Factorisation par fractions continues

Margot Funk, Antoine Hugounet

Février 2021

Table des matières

Introduction

La méthode de factorisation par fractions continues est une variante de la méthode de Fermat-Kraitchik dont les idées (recherche de congruences de carrés $\pmod N$ — N étant le nombre à factoriser — et recherche de relations linéaires sur $\mathbb{F}_2$) demeurent centrales dans les algorithmes de factorisation actuels. Cette méthode fut suggérée en 1931 par D. H. Lehmer et R. E. Powers et permet de manier des quantités plus petites que dans la méthode de Kraitchik originelle. C'est aussi l'une des premières méthodes de factorisation dont la complexité soit sous exponentielle. Bien que la méthode de crible quadratique lui soit aujourd'hui préférée, la méthode des fractions continues obtint des succès d'importance, notamment la première factorisation du nombre de Fermat $F_7 = 2^{2^7} + 1$ le 13 septembre 1970. Cet exploit est dû à M. A. Morrison et J. Brillhart qui implémentèrent la méthode pour la première fois. Dans ce projet, nous implémentons à notre tour la méthode des fractions continues, en langage C, et dans une version très proche de celle de Morrison et Brillhart. La première section de ce texte est donc consacrée à la théorie mathématique sous-jacente ; la deuxième présente l'implémentation en détails ; la troisième expose les résultats numériques obtenus.

1 Théorie

1.1 Fractions continues

1.1.1 Intuition

Intuitivement, une fraction continue est une expression — finie ou infinie — de la forme suivante :

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \dots}}}$$

telle que $a_0 \in \mathbb{Z}$ et $a_i \in \mathbb{N}^*$ pour tout $i \in \mathbb{N}^*$. Si la fraction continue est finie, elle est un rationnel ; si la fraction continue est infinie, on lui associe une valeur en calculant a_0, puis $a_0 + \frac{1}{a_1}$, puis $a_0 + \frac{1}{a_1 + \frac{1}{a_2}}$, et continuant une infinité de fois. La limite de la

suite générée est la « valeur » de la fraction continue. Nous ferons sens plus précis de l'intuition dans la prochaine sous-section.

Les fractions continues permettent d'approcher des réels irrationels par une suite de fractions d'entiers. Par exemple, la fraction $\frac{103993}{33102}$ approche π avec une précision meilleure que le milliardième. Pour générer une telle fraction continue, on part de l'identité $x = x + \lfloor x \rfloor - \lfloor x \rfloor$ et l'on écrit

$$x = \lfloor x \rfloor + \cfrac{1}{\cfrac{1}{x - \lfloor x \rfloor}}.$$

On pose $x_0 = x$ et $x_1 = \frac{1}{x_0 - \lfloor x_0 \rfloor}$ (bien défini par irrationalité de x) et l'on répète la première étape sur x_1 :

$$x = \lfloor x_0 \rfloor + \frac{1}{x_1} = \lfloor x_0 \rfloor + \cfrac{1}{\lfloor x_1 \rfloor + \cfrac{1}{\cfrac{1}{x_1 - \lfloor x_1 \rfloor}}}.$$

Comme le réel x est irrationnel, on peut répéter ce procédé indéfiniment. Nous construisons alors la suite d'éléments *irrationnels* de terme général

$$x_n = \frac{1}{x_{n-1} - \lfloor x_{n-1} \rfloor}, \quad \forall n \geqslant 1.$$

On associe alors à *l'irrationnel* x la fraction continue *infinie*[1]

$$\hat{x}_0 + \cfrac{1}{\hat{x}_1 + \cfrac{1}{\hat{x}_2 + \cfrac{1}{\hat{x}_3 + \dots}}},$$

où l'on a posé

$$\hat{x}_i = \lfloor x_i \rfloor$$

pour tout $i \in \mathbb{N}$.

Notation 1.1. Soit $x \in \mathbb{R}$ un élément irrationel. Notons $x_0 = x$,

$$x_n := \frac{1}{x_{n-1} - \lfloor x_n \rfloor}, \quad \forall n \ \geqslant 1,$$

puis

$$\hat{x}_n := \lfloor x_n \rfloor, \quad \forall n \in \mathbb{N}.$$

1. Lorsque nous aurons correctement défini la notion de fraction continue, cette fraction continue canoniquement associée à x sera notée $\hat{x}$.

Remarque 1.2. La méthode de construction d'une fraction continue *finie* pour un rationnel est la même : il faut simplement s'arrêter lorsque l'on tombe sur un $\hat{x}_n$ vérifiant $\hat{x}_n = \lfloor \hat{x}_n \rfloor$. Cet algorithme termine et s'exécute plus simplement en utilisant l'algorithme d'Euclide ([**wikiu**] ¶ *Les deux fractions continues (finies) d'un rationnel*).

1.1.2 Définition, réduites

Formellement, on peut définir [2] une fraction continue ainsi :

Définition 1.3 (Fraction continue)**.** On appelle *fraction continue* toute suite non vide (finie ou infinie) $(a_i)_{i\in U} \in \mathbb{Z}^U$, $U \subset \mathbb{N}$, d'entiers qui vérifie

$$a_i \geqslant 1, \quad \forall i \in U \setminus \{0\}.$$

Cette suite est alors notée

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \dots}}}.$$

Notation 1.4. Soit $x \in \mathbb{R}$ un irrationel. On note $\hat{x}$ la fraction continue infinie canoniquement associée à x par la méthode exposée dans le premier paragraphe.

À toute fraction continue est associée une suite (finie ou infinie) de fractions « intermédiaires » appelées *réduites* [3].

