Analyse numérique et réduction de réseaux

Ivan Morel[*,**] — **Damien Stehlé**[***] — **Gilles Villard**[****,**]

[*] *Université de Lyon, ÉNS de Lyon, Université de Sydney*
[**] *INRIA, Laboratoire LIP, 46 Allée d'Italie 69364 Lyon Cedex 07, France*
ivan.morel@ens-lyon.fr

[***] *CNRS, Universités de Sydney et Macquarie*
Département de Mathématiques et de Statistiques (F07)
Université de Sydney NSW 2006, Australie
damien.stehle@gmail.com

[****] *CNRS, Université de Lyon, ÉNS de Lyon*
gilles.villard@ens-lyon.fr

RÉSUMÉ. *L'algorithmique des réseaux euclidiens est un outil fréquemment utilisé en informatique et en mathématiques. Elle repose essentiellement sur la réduction LLL qu'il est donc important de rendre aussi efficace que possible. Une approche initiée par Schnorr consiste à effectuer des calculs approchés pour estimer les orthogonalisations de Gram-Schmidt sous-jacentes. Sans approximations, ces calculs dominent le coût de la réduction. Récemment, des outils classiques d'analyse numérique ont été revisités et améliorés, pour exploiter plus systématiquement l'idée de Schnorr et réduire les coûts. Nous décrivons ces développements, notamment comment l'algorithmique en nombres flottants peut être introduite à plusieurs niveaux dans la réduction.*

ABSTRACT. *The algorithmic facet of euclidean lattices is a frequent tool in computer science and mathematics. It mainly relies on the LLL reduction, making worthwile efficiency improvements. Schnorr proved that the LLL algorithm can be speeded up if one computes approximations to the underlying Gram-Schmidt orthogonalisations. Without approximations, these computations dominate the cost of the reduction. Recently, classical tools from the field of numerical analysis have been revisited and improved in order to strengthen Schnorr's approach, and further reducing the costs. We describe these developments, and show especially how floating-point computations may be introduced at various levels in the reduction process.*

MOTS-CLÉS : *Réduction de réseau euclidien, LLL, arithmétique flottante, orthogonalisation de Gram-Schmidt, analyse de perturbation.*

KEYWORDS: *Lattice reduction, LLL, floating-point arithmetic, Gram-Schmidt orthogonalisation, perturbation analysis.*

RSTI - TSI. Volume 29 – n° 1/2010, pages 115 à 144

1. Introduction

Un réseau euclidien est une grille de points régulièrement espacés dans un espace euclidien. Plus précisément, si $\mathbf{b}_1, \ldots, \mathbf{b}_d \in \mathbb{R}^n$ sont linéairement indépendants, le réseau qu'ils engendrent est l'ensemble de leurs combinaisons linéaires entières :

$$L[\mathbf{b}_1, \ldots, \mathbf{b}_d] = \sum_{i \leq d} \mathbb{Z}\mathbf{b}_i = \left\{ \sum_{i \leq d} x_i \mathbf{b}_i, x_i \in \mathbb{Z} \right\}.$$

Dans une telle situation, les vecteurs $\mathbf{b}_i$ forment une base du réseau et l'entier d en est la dimension. Si $d \geq 2$, alors tout réseau donné admet une infinité de bases, liées entre elles par des transformations unimodulaires, c'est-à-dire linéaires à coefficients entiers et inversibles. Soit B (respectivement B') la matrice de $\mathbb{R}^{n \times d}$ ayant pour colonnes les vecteurs linéairement indépendants $\mathbf{b}_1, \ldots, \mathbf{b}_d$ (respectivement $\mathbf{b}'_1, \ldots, \mathbf{b}'_d$). Les deux ensembles de vecteurs engendrent le même réseau si et seulement s'il existe une matrice $U \in \mathbb{Z}^{d \times d}$ de déterminant ± 1 telle que $B' = B \cdot U$. Trouver des bases intéressantes à partir de bases quelconques est le paradigme de la réduction de réseaux. Les réseaux euclidiens sont l'objet de travaux mathématiques depuis plus d'un siècle (Minkowski, 1896), et, depuis la mise au point du célèbre algorithme LLL (Lenstra *et al.*, 1982) au début des années 1980, un effort important a été consacré à leur algorithmique. Dans cet article, nous nous intéresserons principalement à l'efficacité algorithmique de réductions de type LLL. Celles-ci permettent d'obtenir efficacement – en temps polynomial en la taille de la donnée – une base constituée de vecteurs relativement orthogonaux et relativement courts (voir le théorème 2).

Depuis son invention, l'algorithme LLL a donné lieu à de très nombreuses applications. Citons par exemple la factorisation des polynômes à coefficients entiers (Lenstra *et al.*, 1982), améliorée récemment par (van Hoeij, 2001) puis par Novocin dans sa thèse de doctorat (Novocin, 2008) ; la résolution d'équations diophantiennes (Hanrot, 2009) ; plusieurs cryptanalyses de variantes du système de chiffrement à clé publique RSA (May, 2009). L'algorithme LLL est aussi utilisé fréquemment en théorie algorithmique des nombres (Cohen, 1995) et en théorie des télécommunications (Mow, 1994; Hassibi *et al.*, 1998). Ainsi, toute amélioration algorithmique de LLL permet de résoudre plus efficacement de nombreux problèmes.

Donnons un exemple concret et simple (mais assez informel) d'application de l'algorithme LLL. Supposons que l'on dispose de d nombres réels $\alpha_1, \ldots, \alpha_d$ entre lesquels il existe une petite relation linéaire entière : $x_1\alpha_1 + \ldots + x_d\alpha_d = 0$, pour des entiers $x_1, \ldots, x_d$ petits (par exemple, plus petits qu'une certaine borne fixée). On cherche à déterminer cette relation linéaire entière. Considérons le réseau engendré par les vecteurs $\mathbf{b}_i = (C \cdot \alpha_i, 0, \ldots, 0, 1, 0, \ldots, 0)^T$, où la coordonnée égale à 1 est en position $i+1$, et C est un très grand nombre. Le vecteur $x_1\mathbf{b}_1 + \ldots + x_d\mathbf{b}_d$ appartient au réseau engendré par les vecteurs $\mathbf{b}_i$, et est inhabituellement court. En effet, sa première coordonnée est nulle, alors qu'elle serait grande si les x_i n'étaient pas une petite relation linéaire entre les α_i. Aussi, les autres coordonnées, égales aux x_i, sont

petites. L'algorithme LLL permet de trouver un tel vecteur court dans un réseau. Une fois ce vecteur court trouvé, il suffit de lire les x_i dans les coordonnées successives. Lorsque les α_i sont les puissances successives d'un nombre algébrique, cette stratégie permet de trouver son polynôme minimal (voir notamment (Kannan *et al.*, 1984)).

Dans leur article historique, Arjen Lenstra, Hendrik Lenstra Jr. et László Lovász ont montré qu'à partir d'une base $\mathbf{b}_1, \ldots, \mathbf{b}_d \in \mathbb{Z}^n$ d'un réseau, l'algorithme LLL finit en temps $O\left(d^3 n \log \|B\| \mathcal{M}(d \log \|B\|)\right)$, où $\|B\|$ est le maximum des normes euclidiennes des vecteurs $\mathbf{b}_i$ et $\mathcal{M}(x)$ est le coût d'une multiplication de deux entiers de x bits chacun. (Kaltofen, 1983) a ensuite prouvé que LLL finit en fait en temps $O\left(d^4 n \log^2 \|B\| \mathcal{M}(d + \log \|B\|)/(d + \log \|B\|)\right)$. La contribution dominante dans le coût total correspond aux calculs (ou plus exactement mises à jour) des orthogonalisations de Gram-Schmidt des bases successives apparaissant au cours de l'exécution de l'algorithme, en arithmétique rationnelle. Pour accélérer l'algorithme, il est donc naturel d'essayer de remplacer ces rationnels par des approximations flottantes, dont la taille des mantisses est plus faible que celle des rationnels impliqués. Odlyzko a implanté cette idée au début des années 1980, de manière heuristique, dans le but de cryptanalyser le chiffrement asymétrique de Merkle et Hellman (Merkle *et al.*, 1978; Odlyzko, 1984). (Schnorr, 1988) a été le premier à apporter un fondement théorique à une telle stratégie : il décrit un algorithme de type LLL reposant sur une orthogonalisation de Gram-Schmidt approchée, de complexité $O\left(d^3 n \log \|B\| \mathcal{M}(d + \log \|B\|)\right)$. Le principal inconvénient de cet algorithme est que les approximations réelles utilisées requièrent encore $O(d + \log \|B\|)$ bits, rendant incontournable l'utilisation de multiprécision.

