Rapport UE 2I013

Remontée de Hensel et reconstruction rationnelle pour la factorisation et la résolution d'équations polynomiales

Serge Arthur
Berrebi Nathane

Encadrant : M. Jérémy Berthomieu

Janvier 2018 — Septembre 2018

1 Introduction

L'objectif de notre projet est d'étudier la résolution de problèmes qui font intervenir de grands nombres entiers. Pour cela nous avons vu deux méthodes permettant la résolution de systèmes linéaires et la recherche de racines de polynômes, qui sont le théorème Chinois et la Remontée de Hensel. Pour modéliser ces grands entiers nous avons utilisé la bibliothèque GMP. Celle ci possède ses propres variables (mpz, mpf..) qui peuvent dépasser les limites standard du langage. Il n'y a pas de limite pratique à la précision sauf celles impliquées par la mémoire disponible dans la machine sur laquelle GMP s'exécute. Pour notre projet la plupart des calculs effectués sont fais dans $\mathbb{Z}/p\mathbb{Z}$, donc la plupart des variables utilisées sont des entiers. C'est pour cela que nous avons utilisé le type mpz_t qui représente des entiers dans la bibliothèque GMP.

2 Théorème chinois

2.1 Relation de Bézout

Soit a et b deux entiers, d'après la relation de Bézout, il existe u et v deux entiers tel que :

$$au + bv = pgcd(a, b)$$

2.2 Algorithme d'Euclide étendue

Tout d'abord nous avons implémenté l'algorithme d'Euclide étendue qui prend deux entiers a et b en argument et qui renvoie la relation de Bézout associée à ces deux entiers.

2.3 Énoncé du Théorème Chinois

Soient $n_1, n_2, ..., n_k$ des entiers naturels non nuls deux à deux premiers entre eux. Pour tout entier $a_1, a_2, ..., a_k$, tel que $a_i < n_i$ et $\forall i \in \{ 1, ..., k \}$ il existe un unique entier $x \in \mathbb{N}$ modulo $n_1 \times n_2 \times ... \times n_k$ qui vérifie :

$$x \ mod(n_1 \times n_2 \times ... \times n_k) = a_1 \ mod(n_1) = a_2 \ mod(n_2) = ... = a_k \ mod(n_k).$$

2.4 Preuve du théorème Chinois

Montrons par récurrence que pour tout $k \in \mathbb{N}^*$, on peut résoudre un système de k équations (avec des modulos premiers entre eux), par la méthode du théorème Chinois.

Initialisation: Pour k=1: trivial, l'équation est renvoyée.
pour k=2, on a comme système :

$$x = a_1 \ mod(m_1)$$
$$x = a_2 \ mod(m_2)$$

m_1 et m_2 sont des nombres premiers distincts, donc d'après la relation de Bézout il existe u et v qui appartiennent à $\mathbb{Z}$ tel que

$$um_1 + vm_2 = 1$$

$$\Rightarrow um_1(a_1 - a_2) + (a_1 - a_2)vm_2 = (a_1 - a_2)$$

$$\Rightarrow vm_2(a_1 - a_2) + a_2 = a_1 - um_1(a_1 - a_2) = c$$

donc on a bien $c = a_1\ mod(m_1)$ et $c = a_2\ mod(m_2)$ d'où $x = c\ mod(p_1 \times \mathrm{p}_1)$,avec $c \in \mathbb{Z}$)

Hérédité: soit $n \in \mathbb{Z}$ tel que l'on puisse résoudre un système à n équations à l'aide du théorème chinois, montrons que l'on peut résoudre un système à $n+1$ equations.