Définition 1.5 (Réduites formelles)**.** Soit $(X_i)_{i\in\mathbb{N}}$ une suite (infinie) d'indeterminées sur le corps $\mathbb{Q}$. On définit

$$[X_0] = X_0$$

puis par récurrence

$$[X_0, \dots, X_n] = X_0 + \frac{1}{[X_1, \dots, X_n]}.$$

Définition 1.6 (Réduites d'une fraction continue)**.** Soit f une fraction continue.

- Si f est donnée par la suite finie $(a_0, \dots, a_n)$, pour tout $k \in [\![0, n]\!]$ on appelle *k-ième réduite de f* l'élément $[a_0, \dots, a_k]$.
- Si f est donnée par la suite infinie $(a_i)_{i\in\mathbb{N}}$, pour tout $k \in \mathbb{N}$ on appelle *k-ième réduite de f* l'élément $[a_0, \dots, a_k]$.

2. La définition mathématique est descriptive et non prescriptive.
3. L'utilisation d'indéterminées formelles permet dans la définition d'éviter les divisions par zéro.

Exemple 1.7. Soit f la fraction continue infinie donnée par la suite $(1)_{i\in\mathbb{N}}$. La première réduite est $[1] = 1$, la deuxième est

$$[1,1] = 1 + \frac{1}{[1]} = 1 + \frac{1}{1}.$$

Plus généralement, la k-ième réduite de f est de la forme

$$[1,1,\ldots,1] = 1 + \cfrac{1}{1 + \cfrac{1}{\cdots \cfrac{1}{1 + \cfrac{1}{1}}}}.$$

Les réduites de toute fraction continue sont des éléments rationnels, y compris celles de la forme $\hat{x}$ pour un certain irrationel x. De fait, x n'est égal à aucune des réduites de $\hat{x}$. On a toutefois (voir notations 1.1) :

$$x = [\hat{x}_1, \ldots, \hat{x}_{n-1}, x_n], \quad \forall x \in \mathbb{N}. \tag{1}$$

Cette égalité est cruciale dans notre algorithme de factorisation.

Si formellement les fractions continues sont des suites (déf. 1.3), leur représentation graphique permet de les voir trivialement comme des éléments du corps $\mathbb{Q}$. Si f est une fraction continue finie de suite $(a_0, \ldots, a_n)$, le rationnel

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \ldots}}$$

est égal à la dernière réduite $[a_0, \ldots, a_n]$. On dit que f est égale au rationnel $[a_0, \ldots, a_n]$. Pour les fractions continues infinies, ce n'est pas aussi simple.

Définition 1.8. Soient l un réel et f une fraction continue donnée par la suite infinie $(a_i)_{i\in\mathbb{N}}$. On dit que f *converge vers* l ou que f *est le développement en fraction continue de* l et l'on note $f = l$ si la suite des réduites de f converge vers l. Si une fraction continue infinie est égale à un certain réel, on dit qu'elle converge.

Exemple 1.9 (Nombre d'or)**.** On appelle *nombre d'or* et l'on note φ l'unique racine réelle positive du polynôme $X^2 - X - 1 \in \mathbb{Z}[X]$. On a $\varphi = \frac{1+\sqrt{5}}{2} \simeq 1,618$. Comme $\varphi^2 = \varphi + 1$ et que $\varphi \neq 0$, on a $\varphi = 1 + \frac{1}{\varphi} = 1 + \cfrac{1}{1 + \cfrac{1}{\varphi}}$. En réalité, φ admet un

développement en fraction continue donné par

$$\varphi = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \dots}}}.$$

Des raisonnements d'analyse élémentaire ([**wikiu**] ¶ *Bijection entre irrationnels et fractions continues infinies*) permettent de montrer que toute fraction continue infinie converge, et qu'elle converge vers un irrationnel.

Théorème 1.10. *L'application canonique*

$$x \mapsto \hat{x_0} + \cfrac{1}{\hat{x}_1 + \cfrac{1}{\hat{x}_2 + \cfrac{1}{\hat{x}_3 + \dots}}}$$

établit une bijection *entre l'ensemble des nombres réels irrationnels et l'ensemble des fractions continues infinies.*

En particulier, un réel une fraction continue converge vers un réel x si, et seulement si, $f = \hat{x}$. Attention, les réels tout entier ne sont pas en bijection avec les fractions continues (finies ou infinies). En effet, un rationnel est égal à exactement deux fractions continues car $[a_0, \dots, a_n] = [a_0, \dots, a_n - 1, 1]$ ([**wikiu**] ¶ *Les deux fractions continues (finies) d'un rationnel*).

1.1.3 Irrationels quadratiques

L'adaptation de l'algorithme de Fermat-Kraitchik avec les fractions continues que nous verrons plus tard utilise crucialement le développement en fraction continue de $\sqrt{kN}$, où N est le nombre à factoriser et $k \in \mathbb{N}*$ un entier arbitraire. Intéréssons nous aux fractions continues des éléments de cette forme.

Définition 1.11 (Irrationel quadratique)**.** On appelle *irrationel quadratique* tout nombre réel, algébrique sur $\mathbb{Q}$, de degré 2. Un irrationel quadratique est dit *réduit* si son conjugué est dans l'intervalle $]-1, 0[$.

Les fractions continues d'irrationnels quaratiques sont sujettes à des phénomènes de périodicité.

Définition 1.12. Soit $f = (a_i)_{i \in \mathbb{N}}$ une fraction continue. On dit que f est *périodique* si la suite l'est à partir d'un certain rang. Il existe alors un rang $n_0 \in \mathbb{N}$ et une période $p \in \mathbb{N}^*$ tels que

$$a_i = a_{i+p}, \quad \forall i \geqslant n_0.$$

On note alors

$$f = [a_0, \ldots, a_{n_0-1}, \overline{a_{n_0}, \ldots, a_{n_0+p-1}}].$$

On dit que f est *purement périodique* si $n_0 = 0$.

