Projet post-doctoral: évaluation rapide de polynômes

F. Vigneron, Laboratoire de Mathématiques de Reims, URCA

Année 2024-2025

Public visé : titulaire d'un doctorat en *Mathématiques* (analyse, systèmes dynamiques,...) ou en *Informatique* (algorithmique,...). Les bons dossiers présentant un projet personnel indépendant seront aussi considérés avec attention, en particulier autour de l'analyse des équations aux dérivées partielles, des systèmes dynamiques et de la modélisation en physique mathématique, puisque ces thématiques peuvent toujours s'intégrer au sein de notre laboratoire.

Candidature : Déposez vos dossiers de candidature sur

https://www.dropbox.com/request/1fYQZJOZT93k4uKHxydQ

Le dossier doit comporter 1 SEUL fichier au format PDF, contenant votre CV et tout document que vous jugez pertinent à l'appui de votre candidature. La concision est une vertu.

1 Etat de l'art concernant l'évaluation des polynômes

Un polynôme est une expression de la forme

$$P(z) = a_0 + a_1 z + a_2 z^2 + \ldots + a_d z^d = \sum_{k=0}^{d} a_k z^k.$$

Naivement, cette expression comporte $1 + 2 + \ldots + d \simeq d^2$ multiplications. Le seul algorithme d'évaluation efficace mono-point est celui de Hörner qui consiste à réécrire, par exemple

$$1 + 3z - 5z^2 + 6z^3 = 1 + z(3 + z(-5 + 6z)).$$

Ce jeu d'écriture permet, en général, d'évaluer $P(z)$ en seulement $O(d)$ opérations arithmétiques et un théorème [5, 6] garantit l'optimalité de cette complexité. Des algorithmes multi-points existent comme le célèbre algorithme FFT qui permet d'abaisser le coût par point grâce à un certain parallélisme [2] mais dans ce cas, les contraintes géométriques sont extrêmes (les points d'évaluation doivent être tous sur un même cercle). D'autres algorithmes peuvent s'affranchir des contraintes géométriques mais au prix d'un coût mémoire $O(d^2 \log d)$ qui devient prohibitif en haut degré. L'état de l'art vient d'être complètement modifié par l'algorithme FPE [1, 7] qui exploite la précision finie de tout calcul numérique et arrive ainsi à identifier une sous-expression polynomiale $Q_p(z)$ telle que $P(z) \approx Q_p(z)$ au sens de l'arithmétique en précision finie. Cette expression ne comporte, en moyenne, que

$$1.904\sqrt{d(p + 4 + \log_2 d)}$$

termes (où p est la précision). L'évaluation de cette expression peut donc se faire essentiellement $\sqrt{d}$ fois plus rapidement qu'avec la méthode de Hörner. L'implémentation numérique [3] de l'algorithme FPE présente des benchmarks impressionnants avec des gains allant de $\times 2$ à $\times 10$ pour des polynômes de degré $d = 1000$ et des gains encore plus importants lorsque le degré augmente.

2 Objectifs scientifiques & méthodes envisagés

Les expressions polynomiales à plusieurs variables, c'est à dire

$$P(z_1, z_2, \ldots, z_n) = \sum_{k=0}^{d} \left(\sum_{\alpha_1+\ldots+\alpha_n=k} \alpha_{j_1,\ldots,j_n} z_1^{\alpha_1} z_2^{\alpha_2} \ldots z_n^{\alpha_n} \right)$$

présentent un challenge substantiel car le nombre de termes d'une telle expression est de l'ordre de $O(d^n)$. Un polynôme de degré 10 en 6 variables comporte environ un million de termes donc équivaut à un polynôme de degré 10^6 en 1 seule variable. D'autre part, l'algorithme de Hörner présente, dans le cas multi-varié, plusieurs stratégies de factorisation possible.

L'objectif du projet est de généraliser l'algorithme d'évaluation rapide FPE au cas des expressions polynomiales à n variables. Le second objectif est d'implémenter effectivement le nouvel algorithme pour s'assurer qu'il n'y a pas de constante cachée et que le gain théorique est aussi visible en pratique.