En 2005, Nguyen et Stehlé (Nguyen *et al.*, 2005; Stehlé, 2005; Nguyen *et al.*, 2009) ont apporté une réponse à la fois rigoureuse et efficace en pratique à la question de l'algorithme LLL flottant. La précision requise dans les calculs de leur algorithme, L^2, ne dépend plus que de la dimension d. En particulier, en codant les approximations des nombres rationnels par des nombres flottants, la précision requise pour les calculs flottants ne croît que linéairement en d et est indépendante de $\|B\|$. Les flottants disponibles en machine semblent pouvoir être exploités jusqu'à des dimensions relativement élevées (Nguyen *et al.*, 2006). L'algorithme L^2 admet une borne de complexité $O\left(d^2 n \mathcal{M}(d) \log \|B\|(d + \log \|B\|)\right)$. En notant $\beta = \log \|B\|$ et pour un produit d'entiers « lent » en $\mathcal{M}(x) = O(x^2)$ cela donne $O(d^5 n\beta + d^4 n\beta^2)$. Ceci est par exemple meilleur dans un facteur $\log \|B\|$ que la borne donnée par (Kaltofen, 1983). Surtout, le coût de L^2 ne fait pas apparaître de terme cubique en $\log \|B\|$, tel que le terme $O(d^3 n\beta^3)$ présent pour (Schnorr, 1988). À ce jour, l'algorithme L^2 semble être le plus efficace pour obtenir une base LLL-réduite, en théorie et en pratique, plus particulièrement quand la taille des entiers en entrée est grande devant les dimensions d et n. L'algorithme L^2 est intégré dans les bibliothèques de calcul MAGMA (Cannon *et al.*, 2008) et SAGE (Stein, 2009)[1]. Nous pouvons par ailleurs comparer les coûts avec un produit d'entiers asymptotiquement rapide en $\mathcal{M}(x) = O(x \log x \log\log x)$ opé-

1. Nous renvoyons le lecteur à (Stehlé, 2009) pour la mise en œuvre pratique de l'algorithme L^2 de (Nguyen *et al.*, 2006).

rations binaires (voir (Schönhage *et al.*, 1971)), nous noterons aussi $\mathcal{M}(x) = x^{1+\epsilon}$. L'algorithme L^2 a un coût en $O(d^{4+\epsilon}n\beta + d^{3+\epsilon}n\beta^2)$, avec donc le même exposant total 6 que celui des algorithmes de (Schnorr, 1988) ou (Storjohann, 1996). L^2 possède cependant l'avantage du comportement quadratique β^2 devant les termes presque quadratiques $\beta^{2+\epsilon}$ des autres approches.

Un des ingrédients principaux dans la preuve de correction de L^2 consiste en une analyse d'erreur de l'algorithme de Cholesky lorsque l'on utilise une arithmétique flottante. Ainsi, à partir de la matrice (exacte) des produits scalaires des vecteurs $\mathbf{b}_i$, de bonnes approximations des coefficients de l'orthogonalisation de Gram-Schmidt peuvent être calculées en restreignant autant que possible la précision des calculs. L'algorithme L^2 introduit aussi des approximations de nombres rationnels dans la phase dite de proprification (voir en section 6), et généralise ainsi l'utilisation des nombres flottants dans le processus de réduction.

Dans le présent article, nous entreprenons de décrire comment des techniques d'analyse numérique interviennent pour élaborer des algorithmes efficaces de réduction de type LLL. Nous décrirons à un haut niveau le principe de l'algorithme L^2 ainsi que les améliorations et extensions apportées depuis sa mise au point. En particulier, nous montrerons comment les calculs numériques approchés permettent de tester rapidement et rigoureusement si une base donnée est LLL-réduite (Villard, 2007a). Nous verrons aussi comment améliorer la qualité d'une base LLL-réduite (Morel *et al.*, 2009) pour un coût essentiellement indépendant de $\log \|B\|$. Ces travaux sont soit récents soit en cours. Plutôt que nous attarder sur les détails techniques des preuves, notre motivation est d'expliquer les principes généraux sous-jacents. Il s'agit globalement de restreindre et gérer la précision de calculs intermédiaires approchés en nombres flottants, tout en conservant des résultats exacts.

Dans la section 2, nous présentons rapidement les réseaux euclidiens et la réduction LLL. Ensuite, dans la section 3, nous décrivons le type de techniques et de résultats d'analyse numérique qui peuvent être employés dans le contexte de la réduction des réseaux. Dans la section 4, nous présentons un algorithme de vérification qu'une base donnée est LLL-réduite. Nous montrons ensuite, dans la section 5, comment améliorer efficacement la qualité d'une base LLL-réduite. Enfin, dans la section 6, nous nous intéressons au calcul d'une base LLL-réduite à partir d'une base quelconque.

Travaux connexes. (Koy *et al.*, 2001b) et (Schnorr, 2006) étudient l'utilisation des algorithmes de Givens et de Householder reposant sur l'arithmétique flottante pour obtenir de bonnes approximations des coefficients de Gram-Schmidt. Cependant, leurs résultats demeurent heuristiques. Rendre rigoureuses ces stratégies est l'un des objectifs des travaux que nous allons décrire. Par ailleurs, d'autres types d'améliorations algorithmiques de LLL ont été proposés. Citons par exemple (Schönhage, 1984; Storjohann, 1996; Koy *et al.*, 2001a). Notons que ces algorithmes renvoient des bases de qualités (légèrement) dégradées par rapport à LLL pour obtenir des bornes de complexité meilleures. Le principe sous-jacent consiste à réduire le nombre d'opérations arithmétiques en tentant de considérer autant que possible des blocs de vec-

teurs consécutifs plutôt que toute la base. Ces améliorations ne sont pas du même ordre que celles auxquelles nous nous intéressons ici. On peut donc ainsi espérer pouvoir les combiner avec l'utilisation de calculs approchés. Enfin, de nombreux travaux (Schnorr, 1987; Schnorr *et al.*, 1994; Schnorr, 2009; Gama *et al.*, 2006; Gama *et al.*, 2008) améliorent l'algorithme LLL dans le sens où la base renvoyée est de meilleure qualité (constituée de vecteurs plus courts et plus orthogonaux), avec, en contre-partie, un coût plus élevé voire exponentiel. Une fois les techniques d'analyse numérique bien intégrées dans l'algorithme LLL, il sera naturel d'essayer de les adapter à ces extensions.

Notations. Tous les vecteurs seront en gras. Faute de contre-indication, si $\mathbf{b}$ est un vecteur, alors la notation $\|\mathbf{b}\|$ sera la norme euclidienne de $\mathbf{b}$. Si $\mathbf{b}_1$ et $\mathbf{b}_2$ sont deux vecteurs de même dimension, alors nous noterons $\langle \mathbf{b}_1, \mathbf{b}_2 \rangle$ leur produit scalaire. Si $\mathbf{b}$ est un vecteur et k un entier, nous noterons $\mathbf{b}[k]$ la k-ième coordonnée du vecteur $\mathbf{b}$. Si $\mathbf{b}$ est un vecteur de dimenion n et a et b sont deux entiers tels que $1 \leq k_1 \leq k_2 \leq d$, alors la notation $\mathbf{b}[k_1..k_2]$ fera référence au vecteur de dimension $k_2 - k_1 + 1$ constitué des coordonnées d'indices k_1 à k_2 du vecteur $\mathbf{b}$. Si B est une matrice (ou un vecteur), alors sa transposée sera notée B^t. Par défaut, la fonction $\log$ correspondra au logarithme en base 2. Enfin, si x est un réel, alors la notation $\lfloor x \rceil$ désignera un entier le plus proche de x (n'importe lequel s'il y a deux choix).

2. Introduction aux réseaux euclidiens

Dans cette section, nous introduisons la notion de LLL-réduction d'un réseau euclidien. Les sections suivantes montreront comment vérifier, améliorer et atteindre une LLL-réduction efficacement. Pour une introduction plus détaillée aux réseaux euclidiens et à leurs problèmes algorithmiques, nous renvoyons par exemple le lecteur aux premiers chapitres de l'ouvrage (Micciancio *et al.*, 2002).