soit

$$x = a_1 mod(m_1)$$

$$\vdots$$

$$x = a_{n+1} mod(m_{n+1})$$

D'après la propriété de récurrence $\exists c \in \mathbb{Z}$ tel que $c = a_1\ mod(m_1)$, ... , $c = a_n\ mod(m_n)$. On a ainsi :

$$x = c$$

$mod(m_1 \times m_2 \times ... \times m_n)$

$$x = a_{n+1} mod(m_{n+1})$$

Or m_{n+1} est premier avec $m_1 \times m_2 \times ... \times m_n$ donc on peut appliquer la même méthode que dans l'initialisation pour trouver un c' tel que $x = c' mod(m_1 \times m_2 \times ... \times m_n \times m_{n+1})$

2.5 Exemple d'application du théorème Chinois

$x = 1\ mod(2)$
$x = 0\ mod(3)$
$x = 3\ mod(5)$
$x = 6\ mod(7)$
$x = 2\ mod(11)$

On applique le théorème Chinois par couple d'équations. On détaille les calculs du premier couple:

La relation de Bézout entre 3 et 2 donne $u = 1$ et $v = -1$.
Donc $3 + (-1) \times 2 = 1$.
Ce qui implique $c = 3$.
Donc on obtient de ce couple l'équation : $x = 3\ mod(6)$

$x = 3\ mod(5)$
$x = 6\ mod(7)$
$x = 2\ mod(11)$

Puis on obtient :
$x = 3\ mod(30)$
$x = 6\ mod(7)$
$x = 2\ mod(11$

Puis on obtient :
$x = 153\ mod(210)$
$x = 2\ mod(11)$

Enfin on a $x = 783\ mod(2310)$

3 Résolution de systèmes linéaires par le théorème Chinois

3.1 Problématique

Soit $A \in \mathbb{GL}_n(\mathbb{R})$ et $b \in \mathbb{R}^n$. On cherche $x \in \mathbb{R}^n$ tel que $Ax = b$. Pour cela, on cherche $x_i \in \mathbb{N}^n$ tel que $x = x_i\ mod(m_i)$ avec m_i un nombre premier $\forall\ i \in \{1, ..., k\}$ avec k un nombre entier arbitraire positif, puis on utilise le théorème chinois pour obtenir un c tel que $x = c\ mod(m_1 \times ... \times \mathrm{m}_k)$.

3.2 Système de Gauss modulaire

Comme nous l'avons expliqué dans la problématique, les modulos des équations sont des nombres premiers. On peut donc effectuer un système de Gauss sur $\mathbb{Z}/m_i\mathbb{Z}$ avec $i \in \{1, ..., k\}$. En effet $\mathbb{Z}/m_i\mathbb{Z}$ est un corps. Nous pouvons donc appliquer toutes les opérations nécessaires pour la résolution de ce système (addition, multiplication, inversion modulaire).

3.3 Exemple de système de Gauss modulaire

On cherche $x \in \mathbb{R}^3, x = (x_{11}, x_{12}, x_{13})$ tel que $Ax = b$ modulo m.

$$A = \begin{pmatrix} 22 & 6 & 54 \\ 31 & 19 & 2 \\ 11 & 12 & 13 \end{pmatrix} \quad b = \begin{pmatrix} 16 \\ 25 \\ 54 \end{pmatrix} \quad m = 5$$

Méthode de Gauss modulaire :

$$Amod(m) = \begin{pmatrix} 2 & 1 & 4 \\ 1 & 4 & 2 \\ 1 & 2 & 3 \end{pmatrix} \quad bmod(m) = \begin{pmatrix} 1 \\ 0 \\ 4 \end{pmatrix}$$

$L2 \leftarrow L2 - 1\times\ 3\ \times$ L1 (3 est l'inverse de $2\ mod(5)$)
$L3 \leftarrow L3 - 1\times 3\times$L1

$$\rightarrow \begin{pmatrix} 2 & 1 & 4 \\ 0 & 1 & 0 \\ 0 & 4 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}$$

$L3 \leftarrow L3 - 4{\times}1{\times}\mathrm{L2}$

$$\rightarrow \begin{pmatrix} 2 & 1 & 4 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

On obtient une matrice triangulaire supérieure que l'on note A' et un vecteur b'. Résoudre le système $Ax = b$ revient à résoudre le système $A'x = b'$ par un algorithme de remontée.

