

# TP Calcul Numérique

Nicolas BOUTON

December 1, 2020

## Cours

### Notations

- **SAXPY** : Scalar a fois X Plus Y
- **GAXPY** : General matrix A fois X Plus Y
- **DSL** : Domain Specific Language
- **NNZ** : Number of Non Zero

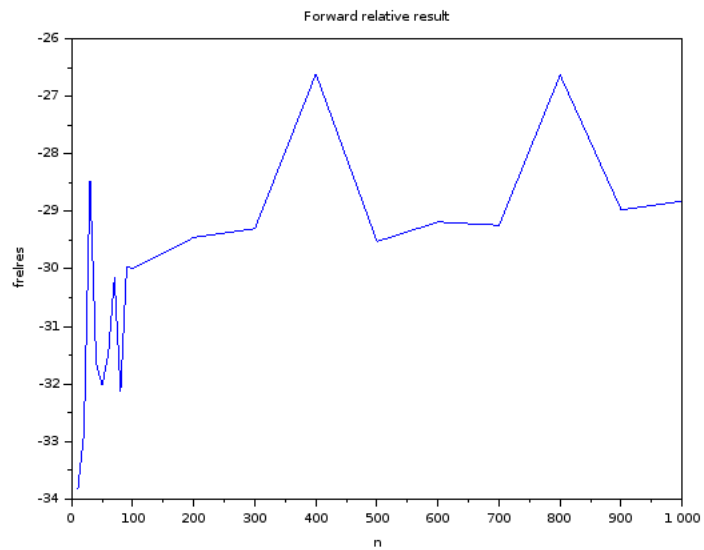
## Exercice 2

Nous remarquons 3 chose :

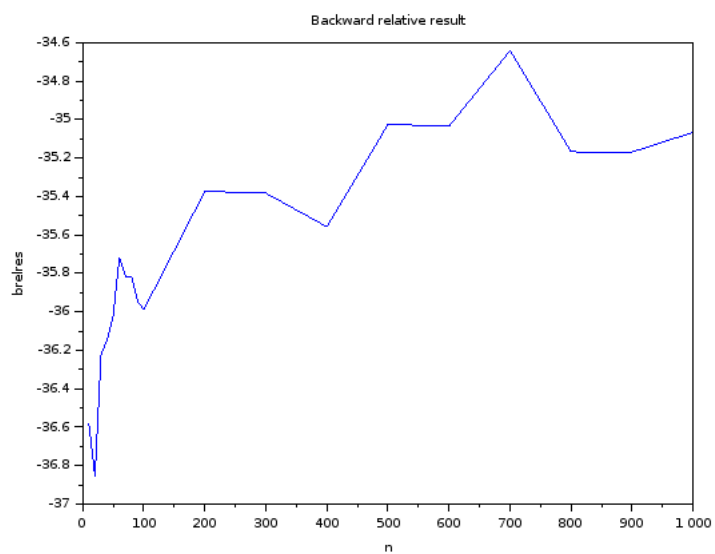
- Augmentation du conditionnement avec la taille
- Il faut considéré l'erreur arrièrè entaché du conditionnement pour avoir une approximation des résultats
- Nous manipulons des algorithmes avec certaines complexité qui peuvent être algorithmique ou bien de mémoire

Graphiques :

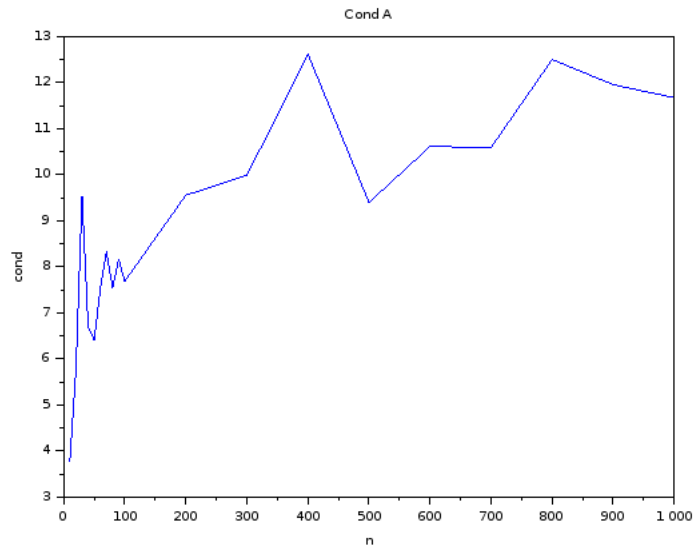
Pour l'erreur relative avant :



