

# Calcul de petites racines d'un polynôme de bas degré à coefficients dans $\mathbb{Z}/n\mathbb{Z}$

Soutenance du projet informatique

NGUYEN Thi Thu Quyen    CHARTOUNY Maya

01/01/2021-30/04/2021

# Contenus

- 1 Introduction
- 2 Réseau euclidien et calcul de sa base réduite par algorithme LLL
- 3 Calcul de petites racines par la nouvelle méthode
- 4 FLINT et implémentation de la nouvelle méthode

# Contenus

- 1 Introduction
- 2 Réseau euclidien et calcul de sa base réduite par algorithme LLL
- 3 Calcul de petites racines par la nouvelle méthode
- 4 FLINT et implémentation de la nouvelle méthode

# Introduction

## Chiffrement RSA

Soit  $x$  le message clair, une méthode naïve de chiffrement RSA de  $x$  est

$$c \equiv x^e \pmod{N}$$

où  $(e, N)$  est la clé publique.

Est-ce qu'on peut retrouver  $x$  efficacement?

# Introduction

En 1996, Coppersmith a répondu que **OUI** pour le cas où  $x$  est "petit".

## Théorème (Coppersmith)

Soit  $p(X) \in \mathbb{Z}[X]$  un polynôme unitaire de bas degré  $k$ . Soit  $N$  un entier. Alors on peut trouver efficacement les entiers  $x$  tels que  $|x| \leq N^{1/k}$  et

$$(1) \quad p(x) = x^k + a_{k-1}x^{k-1} + \dots + a_0 \equiv 0 \pmod{N}$$

La méthode de Coppersmith et la **nouvelle méthode** trouvent  $x$  en temps polynomial par ramener (1) au problème de factoriser un polynôme dans  $\mathbb{Z}[X]$  en construisant le bon **réseau euclidien**.

# Contenus

- 1 Introduction
- 2 Réseau euclidien et calcul de sa base réduite par algorithme LLL
- 3 Calcul de petites racines par la nouvelle méthode
- 4 FLINT et implémentation de la nouvelle méthode

# Réseau euclidien

## Définition

Soient  $n \in \mathbb{N}$ ,  $f_1, \dots, f_n \in \mathbb{R}^n$  où  $f_i = (f_{i1}, \dots, f_{in})$ . Alors,

$$L = \sum_{i=1}^n \mathbb{Z} f_i = \left\{ \sum_{i=1}^n r_i f_i \mid r_1, \dots, r_n \in \mathbb{Z} \right\}$$

est le **réseau euclidien** généré par  $f_1, \dots, f_n$ .

Si de plus, les  $f_i$  sont linéairement indépendants, alors ils forment une **base** de  $L$ .

La **norme** de  $L$  est  $|L| = |\det(f_{ij})_{i,j=1}^n| \in \mathbb{R}$

# Base orthogonale de Gram-Schmidt

## Définition

Soit  $L$  un réseau euclidien et soit  $f = (f_1, \dots, f_n)$  où  $f_i \in \mathbb{R}^n$  pour tout  $i$ , une base quelconque du réseau  $L$ . On définit  $f_i^*$  par la formule suivante:

$$f_i^* = f_i - \sum_{j=1}^i \mu_{ij} f_j^*$$

où  $\mu_{i,j} = \frac{\langle f_i, f_j^* \rangle}{\|f_j^*\|^2}$  pour  $1 \leq j \leq i \leq n$ .

Alors,  $f^* = (f_1^*, \dots, f_n^*)$  est appelée la **base orthogonale de Gram-Schmidt**, notée **GSO**, de la base  $f = (f_1, \dots, f_n)$ .



# Base orthogonale de Gram-Schmidt

## Remarque

On a

$$F = \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix} = \begin{pmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \mu_{n1} & \cdots & 1 \end{pmatrix} \begin{pmatrix} f_1^* \\ \vdots \\ f_n^* \end{pmatrix} = MF^*$$

Alors

$$(2) \quad |L| = |\det(f_1, \dots, f_n)| = |\det(f_1^*, \dots, f_n^*)| = \prod_{i=1}^n \|f_i^*\|$$

# Base réduite de réseau euclidien

## Définition

Soient  $f_1, \dots, f_n \in \mathbb{R}^n$  linéairement indépendants et  $(f_1^*, \dots, f_n^*)$  la base GSO correspondante.