2.1. *Orthogonalisation de Gram-Schmidt*

Soit $\mathbf{b}_1, \ldots, \mathbf{b}_d$ dans $\mathbb{R}^n$ linéairement indépendants. Le procédé de Gram-Schmidt consiste à associer aux $\mathbf{b}_i$ une famille de vecteurs orthogonaux $\mathbf{b}_i^*$ définie récursivement de la manière suivante :

$$\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j<i} \mu_{i,j} \mathbf{b}_j^* \text{ avec } \mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\|\mathbf{b}_j^*\|^2}.$$

Nous appellerons coefficients de Gram-Schmidt des vecteurs $\mathbf{b}_i$ l'ensemble des $\mu_{i,j}$ (pour $i > j$) et des $\|\mathbf{b}_i^*\|^2$. L'orthogonalisation de Gram-Schmidt permet de quantifier à quel point les vecteurs $\mathbf{b}_i$ sont orthogonaux : si les coefficients $|\mu_{i,j}|$ sont petits, et si les longueurs $\|\mathbf{b}_i^*\|$ ne décroissent pas trop vite, alors les vecteurs $\mathbf{b}_i$ sont plutôt orthogonaux.

Si les vecteurs $\mathbf{b}_i$ sont à coefficients entiers, alors les coefficients de Gram-Schmidt sont rationnels et peuvent être calculés en temps polynomial en d, n et $\log \|B\|$ (voir par exemple (Lenstra *et al.*, 1982) et (Erlingsson *et al.*, 1996)). Néanmoins, les numérateurs et dénominateurs peuvent *a priori* être des entiers longs, de taille proportionnelle à $d(\log \|B\| + \log d)$. Ainsi, on préférera éviter autant que possible de calculer exactement les coefficients de Gram-Schmidt.

2.2. *Volume et minima d'un réseau*

Soit L un réseau et $\mathbf{b}_1, \ldots, \mathbf{b}_d \in \mathbb{R}^n$ une base de L. La matrice de Gram $G(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ des vecteurs $\mathbf{b}_i$ est la matrice carrée de dimension d des produits scalaires deux à deux des $\mathbf{b}_i$: si $i, j \leq d$, on a $G_{i,j} = \langle \mathbf{b}_i, \mathbf{b}_j \rangle$. On définit le déterminant du réseau par $\det(L) = \sqrt{\det(G(\mathbf{b}_1, \ldots, \mathbf{b}_d))}$. Cette quantité ne dépend pas du choix de la base $\mathbf{b}_1, \ldots, \mathbf{b}_d$ puisque les bases sont liées entre elles par des transformations unimodulaires. Le déterminant est un invariant du réseau.

Puisque L est discret, il existe un vecteur de $L \setminus \{\mathbf{0}\}$ de norme euclidienne minimale. La norme d'un tel vecteur s'appelle le premier minimum du réseau, et on la note $\lambda_1(L)$. On définit aussi les minima successifs du réseau :

$$\forall i \leq d,\ \lambda_i(L) = \min \{R : \dim(\mathcal{B}(\mathbf{0}, R) \cap L) \geq i\}.$$

Alors que le déterminant du réseau peut être calculé en temps polynomial, les minima sont difficilement estimables précisément lorsque la dimension augmente : approcher $\lambda_1(L)$ à n'importe quelle constante près est NP-difficile sous des réductions randomisées (Khot, 2004). Le premier théorème de Minkowski fournit un lien entre ces deux invariants du réseau. Le deuxième est une généralisation impliquant les minima successifs.

Théorème 1 (Théorèmes de Minkowski) *Soit L un réseau de dimension d. Alors :*

$$\lambda_1(L) \leq \sqrt{d} \cdot (\det L)^{1/d} \quad \textit{et} \quad \prod_{i=1}^{d} \lambda_i(L) \leq \sqrt{d}^d \cdot (\det L).$$

2.3. *La réduction LLL*

La réduction LLL, du nom de ses inventeurs Arjen Lenstra, Hendrik Lenstra Jr. et László Lovász (Lenstra *et al.*, 1982), offre une alternative pratique aux théorèmes de Minkowski : une base LLL-réduite $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ peut être obtenue en temps polynomial, et les vecteurs la constituant satisfont des propriétés proches des théorèmes de Minkowski.

Soit $\delta \in]1/4, 1[$. Soit $\mathbf{b}_1, \ldots, \mathbf{b}_d \in \mathbb{R}^n$ des vecteurs linéairement indépendants. On considère leurs coefficients de Gram-Schmidt. Les vecteurs $\mathbf{b}_i$ sont δ-LLL-réduits si les deux propriétés suivantes sont satisfaites :

1) Pour tous $i > j$, on a $|\mu_{i,j}| \leq 1/2$.

2) Pour tout i, on a $\delta\|\mathbf{b}_i^*\|^2 \leq \|\mathbf{b}_{i+1}^*\|^2 + \mu_{i+1,i}^2\|\mathbf{b}_i^*\|^2$.

La première condition est appelée la *condition de propreté*. La deuxième est la *condition de Lovász* qui signifie qu'après projection sur l'orthogonal de l'espace engendré par $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$, le vecteur $\mathbf{b}_i$ est quasiment plus court que le vecteur $\mathbf{b}_{i+1}$. Ces deux conditions impliquent que les longueurs $\|\mathbf{b}_i^*\|$ ne peuvent pas décroître arbitrairement rapidement.

Théorème 2 *(Lenstra* et al.*, 1982) Soit $\delta \in]1/4, 1[$ et $\alpha = 1/(\delta - 1/4)$. Soit $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ une base δ-LLL-réduite d'un réseau L. Alors les propriétés suivantes sont satisfaites :*

1) Pour tout $i < d$, on a $\|\mathbf{b}_i^*\|^2 \leq \alpha\|\mathbf{b}_{i+1}^*\|^2$.

2) $\|\mathbf{b}_1\| \leq \alpha^{d-1}(\det L)^{1/d}$.

3) $\prod_{i \leq d} \|\mathbf{b}_i\| \leq \alpha^{\frac{d(d-1)}{2}}(\det L)$.

Lenstra, Lenstra et Lovász (1982) décrivent aussi un algorithme qui permet, à partir d'une base $\mathbf{b}_1, \ldots, \mathbf{b}_d \in \mathbb{R}^n$ d'un réseau, de déterminer une base LLL-réduite de ce même réseau. Cet algorithme, depuis appelé l'*algorithme LLL*, s'exécute en temps $O\left(d^3 n \log \|B\| \mathcal{M}(d \log \|B\|)\right)$. L'algorithme de LLL-réduction de Nguyen et Stehlé (2009), que l'on abordera en section 6, admet une borne de complexité de $O\left(d^2 n \mathcal{M}(d) \log \|B\|(d + \log \|B\|)\right)$ (voir la discussion à ce sujet en introduction).

3. Arithmétique flottante et analyse numérique

Nous introduisons ici les outils d'analyse numérique qui nous seront utiles par la suite. Grâce au contexte d'arithmétique flottante standardisée (voir (IEEE, 2008)), ces outils sont rigoureux en ce sens que toutes les constantes qui apparaissent peuvent être explicitées, et qu'ils donnent des majorations d'erreur exactes. Pour des explications plus détaillées, le lecteur intéressé pourra consulter notamment les chapitres 2, 4, 10 et 19 de l'ouvrage (Higham, 2002).

3.1. *Une arithmétique flottante standardisée*

Soit $p > 0$ un entier. Un nombre flottant de précision p est un triplet (s, m, e) où le signe s est dans $\{-1, 1\}$, la mantisse m est un entier dans $[2^{p-1}, 2^p - 1]$ et l'exposant $e \in \mathbb{Z}$ est borné[2]. Le flottant représente le nombre réel $s \cdot m \cdot 2^{e-p}$. On considère également que 0 est un flottant. La norme IEEE-754 a permis de standardiser

2. Ici, pour simplifier, nous ne prendrons pas en compte le fait que l'exposant soit borné, et supposerons implicitement qu'aucun dépassement de capacité n'intervient.

la représentation des flottants et les opérations arithmétiques élémentaires sur ceux-ci, pour deux précisions – la simple précision et la double précision – supportées par une très grande majorité de microprocesseurs. Dans le cas de la double précision, le triplet est représenté sur 64 bits, soit un ou deux mots machine, et la précision p est de 53 bits. On appelle doubles les flottants en double précision. Entre autres choses, la norme spécifie que si le mode d'arrondi choisi est l'arrondi au plus proche, les opérateurs $+, -, \times, \div$ sont conformes au modèle ci-après.