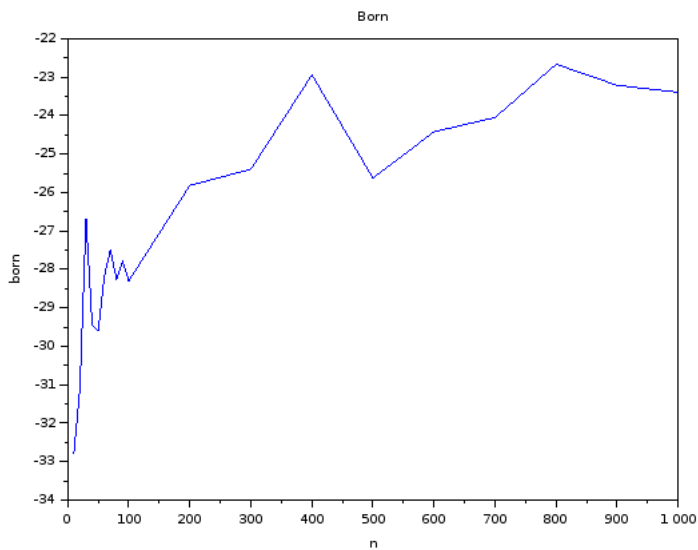
Pour l'erreur relative arrière :



Pour le conditionnement de A :



Pour les bornes :



On voit que les erreurs relatives avant et arrières, ainsi que le conditionnement de  $\mathbf{A}$  et les bornes sont très sensible à la taille de la matrice. De plus on peut apercevoir que c'est courbe suivent une complexité logarithmique. En effet au début l'augmentation de la taille entraîne une augmen-

tation du résultat, alors qu'à partir d'une taille de **100 x 100** la courbe se stabilise.

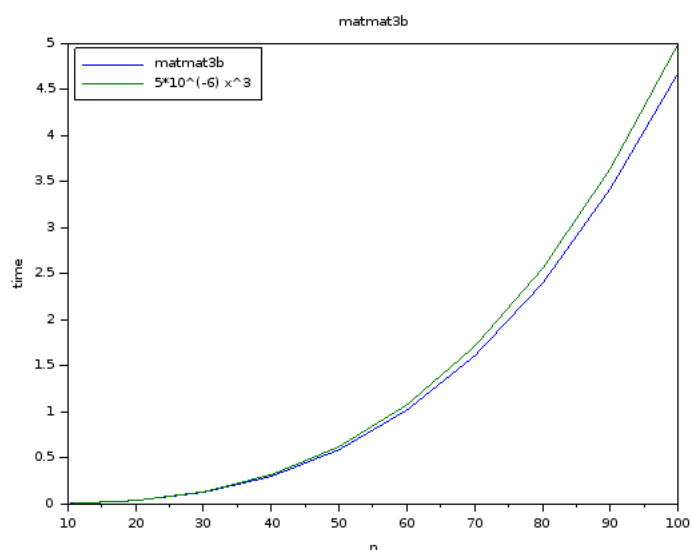
Le code source se trouve dans **exo2.sci**.

## Exercice 3

Le produit **matrice x matrice** à une complexité cubique. Nous remarquons dans un premier temps que l'appel au fonction de **Scilab** pour des calculs **vecteur x vecteur** ou bien **vecteur x matrice** au lieu de déroulé nous même les boucles est beaucoup plus performant. Il est d'autant plus performant si la taille des matrices est grandes. **Scilab** appelle des noyaux de calcul optimisé tel que **BLAS**.

