

**Reed Solomon**

---

**Travail pratique de mathématiques**

**Groupe 10**

Nicolas Albanesi

Darius Briquet

Abivarman Kandiah

Jonas Stirnemann

16/12/2021

Version: 1.0

## Contents

<b>1</b>	<b>Objectif du Travail Pratique</b>	<b>2</b>
<b>2</b>	<b>Message reçu</b>	<b>2</b>
<b>3</b>	<b>Blocs</b>	<b>3</b>
3.1	Manipulation de Polynomes . . . . .	3
3.2	Algorithme d'Euclide étendu . . . . .	4
3.3	Inverse multiplicatif . . . . .	4
3.4	Polynôme de Lagrange . . . . .	4
3.5	Reed Solomon . . . . .	5
3.6	Listes . . . . .	6

## 1

## Objectif du Travail Pratique

Dans ce travail pratique, il nous est demandé de produire un code capable de manipuler des polynômes, exécuter l'algorithme d'euclide étendu ainsi que de calculer un polynôme de Lagrange. Le but étant de rassembler ces blocs afin de décoder un message contenant des erreurs, à l'aide du code de Reed Solomon.

## 2

## Message reçu

### GROUPE 10

Les données du problème:

Le nombre premier : **229**

La longueur du message de base : **25**

Le nombre de points ajoutés : **18**

le nombre maximal d'erreurs : **9**

qui sont situées avant l'indice: **20**

**la liste reçue :** [115, 117, 101, 118, 122, 116, 57, 108, 32, 224, 62, 116, 115, 140, 32, 108, 153, 83, 169, 117, 108, 112, 32, 110, 101, 55, 96, 61, 160, 218, 228, 156, 224, 203, 12, 75, 180, 23, 220, 211, 137, 139, 206]

Dans ce message, nous avons alors un nombre premier *229* nous permettant de travailler dans un corps congruent à ce nombre premier.

Dans la liste reçue, chaque point correspond à un caractère en unicode. Le message de base faisant 25 caractères, 18 caractères ont été rajoutés dont maximum 9 sont faux. L'objectif étant de retrouver le message de base en corrigeant les points erronés.

## 3

## Blocs

## 3.1

## Manipulation de Polynomes

Afin de permettre une manipulation simple des polynômes, nous avons créé une classe **Polynome** permettant d'initialiser un objet polynôme avec de simples coefficients correspondants aux degré respectifs aux index.

## CODE 1: Exemple de création de polynôme

```
1 polynome_test = polynome( [2, 3, 4, 5, 6] )
```

Ce code va alors créer un polynôme de degré 4 de la forme suivante:

$$6x^4 + 5x^3 + 4x^2 + 3x^1 + 2 \quad (1)$$

Nous pouvons également effectuer des opérations simples avec ces objets polynôme

## CODE 2: Addition de polynomes

```
1 # P1 et P2 sont existants : P1[0, 1, 2] et P2[2, 3, 4]
2 P3 = P1.add(P2)
```

$$P3 = 6x^2 + 4x^1 + 2 \quad (2)$$

## CODE 3: Multiplication de polynomes

```
1 # P1 et P2 sont existants : P1[0, 1, 2] et P2[2, 3, 4]
2 P3 = P1.mul(P2)
```

$$P3 = 8x^4 + 10x^3 + 7x^2 + 2x^1 + 0 \quad (3)$$

Il existe également une fonction de classe permettant d'afficher un polynôme dans la console.

## CODE 4: Affichage de polynomes

```
1 # P1 et P2 sont existants : P1[0, 1, 2] et P2[2, 3, 4]
2 # P3 = P1 * P2
3 P3.show()
```

## 3.2

## Algorithme d'Euclide étendu

L'algorithme d'Euclide étendu nous permet de calculer les coefficients de Bézout ( $u$  et  $v$ ) de deux entiers  $a$  et  $b$  tel que

$$\text{PGCD}(a, b) = a * u + b * v \quad (4)$$

## CODE 5: Calcul du PGCD Étendu

```
1 pgcd, u, v = pgcd_etendu(a, b)
```

Ces coefficients seront par la suite utilisés pour calculer l'inverse multiplicatif d'un entier  $a$  dans les entiers modulo un nombre premier  $p$ , voir sous-section 3.3 - Inverse multiplicatif

