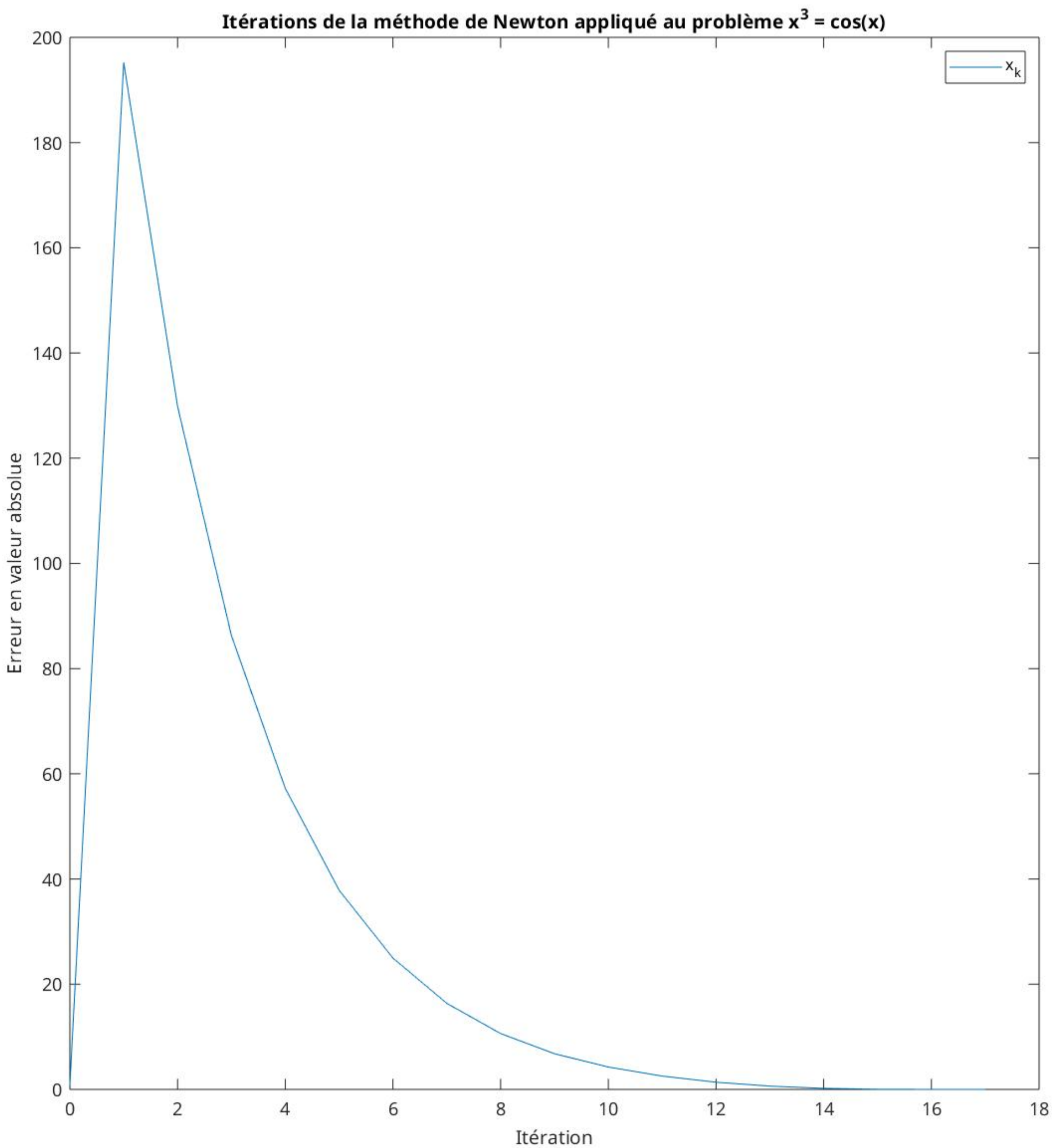

FIGURE 1 – Résolution de $x^3 = \cos(x)$ avec la méthode de Newton non-linéaire