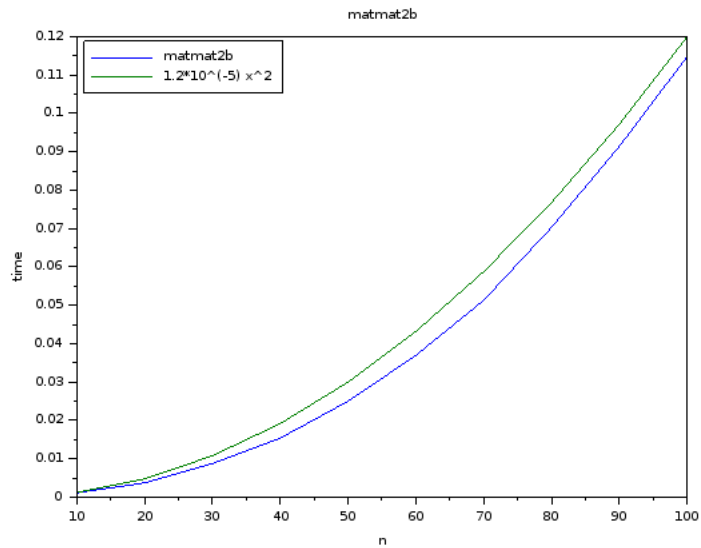
Graphiques :

Pour la fonction **matmat3b** :



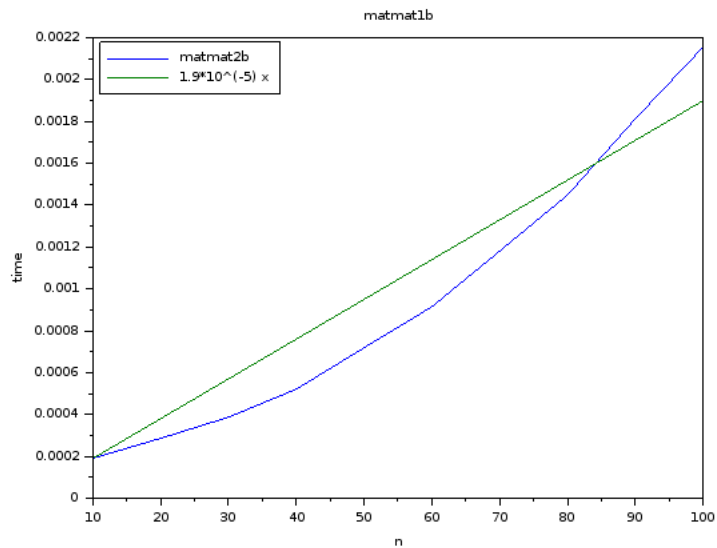
On voit que la fonction suit une complexité **cubique**, car **Scilab** n'appelle pas **BLAS** à cause du fait que nous avons déroulé 3 boucles.

Pour la fonction **matmat2b** :



On voit que la fonction suit une complexité **quadratique**, car **Scilab** appelle une fonction **vecteur x vecteur** de **BLAS** ce qui réduit l'ordre de 1, ce qui augmente la rapidité.

Pour la fonction **matmat1b** :



On voit que la fonction suit une complexité **linéaire**, car **Scilab** appelle

une fonction **vecteur x matrice** de **BLAS** ce qui réduit l'ordre de 2, ce qui augmente la rapidité.

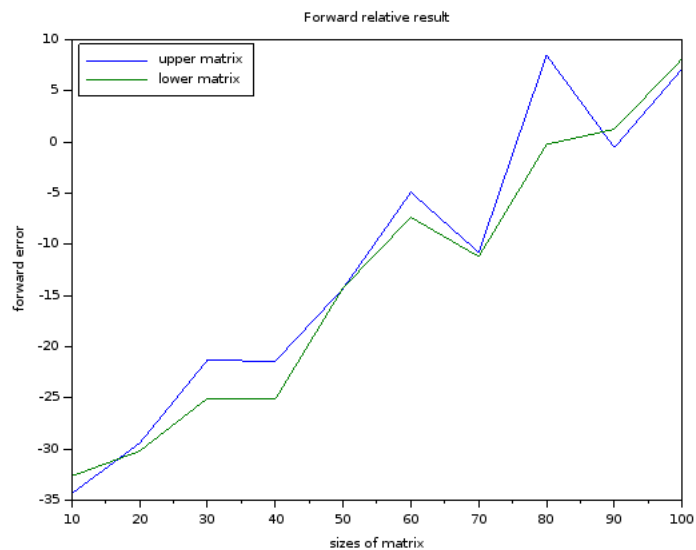
A l'avenir on appellera directement les **BLAS 3** c'est-à-dire le produit **matrice x matrice** car le calcul est plus rapide.

- Le code des fonction se trouve dans **opmat.sci**.
- Le code de test se trouve dans **exo3.sci**.