Exemple 1.13. La fraction continue du nombre d'or est purement périodique de période 1. La fraction continue de l'irrationel $\sqrt{14}$ vaut

$$\sqrt{14} = [4, \overline{2, 1, 3, 1, 2, 8}].$$

Les résultats suivants sont fondamentaux (voir [**wikiu**] ¶ *Fraction continue d'un irrationnel quadratique* pour les preuves).

Théorème 1.14 (Lagrange, 1770)**.** *Un réel irrationel est un irrationel quadratique si, et seulement si, son développement en fraction continue est périodique.*

Théorème 1.15 (Galois, 1829)**.** *Un irrationnel quadratique est réduit si, et seulement si, son développement en fraction continue est purement périodique.*

Théorème 1.16 (Legendre, 1798)**.** *Un réel irrationel est la racine carrée d'un entier* > 1 *si, et seulement si, son développement en fraction continue est de la forme*

$$[a_0, \overline{a_1, a_2, \ldots, a_2, a_1, 2a_0}].$$

Ces phénomènes de périodicité devront être pris en compte dans les paramètres d'entrée de l'algorithme de factorisation. En plus de la suite des réduites, nous aurons besoin d'une autre suite importante.

Lemme 1.17. *Soit x un irrationel quadratique. Alors l'élément $\frac{1}{x-\lfloor x \rfloor}$ est lui aussi un irrationel quadratique.*

Voir [**Lauritzen**] prop. 2.5.2. Fixons N l'entier à factoriser, $k \in \mathbb{N}*$ tel que $\sqrt{kN}$ est un irrationel quadratique et $x := \sqrt{kN}$. D'après l'identité 1 et en reprenant les notations 1.1, nous pouvons donner un développement partiel (jusqu'à un rang donné $n \in \mathbb{N}$) de x en fraction continue

$$x = [\hat{x}_1, \ldots, \hat{x}_{n-1}, x_n].$$

D'après le lemme précédent, x_n est lui aussi un irrationel quadratique, i.e. il est solution d'une équation quadratique. On peut ([**Lauritzen**] dém. de 2.5.8) de fait l'écrire de manière unique

$$x_n = \frac{P_n + x}{Q_n}, \quad P_n, Q_n \in \mathbb{Z}. \tag{2}$$

La n-ième réduite de x étant un nombre rationnel, on peut l'écrire sous la forme $\frac{A_n}{B_n}$, où A_n, B_n sont entiers et la fraction est irréductible bien définie. Pour tout $n \geqslant 1$, on a ([**Lauritzen**] § 2.7)

$$A_{n-1}^2 - kNB_{n-1}^2 = (-1)^n Q_n$$

et donc

$$A_{n-1}^2 \equiv (-1)^n Q_n \pmod{N}. \tag{3}$$

On a également ([**Lauritzen**] dém. de 2.5.8)

$$\begin{cases} P_n < \sqrt{kN} \\ Q_n < 2\sqrt{kN}. \end{cases} \tag{4}$$

Ces notations et identitées seront cruciales dans la suite.

1.2 Factorisation par fractions continues

Dans tout le reste de cette section, N désigne un entier naturel composé impair.

1.2.1 Méthodes de Fermat et Kraitchik

La méthode de factorisation de Fermat part du constat suivant.

Lemme 1.18. *Factoriser N est équivalent à l'exprimer comme différence de deux carrés d'entiers.*

Démonstration. Si $N = u^2 - v^2, u, v \in \mathbb{Z}$ alors $N = (u-v)(u+v)$. Réciproquement si l'on a une factorisation $N = ab$, alors $N = \left(\frac{a+b}{2}\right)^2 - \left(\frac{a-b}{2}\right)^2$. □

La méthode de Fermat exploite cette propriété et se montre particulièrement efficace lorsque N est le produit de deux entiers proches l'un de l'autre. Notons $N = ab$ une factorisation de N avec a et b proches, $r = \frac{a+b}{2}$, $s = \frac{a-b}{2}$ de sorte que

$$N = r^2 - s^2.$$

Comme s est petit en valeur absolue par hypothèse, l'entier r est donc plus grand que $\sqrt{N}$ tout en lui étant proche. Il existe donc un entier positif *u pas trop grand* tel que

$$\lfloor \sqrt{N} \rfloor + u = r$$

et donc tel que $(\lfloor \sqrt{N} \rfloor + u)^2 - N$ soit un carré. Trouver un tel entier u donne alors la factorisation de N. Comme les facteurs de N sont proches l'un de l'autre, on le trouve par essais successifs.

La méthode de Fermat est cependant inefficace lorsque les facteurs de N ne sont pas proches. D'après [**Tale**] ¶ *Fermat and Kraitchik*, la méthode est alors encore plus coûteuse que la méthode des divisions successives. Dans les années 1920, Maurice Kraitchik a amélioré l'efficacité de la méthode de Fermat. Son idée essentielle est que pour factoriser N, il *suffit* de trouver une différence de deux carrés qui soit un multiple de N.