Soit:

$x_{13} = 3\ mod(5)$
$x_{12} = 2\ mod(5)$
$x_{11} = 3{\times}(1\text{-}2\text{-}4 \times 3)\ mod(5)$ (3 est l'inverse de 2 modulo 5)
$x_{11} = 1\ mod(5)$

Donc $x = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$

En effet $A \times x = \begin{pmatrix} 1 \\ 0 \\ 4 \end{pmatrix}\ mod(5) = b\ mod(5)$

3.4 Rôle du théorème Chinois

En utilisant la méthode de Gauss modulaire avec k modulos premiers $(m_1, ..., m_k)$, on obtient k vecteurs de R^n tel que $\forall i \in \{1, .., k\}$ on a $Ax_i = b$ le tout $mod(m_i)$. On veut se ramener à un vecteur x tel que $Ax = b$ le tout $mod(m_1{\times}...{\times}\mathrm{m}_k)$.

Pour cela, nous appliquons le théorème Chinois. En effet, pour obtenir le j^{ieme} élément de x avec $j \in 1, ..., n$, on effectue la méthode du théorème Chinois vue précédemment avec le j^{ieme} élément de tous les x_i.

3.5 Exemple

Soit $x \in R^3$

$x = (1, 1, 1)\ mod(3)$

$x = (1, 2, 3)\ mod(5)$

$x = (1, 9, 9)\ mod(13)$

On applique le théorème Chinois :
$\Rightarrow x = (1, 22, 178)\ mod(195)$

car $x_1 = 1\ mod(3)$, $x_1 = 1\ mod(5)$, $x_1 = 1\ mod(13) \Rightarrow x_1 = 1\ mod(195)$
$x_2 = 1\ mod(3), x_2 = 2\ mod(5), x_2 = 9\ mod(13) \Rightarrow x_2 = 22\ mod(195)$
$x_3 = 1\ mod(3), x_3 = 3\ mod(5), x_3 = 9\ mod(13) \Rightarrow x_3 = 178\ mod(195)$

4 Remontée de Hensel

4.1 Énoncé de la Remontée de Hensel

Soit P un polynôme, p un nombre premier et $x_0 \in \mathbb{N}$ tel que $P(x_0) = 0\ mod(p)$ et $P'(x_0) \neq 0\ mod(p)$. Pour tout $k \in \mathbb{N}^*$, $P(x_k) = 0\ mod(p^{2^k})$ avec $x_k = x_{k-1} - \dfrac{P(x_{k-1})}{P'(x_{k-1})}\ mod(p^{2^k})$.

4.2 Démonstration

Soient $P \in \mathbb{Z}_n[X]$, p un nombre premier et x_0 une racine simple de P modulo p.
Montrons par récurrence : H(i) " Pour tout $i \in \mathbb{N}^*$ x_i est une racine de $P\ mod(p^{2^i})$".

Initialisation: $i = 1$,D'après la formule de Taylor en x_0 On a:

$P(x) = \Sigma_{k=0}^{n} \dfrac{(x - x_0)^k}{k!} \times \mathrm{P}^{(k)}(x_0)$
On pose $x_1 = x_0 - \dfrac{P(x_0)}{P'(x_0)}$ donc
$P(x_1) = \Sigma_{k=0}^{n} \dfrac{(x_1 - x_0)^k}{k!} \times \mathrm{P}^{(k)}(x_0)\ mod(p^2)$
$= P(x_0) + (x_1 - x_0) \times \mathrm{P'(x_0)} + (x_1 - x_0)^2 \times \mathrm{C\ mod(p^2)}$ (avec C un polynôme de degrés $n - 2$)
$= P(x_0) + (x_0 - \dfrac{P(x_0)}{P'(x_0)} - x_0) \times \mathrm{P'(x_0)} + (x_1 - x_0)^2 \times \mathrm{C\ mod(p^2)}$
$= P(x_0) - P(x_0) + (x_1 - x_0)^2 \times \mathrm{C\ mod(p^2)}$ Or $(x_1 - x_0)^2\ mod(p^2) = -(\dfrac{P(x_0)^2}{P'(x_0)^2})\ mod(p^2)$
$= 0\ mod(p^2)$ Car x_0 est une racine de $P\ mod(p)$.
Donc $P(x_0)^2 = 0\ mod(p^2)$.