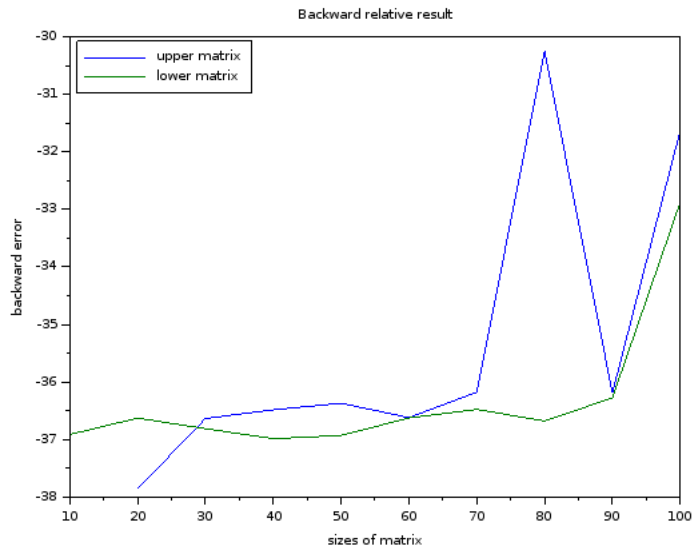
## Exercice 4

Graphiques :

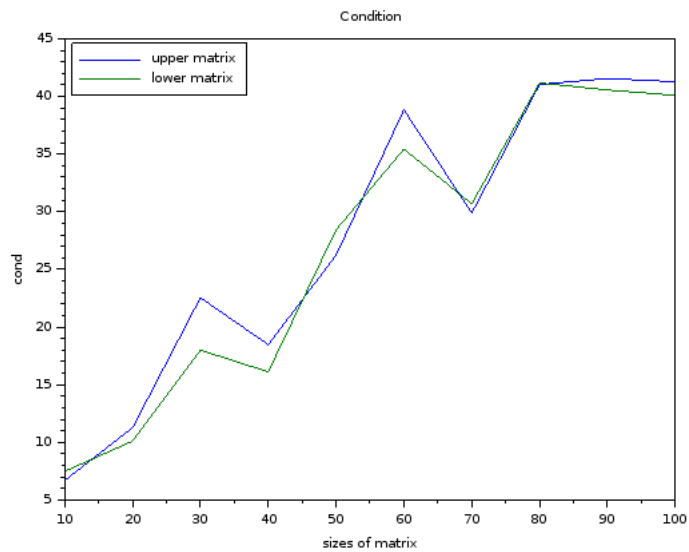
Pour l'erreur relative avant pour les 2 matrices :



Pour l'erreur relative arrière pour les 2 matrices :



Pour le conditionnement des 2 matrices :



On sait que l'erreur relative avant est borné avec la formule suivante : **forward error**  $\leq$  **condition number** x **backward error** et donc cela explique pourquoi nos erreurs relatives avant se dégrade très vite lorsque la taille augmente car le conditionnement augmente très vite et les résultats

sont entaché par le conditionnement qui est quand à lui borné par la précision de la machine qui est ici  $10^{-16}$ .

Donc après une certaine taille l'erreur relative avant n'est plus une erreur étant donné qu'il dépasse même le plus petit nombre entier représentable qui est **1**. L'erreur est acceptable tant qu'elle est inférieure à la précision machine qui est de  $10^{-16}$ .

Donc pour les erreurs relatives arrières, elles sont acceptables car elles sont inférieure à la précision machine qui est de  $10^{-16}$ .

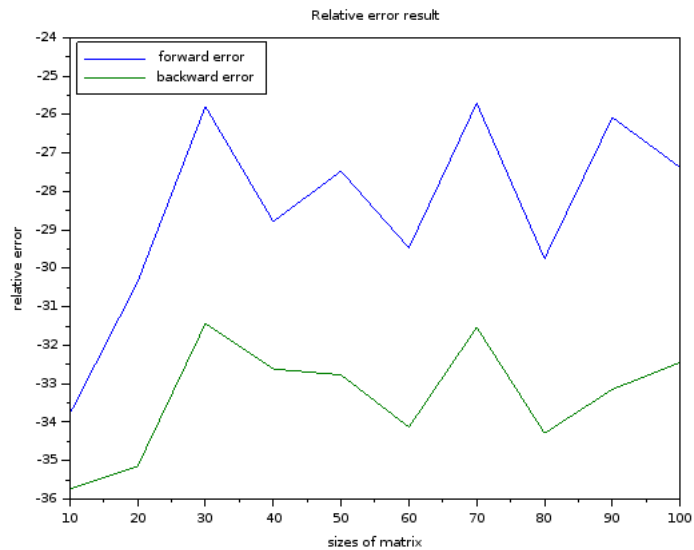
## Exercice 5

Complexité de la méthode de **Gauss** :