Alors  $(f_1, \dots, f_n)$  est une **base réduite** du réseau euclidien  $L$  si:

$$(3) \quad \|f_i^*\|^2 \leq 2 \|f_{i+1}^*\|^2 \text{ pour } 1 \leq i < n$$

## Remarque

Si  $(f_1, \dots, f_n)$  est une **base réduite** du réseau euclidien  $L$  et  $(f_1^*, \dots, f_n^*)$  sa base GSO, alors  $f_1 = f_1^*$  est un vecteur court du réseau euclidien  $L$ . De plus, on a

$$(4) \quad \|f_1\| \leq 2^{(n-1)/4} |L|^{1/n}$$

$$(5) \quad \|f_n^*\| \geq 2^{-(n-1)/4} |L|^{1/n}$$

*Preuve:* (4)  $\|f_1\|^{2n} = \|f_1^*\|^{2n} \stackrel{(3)}{\leq} 2^{\sum_{i=0}^{n-1} i} \prod_{i=1}^n \|f_i^*\| \stackrel{(2)}{=} 2^{\frac{n(n-1)}{2}} |L|$

(5) même preuve ■

# Algorithme LLL

---

## Algorithme 1: Réduction de la base

---

**Input:** Vecteurs lignes linéairement indépendants  $f_1, \dots, f_n \in \mathbb{Z}^n$

**Output:** Une base réduite  $(g_1, \dots, g_n)$  du réseau  $L = \sum_{1 \leq i \leq n} \mathbb{Z} f_i \subset \mathbb{Z}^n$

---

```

1 for  $i \leftarrow 2$  to  $n$  do  $g_i \leftarrow f_i$ ;
2 Calculer la GSO  $G^*, M \in \mathbb{Q}^{(n \times n)}$ ,  $i \leftarrow 2$ 
3 while  $i \leq n$  do
4   for  $j = i - 1, i - 2, \dots, 1$  do  $g_i \leftarrow g_i - \lfloor \mu_{ij} \rfloor g_j$ ;    /* pour mettre à jour la
      GSO */
5   ;
6   if  $j > 1$  and  $\|g_{i-1}^*\|^2 > 2\|g_i^*\|^2$  then  $i \leftarrow i - 1$ ;    /* échanger  $g_i$  et  $g_{i-1}$  et
      mettre à jour la GSO */
7   ;
8   else  $i \leftarrow i - 1$ ;
9 return  $g_1, \dots, g_n$ 

```

---

Figure: LLL

# Contenus

- 1 Introduction
- 2 Réseau euclidien et calcul de sa base réduite par algorithme LLL
- 3 Calcul de petites racines par la nouvelle méthode
- 4 FLINT et implémentation de la nouvelle méthode

$$p(x) = x^k + a_{k-1}x^{k-1} + \cdots + a_0 \equiv 0 \pmod{N}$$

## Théorème (Coppersmith)

Soit  $p(X) \in \mathbb{Z}[X]$  un polynôme unitaire de bas degré  $k$ . Soit  $N$  un entier. Alors on peut trouver efficacement les entiers  $x$  tels que  $|x| \leq N^{1/k}$  et

$$(6) \quad p(x) = x^k + a_{k-1}x^{k-1} + \cdots + a_0 \equiv 0 \pmod{N}$$

*Preuve: Si  $N$  est premier?* Pour  $x \in \mathbb{Z}/N\mathbb{Z}$  la solution de (6) on a  $p(x) = 0$  et  $x^N - x = 0$ . Alors  $(X - x)$  est un **facteur commun de  $p(X)$  et  $X^N - X$  dans  $\mathbb{Z}/N\mathbb{Z}[X]$** . Trouvons  $X - x$  en temps polynomial par l'algorithme de Berlekamp qui cherche le  $\text{pgcd}(p(X), X^N - X)$ .

*Si  $N$  est composé? Si la factorisation  $N = \prod p_i^{\alpha_i}$  est connue*, trouvons  $x \equiv x_i \pmod{p_i}$  où  $p(x_i) \equiv 0 \pmod{p_i}$  par théorème du reste chinois.