Lemme 1.19. *Connaître deux entiers $u, v \in \mathbb{Z}$ tels que $u^2 \equiv v^2 \pmod N$ et $u \not\equiv \pm v \pmod N$ fournit une factorisation de N. Plus spécifiquement, les entiers $\mathrm{pgcd}(u-v, N)$ et $\mathrm{pgcd}(u+v, N)$ sont des facteurs non triviaux de N.*

Démonstration. Posons $g = \mathrm{pgcd}(u-v, N)$ et $g' = \mathrm{pgcd}(u+v, N)$. Comme $u \not\equiv \pm v \pmod N$, on a $g < N$ et $g' < N$. Enfin ni g et g' ne sont réduits à 1 : si l'un deux l'est, l'autre vaut N, contradiction. Donc g et g' sont tous deux des facteurs non triviaux de N. □

Remarque 1.20. Dans l'algorithme, nous nous contenterons de chercher des u, v tels que N divise la différence de leurs carrés, sans vérifier s'ils vérifient $u \not\equiv \pm v \pmod N$. Comme le polynôme $X^2 - v^2 \in \mathbb{Z}/N\mathbb{Z}$ a exactement quatre racines, il y a « une chance sur deux » pour que u et v nous fournissent un facteur non trivial de N.

La factorisation de N se fait donc en trouvant de tels couples (u, v). Kraitchik cherche dans un premier temps des couples $(u_i, Q_i)_{1 \leqslant i \leqslant r}$ vérifiant

$$u_i^2 \equiv Q_i \pmod N$$

et tels que l'entier $\prod_{i=1}^r Q_i$ soit un carré (dans $\mathbb{Z}$). Posant $u = \prod_{i=1}^r u_i$ et $v = \sqrt{\prod_{i=1}^r Q_i}$, il vient

$$u^2 \equiv v^2 \pmod N.$$

Obtenir des congruences de la forme $u_i^2 \equiv Q_i \pmod N$ ne pose pas de difficulté. Kraitchik propose d'utiliser le polynôme $Q := X^2 - N \in \mathbb{Z}[X]$. En se donnant des éléments $u_1, \ldots, u_s \in \mathbb{Z}$ et en posant $Q_1 = Q(u_1), \cdots, Q_s = Q(u_s)$, on a bien les congruences souhaitées. Comment exhiber une sous-famille de $\{Q_1, \cdots, Q_s\}$ dont le produit est un carré ?

1.2.2 Recherche de congruences de carrés

Morrison et Brillhart répondent à cette question dans [**MB**]. Leur méthode est basée sur la connaissance de congruences de la forme

$$u_i^2 \equiv Q_i \pmod N,$$

où $|Q_i|$ est suffisamment petit pour être factorisé. On fixe pour cela $B = \{p_1, \ldots, p_m\}$ une base de factorisation, c'est à dire un ensemble non vide fini de nombres premiers, et l'on dit qu'un entier $Q \in \mathbb{N} \setminus \{0, 1\}$ est *B-friable* si tous ses facteurs premiers sont dans B. La connaissance de suffisamment d'entiers Q_i qui sont B-friables permet de factoriser N. Soit $Q \in \mathbb{Z}$ un entier dont la valeur absolue est B-friable, il s'écrit alors

$$Q = (-1)^{v_0} \prod_{i=1}^{m} p_i^{v_{p_i}(Q)}.$$

Puisque les éléments de B sont fixés et en nombre fini, l'élément Q peut être vu comme le vecteur des valuations $(v_{p_m}(Q), \ldots, v_{p_1}(Q), v_0) \in \mathbb{N}^{m+1}$.

Définition 1.21. On appelle *B-vecteur exposant de Q*, ou simplement *vecteur exposant de B* si aucune confusion n'est possible, et l'on note $v_B(Q)$ le vecteur [4]

$$v_B(Q) := (v_{p_m}(Q), \ldots, v_{p_1}(Q), v_0) \in \mathbb{F}_2^{m+1}.$$

Connaissant suffisamment d'entiers Q_i étant B-friables, on peut en extraire une sous-famille dont le produit est un carré.

Proposition 1.22. *Soit F une famille d'entiers dont les valeurs absolues sont B-friables. Si*

$$\#F \geqslant \#B + 2$$

alors on peut extraire une sous-famille de F dont le produit des éléments est un carré.

Démonstration. Posons $F = \{Q_1, \ldots, Q_k\}$ (de sorte que $k = \#F$) et $B = (p_1, \ldots, p_m)$ (de sorte que $\#B = m$). Par hypothèse de friabilité, on peut associer à chaque Q_j, $1 \leqslant j \leqslant k$, un B-vecteur exposant. Fixons $j, j' \in [\![1, k]\!]$. L'entier Q_j est un carré si, et seulement si, les composantes de son vecteur de valuations $(v_{p_m}(Q), \ldots, v_{p_1}(Q), v_0) \in \mathbb{N}^{m+1}$ sont paires, i.e. si son B-vecteur exposant est nul. Par propriété des valuations, le B-vecteur exposant associé au produit $Q_j \cdot Q_{j'}$ est le vecteur somme $v_B(Q_j) + v_B(Q_{j'})$. Autrement dit, le produit d'une sous-famille $\{Q_{j_1}, \ldots, Q_{j_s}\}$ de F est un carré si, et seulement si, la somme des B-vecteurs exposants $v_B(Q_{j_1}), \ldots, v_B(Q_{j_s})$ est nulle. Soit V le $\mathbb{F}_2$-espace vectoriel $\mathbb{F}_2^{m+1}$, qui est de dimension $m + 1$. Comme $k \geqslant m + 2$, la famille $\{v_B(Q_1), \ldots, v_B(Q_k)\}$ est liée dans V et il existe de fait des éléments $l_1, \ldots, l_k \in \mathbb{F}_2$ tels que

$$\sum_{j=1}^{k} l_j v_B(Q_j) = 0.$$

L'élément $\prod_{j=1}^{k} Q_j^{l_j}$ est alors un carré. □

4. Notez qu'il s'agit d'un élément de $\mathbb{F}_2^{m+1}$: seule la parité des valuations nous intéresse. L'élément v_0 est placé à droite et non au début car il correspondra au bit de poids faible du B-vecteur exposant dans le code.