D'où $P(x_1) = 0\ mod(p^2)$.

La propriété est vraie pour $i = 1$.

Hérédité : Soit $i - 1 \in N$ tel que H(i-1) soit vraie, montrons H(i).

D'après la formule de Taylor en x_{i-1} on a :
$P(x) = \Sigma_{k=0}^{n} \dfrac{(x - x_{i-1})^k}{k!} \times (P^{(k)}(x_{i-1}))$ donc

$P(x_i) = \Sigma_{k=0}^{n} \frac{(x_i - x_{i-1})^k}{k!} \times (P^{(k)}(x_{i-1}))\ mod(p^{2^i})$
$= P(x_{i-1}) + (x_i - x_{i-1}) \times P'(x_{i-1}) + (x_i - x_{i-1})^2 \times C\ mod(p^{2^i})$
$= P(x_{i-1}) + (x_{i-1} - \frac{P(x_{i-1})}{P'(x_{i-1})} - x_{i-1}) \times P'(x_{i-1} + (x_i - x_{i-1})^2 \times C\ mod(p^{2^i})$
$= P(x_{i-1}) - P(x_{i-1}) + (x_i - x_{i-1})^2 \times C\ mod(p^{2^i})$
Or $((x_i - x_{i-1})^2)\ mod(p^{2^i})$
$= -(\frac{P(x_{i-1})^2}{P'(x_{i-1})^2})\ mod(p^{2^i})$
$= 0\ mod(p^{2^i})$ Car x_{i-1} est une racine de $P\ mod(p^{2^{i-1}})$, d'après la propriété de récurrence donc $P(x_{i-1})^2 = 0\ mod(p^{2^i})$.

D'où $P(x_i) = 0\ mod(p^{2^i})$.
Donc la propriété est vraie $\forall i \in \mathbb{N}$.

4.3 Exemple

Soit le polynôme $P(x) = -5 \times x^3 + 3 \times x^2 + 5 \times x - 12$, cherchons une racine avec un modulo supérieur à 1000.

On prend $m = 3$, $P(0) = -12 = 0\ mod(3)$ donc on prend $x_0 = 0$.
$x_1 = x_0 - \frac{P(x_0)}{P'(x_0)}\ mod(9)$
$x_1 = \frac{3}{5}\ mod(9) = 3 \times 5^{-1}\ mod(9) = 3 \times 2\ mod(9) = 6\ mod(9)$
donc $x_1 = 6$ et on a bien $p(x_1) = -954 = 0\ mod(9)$
$x_2 = x_1 - \frac{P(x_1)}{P'(x_1)}\ mod(81)$
$x_2 = 6 - \frac{63}{13}\ mod(81)$
$x_2 = 6 - 63{\times}25 \mod(81)$
$x_2 = -30 mod(81) = 51\ mod(81)$
en effet, $P(51) = -655209 = 0\ mod(81)$ donc x_2 est bien une racine de P modulo 81.

$x_3 = x_2 - \frac{P(x_2)}{P'(x_2)}\ mod(6561)$

$x_3 = 51 - \frac{5670}{5899}\ mod(6561)$

$x_3 = 51 + 5670 \times 1001\ mod(6561)$

$x_3 = 456 mod(6561)$

On a ainsi trouvé une racine de P modulo 6561 (qui est un modulo supérieur à 1000).