**Sinon:** l'approche par la nouvelle méthode.

## Nouvelle méthode

Soit  $h \geq 2$  un entier,  $\tilde{X}$  un entier qui sert comme borne pour la solution  $x$ . Soit  $M_{hk \times hk} = (m_{i,j})_{0 \leq i,j < hk} = (e_{i,j} \tilde{X}^j)_{0 \leq i,j < hk}$  une matrice triangulaire inférieure, i.e  $e_{i,j} = 0$  pour  $j > i$ , les  $e_{i,j}$  pour  $j \leq i$  satisfont

$$q_i(X) = q_{u,v}(X) := N^{h-1-v} X^u (p(X))^v = \sum_{0 \leq j \leq i} e_{i,j} X^j \quad \text{pour} \quad \begin{cases} v = \lfloor i/k \rfloor \\ u = i - kv \end{cases}$$

Ou,  $q_i(x) = e_{i,i} X^i + \cdots + e_{i,0}$ , où  $e_{i,i} = N^{h-1-\lfloor i/k \rfloor}$ . De plus, on a

$$q_i(x) = q_{u,v}(x) := N^{h-1-v} x^u (p(x))^v \equiv 0 \pmod{N^{h-1}}$$

### Remarque

Construisons  $r(X) = \sum c_i q_i(X)$  et tel que  $r(x) = 0$ .

# Nouvelle méthode

Alors  $M$  s'écrit

$$\begin{pmatrix} e_{0,0} & 0 & 0 & \cdots & 0 \\ e_{1,0}\tilde{X} & e_{1,1}\tilde{X} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ e_{hk-1,0}\tilde{X}^{hk-1} & e_{hk-1,1}\tilde{X}^{hk-1} & e_{hk-1,2}\tilde{X}^{hk-1} & \cdots & e_{hk-1,hk-1}\tilde{X}^{hk-1} \end{pmatrix}$$

Par ailleurs,  $e_{i,i}\tilde{X}^i = N^{h-1-\nu}\tilde{X}^i$ , donc

$$\begin{aligned} \det M &= \prod_{i=0}^{hk-1} e_{i,i}\tilde{X}^i = N^{\sum_{i=0}^{hk-1} (h-1-\lfloor \frac{i}{k} \rfloor)} \tilde{X}^{\sum_{i=0}^{hk-1} i} \\ (7) \quad &= N^{hk(h-1)/2} \tilde{X}^{hk(hk-1)/2} \end{aligned}$$



# Nouvelle méthode

Utilisons l'algorithme LLL pour trouver la base réduite du réseau euclidien engendré par des lignes de  $M$  dont le vecteur court  $b_1 = cM$ . Alors

$$\begin{aligned}
 \|b_1\|_2 &\geq \frac{1}{\sqrt{hk}} \|b_1\|_1 = \frac{1}{\sqrt{hk}} \|cM\|_1 \quad (\text{l'inégalité de Cauchy-Schwarz}) \\
 &\geq \frac{1}{\sqrt{hk}} (|\sum_{i=0}^{hk-1} c_i m_{i,0}| + |\sum_{i=0}^{hk-1} c_i m_{i,1}| + \cdots + |\sum_{i=0}^{hk-1} c_i m_{i,hk-1}|) \\
 &\geq \frac{1}{\sqrt{hk}} (|\sum_{i=1}^{hk} c_i e_{i,0}| + |(\sum_{i=0}^{hk-1} c_i e_{i,1})\tilde{X}| + \cdots + |(\sum_{i=0}^{hk-1} c_i m_{i,hk-1})\tilde{X}^{hk-1}|) \\
 &\geq \frac{1}{\sqrt{hk}} (|\sum_{i=0}^{hk-1} c_i e_{i,0}| + |(\sum_{i=0}^{hk-1} c_i e_{i,1})X| + \cdots + |(\sum_{i=0}^{hk-1} c_i m_{i,hk-1})X^{hk-1}|)
 \end{aligned}$$

(8)

