Projet d'informatique 2

Bregeon Pierre / Schobert Néo

2 avril 2021

Table des matières

1 Introduction

Le projet se concentre sur la recherche des racines d'un polynôme unitaire. On s'interesse ici à la méthode de Durand-Kerner.

Cette méthode commence par limiter le rayon de recherche de ces racines en constatant que si λ est une racine de P (P est le polynôme concerné), alors $|\lambda| \leq 1 + \max_{0 \leq i \leq p-1} |a_i|$.

On se concentre alors pour la suite du projet sur les complexes répondant à la condition citée ci-dessus.

Pour la suite, on applique simplement l'algorithme de Durand-Kerner telle que décrite dans l'énoncé.

1. On initialise $deg(P)$ nombres complexes distincts qui ne sont pas racine de notre polynôme
2. On construit alors $deg(P)$ suites par récurrence comme définies dans l'énoncé.
3. On distingue deux cas :
   — Si la méthode de Durand-Kerner converge, on fait un calcul de limite. Les limites de ces suites tendent chacune vers une racine de P distincte. (Non prouvé mais admis ici)
   — Sinon, on ne peut rien en déduire. (On sait tout de même qu'il existe au moins une racine si $deg(P) \geq 1$ (Théorème d'Alembert))

Il nous faut aussi représenter notre polynôme P de la manière la plus adaptée possible. Nous avons donc choisi la structure du tableau. Ce tableau contiendra tous les coefficients du polynôme P dans l'ordre des degrés décroissants **exprimés en notation complexe (exemple : $1 = 1 + 0j$)** [1] (sauf le premier qui vaut 1 car P est unitaire). La structure du tableau a été choisi car il s'agit d'une structure statique et avec une indexation bien définie et faite pour. Les listes auraient pu convenir mais elles n'ont pas été retenues car même si en python, elles ont une indexation adaptée, leur aspect dynamique nous est inutile.

2 Limitation du rayon de recherche

Comme dit antérieurement [2], si λ est une racine de P (P est le polynôme concerné), alors :

$$|\lambda| \leq 1 + \max_{0 \leq i \leq p-1} |a_i|$$

. (avec $\{a_i, i \in [\![1, p-1]\!]\}$ l'ensemble des coefficients de P)

2.1 Algorithme

Le but ici est de limiter notre recherche.

Sachant que l'on recherche les racines d'un polynôme et qu'elles se situent toutes dans le disque $D = \{z \in \mathbb{C} / |z| \leq 1 + \max_{0 \leq i \leq p-1} |a_i|\}$, nous allons déterminer la valeur de $1 + \max_{0 \leq i \leq p-1} |a_i|$.

La fonction calcul_R(tab) du fichier .py permet alors de faire cela.

2.2 Complexité

La fonction calcul_R fait :
— deux affectations
— une boucle de longueur deg(P)
— une condition et une affectation
— une opération dans le return

1. On les mettra en notation complexe pour éviter les problèmes liés aux conversions complexe / float
2. cf Introduction

La complexité est alors au pire (en comptant 1 pour une affectation un calcul ou un test) :

$$c = 2 + deg(P) * 2 + 1 = 3 + 2 * deg(P) = \mathcal{O}(deg(P))$$

3 Algorithme de Durand-Kerner

Après un court raffinage, nous avons constaté que l'algorithme de Durand-Kerner doit être séparé en 3 fonctions :

— Un première qui permettra de calculer $P(x_n^k)$ appelée ici P_el dans le fichier .py
— Une seconde qui permettra de générer un complexe aléatoire (fonction facultative)
— La troisième qui fera le reste de l'algorithme

3.1 Algorithme

Notre fonction prend deux arguments : le premier est le tableau représentant le polynôme, le second est un argument de précision, qui permet à l'utilisateur de choisir le degré de précision des racines en fonction de la complexité. On rappelle de plus que la complexité de la fonction croit avec le degré du polynôme entré. (Il ne faut donc pas choisir un epsilon trop important)

3.2 Complexité

On comptera 1 pour chaque affectation ou opération.
La première fonction (P_el) fait :
— deux affectation
— une boucle de longueur $deg(P)$ dans laquelle deux opérations et une affectation est faite
.

La complexité associée est alors :

$$c = 2 + deg(P) * 3 = \mathcal{O}(deg(P))$$

La seconde fonction (random_complex_de_D)) fait seulement 3 opérations ($\mathcal{O}(1)$)
La troisième fonction (suite$_Durand_Kerner1)fait$:
trois affectations et une copie de tableau de taille $deg(P)$ [3]

boucle de longueur $deg(P)$ dans laquelle on a un while à condition quasiment toujours fausse (la terminaison n'est pas assurée mais la probabilité fait que l'on peut considérer sa complexité en $\mathcal{O}(1)$)

Une deuxième boucle (qui fera le coeur de notre complexité) dans laquelle on a :

4 Utilisation et visualisation de la méthode de Durand-Kerner

5 Dépassements

6 Conclusion

3. la copie de tableau est en $\mathcal{O}(deg(P))$