On pourra s'appuyer sur les méthodes d'analyse convexe développées dans [1]. En ce qui concerne la complexité de l'algorithme FPE, celle-ci peut se ramener à un problème géométrique à savoir calculer la largeur moyenne d'une calotte concave d'épaisseur δ. En 1D, on peut montrer que cette épaisseur est de l'ordre de $\sqrt{\delta}$ (figure 1). Dans le cas multi-dimensionnel, certains aspects techniques de la preuve ne sont plus valables et une nouvelle preuve doit être élaborée. En ce qui concerne la certification du résultat dans l'arithmétique de précision finie, les outils restent valables, mais la multiplicité des termes d'un même degré peut nécessiter de prendre des marges de sécurité substantiellement plus grandes qu'en 1D (le correcteur $\log_2 d$ ci-dessus). Ces marges sont à identifier.

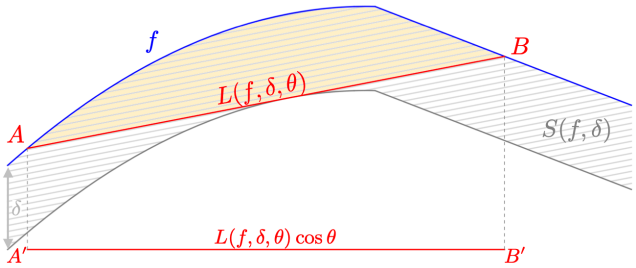

Fig. 1 - Calotte concave d'épaisseur δ en 1D.

3 Programme de travail

Les résultats théoriques [1, 7] et l'implémentation existante [3] constituent une fondation solide. Des simulations numériques préliminaires suggèrent qu'il est raisonnable de s'attendre à un coût moyen $O(d^{n/2})$ pour la généralisation à n variables de l'algorithme FPE.

Le travail se fera selon deux axes. Le premier axe, théorique, concerne le développement de l'algorithme FPE généralisé. Il comporte les étapes suivantes:

- Formulation d'un algorithme FPE_n pour les polynômes à n variables.

- Preuve de la complexité en $O(d^{n/2}(p + n \log_2 d))$ par géométrie convexe. Estimation précise de la constante et construction d'exemples qui en assurent l'optimalité.
- Preuve de l'exactitude du résultat dans l'arithmétique de précision finie.
- Identifier les algorithmes existants (et leurs implémentations éventuelles) dans le cas multivarié. Comparer avec nos résultats.

Le second axe concerne l'implémentation numérique avec les étapes suivantes:

- Portage du code FPE existant pour améliorer sa visibilité (exemple: librairie Python).
- Réflexion sur l'implémentation multi-variée (format des données, calcul de l'envelope concave, stratégie d'évaluation du polynôme résiduel).
- Implémentation effective de notre algorithme dans le cas multi-varié. Test d'exactitude puis benchmarks de vitesse mono-CPU (nécessite d'utiliser Roméo pour paralléliser les benchmarks et obtenir un échantillon statistiquement significatif). Documentation du code.
- Implémentation d'une librairie pour calcul multi-précision sur GPU. Adaptation du code (existant et multi-varié). Test d'exactitude & benchmark de vitesse du code parallèle GPU.
- Applications à l'interpolation polynomiale, à la recherche de zéros, etc.

Références

[1] R. Anton, N. Mihalache, F. Vigneron. *Fast evaluation of polynomials* (2022) [Hal-03820369]
[2] Cooley, James W.; Tukey, John W. *An algorithm for the machine calculation of complex Fourier series.* Math. Comput. 19, 90 (1965)
[3] N. Mihalache, F. Vigneron. *FPE : a Fast Polynomial Evaluator* [GitHub]
[4] N. Mihalache, F. Vigneron. *How to split a tera-polynomial* (2021) [JCAD 2021]
[5] A.M. Ostrowski. *On two problems in abstract algebra connected with Hörner's rule.* Studies in Mathematics and Mechanics, (1954)
[6] V. Ja. Pan. *On means of calculating values of polynomials.* Russian Math. Surveys, 21 (1966)
[7] F. Vigneron. Présentation de FPE au *Journées Calcul Données* (2023) [JCAD 2023]