Modèle flottant. Si a et b sont deux doubles et si op $\in \{+, -, \times, \div\}$, alors la valeur renvoyée par l'opération a op b est le double le plus proche (celui de mantisse paire en cas d'ambiguité) de la valeur exacte a op b. De même, si a est un double, alors l'opérateur $\sqrt{a}$ renvoie le flottant le plus proche de la valeur exacte $\sqrt{a}$.

Pour distinguer l'opérateur flottant de l'opérateur exact, nous noterons a op b pour l'opération exacte et $\diamond(a \text{ op } b)$ pour l'opérateur flottant. Le modèle flottant s'étend naturellement à toute précision p. Nous appelerons ulp (*unit in last place*) la quantité 2^{-p}. Si x est un nombre réel et que l'on connaît $\diamond(x)$, alors nous dirons que x est connu à l'ulp près. La généralisation du modèle flottant à toute précision p est plus contraignante que le modèle suivant, très utilisé en analyse numérique.

Modèle numérique. Si a et b sont deux flottants de précision p et si op $\in \{+, -, \times, \div\}$, alors $|\diamond(a \text{ op } b) - (a \text{ op } b)| \leq 2^{-p} \cdot |a \text{ op } b|$. De même, on a $|\diamond(\sqrt{a}) - \sqrt{a}| \leq 2^{-p} \cdot |\sqrt{a}|$.

Ces deux modèles sont définis rigoureusement et permettent d'obtenir des preuves complètes de bornes d'erreurs. Ils sont fidèlement implantés par la plupart des processeurs pour la double précision, et par plusieurs bibliothèques pour des précisions p arbitraires, dont MPFR (Fousse *et al.*, 2007).

3.2. *Orthogonalisation de Gram-Schmidt et factorisation QR*

Si l'on utilise une arithmétique rationnelle, l'algorithme d'orthogonalisation qui dérive directement de la définition de l'orthogonalisation de Gram-Schmidt de la section 2 fait intervenir des numérateurs et des dénominateurs de grandes tailles. Afin d'éviter de coûteuses manipulations en multiprécision, il est naturel d'essayer de se contenter d'approcher les nombres rationnels en utilisant une arithmétique flottante, si possible (suivant la taille de la base en entrée) avec la double précision. L'orthogonalisation de Gram-Schmidt a été très étudiée en analyse numérique (Higham, 2002), elle est en effet un passage incontournable pour résoudre les problèmes de moindres carrés.

L'orthogonalisation de vecteurs linéairement indépendants $\mathbf{b}_1, \ldots, \mathbf{b}_d \in \mathbb{R}^n$ correspond à la factorisation QR de la matrice $n \times d$ dont les colonnes sont les $\mathbf{b}_i$. Soit B une matrice $n \times d$ de rang d. Cette matrice se décompose de manière unique $B = QR$ avec $Q \in \mathbb{R}^{n \times d}$ constituée de vecteurs colonnes orthonormaux et $R \in \mathbb{R}^{d \times d}$ triangulaire supérieure avec des coefficients diagonaux positifs. Les colonnes de Q cor-

respondent aux vecteurs $\mathbf{b}_i^*/\|\mathbf{b}_i^*\|$, on a aussi $\|\mathbf{b}_i^*\| = R_{i,i}$ et $\mu_{i,j} = R_{j,i}/R_{j,j}$ pour tous $i > j$.

Pour calculer la factorisation QR d'une matrice, on utilise le plus fréquemment les algorithmes de Householder, Givens ou Gram-Schmidt modifié. Ces algorithmes ont des propriétés numériques similaires. L'algorithme 1 est une description de l'algorithme de Householder. Quand l'algorithme est appelé avec une précision de calcul p constante on note $\widetilde{R}$ l'approximation de R obtenue. Si l'on considère le facteur R de la factorisation, ces algorithmes sont stables dans le sens inverse. C'est-à-dire l'approximation $\widetilde{R}$ calculée est le vrai facteur R d'une matrice $\widetilde{B}$ proche de la matrice B donnée en entrée. Par exemple, pour l'algorithme 1, on a le résultat suivant.

Théorème 3 *Soit $B \in \mathbb{R}^{n\times d}$ une matrice de rang d donnée en entrée à l'algorithme 1. Supposons que les calculs soient flottants et effectués avec une précision p telle que $30d(n+9)2^{-p} \leq 1$. Soit $\widetilde{R} \in \mathbb{R}^{d\times d}$ la matrice renvoyée. Alors il existe une matrice $Q \in \mathbb{R}^{n\times d}$ de colonnes orthonormales telle que les colonnes de la matrice $\Delta B = B - Q\widetilde{R}$ satisfassent les relations suivantes :*

$$\forall j \leq d,\ \|\Delta\mathbf{b}_j\| \leq 30d(n+9)2^{-p} \cdot \|\mathbf{b}_j\|.$$

Pour ce théorème classique le lecteur se référera par exemple au théorème 19.4 de (Higham, 2002). Les constantes sont calculées explicitement dans (Chang *et al.*, 2009). Surtout d'un point de vue pratique, elles permettent de borner les erreurs très précisement. Des théorèmes analogues peuvent être établis pour le calcul du facteur R par l'algorithme de Givens ou de Gram-Schmidt modifié.

Entrée : Une matrice $B \in \mathbb{R}^{n\times d}$.
Sortie : Une matrice $R \in \mathbb{R}^{d\times d}$.

1. R:=B. Pour i de 1 à d, faire
2. $\quad\mathbf{r}$:=$\mathbf{r}_i[i..n]$, $\mathbf{v}$:=$\mathbf{r}$.
3. $\quad\sigma_i$:=sign($\mathbf{r}[1]$), s:=$\sigma_i \cdot \|\mathbf{r}\|$.
4. $\quad\mathbf{v}[1]$:=$(-\sum_{j=2}^{n-i+1} \mathbf{r}[j]^2)/(\mathbf{r}[1]+s)$.
5. $\quad\mathbf{v}$:=$\frac{1}{\sqrt{-s\cdot\mathbf{v}[1]}} \cdot \mathbf{v}$.
6. $\quad$Pour j de i à d, faire
7. $\quad\quad\mathbf{r}_j[i..n] = \mathbf{r}_j[i..n] - (\mathbf{v}^t\mathbf{r}_j[i..n]) \cdot \mathbf{v}$.
8. $\quad\quad\mathbf{r}_j[i] = \sigma_i \cdot \mathbf{r}_j[i]$.
9. Renvoyer les d premières lignes de R.

Algorithme 1. *L'algorithme de Householder*

Pour obtenir des termes explicites dans le théorème 3, il convient de préciser exactement comment les opérations flottantes sont effectuées. À l'étape 3 de l'algorithme, on commence par calculer le carré de la norme du vecteur, en effectuant la somme

des carrés des coordonnées du plus grand indice vers le plus petit – pour réutiliser l'avant-dernière somme de carrés à l'étape 4 – puis on prend la racine du résultat. La multiplication par le signe de $\mathbf{r}[1]$ peut sembler anodine, mais elle est cruciale pour garantir la stabilité inverse de l'algorithme. Le numérateur apparaissant à l'étape 4 a été calculé à l'étape 3 : on réutilise cette valeur. À l'étape 7, on commence par effectuer le produit scalaire $\mathbf{v}^t \mathbf{r}_j[i..n]$ (en calculant séquentiellement la somme impliquée), puis le reste des calculs se fait coordonnée par coordonnée.

Pour justifier le fait que la matrice $\widetilde{R}$ calculée soit proche du facteur R de la matrice B donnée en entrée, le théorème 3 doit être complété. Cette proximité de $\widetilde{R}$ et R n'est d'ailleurs en générale pas vraie : l'algorithme de Householder n'est pas stable dans le sens direct. Pour que ce soit le cas, il suffit que la matrice B soit « bien conditionnée » vis-à-vis du facteur R. Cela signifie que si une matrice B' est proche de B, en un sens qui doit être précisé, alors les facteurs R de B et B' sont proches. Cette propriété est généralement établie par une étude de perturbation (voir (Higham, 2002, section 19.2) et les références qui y sont données). Chang, Stehlé et Villard (2009) montrent que si la matrice B est LLL-réduite – même pour une LLL-réduction affaiblie – alors la propriété est vérifiée.