# Nouvelle méthode

## Remarque

Posons

$$\begin{aligned}
 r(X) &= \left( \sum_{i=0}^{hk-1} c_i e_{i,0} \right) + \left( \sum_{i=0}^{hk-1} c_i e_{i,1} \right) X + \cdots + \left( \sum_{i=0}^{hk-1} c_i m_{i,hk-1} \right) X^{hk-1} \in \mathbb{Z}[X] \\
 &= c_0 e_{0,0} + c_1 \sum_{j=0}^1 e_{1,j} X^j + \cdots + c_{hk-1} \sum_{j=0}^{hk-1} e_{hk-1,j} X^j \\
 &= c_0 q_0(X) + c_1 q_1(X) + \cdots + c_{hk-1} q_{hk-1}(X) \\
 \text{Alors } r(x) &= \sum_{i=0}^{hk-1} q_i(x) \equiv 0 \pmod{N^{h-1}}.
 \end{aligned}$$

# Nouvelle méthode

Alors pour la solution  $x$  dont

$$(9) \quad |x| \leq \tilde{X} = \tilde{X}(h) = \lfloor (2^{-1/2}(hk)^{-1/(hk-1)} N^{(h-1)/(hk-1)}) \rfloor$$

et par l'inégalité (4) on a

$$\begin{aligned} |r(x)| &\stackrel{(8)}{\leq} \sqrt{hk} \|b_1\|_2 \\ &\stackrel{(4)}{\leq} 2^{(hk-1)/4} |\det M|^{1/(hk)} \sqrt{hk} \stackrel{(7)}{=} 2^{(hk-1)/4} N^{(h-1)/2} \tilde{X}^{(hk-1)/2} \sqrt{hk} \\ &\stackrel{(9)}{\leq} N^{h-1} \end{aligned}$$

et  $r(x) \equiv 0 \pmod{N^{h-1}}$ , donc  $r(x) = 0$ . On trouvera des solutions de (6) en cherchant des racines entières de  $r(X)$  par des algorithmes de factorisation dans  $\mathbb{Z}[X]$  en temps polynomial. ■

# Nouvelle méthode

## Remarque

- On peut trouver des racines entières de  $r(X)$  efficacement par l'algorithme de Zassenhaus et van Hoeij. Il reste à vérifier si qu'elle sont solutions de (6).
- D'après (9) on a

$$\tilde{X}(h) \xrightarrow{h \rightarrow \infty} N^{1/k}$$

Donc il faut préciser que les méthodes concernant cette borne ne trouvent que des solutions du (6) jusqu'à  $O(N^{1/k})$ , autrement dit des petites racines.

Example:  $p(X) = X^2 + 14X + 19 \equiv 0 \pmod{35}$

Prenons  $h = 3, \tilde{X} = 2$  alors  $q_i(X) = q_{u,v}(X) := 35^{3-1-v} X^u (p(X))^v$  avec  $k = 2, 0 \leq i < hk = 6, v = \lfloor i/k \rfloor$  et  $u = i - kv$

$$q_0 = q_{0,0} = 1225$$

$$q_1 = q_{0,1} = 1225X$$

$$q_2 = q_{1,0} = 665 + 490X + 35X^2$$

$$q_3 = q_{1,1} = 665X + 490X^2 + 35X^3$$

$$q_4 = q_{2,0} = 361 + 532X + 234X^2 + 28X^3 + X^4$$

$$q_5 = q_{2,1} = 361X + 532X^2 + 234X^3 + 28X^4 + X^5$$

Example:  $p(X) = X^2 + 14X + 19 \equiv 0 \pmod{35}$

$$\text{Alors } M = \begin{pmatrix} 1225 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1225\tilde{X} & 0 & 0 & 0 & 0 \\ 665 & 490\tilde{X} & 35\tilde{X}^2 & 0 & 0 & 0 \\ 0 & 665\tilde{X} & 490\tilde{X}^2 & 35\tilde{X}^3 & 0 & 0 \\ 361 & 532\tilde{X} & 234\tilde{X}^2 & 28\tilde{X}^3 & \tilde{X}^4 & 0 \\ 0 & 361\tilde{X} & 532\tilde{X}^2 & 234\tilde{X}^3 & 28\tilde{X}^4 & \tilde{X}^5 \end{pmatrix}$$

