Calculs de PGCD de polynômes connus inexactement

Olivier Bouillot, L.I.G.M., équipe de combinatoire et calcul formel

27 septembre 2024

1 Motivation

Souvent, une simulation physique se termine en déterminant numériquement les zéros d'une expression approchant la réalité. Malheureusement, la plupart des algorithmes de recherche numérique de zéros se comportent mal en présence de zéros multiples ; c'est le cas de la méthode de Newton et de sa généralisation, la méthode de Durand-Kerner permettant de trouver simultanément tous les zéros d'un polynôme.

Dans le cas simple des polynômes, il est tentant de trouver un nouveau polynôme ayant les même zéros que ceux recherchés, mais que ceux-ci soient tous simples ! Théoriquement, cela se fait en calculant un *plus grand diviseur commun* (PGCD) au polynôme étudié et à sa dérivée.

Malheureusement, le calcul d'un plus grand diviseur commun à deux polynômes à coefficients réels est un exemple où des difficultés imprévues par la théorie s'invitent en pratique en raison de la manipulation des nombres flottants... : des erreurs d'arrondis s'accumulent et se propagent jusqu'à produire un résultat faux. Un exemple classique d'erreurs d'arrondis est donné par le code Python suivant :

```
>>> 0.1 + 0.1 + 0.1 == 0.3
False
```

Calculer un PGCD en arithmétique des flottants revient donc à relever le défi de comprendre comment se créent des erreurs d'arrondis, comment elles se propagent et comment on peut les maîtriser.

Malgré ces difficultés, l'objectif du projet est de développer **un algorithme permettant de réaliser le calcul d'un plus grand diviseur commun à deux polynômes connus de manière imparfaite**. Une telle boite noire serait particulièrement utile dans de nombreux domaines scientifiques (ingénierie, robotique, vision par ordinateur, restauration d'images, théorie du contrôle, sans même parler des applications en mathématiques ou en informatique) !

2 Plus grand diviseur commun à deux polynômes - Exemples

La notion de plus grand diviseur commun à deux polynômes est une notion identique à celle de plus grand diviseur commun à deux nombres entiers : si P et Q sont deux polynômes, G est leur PGCD lorsqu'il existe deux polynômes $\widetilde{P}$ et $\widetilde{Q}$ tel que :

(1) $P = G \cdot \widetilde{P}$;
(2) $Q = G \cdot \widetilde{Q}$;
(3) G est un polynôme de degré maximum parmi tous les polynômes qui vérifient les points (1) et (2).

Lorsque les polynômes P et Q sont connus qu'imparfaitement, on souhaite trouver un PGCD de deux polynômes $\widetilde{P}$ et $\widetilde{Q}$ respectivement *"très proche"* de P et Q.

3 Un problème instable...

En théorie, pour calculer un PGCD, il suffit d'appliquer l'algorithme d'Euclide : la question semble donc facile. Dans le cas des polynômes, c'est en fait beaucoup plus complexe que la théorie ne le prévoit, car les coefficients des polynômes ne sont, la plupart du temps, connu qu'inexactement$\cdots$

Illustrons cela en imaginant que nous travaillons à deux décimales près.

Considérons les polynômes $A(X) = X + \sqrt{2}$ et $B(X) = (X + \sqrt{2})^2$. Bien sur, leur PGCD est $X + \sqrt{2}$.

• Si $\sqrt{2} \approx 1.41$ et $2\sqrt{2} \approx 2.82$, alors $A(X)$ est représenté en machine par le polynôme $\widetilde{A}(X) = X + 1.41$ et $B(X)$ par $\widetilde{B}(X) = X^2 + 2.82 + 2$. L'algorithme d'Euclide permet alors de trouver qu'un PGCD de $\widetilde{A}(X)$ et $\widetilde{B}(X)$ est $X + 1.41$: on retrouve bien un polynôme qui approxime $X + \sqrt{2}$.
• Néanmoins, si $\sqrt{2} \approx 1.41$ et $2\sqrt{2} \approx 2.83$, alors $\widetilde{A}(X) = X + 1.41$ et $\widetilde{B}(X) = X^2 + 2.83 + 2$. On trouve cette fois qu'un PGCD de $X + 1.41$ et $X^2 + 2.83X + 2$ est 1 ! Ce qui signifierait que $A(X)$ et $B(X)$ n'auraient pas de facteur commun... Le travail ne simplification n'aurait tout bonnement pas lieu !

L'exemple précédent illustre bien le fait que le calcul numérique d'un plus grand diviseur commun entre polynômes est un problème **instable**, donc difficile : *de petites perturbations sur les données peuvent engendrer une grande perturbation sur le résultat final*, en détruisant par exemple le degré exact du PGCD calculé !

En conséquence, lorsque les coefficients des polynômes sont tronquées à une certaine précision, comme cela est fait avec l'arithmétique des flottants, il est extrêmement probable que l'algorithme d'Euclide se termine avec un dernier reste non nul qui soit égale à 1, *i.e.* échouer à trouver un facteur commun aux deux polynômes, alors qu'il peut y en avoir...

4 Objectif du projet

L'objectif du projet est de :

1. construire un corpus d'exemples de polynômes dont on connaît leur PGCD à l'avance, comprenant notamment des polynômes produits aléatoirement ;
2. tester ce corpus sur différents algorithmes disponibles dans la littérature afin de vérifier leur efficacité et détecter leurs faiblesses ;
3. produire une bibliothèque de calcul compatible avec Python d'un ou plusieurs algorithmes de calcul du PGCD de deux polynômes connu imparfaitement après avoir sélectionné un ou plusieurs algorithmes dans la littérature, voir même proposé un nouvel algorithme ;
4. analyser les résultat.

On pourra notamment tester l'algorithme d'Euclide (cf. [3]) et ses modifications utilisant les *Polynomial Sequences of Remainders* (cf. [1]) ainsi que d'autres algorithmes reposant sur des manipulations de matrices comme le *résultant* et la *réduction en valeurs singulières* (cf. [2] et [4]).

L'étudiant choisissant ce sujet développerait de solides connaissances en optimisation de code et en arithmétique des flottants. D'un point de vue scientifique, un tel projet permettait d'apporter à la communauté un état de l'art, avec en perspective, une publication envisageable sous forme d'un *survey*.

Références

[1] M. Hegland *Numerical methods for computing the greatest common divisor of univariate polynomials using floating point arithmetic*, *in* Proceddings of the 2018 Computational Techniques and Applications Conference, CTAC 2018, Anziam J. 60, p.127-139.

[2] R. M. Corless, P. M. Gianni, B. M. Trager, S. M. Watt. *The singular value decomposition for polynomial systems. in* Proceedings of the 1995 International Symposium on Symbolic and Algebraic Computation, ISSAC 1995, p. 195–207, ACM, 1995.

[3] Donald E. Knuth *Seminumerical Algorithms*. The Art of Computer Programming. Vol. 2 (Third ed.). Reading, Massachusetts : Addison-Wesley., p. 425-426.

[4] Z. Zeng *The numerical greatest common divisor of univariate polynomials. in* Randomization, relaxation and complexity in polynomial equation solving, vol. 556 of Contemp. Math., p. 187–217, Amer. Math. Soc., 2011.