Théorème 4 *Soit $\eta \in [1/2, 1[, \theta \in [0, 1-\eta[$ et $\delta \in]\eta^2, 1[$. Soit $B \in \mathbb{R}^{n \times d}$ de rang d dont le facteur R satisfait $|R_{i,j}| \leq \eta R_{i,i} + \theta R_{j,j}$ et $\delta R_{i,i}^2 \leq R_{i,i+1}^2 + R_{i+1,i+1}^2$ pour tous $i < j$. On considère $\epsilon \geq 0$ tel que $6nd^{3/2}\rho^{d+1} \cdot \epsilon < 1$ où*

$$\rho = (1 + \eta + \theta)\left(\theta\eta + \sqrt{(1+\theta^2)\delta - \eta^2}\right)/(\delta - \eta^2).$$

Si $\Delta B \in \mathbb{R}^{n \times d}$ est telle que pour tout i on ait $\|\Delta \mathbf{b}_i\| \leq \epsilon \cdot \|\mathbf{b}_i\|$, et si $R + \Delta R$ est le facteur R de la matrice $B + \Delta B$, alors :

$$\forall j \leq d,\ \|\Delta \mathbf{r}_j\| \leq 10nd\rho^j \epsilon \cdot R_{j,j}.$$

Le théorème exprime qu'une base LLL-réduite est bien conditionnée en ce sens que la perte de précision qui résulte sur $\widetilde{R}$ est essentiellement linéaire en d, c'est-à-dire indépendante de $\log \|B\|$. Sous les conditions d'application du théorème, on obtient qu'une perturbation par colonnes de la matrice B entraîne une perturbation du facteur R bornée par colonnes. Notons que le théorème 3 sur la stabilité inverse de l'algorithme de Householder affirme que la matrice $B + \Delta B$ dont $\widetilde{R}$ est le facteur R est une perturbation par colonnes de la matrice B donnée en entrée. Les deux théorèmes peuvent ainsi être combinés pour fournir un résultat de stabilité dans le sens direct.

Dans (Chang *et al.*, 2009), il est démontré que pour une base perturbée qui n'est pas réduite, le nombre de bits de précision perdus lors du calcul du facteur R peut être de l'ordre de

$$d + \log \prod_{\substack{j < d \\ R_{j,j} > R_{j+1,j+1}}} \frac{R_{j,j}}{R_{j+1,j+1}}.$$

Cette propriété signifie qu'essentiellement $\log R_{j,j} - \log R_{j+1,j+1}$ bits de précision sont perdus à chaque fois que le coefficient diagonal du facteur R décroît. Si la condition de Lovász est satisfaite, on retrouve bien que le nombre de bits de précision perdus est $O(d)$.

En prenant $\theta = 0$ et $\eta = 1/2$ on voit que l'hypothèse du théorème 4 sur la matrice B est plus faible qu'une hypothèse de LLL-réduction. Cet affaiblissement de l'hypothèse a été choisi pour donner une certaine circularité au résultat : si la matrice B satisfait les hypothèses avec des paramètres η, θ et δ, alors la matrice perturbée $B + \Delta B$ satisfait les hypothèses pour des paramètres $\eta' = \eta + O(1)\rho^d\epsilon, \theta' = \theta + O(1)\rho^d\epsilon$ et $\delta' = \delta - O(1)\rho^d\epsilon$. En prenant ϵ décroissant exponentiellement avec la dimension d – ce ce qui correspond à choisir une précision linéaire en d pour les calculs flottants, alors cet affaiblissement des facteurs peut être rendu arbitrairement petit. Cette circularité du théorème 4 fournit notamment une notion de « réduction numérique » qui se conserve de manière itérative.

Par ailleurs, une base $\mathbf{b}_1, \ldots, \mathbf{b}_d$ dont la matrice B correspondante satisfait les hypothèses du théorème 4 a des propriétés semblables aux bases LLL-réduites.

Théorème 5 *(Chang* et al.*, 2009) Soit $\eta \in [1/2, 1[, \theta \in [0, 1 - \eta[$ et $\delta \in]\eta^2, 1[$. Soit $\mathbf{b}_1, \ldots, \mathbf{b}_d \in \mathbb{R}^n$ linéairement indépendants. Supposons que le facteur R correspondant à la matrice dont les colonnes sont les $\mathbf{b}_i$ satisfasse les conditions $|R_{i,j}| \leq \eta R_{i,i} + \theta R_{j,j}$ et $\delta R_{i,i}^2 \leq R_{i,i+1}^2 + R_{i+1,i+1}^2$ pour tous $i < j$. Nous dirons qu'une telle base est (δ, η, θ)-LLL-réduite. Si on pose $\alpha = \eta\theta + \sqrt{(1+\theta^2)\delta - \eta^2}/(\delta - \eta^2)$, alors :*

1) Pour tout $i < d$, on a $\|\mathbf{b}_i^*\|^2 \leq \alpha \|\mathbf{b}_{i+1}^*\|^2$.

2) $\|\mathbf{b}_1\| \leq \alpha^{d-1} (\det L)^{1/d}$.

3) $\prod_{i \leq d} \|\mathbf{b}_i\| \leq \alpha^{\frac{d(d-1)}{2}} \det(L)$.

Nous illustrons la différence entre la $(\delta, \eta, 0)$-LLL-réduction et la (δ, η, θ)-LLL-réduction à la figure 1. Le dessin de gauche correspond à un paramètre θ nul, alors que le dessin de droite correspond à un paramètre θ non-nul. Dans chaque dessin, le vecteur $\mathbf{b}_1$ est fixé et la base $(\mathbf{b}_1, \mathbf{b}_2)$ est réduite si et seulement si $\mathbf{b}_2$ est dans une des zones hachurées. Le théorème 5 (à comparer au théorème 2) signifie que du point de vue des propriétés impliquées par les conditions de réduction, il est suffisant de s'intéresser à la notion affaiblie de (δ, η, θ)-LLL-réduction.

3.3. *Orthogonalisation de Gram-Schmidt et factorisation de Cholesky*

Soit $G \in \mathbb{R}^{d \times d}$ une matrice symétrique définie positive. Alors G s'écrit de manière unique sous la forme $G = R^t R$ avec R une matrice triangulaire supérieure à coefficients diagonaux positifs. Cette décomposition s'appelle la factorisation de Cholesky. Celle-ci est reliée à la factorisation QR, et donc à l'orthogonalisation de

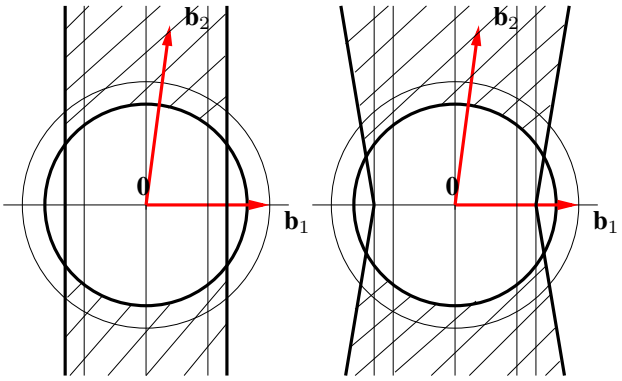

Figure 1. *Comparaison entre $(\delta, \eta, 0)$-LLL-réduction et (δ, η, θ)-LLL-réduction*

Gram-Schmidt, de la manière suivante. Soit $B \in \mathbb{R}^{n \times d}$ de rang d et $B = QR$ sa factorisation QR. Alors $B^t B$ est symétrique définie positive et sa factorisation de Cholesky est $B^t B = R^t R$. Ainsi, si $\mathbf{b}_1, \ldots, \mathbf{b}_d \in \mathbb{R}^n$ sont linéairement indépendants, leurs coefficients de Gram-Schmidt $\mu_{i,j}$ et $\|\mathbf{b}_i^*\|^2$ peuvent s'obtenir par le calcul de la factorisation de Cholesky de leur matrice de Gram $G(\mathbf{b}_1, \ldots, \mathbf{b}_d)$.