## 3.3

## Inverse multiplicatif

Afin de simplifier et accélérer nos calculs et de travailler avec des nombres d'ordre correct, nous travaillons dans un corps congruent un nombre Premier. Nous avons alors produit une fonction permettant de trouver l'inverse multiplicatif  $\text{mod } p$  d'un nombre  $a$ , ce qui veut dire qu'au lieu de diviser un nombre par une approximation en float, il est possible de le multiplier par l'inverse modulaire. Pour trouver cet inverse, il suffit de trouver les coefficients de Bézout de ce nombre  $a$  avec le modulo  $p$ . Le premier coefficient donné correspond alors à l'inverse multiplicatif du nombre  $a \text{ mod } p$ .

$$a * u \pmod{p} = 1 \quad (5)$$

## 3.4

## Polynôme de Lagrange

Le polynôme de Lagrange est un polynôme  $L(x)$  de degré  $n$  qui est calculé en fonction d'une liste de  $n + 1$  points.

$$l_i(x) = \prod_{k=0, k \neq i}^n \frac{x - x_k}{x_i - x_k} \quad (6)$$

$$L(x) = \sum_{i=0}^n y_i \cdot l_i(x) \quad (7)$$

## CODE 6: Calcul du polynôme de Lagrange

```
1 # Exemple d'une liste : [(0, 1), (1, 2), (2, 3), (3, 4), (4, 7)]
2 polynome = lagrange_compute(list_points)
```

$$210x^4 + 114x^3 + 20x^2 + 115x^1 + 1 \quad (8)$$

## 3.5

## Reed Solomon

Voici une liste des étapes afin de corriger les erreurs d'un message reçu grâce au code de Reed-Solomon

Les données du problème:

Le nombre premier : **229**

La longueur du message de base : **25**

Le nombre de points ajoutés : **18**

Le nombre total de points : **43**

Le nombre maximal d'erreurs : **9**

qui sont situées avant l'indice: **20**

### 1 - Génération de listes de points

Une liste contenant des listes de **25** Points (x, y) est générée contenant alors différentes combinaisons de points toutes avec les **23** derniers points identiques (car nous sommes assurés qu'ils ne contiennent pas d'erreur).

### 2 - Générer le Polynôme de Lagrange

Une fois la liste de points générée, le polynome de Lagrange correspondant est calculé.

### 3 - Compter les erreurs

On va alors tester avec ce polynôme de Lagrange, le nombre de points qui ne correspondent pas aux points reçus. Le polynome testé avec lequel on trouve le moins d'erreurs est le bon polynôme. (Un peu plus rapide dans le code car dans notre cas, il est possible de s'arrêter quand on trouve 9 erreur ou moins)

### 4 - Décoder le message initial

Une fois le bon Polynôme trouvé, il nous suffit de régénérer les 25 premiers points de la liste en les évaluant avec le polynôme. Puis nous pouvons convertir ces points en caractère ASCII afin de reformer la phrase initiale.

### 5 - Récupérer le message final

Message reçu (avec des erreurs) : suevzt9l à>ts lS@ulp ne

Message décodé (sans les erreurs) : srevne'l à tse li sulp ne

'En plus il est à l'envers'

$$\begin{aligned}
 &99x^{24} + 131x^{23} + 38x^{22} + 185x^{21} + 165x^{20} \\
 &189x^{19} + 219x^{18} + 220x^{17} + 25x^{16} + 104x^{15} + \\
 &211x^{14} + 132x^{13} + 5x^{12} + 167x^{11} + 202x^{10} + \\
 &220x^9 + 190x^8 + 28x^7 + 45x^6 + 140x^5 + 155x^4 + 2
 \end{aligned} \tag{9}$$

## 3.6

## Listes

## CODE

1	Exemple de création de polynôme . . . . .	3
2	Addition de polynomes . . . . .	3
3	Multiplicaiton de polynomes . . . . .	3
4	Affichage de polynomes . . . . .	3
5	Calcule du PGCD Étendu . . . . .	4
6	Calcul du polynôme de Lagrange . . . . .	4