5 Test

5.1 Comparaison des deux méthodes

Pour comparer l'efficacité des deux méthodes, nous tentons de résoudre le même problème avec celles-ci. Le problème en question est la recherche d'une racine modulaire d'un polynôme avec un modulo supérieur à une certaine limite.

Tout d'abord nous avons implémenté une fonction qui cherche une racine modulaire en parcourant tous les entiers entre 0 et le modulo et qui évalue le polynôme avec chaque entier jusqu'à que l'on trouve un multiple du modulo. Cette fonction sera utilisée dans les deux programmes.

Pour la remontée de Hensel, nous posons un nombre premier p, nous cherchons la racine x_0 du polynôme modulo p avec la fonction expliquée juste au-dessus. Puis, nous appliquons la remontée de Hensel jusqu'à obtenir un p^{2^n} supérieur à la limite imposée par l'utilisateur.

Pour le théorème Chinois, on fait le crible d'Eratostene de manière à obtenir la liste des nombres premiers inférieurs à 10^6.

Ensuite on utilise une fonction qui calcule *prod*, le produit de tous les nombres de cette liste jusqu'au moment ou celui-ci dépasse la limite souhaitée.

On cherche une racine pour chaque nombre premier utilisé pour construire *prod* en utilisant la même fonction que pour la remontée de Hensel. On obtient une liste de modulo associée à des valeurs, à laquelle on applique le théorème chinois pour obtenir une unique valeur modulo *prod*.

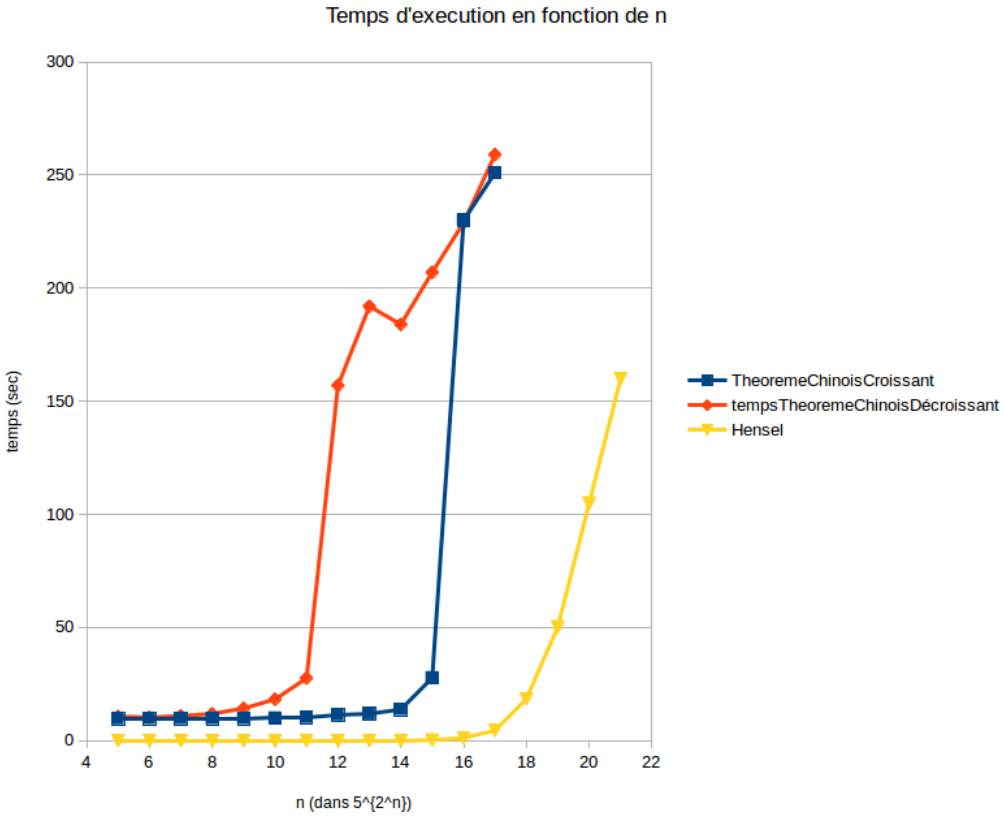
Voici les différents tests:

Le premier, intitulé Théorème Chinois Croissant utilise la méthode du théorème Chinois comme décrite précédemment et calcul *prod* (le produit des nombres premiers de la liste) en partant du début de la liste jusqu'à ce que *prod* soit supérieur à la limite.