(Nguyen *et al.*, 2009) utilisent l'algorithme 2 pour calculer les coefficients d'orthogonalisation de Gram-Schmidt à partir de la matrice de Gram exacte. Notons qu'il s'agit en fait d'une décomposition LU de la matrice de Gram, ce qui permet de ne pas utiliser de racine carrée. De même que pour l'algorithme de Householder, il convient de lever les ambiguités lorsque l'on utilise des opérations flottantes. Dans ce cas, à l'étape 3, la somme est calculée séquentiellement.

Entrée : Une matrice de Gram $G(\mathbf{b}_1, \ldots, \mathbf{b}_d) \in \mathbb{R}^{d \times d}$ (symétrique et définie positive).
Sortie : Des coefficients $\mu_{i,j}$ et $\|\mathbf{b}_i^*\|^2$ pour $i \geq j$.

1. Pour i de 1 à d, faire
2. Pour j de 1 à i, faire :
3. $t_{i,j} := G_{i,j} - \sum_{k=1}^{j-1} t_{i,k}\mu_{j,k}$.
4. $\mu_{i,j} := t_{i,j}/t_{j,j}$.
5. $\ell_i := t_{i,i}$.
6. Renvoyer les $\mu_{i,j}$ et les ℓ_i.

Algorithme 2. *L'orthogonalisation de Gram-Schmidt à partir de la matrice de Gram*

Le résultat suivant correspond à une analyse d'erreur dans le sens direct sous l'hypothèse que la base d'entrée soit $(\delta, \eta, 0)$-LLL-réduite.

Théorème 6 *(Nguyen* et al.*, 2009) Soit* $\eta \in [1/2, 1[$ *et* $\delta \in]\eta^2, 1[$*. Soit* d *vecteurs linéairement indépendants* $\mathbf{b}_1, \ldots, \mathbf{b}_d \in \mathbb{R}^n$ *et* $\mu_{i,j}$ *les coefficients de leur orthogonalisation de Gram-Schmidt. Supposons que les vecteurs* $\mathbf{b}_i$ *soient* $(\delta, \eta, 0)$*-LLL-réduits, et que leur matrice de Gram soit connue à l'ulp près. Supposons également que les calculs soient flottants avec une précision* p *qui satisfasse* $d^2\gamma^d 2^{-p+6} \leq \epsilon$*, avec* $\gamma = \frac{(1+\eta)^2+\epsilon}{\delta-\eta^2}$ *et* $\epsilon \in]0, 1/2]$*. En notant* $\Delta\|\mathbf{b}_i^*\|^2$ *et* $\Delta\mu_{i,j}$ *les différences entre les quantités exactes et celles renvoyées par l'algorithme 2 en précision* p*, alors on a pour tous* $j < i$ *:*

$$\Delta\|\mathbf{b}_j^*\|^2 \leq 8d\gamma^j 2^{-p} \cdot \|\mathbf{b}_j^*\|^2 \quad \textit{et} \quad |\Delta\mu_{i,j}| \leq 32d\gamma^j 2^{-p}.$$

Le théorème 6 est analogue à la combinaison des théorèmes 3 et 4. Il pourrait être formulé avec une propriété de circularité semblable à celle que nous avons observée pour le théorème 4. En prenant une précision p linéaire en la dimension d, on peut faire en sorte que les coefficients de Gram-Schmidt calculés satisfassent les mêmes propriétés de LLL-réduction que la base d'entrée, avec un affaiblissement des paramètres arbitrairement faible. Considérer la matrice de Gram exacte dans le théorème 6 implique cependant une différence quant à l'erreur directe sur les coefficients de l'orthogonalisation. Nous allons le voir ci-après.

3.4. *Comparaison entre l'approche factorisation QR et l'approche matrice de Gram*

D'un point de vue général, les deux approches sont similaires. Les algorithmes en question sont stables dans le sens inverse, et le fait que les bases données en entrée soient LLL-réduites garantit que le conditionnement soit borné indépendamment de $\|B\|$. Relativement à ce dernier critère, les algorithmes sont donc stables dans le sens direct.

Concernant les entrées requises par les deux approches, la factorisation de Cholesky est appliquée à la matrice de Gram exacte de la base. Supposons que la base soit modifiée au cours de l'exécution d'un autre algorithme, comme dans le cas d'une LLL-réduction, et que l'on ait besoin de calculer des coefficients de Gram-Schmidt régulièrement. Alors toute opération sur la matrice de la base doit être effectuée parallèlement sur la matrice de Gram, et par exemple quand le réseau considéré est à coordonnées entières, cela peut considérablement augmenter le coût de l'arithmétique entière.

Pour ce qui est du nombre d'opérations effectuées, on peut vérifier (voir (Higham, 2002, chap. 19)) que l'algorithme de Householder fait intervenir de l'ordre de $2nd^2 - \frac{2}{3}d^3$ opérations flottantes, alors que l'algorithme 2 en effectue de l'ordre de $\frac{1}{3}d^3$. Ainsi, de ce point de vue, ce dernier est substantiellement préférable. En contrepartie, la précision requise *a priori* (en appliquant les théorèmes 4 et 6) par l'algorithme 2 est essentiellement le double de celle requise par l'algorithme de Householder. En effet, observons les constantes ρ et γ des théorèmes 4 et 6. Les précisions requises sont

asymptotiquement linéaires en la dimension d, et les coefficients de proportionalité sont $\log \rho$ et $\log \gamma$. Si le facteur θ du théorème 4 et le facteur ϵ du théorème 6 tendent tous les deux vers 0, alors on voit que $\gamma \approx \rho^2$. Cela signifie que l'algorithme 2 requiert une précision environ double de l'algorithme de Householder. (Schnorr, 2006) arrive à la même observation, et en déduit que l'algorithme de Householder est numériquement préférable. Cette différence n'a pas d'importance en pratique pour les petites dimensions quand la double précision n'induit pas ou peu de surcoût, mais cela peut devenir significatif quand la dimension est telle que la double précision ne suffit plus. Le choix de l'approche QR ou par matrice de Gram doit cependant tenir compte de tous les aspects, et notamment du point suivant.

Les deux approches diffèrent structurellement du point de vue de l'erreur directe sur les coefficients de Gram-Schmidt. Dans les deux cas, l'erreur sur le terme $\|\mathbf{b}_i^*\|^2$ est relative. Par contre, dans le cas de la factorisation de Cholesky, l'erreur portant sur $\mu_{i,j}$ est absolue, alors qu'avec la factorisation QR l'erreur portant sur $\mu_{i,j}$ peut être proportionnelle à $R_{i,i}/R_{j,j} = \|\mathbf{b}_i^*\|/\|\mathbf{b}_j^*\|$. Cela ne provient pas d'une faiblesse de l'analyse, mais semble intrinsèque. Par exemple soit

$$B = R = \begin{bmatrix} 1 & 1 \\ 0 & r \end{bmatrix}.$$

Pour ϵ petit considérons la perturbation $\widetilde{B}$ suivante de B ainsi que sa décomposition QR

$$\widetilde{B} = \begin{bmatrix} 1 & 1 \\ \epsilon & r \end{bmatrix} = \widetilde{Q}\widetilde{R} = \widetilde{Q} \begin{bmatrix} 1 & 1+r\epsilon \\ 0 & r-\epsilon \end{bmatrix} + O(\epsilon^2),$$

où l'élement $\widetilde{R}_{1,2}$ correspond à

$$\widetilde{\mu}_{2,1} = \frac{\langle \widetilde{\mathbf{b}}_1, \widetilde{\mathbf{b}}_2 \rangle}{\|\widetilde{\mathbf{b}}_1\|^2} = \frac{1+r\epsilon}{1+\epsilon^2} = 1 + r\epsilon + O(\epsilon^2).$$

On constate que le calcul du produit scalaire $\langle \widetilde{\mathbf{b}}_1, \widetilde{\mathbf{b}}_2 \rangle$ induit le terme d'erreur $r\epsilon$. En choisissant r, ce terme peut être rendu arbitrairement grand devant $\mu_{2,1} = 1$. En revanche, la matrice de Gram associée à B est

$$G(\mathbf{b}_1, \mathbf{b}_2) = R^t R = \begin{bmatrix} 1 & 1 \\ 1 & 1+r^2 \end{bmatrix},$$

et, même avec une perturbation en ϵ, aucun terme en r ne sera impliqué dans le calcul du terme hors diagonal lors de la factorisation de Cholesky. Le fait que la matrice de Gram soit connue « exactement » (à l'ulp près) – et par conséquent les produits scalaires – dans les hypothèses du théorème 6 et dans l'algorithme L^2 (voir en section 6), est un élément important de l'analyse de L^2. Que l'erreur portant sur les $\mu_{i,j}$ ne soit pas absolue dans le cas de l'algorithme de Householder, implique que l'on ne puisse pas toujours vérifier que la base soit $(\delta, \eta, 0)$-LLL-réduite. C'est la raison pour laquelle le facteur de réduction θ a été introduit au théorème 4. Pour autant que la matrice de Gram soit connue avec une bonne qualité numérique, le résultat du théorème 6 semble donc plus facilement exploitable dans le contexte de la LLL-réduction.