- en temps :  $\frac{2}{3}n^3$
- en espace :  $O(n^2)$

Graphiques :

Pour les erreurs relatives :



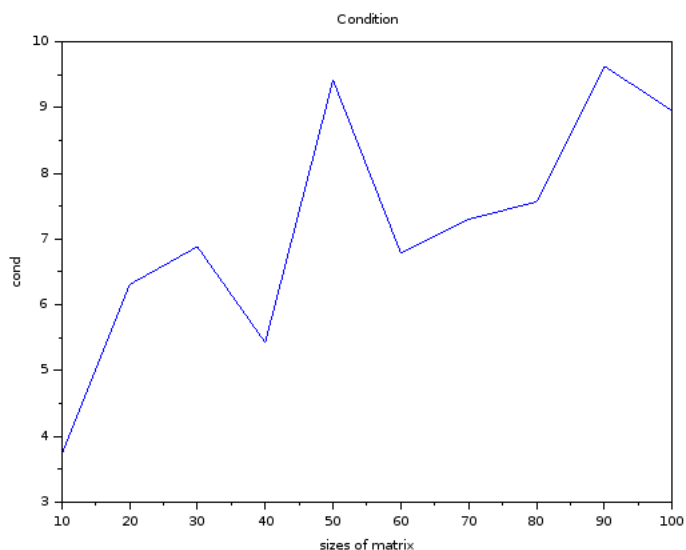
Les erreurs relatives avant et arrières sont acceptables car elles sont inférieure à la précision machine qui est de  $10^{-16}$ .



Et on a toujours l'erreur avant qui est plus élevée que l'erreur arrière mais qui reste proportionnelle, c'est-à-dire que si l'une baisse ou augmente l'autre fait de même étant donné qu'on a la relation suivante qui borne l'erreur arrière :

$$\text{forward error} \leq \text{condition number} \times \text{backward error}$$

Pour le conditionnement :



Le conditionnement pour une taille de matrice **100 x 100** est de  $10^{10}$  ce qui peut donner un résultat avec au minimum une précision de  $10^6$  avec une précision machine de  $10^{16}$ .

## Exercice 6

Complexité de la méthode **LU** :

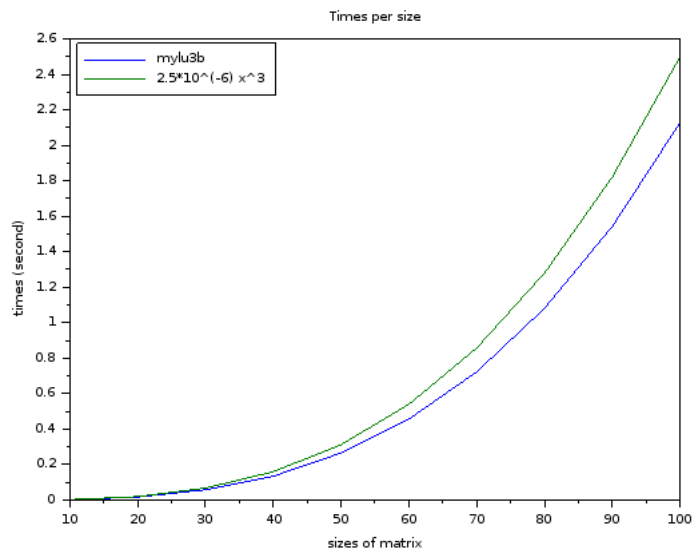
- en temps :  $\frac{2}{3}n^3$  (mais 1 seul fois car indépendant du second membre)
- en espace :  $O(n^2)$  (forme compact)

Remarques sur le pivot :

- éviter les pivots proches de zéro au début, ce qui implique d'utiliser des méthodes pour choisir les pivots

