# Compte-rendu programmation

Sofiane DJERBI
Aksel PERRIGAULT

October 29, 2021

## 1 Conception des algorithmes

Nous avons opté pour une modélisation des polynomes sous forme de structure.
Chaque opération retourne un pointeur, et ne mute pas coefficients des polynômes de base, afin d'éviter les effets de bord.
Il y a un test unitaire de comparaison avec l'algorithme naïf, afin de s'assurer du bon fonctionnement de chaque algorithme. Ainsi qu'une fonction de benchmark et un wrapper en python.

## 2 Comparaison des algorithmes

Nous plottons ici la performance de chaque algorithme en fonction de son degré.

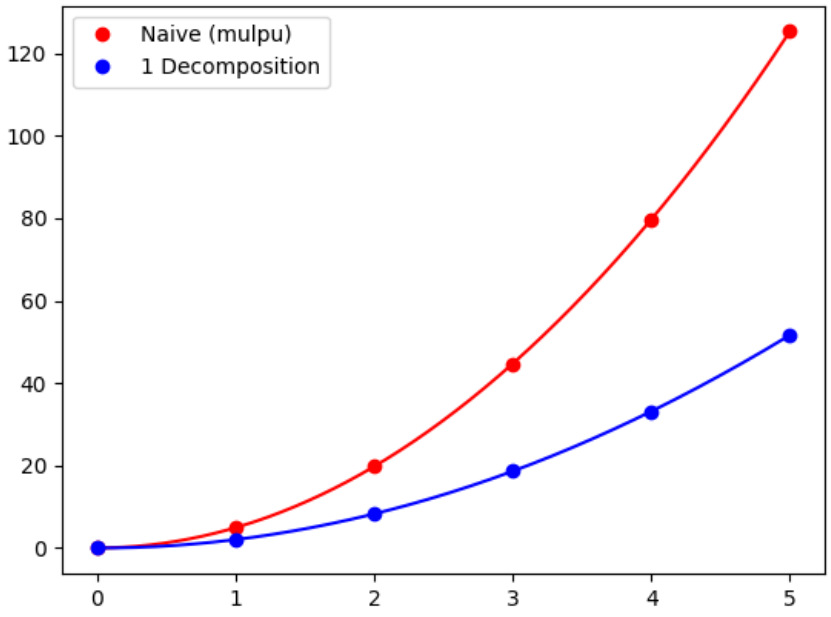


Figure 1: Comparaison des performances.
X: Degré du polynome (1 = degré 100 000)
Y: Temps en secondes

On remarque qu'avec une decomposition, on obtient de bien meilleurs résultats. En effet, comme la complexité naïve est de $O(n^2)$ (avec n le degré du polynome). Il est intéressant, même pour des poynômes de bas degré, de se ramener à un problème en $O((\frac{n}{2})^2)$.

# 3 Pistes d'amélioration

Nous avons eu un problème avec l'algorithme récursif de Karatsuba.
En effet, trop d'appels récursifs appelant malloc et free causent un gros problème de mémoire et de performance.
Il pourrait être envisageable d'implémenter des opérations "in-place" sur les polynômes.
Par ailleurs, comme l'algorithme de Karatsuba produit de bons résultats sur des polynômes de haut degré, il est intéressant de mettre un threshold élevé afin d'éviter d'utiliser Karatsuba sur des polynômes de trop petit degré, ce qui pourrait s'avérer moins efficace.