4. Certifier qu'une base est réduite

Nous nous plaçons dans le contexte où le calcul approché est utilisé dans un algorithme pour calculer un résultat ou une propriété exacte. Afin d'assurer que l'algorithme soit correct, une précision suffisante pour les calculs peut être déterminée *a priori* en fonction des dimensions et de la taille des entrées. C'est par exemple le champ d'application des théorèmes 4 et 6.

Il existe cependant de nombreuses situations où la précision n'est pas complètement maîtrisée. C'est bien sûr le cas du calcul purement numérique. C'est aussi le cas où – par souci d'efficacité – la précision est volontairement limitée, et que l'on transforme un algorithme correct en heuristique. Nous nous référons par exemple aux différents modes d'exécution de L^2 (Stehlé, 2009). Cela peut être le cas aussi quand on cherche à déterminer une précision suffisante de manière dynamique, en augmentant progressivement la précision des calculs. Dans toutes ces situations se pose la question de certifier un résultat ou une propriété, c'est-à-dire d'être certain que la précision utilisée ait été suffisante. Le certificat doit être rapide pour ne pas pénaliser l'algorithme dans lequel il est appelé, ou l'heuristique qu'il complète.

Cette certification – ou preuve – de quantités ou de propriétés est l'objet du *calcul auto-validant* à la (Rump, 2005) que nous appliquons ici au test de LLL-réduction. La standardisation IEEE (voir section 3) permet de développer des certificats de propriétés ou des algorithmes de calcul de bornes d'erreur qui utilisent uniquement le calcul flottant. Ces derniers peuvent se révéler du même ordre de coût que le calcul numérique, et donner des réponses satisfaisantes pour une large plage d'entrées. Le lecteur pourra se référer par exemple à (Oishi *et al.*, 2002) pour vérifier la solution d'un système linéaire, ou à (Rump, 2006) pour vérifier qu'une matrice est définie positive.

Nous exposons ici les principes du certificat de LLL-réduction proposé dans (Villard, 2007a) qui s'exécute en $8nd^2 + \frac{10}{3}d^3 + O(nd)$ opérations flottantes. Suivant d et n, ce certificat ne coûte donc qu'entre quatre et neuf fois plus qu'un calcul de R par factorisation QR numérique de type Householder, et aura un temps négligeable dans un algorithme de LLL-réduction. Si la base en entrée est réduite alors le certificat retourne « oui » en général. Si la base n'est pas réduite, ou si la précision flottante utilisée par le certificat n'est pas suffisante relativement aux dimensions et aux propriétés numériques de la base, alors le certificat retourne « décision impossible ». Ainsi, le certificat peut ne pas aboutir mais ne renvoie jamais de réponse erronée. Nous avons vu que la réduction se définit à partir des coefficients de l'orthogonalisation. Le certificat se décompose en un calcul de bornes d'erreurs sur ces derniers, puis en la combinaison de ces bornes pour prendre une décision.

4.1. *Bornes d'erreur pour le facteur R de la factorisation QR*

Étant donné une matrice $B \in \mathbb{R}^{n\times d}$ de rang d, et une matrice triangulaire supérieure inversible $\widetilde{R} \in \mathbb{R}^{d\times d}$, on se pose la question de borner la distance entre $\widetilde{R}$ et le

facteur R de B par le calcul d'une matrice $H \in \mathbb{R}^{d\times d}$ qui vérifie $|\widetilde{R} - R| \leq H|\widetilde{R}|$. Les valeurs absolues et l'inégalité – ici et dans ce qui suit – sont prises coefficient par coefficient. La matrice H fournit donc une borne dite *componentwise* qui, si $\widetilde{R}$ est le résultat d'un algorithme approché de factorisation QR, donne une borne sur l'erreur directe de l'algorithme. L'intérêt de l'approche – c'est celui du calcul auto-validant – est de s'appliquer indépendamment de la façon dont $\widetilde{R}$ est calculée[3]. Pour que la borne d'erreur soit intéressante elle doit se fonder sur une analyse de perturbation fine. En étendant celle de (Sun, 1992) nous arrivons au résultat suivant.

Théorème 7 *(Villard, 2007a) Soit $B \in \mathbb{R}^{n\times d}$ de rang d et $R \in \mathbb{R}^{d\times d}$ son facteur R. Soit $\widetilde{R} \in \mathbb{R}^{d\times d}$ une matrice triangulaire supérieure inversible et posons*

$$G = |(\widetilde{R}^t)^{-1}B^tB\widetilde{R}^{-1} - I|.$$

Alors, si le rayon spectral $\rho(G)$ de G (plus grand module de valeur propre) vérifie $\rho(G) < 1$, on a :

$$|\Delta R| = |\widetilde{R} - R| \leq \text{triu}\left(G(I-G)^{-1}\right)|\widetilde{R}|$$

où triu(M) *d'une matrice M est sa partie triangulaire supérieure.*

Remarquons que ce théorème donne une borne qui va dépendre de B et de la matrice $\widetilde{R}$ effectivement calculée. Il va ainsi conduire en général à des résultats meilleurs que ceux qui pourraient être déduits des bornes de perturbation au pire et en norme des théorème 4 et 6. Afin de développer un algorithme de calcul de l'erreur à partir du théorème 7 on voit qu'il faut aussi manipuler l'inversion de matrices, puisque $\widetilde{R}^{-1}$ et $(I-G)^{-1}$ interviennent. En nous inspirant de (Oishi *et al.*, 2002) nous montrons que l'on peut se ramener au cas de l'inversion de matrices de type $I - M$. Si $\rho(M) < 1$, alors $I - M$ est inversible et on peut utiliser que

$$|(I-M)^{-1}| \leq |I + M| + \mathbf{1} \cdot \frac{\|M\|_\infty^2}{1 - \|M\|_\infty}$$

après développement en série, et où $\mathbf{1}$ est la matrice dont tous les coefficients sont égaux à 1 avec les dimensions appropriées (Villard, 2007a). Notons que puisque $\rho(M) \leq \|M\|$ pour n'importe quelle norme matricielle consistante (voir (Higham, 2002, chap. 6)), tester l'inversibilité de $I - M$ peut se faire en testant si $\|M\|_\infty < 1$. Vérifier les hypothèses du théorème 7 et majorer l'erreur se ramène ainsi essentiellement à majorer des additions, des produits, ou des normes infinies de matrices.

4.2. *Calcul des bornes d'erreur sous le standard IEEE*

Grâce au standard IEEE, on peut utiliser le changement d'arrondi pour majorer des expressions arithmétiques en utilisant le calcul en nombres flottants. Par exemple,

3. Voir les observations de (Rump, 2005) quant à une analyse par intervalles directe qui serait très peu productive dans notre contexte.

pour a et b deux nombres flottants, une borne r sur $|a \text{ op } b|$ où op $\in \{+, -, \times, \div\}$ peut se calculer à l'aide du (pseudo-)programme

$$\begin{array}{ll} \texttt{fixer-arrondi(haut);} & \overline{r} := \diamond(a \text{ op } b); \\ \texttt{fixer-arrondi(bas);} & \underline{r} := \diamond(a \text{ op } b); \quad r := \max\{|\overline{r}|, |\underline{r}|\}; \end{array}$$

où l'instruction `fixer-arrondi` détermine le mode d'arrondi pour toutes les instructions qui suivent jusqu'à un prochain appel. Pour a et b vus comme des réels, le flottant – donc réel – r est une borne rigoureuse sur le résultat de op. Cette démarche s'étend aux opérations scalaires et matricielles dont nous avons besoin (Rump, 2005), et, à partir du théorème 7, nous permet obtenir un programme de majoration de $|\Delta R|$ en calcul flottant. Son coût, étant donné B et $\widetilde{R}$ au format flottant, est de $6nd^2 + 4d^3 + O(nd)$ opérations (Villard, 2007a).