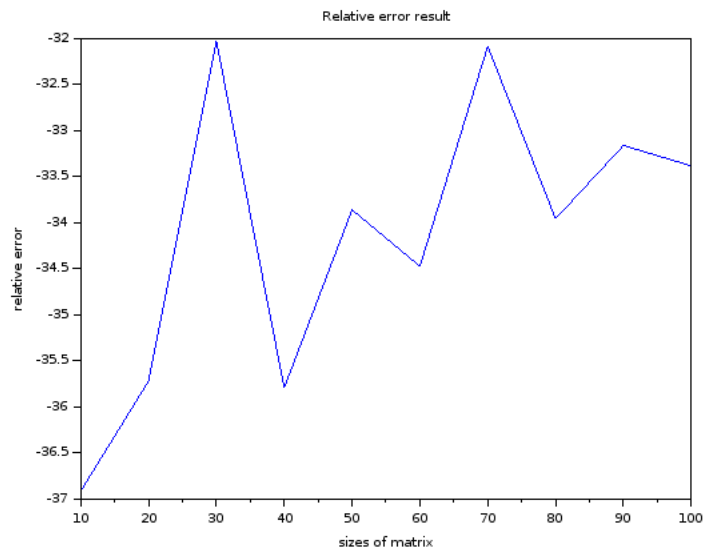
- on privilégiera généralement le **pivot partiel** si on veut assurer une stabilité numérique, car le **pivot complet** est trop coûteux et le gain en stabilité n'est pas assez bon

Temps :

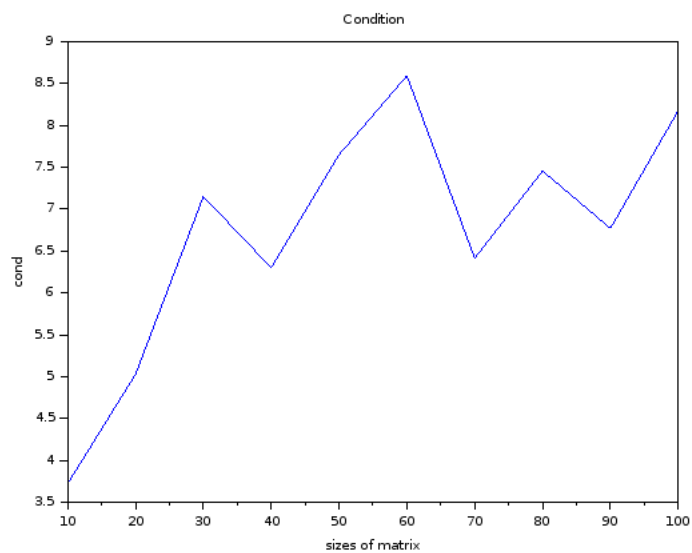


- Nous pouvons apercevoir que le temps suit une complexité cubique, ce qui est normal car nous avons déroulé les 3 boucles et donc nous n'avons fait appel à aucun noyau de calcul optimisé.

Erreur relative avant :



Conditionnement :



Nous remarquons :

- l'erreur relative avant est proportionnelle au conditionnement (normale car elle est dépendante de celle-ci)

- pour des matrices **100 x 100** ou moins, l'erreur est respectables étant donné qu'elle est inférieure à la précision machine ( $10^{-16}$ )

## TD 2

### Exercice 1

#### Algorithme

---

**Algorithm 1:** Détermine la matrice compact LU

---

**Data:**  $A \in \mathbb{R}^{n \times n}$

**Result:**  $LU \in \mathbb{R}^{n \times n}$

$LU(1, 1) = A(1, 1);$

**for**  $i = 2 : n$  **do**

    // Compute  $d_i$ ;

$LU(i, i) = A(i, i) - \frac{A(i-1, i)A(i, i-1)}{LU(i-1, i-1)};$

    // Compute  $c_i$ ;

$LU(i-1, i) = A(i-1, i);$

    // Compute  $e_i$ ;