Étant données des congruences de la forme $u_i^2 \equiv Q_i \pmod N$, la preuve de la proposition fournit un procédé d'algèbre linéaire pour extraire une sous-famille des Q_i dont le produit des éléments est un carré. On trouve tout d'abord des $Q_{i_1}, \ldots, Q_{i_k}$ dont la valeur absolue est B-friable [5]. Soit M la matrice

$$M := \begin{pmatrix} v_B(Q_{i_1}) \\ \vdots \\ v_B(Q_{i_k}) \end{pmatrix} \in \mathcal{M}_{k,\#B+1}(\mathbb{F}_2),$$

soient $l_1, \ldots, l_k$ les éléments de $\mathbb{F}_2$ donnés dans la preuve de la proposition tels que

$$\prod_{j=1}^{k} Q_{i_j}^{l_{i_j}}$$

est un carré. Le vecteur $(l_1, \ldots, l_k)$ est un élément du noyau de la matrice transposée de M. Il peut donc être exhibé par pivot de Gauß.

1.2.3 Utilisation des fractions continues

L'introduction des fractions continues est motivée par le constat suivant. Si

$$u_i^2 = Q_i + kNb^2, \quad u_i, Q_i, b \in \mathbb{Z}, k \in \mathbb{N}^*$$

de telle sorte que $|Q_i|$ soit petit, alors

$$\left(\frac{u_i}{b}\right)^2 - kN = \frac{Q_i}{b^2}$$

est petit en valeur absolue et $\frac{u_i}{b}$ est une bonne approximation de $\sqrt{kN}$ ([**Cohen**], p. 478). Fixons k un entier naturel non nul, posons $x = \sqrt{kN}$ et reprenons les notations 1.1 et celles développées à la fin de la sous-section 1.1.3. En vertu de l'identité

$$A_{n-1}^2 \equiv (-1)^n Q_n \pmod N,$$

nous appelons *méthode de factorisation des fractions continues* la méthode de Kraitchik dans laquelle les entiers u_i sont donnés par les A_{i-1} et les Q_i par les $(-1)^i Q_i$.

L'intérêt de la méthode des fractions continues réside en la majoration $Q_n \leqslant 2\sqrt{kN}$ (4). À l'inverse, les $x^2 - N, x \in \mathbb{N}$ de Kraitchik ont une croissance linéaire de pente $2\sqrt{N}$ lorsque x s'éloigne de $\sqrt{N}$. Pour une base de factorisation B fixée, les Q_n auront donc plus de chance d'être B-friables. Or, l'étape la plus coûteuse de l'algorithme est celle de la recherche des termes B-friables par divisions successives. Notons d'autre part qu'il est facile de générer le développement en fraction continue de x et les paires (A_{n-1}, Q_n) grace à un algorithme itératif dû à Gauß et exposé dans [**MB**], p. 185. Qui plus est, on a le résultat suivant ([**MB**], p. 191).

5. Nous le ferons en factorisant les Q_i à disposition par divisions successives.

Proposition 1.23. *Les diviseurs premiers p de Q_n vérifient nécessairement*

$$\left(\frac{kN}{p}\right) \in \{0, 1\}.$$

Démonstration. Soit p un diviseur premier de Q_n. Alors $A_{n-1}^2 \equiv kNB_{n-1}^2 \pmod{p}$. Comme $\text{pgcd}(A_{n-1}, B_{n-1}) = 1$, p ne peut pas diviser B_{n-1} (sinon $A_{n-1}^2 \equiv 0 \pmod{p}$ et p diviserait aussi A_{n-1}). L'entier B_{n-1} est donc inversible modulo p et $\left(\frac{A_{n-1}}{B_{n-1}}\right)^2 \equiv kN \pmod{p}$. □

1.2.4 Quelques éléments pour appréhender la complexité de la méthode

2 Explication du programme

Dans cette §, nous décrivons notre implémentation de la méthode de factorisation des fractions continues.

2.1 Architecture du programme

2.1.1 Terminologie

Énonçons pour commencer quelques définitions qui seront utiles pour décrire le code. **réf vers 1.2.3 pour les notations**

Définition 2.1. On nomme *paire* (A, Q) tout couple $(A_{n-1}, Q_n), n \geqslant 1$.

Définition 2.2. Un ensemble de paires (A, Q) indexé par $n_1, \ldots, n_r$ est dit *valide* si le produit $\prod_{i=1}^{r}(-1)^{n_i}Q_{n_i}$ est un carré (dans $\mathbb{Z}$).

Définition 2.3. Si B est la base de factorisation utilisée par le programme, on désignera par l'expression *vecteur exposant associé à* Q_n le B-vecteur exposant $v_B((-1)^nQ_n)$.

2.1.2 Structure générale

Le programme comprend deux étapes principales. La première consiste à générer, à partir du développement en fractions continues de $\sqrt{kN}$ **réf section**, des paires (A, Q) avec Q_n friable pour une base de factorisation préalablement déterminée et fixée. On associe à chaque Q_n ainsi produit son vecteur exposant `mpz_t exp_vect`. Ce vecteur permet de retenir les nombres premiers qui interviennent dans la factorisation de Q_n avec une valuation impaire. Dans le but d'augmenter le nombre de paires (A, Q) acceptées lors de cette étape, nous avons implémenté la *large prime variation*. Cette variante permet d'accepter une paire si son Q_n se factorise grâce aux premiers de la base de factorisation fixée et à un nombre premier supplémentaire. Les fonctions de cette phase

de collecte sont rassemblées dans le fichier `step_1.c`. Elles font appel, pour mettre en oeuvre la *large prime variation*, aux fonctions du fichier `lp_var.c`.

