Atelier 9

Objectifs de formation
— Multiplier des polynômes.
— Interpoler un polynôme.
— Comparer des vitesses d'exécution d'algorithmes.

Dans tout l'atelier, un polynôme f à coefficients dans $\mathbb{R}$ sera représenté par un tableau de flottants double précision `double* coeff_f` de sorte que le coefficient de degré i de f soit stocké dans `coeff_f[i]`.

1. : **30 min**
   — Implémenter la fonction d'évaluation par la méthode de Horner du premier ordre du polynôme f en un point a dont la signature est
   ```
   void eval_Horner_1(double a,
                      double* coeff_f,
                      int n,
                      double* f_a);
   ```
   Le pointeur `f_a` permet de stocker le résultat.
   — Implémenter la fonction d'évaluation par la méthode de Horner du second ordre du polynôme f en un point a et son opposé dont la signature est
   ```
   void eval_Horner_2(double a,
                      double* coeff_f,
                      int n,
                      double* f_a,
                      double* f_minus_a);
   ```
   — Comparer les temps de calculs pour des polynômes de degré 2^i avec i allant de 0 à 16 et des coefficients tirés aléatoirement dans $[-1, 1]$.
2. : **60 min**
   — Implémenter une fonction
   ```
   void mulpol (double* coeff_f, int n, double* coeff_g, int p,
                double* coeff_h, int* q)
   ```
   calculant le produit h, de degré q, de deux polynômes f et g, de degrés respectifs n et p.
3. : **90 min donnant lieu à un rendu**
   — En utilisant le code de résolution de système linéaire déjà existant, écrire une fonction

```
void interpol_linalg (double* a,
                      double* f_a,
                      int n,
                      double* coeff_f)
```

calculant le polynôme f de degré minimal inférieur ou égal à n satisfaisant $f(a_i) = y_i$ pour tout a_i dans `a` et y_i dans `f_a`.

— Écrire une fonction

```
void interpol_Lagrange (double* a,
                        double* f_a,
                        int n,
                        double* coeff_f)
```

résolvant le même problème mais en utilisant la formule du polynôme interpolateur de Lagrange.

— Comparer les temps de calculs pour $n = 2^i$ avec i allant de 0 à 16 et des entrées dans `a` et `f_a` tirées aléatoirement dans $[-1, 1]$.