$$M \xrightarrow{LLL} B = \begin{pmatrix} 3 & 8\tilde{X} & -24\tilde{X}^2 & -8\tilde{X}^3 & -1\tilde{X}^4 & 2\tilde{X}^5 \\ 49 & 50\tilde{X} & 0 & 20\tilde{X}^3 & 0 & 2\tilde{X}^5 \\ 115 & -83\tilde{X} & 4\tilde{X}^2 & 13\tilde{X}^3 & 6\tilde{X}^4 & 2\tilde{X}^5 \\ 61 & 16\tilde{X} & 37\tilde{X}^2 & -16\tilde{X}^3 & 3\tilde{X}^4 & 4\tilde{X}^5 \\ 21 & -37\tilde{X} & -14\tilde{X}^2 & 2\tilde{X}^3 & 14\tilde{X}^4 & -4\tilde{X}^5 \\ -201 & 4\tilde{X} & 33\tilde{X}^2 & -4\tilde{X}^3 & -3\tilde{X}^4 & \tilde{X}^5 \end{pmatrix}$$

Alors  $r(X) = 3 + 8X - 24X^2 - 8X^3 - 1X^4 + 2X^5$ ,  
on a  $r(3) = 0 \equiv p(3) \pmod{35}$ .

Example:  $p(X) = X^2 + 14X + 19 \equiv 0 \pmod{35}$

### Remarque

On constate que la borne  $\tilde{X} = \tilde{X}(h) = 2 < 3 = x$  qui est la solution. Ce phénomène se passe souvent, on dit que la borne  $\tilde{X}(h)$  est pessimiste.

# Contenus

- 1 Introduction
- 2 Réseau euclidien et calcul de sa base réduite par algorithme LLL
- 3 Calcul de petites racines par la nouvelle méthode
- 4 FLINT et implémentation de la nouvelle méthode



# Librairie FLINT

Flint est une librairie dans C qui permet de faire de la théorie des nombres. Actuellement, elle est maintenue par William Hart, son site officiel est <http://www.flintlib.org/>.

- **Installation**

FLINT est compatible avec *Windows, Linux, macOS*. Pour utilisateurs *macOS, Linux*, on peut l'installer simplement via **Homebrew**. Autre méthodes d'installation se trouvent dans **README, INSTALL** du package FLINT téléchargé depuis son site officiel.

- **Utilisation globale**

FLINT est généralement utilisée pour l'arithmétique avec: *des nombres, des polynômes, des séries et des matrices*, sur de nombreux anneaux.

# Implémentation de la nouvelle méthode

FLINT fournit des **structures polynomiales, matricielles** (dans `fmpz_poly.h`) et **des algorithmes de factorisation en temps polynomial dans  $\mathbb{Z}[X]$**  (dans `fmpz_poly_factor.h`). Elle possède aussi **l'algorithme LLL** (dans `fmpz_lll.h`) qui sort la base réduite depuis une base de réseau euclidien quelconque. Les mentionnés contribuent à l'implémentation de la nouvelle méthode.

# Implémentation de la nouvelle méthode

La compilation se fait par le Makefile.

```
CC=gcc
CFLAGS=-Wall

GMP_INCLUDE=-I/Users/quyennguyen/Programs/libs/gmp/6.2.1/include
GMP_LIBS=-L/Users/quyennguyen/Programs/libs/gmp/6.2.1/lib -lgmp
FLINT_INCLUDE=-I/usr/local/Cellar/flint/2.7.0/include/flint
FLINT_LIBS=-L/usr/local/Cellar/flint/2.7.0/lib -lflint

test7 : test7.c
    ${CC} ${CFLAGS} -o test7 test7.c ${FLINT_INCLUDE} ${FLINT_LIBS}
```

Figure: Makefile

# Implémentation de la nouvelle méthode

Le syntaxe du programme compilé est

```
./test7 a b c d
```

- $N \leftarrow a$
- $h \leftarrow b$
- Si  $c = 0$  alors  $\tilde{X} \leftarrow \tilde{X}(h)$  comme calculé dans (9) sinon  $\tilde{X} \leftarrow c$
- Si  $d = \text{default}$  (Figure: 3) alors  $p(X) \leftarrow X^2 + 14X + 19$  sinon  $d = \text{test}$  (Figure: 4) alors le programme lit  $p(X)$  sous le syntaxe

$$k+1 \quad a_0 \ a_1 \ \dots \ a_k$$

depuis le clavier.