Ces données sont traitées lors de la seconde phase dans l'espoir de trouver un facteur non trivial de N. Il s'agit de trouver des ensembles valides de paires (A, Q) parmi les paires trouvées en première phase. Cela se fait par pivot de Gauß sur la matrice dont les lignes sont formées des vecteurs exposants **réf à avant**. Chaque ensemble de paires (A, Q) valide trouvé fournit une congruence de la forme $A^2 \equiv Q^2 \pmod{N}$. Cette dernière permet éventuellement de trouver un facteur non trivial de N. Les fonctions de cette phase sont regroupées dans le fichier `step_2.c`.

Avant d'effectuer la première étape, il faut se doter d'une base de factorisation, ce qui est fait avec les fonctions de `init_algo.c`. Ces dernières se chargent plus généralement de l'initialisation et du choix par défaut des paramètres, comme la taille de la base de factorisation ou l'entier k.

Finalement, en mettant bout à bout les deux étapes, la fonction `contfract_factor` du fichier `fact.c` recherche un facteur non trivial de N et `print_results` affiche les résultats.

2.1.3 Entrées et sorties

Nous avons regroupé dans une structure `Params` les paramètres d'entrée de la fonction de factorisation, à savoir :

- `N` : le nombre à factoriser, supposé produit de deux grands nombres premiers.
- `k` : le coefficient multiplicateur.
- `n_lim` : le nombre maximal de paires (A, Q) que l'on s'autorise à calculer. Ce nombre prend en compte toutes les paires produites, non uniquement les paires dont le Q_n est friable ou résultant de la *large prime variation*.
- `s_fb` : la taille de la base de factorisation.
- `nb_want_AQp` : le nombre désiré de paires (A, Q) avec Q_n friable ou résultant de la *large prime variation*.
- des booléens indiquant si la *early abort strategy* ou la *large prime variation* doivent être utilisées et des paramètres s'y rapportant.

Le programme stocke dans une structure `Results` un facteur non trivial de `N` trouvé (si tel est le cas) ainsi que des données permettant l'analyse des performances de la méthode.

Remarque 2.4. L'efficatité de la méthode dépend du choix des paramètres ci-dessus. Pour avoir plus de latitude dans les tests, nous les considérons comme des paramètres

d'entrée du programme. C'est pourquoi notre programme ne s'attèle pas à la factorisation complète d'un entier, qui aurait nécessité une sous-routine déterminant des paramètres optimaux en fonction de la taille de l'entier dont on cherche un facteur.

Remarque 2.5. Notre programme n'est pas supposé prendre en entrée un nombre admettant un petit facteur premier (inférieur aux premiers de la base de factorisation par exemple). En effet, comme il ne teste pas au préalable si `N` est divisible par de petits facteurs, il mettra autant de temps à trouver un petit facteur qu'un grand facteur.

2.2 Pivot de Gauss et recherche d'un facteur non trivial : `step_2.c`

Avant de nous pencher sur les détails de la phase de collecte, regardons l'implémentation de la seconde phase, qui aide à mieux comprendre la forme sous laquelle nous collectons les données.

2.2.1 Utilisation des données collectées

A l'issue de la première phase, on espère avoir collecté `nb_want_AQp` paires (A, Q) avec Q_n friable [6]. Le nombre réel de telles paires est stocké dans le champ `nb_AQp` d'une structure `Results` (voir sous-§ précédente). Une paire (A, Q) collectée est caractérisée par :

— la valeur A_{n-1},

— la valeur Q_n,

— le vecteur exposant associé à Q_n,

— un vecteur historique (voir ci-contre).

Les données de ces `nb_AQp` paires sont stockées dans quatre tableaux : `mpz_t *Ans`, `mpz_t *Qns`, `mpz_t *exp_vects` et `mpz_t *hist_vects`. À un indice correspond une paire (A, Q) donnée. Le vecteur historique sert à indexer les paires collectées pour former un analogue de la matrice identité utilisée pendant le pivot de Gauss. Plus précisément, `hist_vects[i]` est, avant pivot de Gauss, le vecteur $(e_{l-1}, \cdots, e_0)$ où $l =$ `nb_AQp` et $e_j = \delta_{ij}$. À partir de ces quatre tableaux, la fonction `find_factor` cherche un facteur de `N` selon la méthode des congruences de carrés. Elle utilise pour cela les fonctions auxiliaires `gauss_elimination` et `calculate_A_Q`.

6. Les paires peuvent aussi résulter de la *large prime variation*, cela n'a aucune incidence sur les fonctions de cette partie.

2.2.2 La fonction `gauss_elimination`

La fonction `gauss_elimination` effectue un pivot de Gauss sur les éléments de `mpz_t *exp_vects`, vus comme les vecteurs-lignes d'une matrice. Comme pour un pivot de Gauss classique, les calculs effectués sur les vecteurs exposants sont reproduits en parallèle sur la matrice identité, c'est-à-dire sur les éléments de `mpz_t *hist_vects`. Si le *xor* de deux vecteurs exposants donne le vecteur nul, cela signifie qu'une relation de dépendance a été trouvée. On inscrit alors dans un tableau l'indice de ce vecteur nul. Le vecteur historique dudit indice indique les paires (A, Q) de l'ensemble valide trouvé. La procédure que nous avons implémentée est décrite ci-dessous.