Le second, intitulé Théorème Chinois Décroissant utilise la méthode du théorème Chinois comme décrite précédemment et calcul *prod* en partant du dernier nombre de la liste et en parcourant la liste de manière décroissante jusqu'à ce que *prod* soit supérieur a la limite.

Le troisième, intitulé Hensel, effectue la remontée de Hensel comme expliqué précédemment.

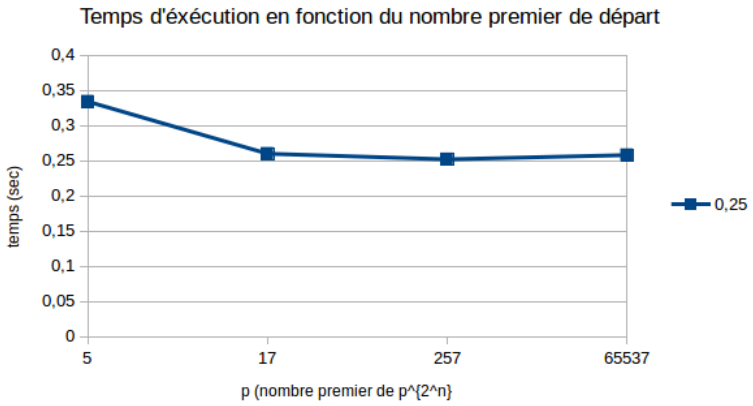
Premier graphique:



Analyse: On prend $p = 5$ le nombre premier. On fait varier n entre 5 et 18 puis pour chaque n on effectue un lancement de chaque test. On a donc L qui varie entre 5^{2^5} et on voit sans surprise que la méthode de la remontée de Hensel est extrêmement plus rapide, de l'ordre de 10^4 pour les n petits (Hensel le fait de manière quasi instantanée). Puis lorsqu'on dépasse n=17 l'exécution de Théorème Chinois Croissant et Théorème Chinois Décroissant n'est pas aboutie, le terminal affiche "processus arrêté" alors que Hensel le fait en moins de 25 secondes(Hensel s'exécute sur un temps raisonnable jusqu'à n=21).

En ce qui concerne les deux tests qui utilisent la méthode du théorème Chinois, on remarque une différence qui avantage l'algorithme qui utilise l'ordre croissant (sûrement à cause de la fonction qui cherche une racine modulaire qui prend plus de temps pour de gros modulos que pour des petits). Entre $n = 12$ et $n = 15$ le test Décroissant est dix fois plus lent que le Croissant mais cette différence disparaît à l'itération suivante.

Second graphique:

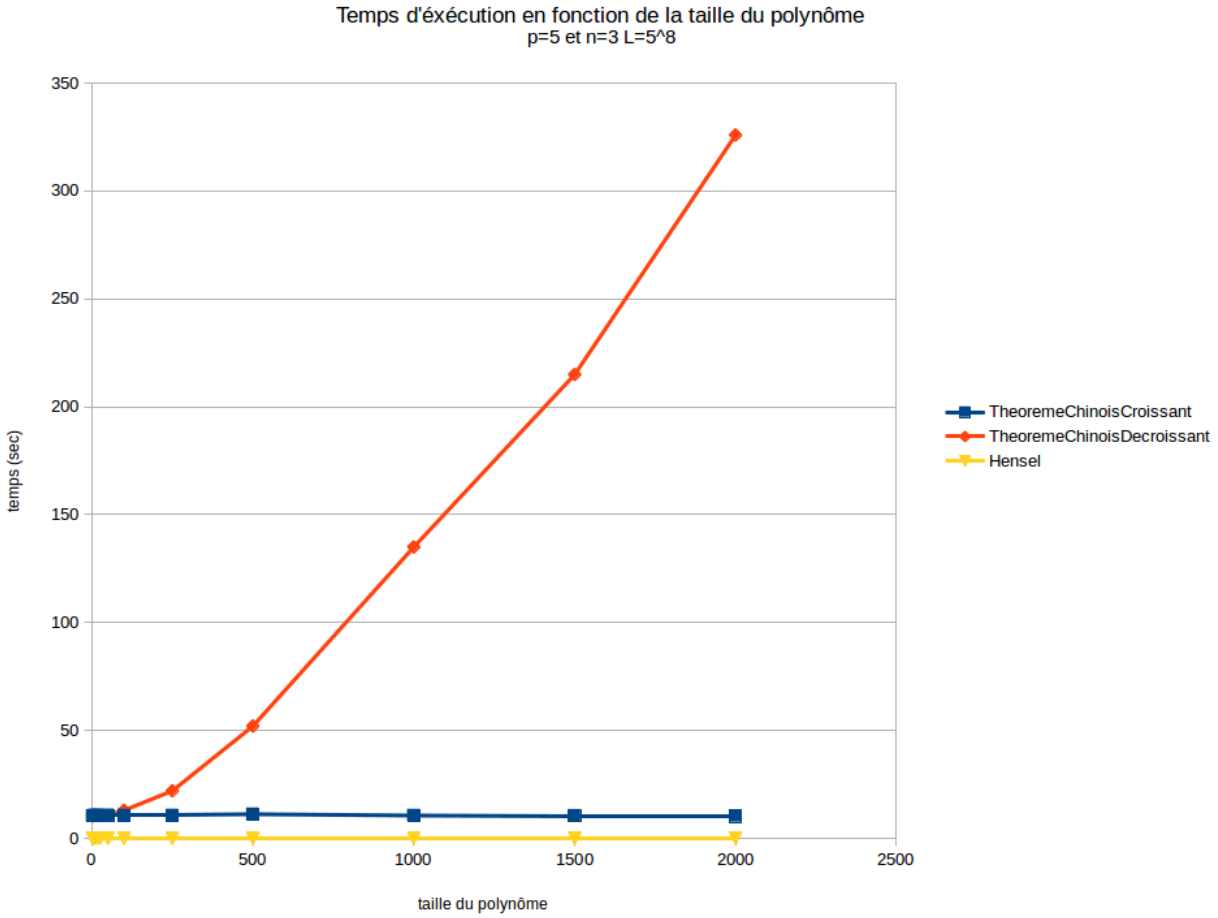


Analyse: Pour chaque lancement de programme on a $L \approx 2^{2^{16}} \approx 5^{2^{15}} \approx 17^{2^{14}} \approx 257^{2^{13}} \approx 65537^{2^{12}}$

A chaque exécution de Hensel on fait varier le nombre premier p et on prend un n de manière à garder la limite $L = p^{2^n}$ à peu près la même.

On remarque que la différence entre les temps d'exécution de chaque test est négligeable. Donc le nombre premier avec lequel on commence la remontée de Hensel n'a pas d'influence sur le temps d'exécution.

Troisième graphique:



Analyse: Pour chaque test on fait varier la taille du polynôme en fonction du temps; pas de variation pour Hensel et TheoremeChinoisCroissant. Cependant pour theoremeChinoisDeccroissant cela est dû à la fonction de recherche de racines.

5.2 Solution rationnelle

Avec la méthode du théorème Chinois ainsi qu'avec la remontée de Hensel on peut trouver des solutions entières. Par exemple, lorsqu'on recherche une racine d'un polynôme qui a une racine entière, la valeur retournée par ces deux méthodes sera toujours la même dès que le modulo sera supérieur à la solution. La valeur qui sera systématiquement retournée est la racine recherchée.