$LU(i, i-1) = \frac{A(i, i-1)}{LU(i-1, i-1)};$

**end**

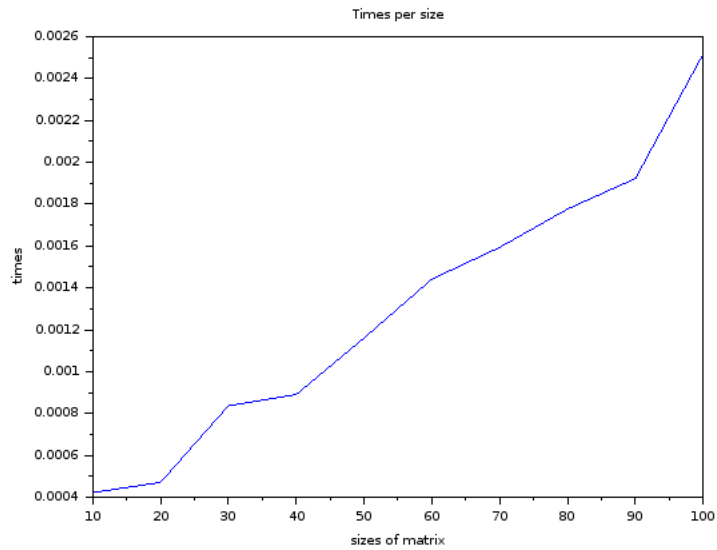
---

#### Complexité de l'algorithme

- en temps :  $3n$
- en espace :  $2n^2$

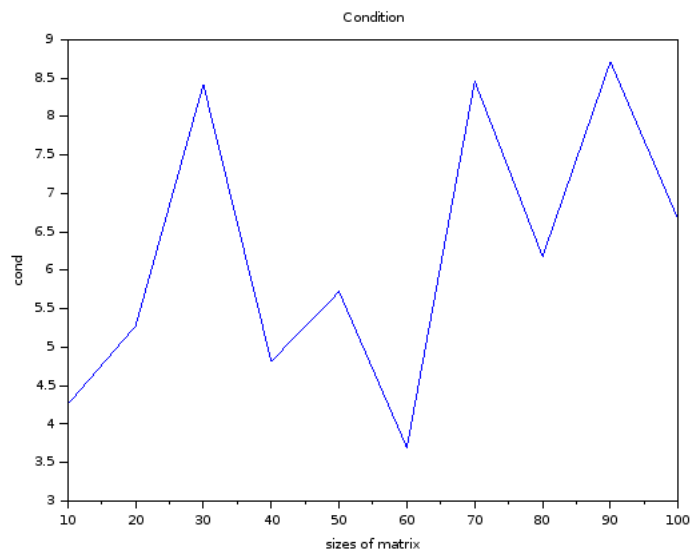
#### Graphiques

Temps :



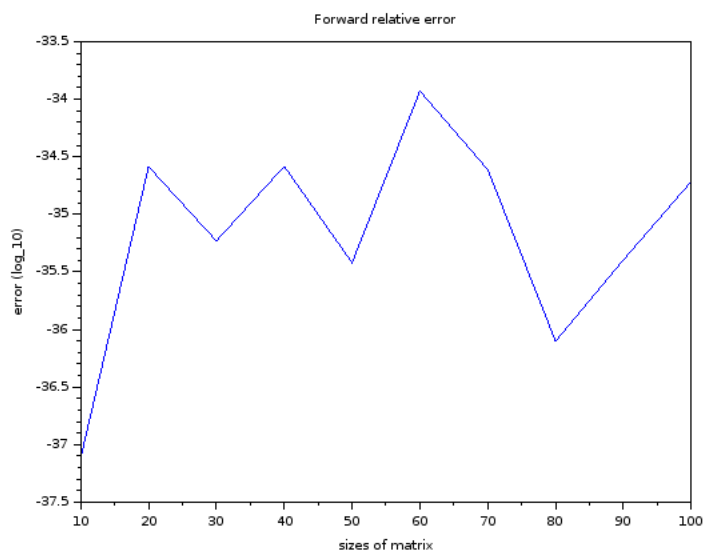
- Nous pouvons apercevoir que le temps est plutôt linéaire.

Conditionnement :



Le conditionnement reste bon pour des tailles de matrice entre 10 et 100 étant donné qu'il ne dépasse pas  $10^{10}$ , ce qui rendrait l'erreur pour un double une erreur pour un float.

### Erreur relative avant :



L'erreur relative arrière est respectable étant donné qu'elle est de l'ordre de  $10^{-34}$  pour des matrices de taille **100 x 100** ce qui donne des bornes pour l'erreur vers  $10^{-24}$  (étant donné que le conditionnement est de  $\approx 10^{10}$ ). Et cette borne est également respectable comparé à la précision machine de  $10^{-16}$ .

## Annexe

Dépôt github : [https://github.com/Sholde/CN/tree/master/partie\\_2](https://github.com/Sholde/CN/tree/master/partie_2)