Algorithme 1 : PIVOT DE GAUSS

Entrées : tableau exp_vects$[0 \cdots nb_AQp-1]$ des vecteurs exposants, tableau hist_vects$[0 \cdots nb_AQp-1]$ des vecteurs historiques

Sorties : hist_vects$[0 \cdots nb_AQp-1]$ après le pivot, le nombre nb_lin_rel de relations linéaires trouvées, lin_rel_ind$[0 \cdots nb_lin_rel-1]$ contenant les indices des lignes où une relation linéaire a été trouvée

créer tableau msb_ind$[0 \cdots nb_AQp-1]$
créer tableau lin_rel_ind
$nb\_lin\_rel \leftarrow 0$

`/* Initialisation du tableau MSB_IND : Msb(x) renvoie 0 si x est nul, l'indice du bit de poids fort de x sinon. Les indices des bits sont numérotés de 1 à l'indice du bit de poids fort. */`

pour $i \leftarrow 0$ ***à*** nb_AQp-1 **faire**
    msb_ind$[i] \leftarrow$ MSB(exp_vects$[i]$)

pour $j \leftarrow MAX$(msb_ind) ***à*** 1 **faire**
    $pivot \leftarrow \begin{cases} \min\left\{i \in [\![0, nb\_AQp-1]\!] \middle| \text{msb\_ind}[i] = j\right\} \\ \varnothing \text{ si pour tout } i \in [\![0, nb\_AQp-1]\!], \text{msb\_ind}[i] \neq j \end{cases}$
    si $pivot \neq \varnothing$ **alors**
        pour $i \leftarrow pivot+1$ ***à*** nb_AQp-1 **faire**
            si msb_ind$[i] = j$ **alors**
                exp_vects$[i] \leftarrow$ exp_vects$[i] \oplus$ exp_vects$[pivot]$
                hist_vects$[i] \leftarrow$ hist_vects$[i] \oplus$ hist_vects$[pivot]$
                msb_ind$[i] \leftarrow$ MSB(exp_vects$[i]$)
                si exp_vects$[i] = 0$ **alors**
                    ajouter i au tableau lin_rel_ind
                    $nb\_lin\_rel \leftarrow nb\_lin\_rel + 1$

retourner hist_vects$[0 \cdots nb_AQp-1]$, lin_rel_ind$[0 \cdots nb_lin_rel-1]$, nb_lin_rel

2.2.3 La fonction `calculate_A_Q`

Une fois les indices des vecteurs historiques indiquant un ensemble valide de paires (A, Q) récupérés, la fonction `find_factor` appelle la fonction `calculate_A_Q` pour calculer des entiers A et Q vérifiant $A^2 \equiv Q^2 \pmod{N}$. Elle lui donne en argument un de ces vecteurs historiques et les données des tableaux `Ans` et `Qns`.

Notons l l'entier `nb_AQp` et $(e_{l-1}, \cdots, e_0) \in \mathbb{F}_2^l$ le vecteur historique donné en argument de la fonction. Le calcul de

$$A := \prod_{n=0}^{l-1} Ans[i]^{e_i} \pmod N$$

ne pose pas de difficulté. Pour le calcul de

$$Q := \sqrt{\prod_{n=0}^{l-1} Qns[i]^{e_i}} \pmod N,$$

on utilise l'algorithme proposé par Morrison et Brillhart.

Algorithme 2 : Extraction de racine carrée

Entrées : Des entiers $Q_1, \cdots, Q_r \in \mathbb{Z}$ tels que $\prod_{i=1}^r Q_i$ est un carré
Sorties : $\sqrt{\prod_{i=1}^r Q_i} \pmod N$

$Q \leftarrow 1$
$R \leftarrow Q_1$
pour $i \leftarrow 2$ ***à*** r **faire**
  $X \leftarrow \text{pgcd}(R, Q_i)$
  $Q \leftarrow XQ \pmod N$
  $R \leftarrow \frac{R}{X} \cdot \frac{Q_i}{X}$
$X \leftarrow \sqrt{R}$
$Q \leftarrow XQ \pmod N$
retourner Q

Pour démontrer la correction de l'algorithme, on peut utiliser l'invariant de boucle

$$Q\sqrt{R.Q_i \cdots Q_r} \equiv \sqrt{\prod_{i=1}^{r} Q_i} \pmod N,$$

dont la conservation découle de l'égalité

$$Q\sqrt{R.Q_i \cdots Q_r} \equiv QX\sqrt{\frac{R}{X}\frac{Q_i}{X}Q_{i+1} \cdots Q_r} \pmod N.$$

2.3 Collecte des paires (A, Q) : `step_1.c` et `lp_var.c`

Décrivons à présent la phase de collecte des données. Concernant les vecteurs historiques, il suffit d'initialiser à la fin de la collecte `hist_vects[i]` pour `0 <= i < nb_AQp`. C'est ce que fait la fonction `init_hist_vects`. La collecte des autres données requiert un peu plus d'explications.

2.3.1 La fonction `create_AQ_pairs`

Sachant que seules les paires (A, Q) dont on a pu factoriser Q_n nous intéressent pour la seconde phase, nous avons décidé de ne stocker que celles-ci. Ce choix a un autre avantage : ayant fixé `nb_want_AQp` le nombre de paires (A, Q) factorisées désirées, on peut arrêter le développement en fraction continue dès que ce nombre est atteint. Au fur à mesure du développement de $\sqrt{kN}$ en fraction continue, il faut donc tester si le Q_n qui vient d'être calculé est factorisable. Si c'est le cas, on crée son vecteur exposant et ajoute les données de la paire aux tableaux `Ans`, `Qns` et `exp_vects`. Tout ceci est géré dans la longue fonction `create_AQ_pairs`, qui utilise les sous-routines `is_Qn_factorisable` et `init_exp_vect`.