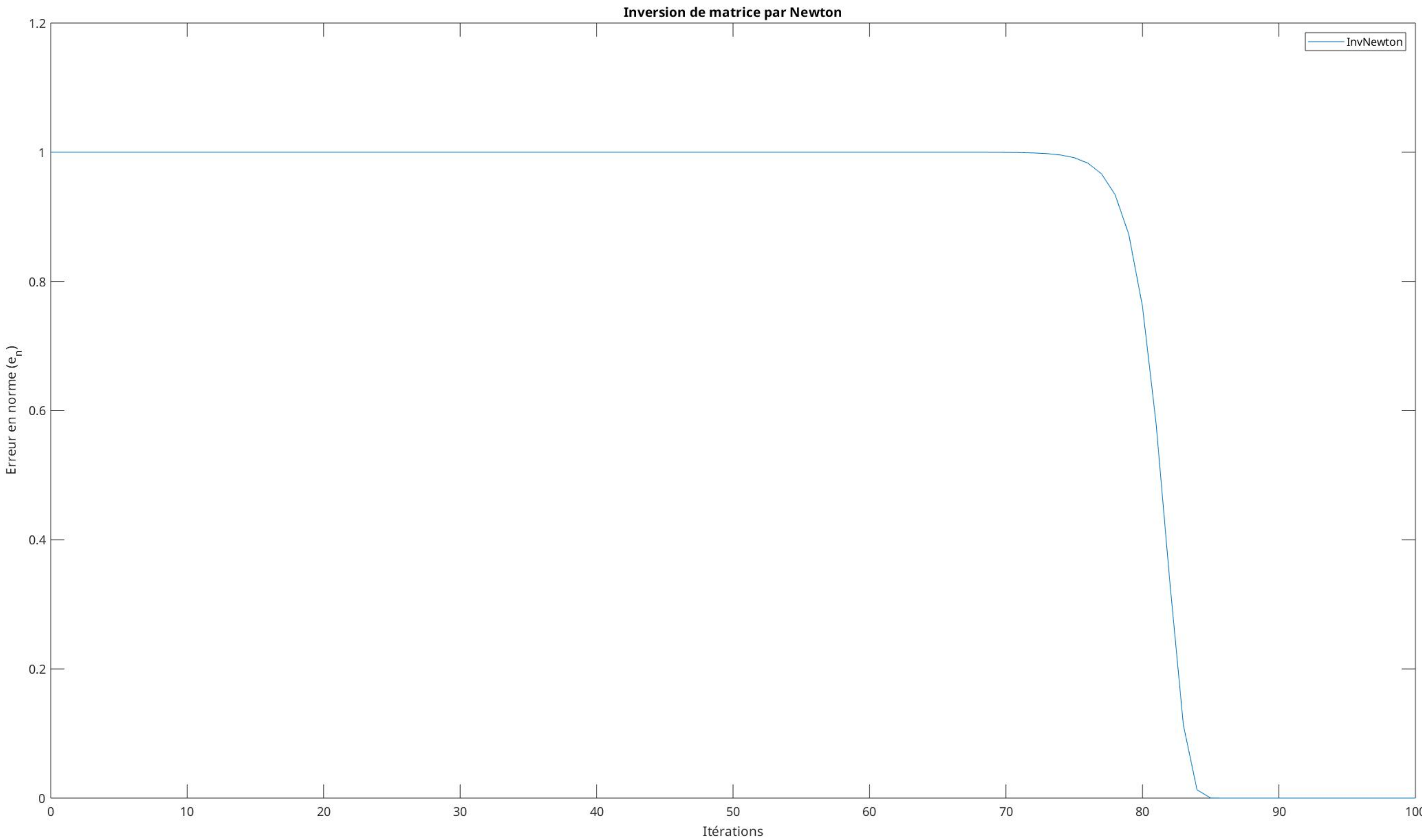

FIGURE 2 – Inversion de matrice avec la méthode de Newton sur la matrice de Vandermonde $[[1, 10]]$