Cependant, on peut aussi trouver des solutions rationnelles à l'aide des calculs modulaires. En effet lorsqu'on a un $x = b\ mod(m)$, on peut trouver un R et un V tel que $b = R/V\ mod(m)$.

preuve :

Soit $x = b\ mod(m)$,et $L < b$. Cherchons R et V tel que$R < L$ et $b = \dfrac{R}{V}\ mod(m)$.

$b = \dfrac{R}{V}\ mod(m) \Rightarrow$ il existe k tel que $b = \dfrac{R}{V} + k \times m \Rightarrow b \times v = R + k \times v \times m \Rightarrow b \times v - k \times v \times m = R$.

Puis on applique l'algorithme d'Euclide jusqu'à obtenir $R < L$.

exemple :
Soit P un polynôme tel que $P(x) = 2x - 1$, on cherche une racine de P avec la remontée de Hensel pour $m = 5$.

$P'(x) = 2$

$P(3) = 2 \times 3 - 1 = 5 = 0\ mod(5)$, donc $x_0 = 3$.

$x_1 = x_0 - P(x_0) \times P'(x_0)^{-1}\ mod(25)$

$x_1 = 3 - 5 \times (2)^{-1}\ mod(25)$

$x_1 = 3 - 5 \times 13\ mod(25)$

$x_1 = 13\ mod(25)$

$x_2 = x_1 - P(x_1) \times P'(x_1)^{-1}\ mod(625)$

$x_2 = 13 - 25 \times (2)^{-1}\ mod(625)$

$x_2 = 13 - 25 \times 313\ mod(625)$

$x_2 = 313\ mod(625)$

Or $3 = \frac{1}{2}\ mod(5)$, $13 = \frac{1}{2}\ mod(25)$ et $313 = \frac{1}{2}\ mod(625)$... on peut donc en conclure que $\frac{1}{2}$ est une racine de P.

6 Conclusion

Les difficultés que nous avons rencontrées étaient diverses.
En terme de code elles se traduisaient principalement par des erreurs de segmentation et ce souvent lors de la phase de "traduction" de notre code, c'est à dire le passage à GMP. Nous avons effectué le passage à GMP après avoir terminé la partie fondamentale du projet, car la remontée d'Hensel fait croître très vite le modulo, et pour effectuer des tests significatifs nous avions besoin de dépasser les limite standard du logiciel. Notre maîtrise de la bibliothèque était faible et donc entraînait une cascades d'erreurs.
Nonobstant, cet acquis est important pour nous car nous savons à présent par quel moyen pousser nos tests sans limite.

Nous avons aussi appris a utiliser LaTeX pour écrire ce rapport qui est un langage que nous n'avions jamais utiliser jusqu'à présent.

Concernant les notions vues pendant ce projet, la première relative au théorème Chinois nous était familière, étant donné que nous l'avions déjà étudié au cours l'année précédente. Ce projet nous a permis de voir d'autres exemples d'applications comme résoudre un système linéaire modulaire.
L'assimilation de la méthode de la remontée de Hensel était plus délicate. Bien qu'abstraite notre encadrant a su nous apporter les outils nous permettant d'aborder au mieux cette notion.

Le calcul de complexité des algorithmes était d'une difficulté certaine, l'expérimentation nous à permis de rendre compte de l'efficacité de la remontée de Hensel par rapport à la méthode du théorème Chinois. En effet le temps d'exécution de la remontée de Hensel est bien meilleur quel que soit le choix du paramètre à faire varier. De plus il n'est pas nécessaire d'utiliser une liste de nombre premier.
Bien que l'optimisation du temps d'exécution de la remontée de Hensel est une évidence expérimentale par rapport au théorème Chinois, il reste néanmoins une limite pratique, pour certaines valeurs l'ordinateur ne finit pas l'exécution de Hensel.