2.3.2 La *early abort strategy*

La fonction `is_Qn_factorisable` teste si un Q_n est friable [7] par divisions successives avec les premiers de la base de factorisation. Il est possible d'améliorer les performances de la fonction en l'empêchant de poursuivre les divisions successives si, après un certain nombre de tentatives, on juge la partie non factorisée de Q_n trop grande. On fixe pour cela une borne `eas_bound_div` (proportionnelle à la borne déjà connue $2\sqrt{kN}$), un palier `eas_cut` et l'on arrête les divisions successives sur Q_n si Q_n reste plus grand que `eas_bound_div` après `eas_cut` tentatives de divisions.

2.3.3 La *large prime variation*

Étant donnée une base de factorisation $B = \{p_1, \cdots, p_m\}$, la *large prime variation* consiste à accepter lors de la collecte, non seulement des Q_n B-friables mais aussi des Q_n produits d'un entier B-friable et d'un entier lp_n inférieur à p_m^2. On dira que Q_n est *presque friable* et l'on appellera *grand premier (large prime)* le premier lp_n en question.

Pour que des Q_n presque friables soient exploitables, il faut qu'ils aient un grand premier lp en commun. En effet, si on trouve deux entiers presque friables $Q_{n_1} = X_{n_1} lp$ et $Q_{n_2} = X_{n_2} lp$, on peut former une nouvelle paire (A, Q) avec laquelle on peut travailler pour chercher une congruence de carrés. Remarquons pour cela que :

$$\begin{cases} A_{n_1-1}^2 \equiv (-1)^{n_1} X_{n_1} lp \pmod N, \\ A_{n_2-1}^2 \equiv (-1)^{n_2} X_{n_2} lp \pmod N. \end{cases}$$

Multiplier entre elles ces deux identités donne :

$$(A_{n_1-1}A_{n_2-1})^2 \equiv \underbrace{(-1)^{n_1+n_2} X_{n_1} X_{n_2}}_{\substack{\text{associé au vecteur exposant} \\ v_B\left((-1)^{n_1}X_{n_1}\right)+v_B\left((-1)^{n_2}X_{n_2}\right)}} \underbrace{lp^2}_{\text{carré qui ne pose pas problème}} \pmod N.$$

7. Ou presque friable, voir ¶ suivant.

On forme donc la nouvelle paire $(A_{n_1-1}A_{n_2-1} \pmod N, Q_{n_1}Q_{n_2})$ associée au vecteur exposant $v_B\Big((-1)^{n_1}X_{n_1}\Big) + v_B\Big((-1)^{n_2}X_{n_2}\Big)$. Elle sera traitée lors de la deuxième phase exactement de la même manière que les paires « classiques ».

En pratique, pour repérer les paires qui ont le même grand premier, nous constituons au fur et à mesure de la collecte une liste chainée dont les nœuds stockent les données d'une paire dont le Q_n est presque friable (les entiers Q_n, A_{n-1}, le vecteur exposant et le grand premier associé à Q_n). Nous maintenons cette liste triée par taille des grands premiers. Lorsque survient un Q_n presque friable, il est repéré par la fonction `is_Qn_factorisable` qui fournit également son grand premier lp. La liste chainée est alors parcourue pour savoir si l'on a déjà rencontré ce lp. Deux cas se présentent alors. Si lp est absent de la liste, on crée à la bonne place un nœud. Si lp est déjà présent dans la liste, au lieu de rajouter un nœud, on utilise le nœud possédant ce lp pour obtenir une nouvelle paire (A, Q) selon la méthode énoncée plus haut et ajoute ses composantes aux tableaux `Ans`, `Qns` et `exp_vects`. La fonction `insert_or_elim_lp` se charge de cela.

3 Résultats expérimentaux

Les fonctions dont nous nous sommes servi pour tester le programme se trouvent dans le fichier `test.c`. Nous avons effectué nos tests sur des entiers générés par la fonction `rand_N`. Ces entiers sont de la forme $N = pq$ avec p et q des premiers aléatoires et de taille semblable.

3.1 Choix de la taille de la base de factorisation

Nous avons cherché à déterminer expérimentalement, selon la taille de l'entier N à factoriser, une bonne taille pour la base de factorisation lorsque la *large prime variation* et la *early abort strategy* sont utilisées. Pour une taille d'entier fixée, nous avons inscrit dans un fichier le temps que met le programme pour plusieurs tailles de bases de factorisation (voir les fichiers *.txt). Nous donnons ci-dessous des tailles appropriées (il ne s'agit que d'un ordre de grandeur, la taille optimale varie d'un entier à l'autre). Ce sont elles qui sont renvoyées par la fonction `choose_s_fb`.

nombre de bits de N	taille de la base de factorisation
70	truc
80	truc
90	truc
100	truc
110	ttuc
120	truc
130	truc
140	truc
150	truc

3.2 Factorisation de F_7

Notre programme permet de factoriser $F_7 = 2^{128} + 1$. La fonction `fact_F7` prend en entrée des booléens indiquant si la *large_prime variation (lp)* et la *early abort strategy (eas)* doivent être utilisées et lance cette factorisation. Avec le paramètre `k` retourné par la fonction `choose_k` - soit $k = 38$ - et les paramètres `eas_cut` et `eas_coeff` valant respectivement 50 et 1000000, nous obtenons les résultats suivants :

variantes	`s_fb`	`nb_want_AQp`	`nb_AQp`	`last_n`	temps de calcul
lp et *eas*					
lp					
eas					
sans variante					

`last_n` est l'indice de la dernière paire (A, Q) calculée.

3.3 Temps de calcul (avec les deux variantes)

Ce graphique représente le temps moyen mis par le programme (sur ... tests) en fonction du nombre de bits de l'entier que l'on souhaite factoriser. La *large prime variation* et la *early abort strategy* ont été utilisées et les paramètres ont été choisis par les fonctions `choose_s_fb` et `choose_k`.

comparer quand k choisi par la fonction et k = 1 : prévu de calculer le rapport des deux temps