# Implémentation de la nouvelle méthode

```
[quyennguyen@Quyens-MacBook-Air flint % ./test7 35 3 0 default
hk h k N 6 3 2 35
X 2
p: x^2+14*x+19
q0: 1225
q1: 1225*x
q2: 35*x^2+490*x+665
q3: 35*x^3+490*x^2+665*x
q4: x^4+28*x^3+234*x^2+532*x+361
q5: x^5+28*x^4+234*x^3+532*x^2+361*x
M: 6 6 1225 0 0 0 0 0 2450 0 0 0 0 665 980 140 0 0 0 0 1330 1960 280 0 0 361
1064 936 224 16 0 0 722 2128 1872 448 32
M: 6 6 3 16 -96 -64 -16 64 49 100 0 160 0 64 -128 50 -20 80 160 32 -201 8 132
-32 -48 32 -83 -142 52 8 32 160 61 32 148 -128 48 128
r: 2*x^5-x^4-8*x^3-24*x^2+8*x+3
3
```

Figure:  $c = 0$ ,  $d = \text{default}$ ,  $x = 3$

# Implémentation de la nouvelle méthode

```
[mayachartouny@Maya Nouv % ./test7 35 3 3 test
3 19 14 1
hk h k N 6 3 2 35
X 3
p: x^2+14*x+19
1225
1225*x
35*x^2+490*x+665
35*x^3+490*x^2+665*x
x^4+28*x^3+234*x^2+532*x+361
x^5+28*x^4+234*x^3+532*x^2+361*x
M: 6 6 1225 0 0 0 0 0 0 3675 0 0 0 0 665 1470 315 0 0 0 0 1995 4410 945 0 0 36
1 1596 2106 756 81 0 0 1083 4788 6318 2268 243
M: 6 6 -201 12 297 -108 -243 243 -3 -24 216 216 81 -486 -243 324 -81 -81 324 -
243 193 411 216 81 81 243 387 -63 297 -378 -243 0 -66 399 -36 189 -486 0
r: x^5-3*x^4-4*x^3+33*x^2+4*x-201
3
```

Figure:  $c = 3$ ,  $d = \text{test}$ ,  $x = 3$

# Implémentation de la nouvelle méthode

```
quyennnguyen@Quyens-MacBook-Air flint % ./test7 8619 6 8 default
hk h k N 12 6 2 8619
X 8
p: x^2+14*x+19
q0: 0
q1: 0
q2: 5518582289439921*x^2+77260152052158894*x+104853063499358499
q3: 5518582289439921*x^3+77260152052158894*x^2+104853063499358499*x
q4: 640281040659*x^4+17927869138452*x^3+149825763514206*x^2+340629513630588*x+231141455677899
q5: 640281040659*x^5+17927869138452*x^4+149825763514206*x^3+340629513630588*x^2+231141455677899*x
q6: 74287161*x^6+3120060762*x^5+47915218845*x^4+322406278740*x^3+910389158055*x^2+1126341935082*x+509535637299
q7: 74287161*x^7+3120060762*x^6+47915218845*x^5+322406278740*x^4+910389158055*x^3+1126341935082*x^2+509535637299*x
q8: 8619*x^8+482664*x^7+10790988*x^6+122113992*x^5+734942130*x^4+2320165848*x^3+3895546668*x^2+3310592376*x+1123236699
q9: 8619*x^9+482664*x^8+10790988*x^7+122113992*x^6+734942130*x^5+2320165848*x^4+3895546668*x^3+3310592376*x^2+1123236699*x
q10: x^10+70*x^9+2055*x^8+32760*x^7+307410*x^6+1732164*x^5+5840790*x^4+11826360*x^3+14095245*x^2+9122470*x+2476099
q11: x^11+70*x^10+2055*x^9+32760*x^8+307410*x^7+1732164*x^6+5840790*x^5+11826360*x^4+14095245*x^3+9122470*x^2+2476099*x
zsh: segmentation fault ./test7 8619 6 8 default
```

Figure: Faux calculs de  $q_0$  et  $q_1$  et segmentation fault

# Implémentation de la nouvelle méthode

## Remarque

- L'affichage ne sert qu'à vérifier le bon fonctionnement du programme.
- Comme on voulait profiter des fonctions de FLINT, quelques détails du programme ne sont pas optimisés. Par ailleurs, la limitation des tests nous empêche de conclure des résultats statistiques fiables sur la performance du programme.
- Le code du programme est disponible sur [github](#). Le code explique rapidement les structures et calculs utilisés.