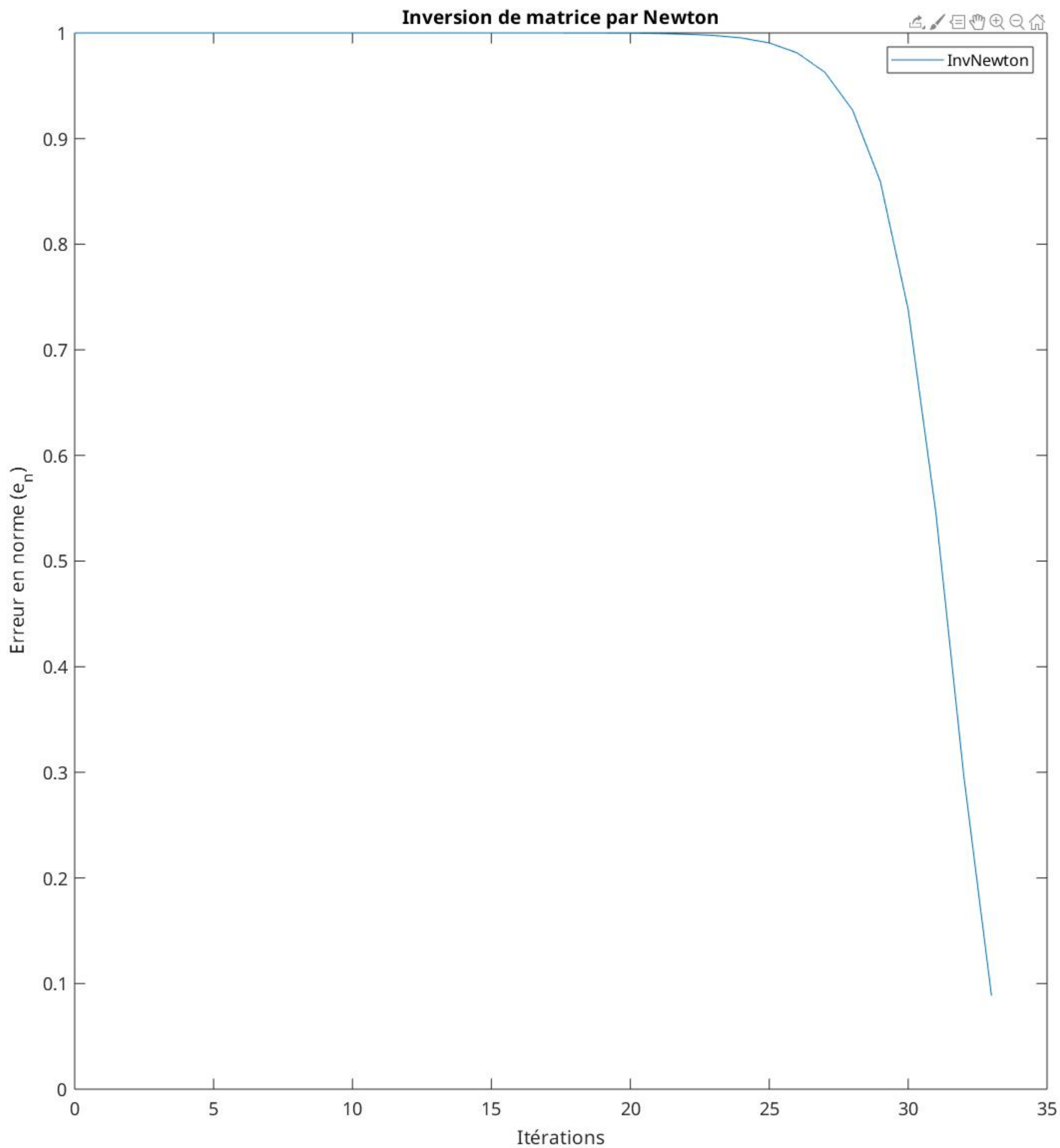

FIGURE 3 – Inversion de matrice avec la méthode de Newton sur une matrice aléatoire d'ordre 1000

