1 Cas en 1D :

Dans le cas 1D, on a eu l'idée d'utiliser les polynomes de Vandermonde, avec le problème d'ajouter un point pour ne pas avoir comme solution le polynome nul

$$\begin{pmatrix} 1 & x_0 & \dots & x_0^n \\ \vdots & & \vdots & \\ 1 & x_{n-1} & \dots & x_{n-1}^n \\ 1 & x_b & \dots & x_b^{n-1} \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_{n-1} \\ a_n \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}$$

avec $x_b = \frac{1}{n} \sum_{i=0}^{n-1} x_i$
Cela permet d'avoir un système de Vandermonde résolvable en inversant la matrice.

2 Cas en 2D :

Les polynômes que nous avons trouvé à la fin de la séance sont :

2.1 Prosition de Gabriel :

$$P_1(x,y) = \prod_{i=0}^{n-1} ((x-x_i)^2 + (y-yi)^2)$$

2.2 Proposition de Mariem :

$$P_2(x,y) = \prod_{i=0}^{n-1} ((x-x_i) + i(y-yi))$$

Si l'on passe au module P_2, on retrouve l'expression de P_1, il est plus rapide de trouver les coefficients de P_2 mais le calcul de $P_2(x,y)$ après ça demande plus de temps (polynome dans$\mathbb{C}$).

2.3 Proposition matricielle :

Début d'idée

$$P_3(x,y) = \sum_{i,j=0}^{n} a_{ij} x^i y^j$$

$$\begin{pmatrix} 1 & x_0 & y_0 & x_0 y_0 & \dots & x_0^n y_0^n \\ \vdots & & & & \vdots & \\ 1 & x_{n-1} & y_{n-1} & x_{n-1} y_{n-1} & \dots & x_{n-1}^n y_{n-1}^n \\ 1 & x_b & y_b & x_b y_b & \dots & x_b^n y_b^n \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_{n-1} \\ a_n \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}$$

Où x_b et y_b sont définis de manière similaire à x_b au début du document. Dans cette méthode il nous faut inverser la matrice, contrairement au cas en 1D avec de vraies matrices de Vandermonde cela n'est pas théoriquement assuré.

L'implémentation

Soit $(p_i)_{i \in chips}$ La fonction principale est set polynome xpy numpy matrix. Elle prend en entrée coords, un array de coordonnées où le polynôme à construire est nul.

Etape 1 : Construire matrice avec u

$$\big(A_n =\big) \tag{1}$$