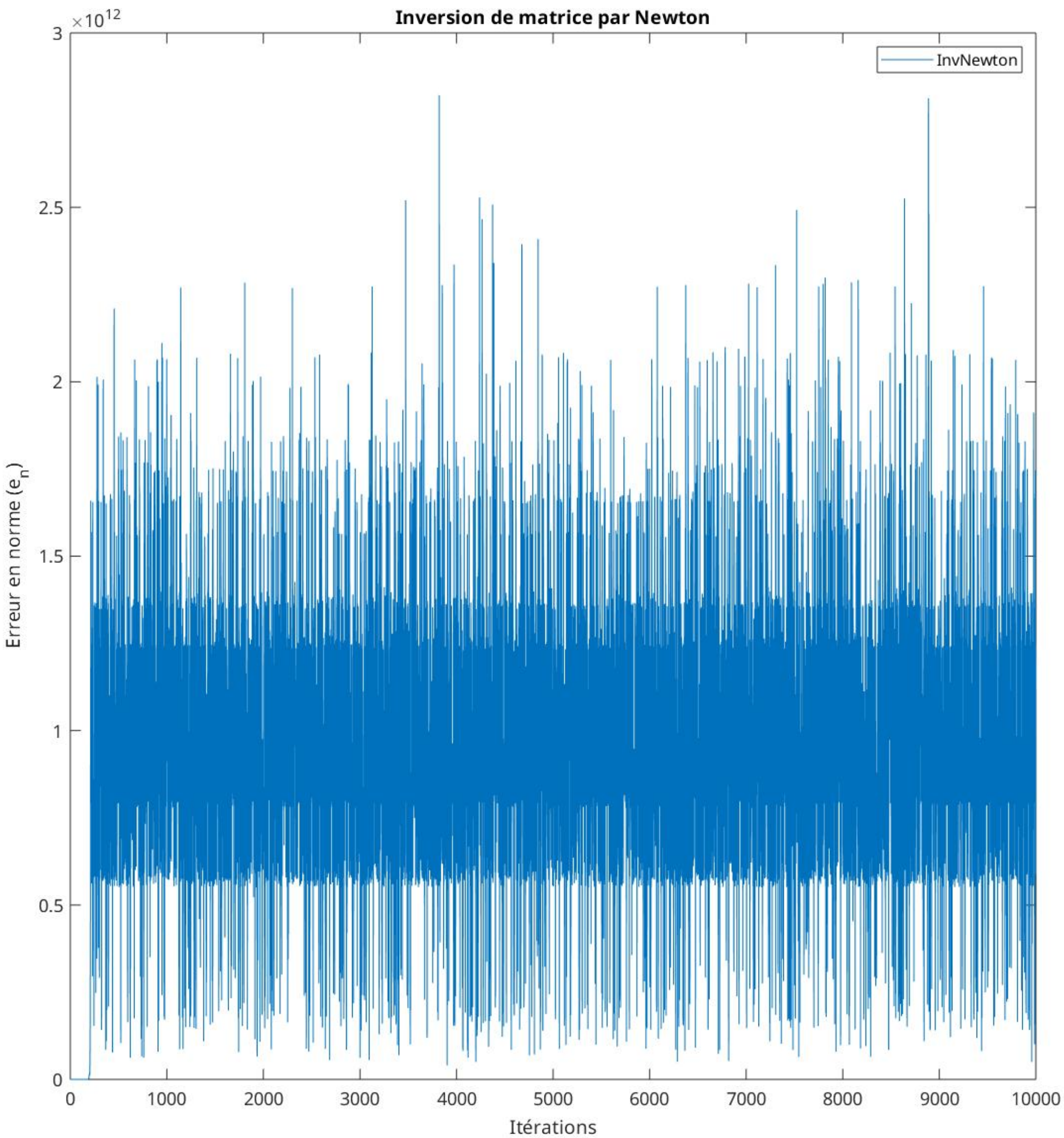

FIGURE 4 – Inversion de matrice avec la méthode de Newton sur la matrice de Vandermonde $[[1, 20]]$ $(\kappa(A) \sim 1.803307965815822 \times 10^{31})$