4.3. *Certificat de LLL-réduction*

Une base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ par exemple dans $\mathbb{Z}^n$ dont on veut tester la LLL-réduction, correspond une matrice B à coefficients flottants. Afin de prendre en compte la conversion du domaine de la base vers les flottants, il faut introduire un petit intervalle d'incertitude autour de B. Le certificat – alors purement numérique – consiste à : calculer une approximation $\widetilde{R}$ du facteur R de B ; appliquer les résultats ci-dessus pour calculer une matrice E telle de $|\widetilde{R} - R| \leq E$; tester la réduction à partir des valeurs approchées et de la borne d'erreur. Il nous reste à voir ce dernier point. Une valeur approchée et une borne d'erreur donnent un encadrement de la valeur exacte inconnue. On peut donc éventuellement certifier la (δ, η, θ)-LLL-réduction en testant si les inégalités de la définition du théorème 5 sont vraies, à partir des encadrements calculés. Par exemple, en se fondant à nouveau sur le changement de l'arrondi, la condition de Lovász

$$\sqrt{\delta - (R_{i,i+1}/R_{i,i})^2}\, R_{i,i} \leq R_{i+1,i+1}$$

peut s'écrire (en supposant δ flottant) :

$$\begin{array}{ll} \texttt{fixer-arrondi(haut);} & t_i := \diamond(\widetilde{R}_{i,i} + E_{i,i}); \\ \texttt{fixer-arrondi(bas);} & t_{i+1} := \diamond(\widetilde{R}_{i+1,i+1} - E_{i+1,i+1}); \\ & t := \diamond\left(\diamond(\diamond(|\widetilde{R}_{i,i+1}| - E_{i,i+1})/t_i)^2 - \delta\right); \\ \texttt{fixer-arrondi(haut);} & t := -t; \quad t := \diamond(\sqrt{t} \times t_i); \\ & t \leq t_{i+1}? \end{array}$$

De nombreuses questions doivent être prises en compte en pratique, notamment pour l'utilisation de fonctionnalités rapides en algèbre linéaire. Néanmoins le surcoût par rapport au calcul de E est avant tout celui d'une orthogonalisation, ce qui conduit à un coût global de $8nd^2 + \frac{10}{3}d^3 + O(nd)$.

Nous nous référons à (Villard, 2007a; Villard, 2007b) pour des résultats expérimentaux du calcul d'erreur et du certificat. En pouvant certifier la réduction de bases pour $d = n$ jusqu'à 300 (de type sac à dos, voir (Nguyen *et al.*, 2006)) avec des flottants double précision ($p = 53$ bits), nous montrons la pertinence de l'approche en comparaison des bornes théoriques comme celles des théorèmes 4 et 6. Étudier de manière plus générale, en fonction des dimensions, la précision suffisante afin que le certificat détecte systématiquement les bases réduites reste cependant une question entière.

5. Améliorer la réduction d'une base réduite

La qualité d'une base LLL-réduite dépend de la valeur des paramètres de réduction choisis (voir théorème 5). Le facteur α – fonction de δ, η, et θ – guide les longueurs respectives des vecteurs et en particulier, plus la valeur de δ est proche de 1, meilleure est la réduction. Toutefois, l'analyse dans le pire cas de l'algorithme LLL indique que choisir une valeur élevée pour δ rend le processus de réduction plus lent, le nombre d'itérations dépend en effet de $\log_{1/\delta} \|B\|$ (Lenstra *et al.*, 1982). C'est pourquoi (LaMacchia, 1991, p. 70), reprenant une idée de Dake He, a proposé de procéder à la réduction de manière progressive. L'idée est d'effectuer la majeure partie de la réduction lors d'un premier passage de l'algorithme LLL avec des paramètres $(\delta_0, \eta_0, \theta_0)$ qui permettent une réduction rapide, mais n'assurent qu'une qualité modérée en sortie. Puis l'algorithme LLL est à nouveau employé sur cette base, qui est donc déjà LLL-réduite, mais avec des paramètres plus contraignants $(\delta_1, \eta_1, \theta_1)$. Les paramètres $(\delta_0, \eta_0, \theta_0)$ et $(\delta_1, \eta_1, \theta_1)$ définissent deux facteurs α_0 et α_1 dans le théorème 5, et $\alpha_1 < \alpha_0$ signifie que la réduction est plus forte.

On s'intéresse ici au deuxième passage de l'algorithme. Le fait de savoir que la base en entrée du deuxième appel à LLL est LLL-réduite donne des informations sur le comportement des longueurs des orthogonalisés de Gram-Schmidt qui permettent d'accélérer la réduction. Nous présentons ici un algorithme qui effectue cette deuxième réduction, dont la borne de complexité, de même exposant total que LLL, dépend quasi-exclusivement de la dimension d (et donc moins de la taille $\log \|B\|$ des entrées). On note $\mathcal{L}(d, n, \beta)$ le coût général de la réduction d'une base de d vecteurs de $\mathbb{Z}^n$ avec $\beta = \log \|B\|$, en incluant le calcul de la matrice de transformation. Afin de simplifier l'écriture de l'estimation de complexité, on supposera ici que le nombre d de vecteurs n'est pas trop petit devant n, à savoir que $d = \Omega(\log n)$. Sans cette hypothèse, des termes en $d + \log n$ apparaissent.

Théorème 8 *(Morel* et al.*, 2009) Soit $B \in \mathbb{Z}^{n \times d}$ une base $(\delta_0, \eta_0, \theta_0)$-LLL-réduite avec comme premier jeu de paramètres*

$$\eta_0 \in [1/2, 1[,\ \theta_0 \in [0, 2^{-\Omega(d)}[,\ \delta_0 \in]\eta_0^2, 1[.$$

Pour l'amélioration de la qualité, soit le jeu de paramètres cible

$$\eta_1 \in]1/2 + 2^{-O(d)}, 1[,\ \theta_1 \in]\Omega(1), 1 - \eta_1[,\ \delta_1 \in]\eta_1^2, 1 - 2^{-O(d)}[.$$

Alors il existe un algorithme qui calcule en $O(\mathcal{L}(d,n,d) + nd\mathcal{M}(d)\log\|B\|)$ opérations élémentaires une base $(\delta_1,\eta_1,\theta_1)$-LLL-réduite du réseau engendré par B.

Nous allons voir que le coût donné pour l'amélioration de la réduction à partir de B, correspond essentiellement au coût d'une réduction LLL pour une base $\widehat{B}$ telle que $\log\|\widehat{B}\| = O(d)$ (donnant lieu au terme en $\mathcal{L}(d,n,d)$), avec un produit additionnel de B par une matrice $d \times d$. Il est requis que θ_0 soit raisonnablement petit afin de pouvoir borner efficacement la norme de la transformation nécessaire pour réduire la base. Les paramètres optimaux ($\delta_1 = 1, \eta_1 = 1/2$ et $\theta_1 = 0$) ne peuvent être atteints qu'avec une relaxation liée à la précision, linéaire en d, utilisée dans l'algorithme. De plus, la mise à l'échelle employée dans le *cas général* (voir ci-dessous) rend nécessaire de restreindre davantage le champ des valeurs de θ_1.

5.1. *Description de l'algorithme*

Si l'on tronque les données en entrée, nous avons vu que l'analyse de perturbation de R donnée au théorème 4 indique une perte de précision au plus linéaire en d pour une base initialement réduite. Cela nous permet d'appliquer le principe suivant. On approche la base d'entrée B ($(\delta_0,\eta_0,\theta_0)$-LLL réduite) par une base $\widetilde{B}$ d'un réseau proche de celui représenté par B, en ne gardant que les $\ell = \Theta(d)$ premiers bits de chaque vecteur de B. C'est-à-dire que l'on a

$$\widetilde{\mathbf{b}}_i = 2^{\ell}\lfloor 2^{-\ell}\mathbf{b}_i \rfloor, \text{ et donc : } \|\widetilde{\mathbf{b}}_i - \mathbf{b}_i\| \leq 2^{-\Theta(d)}\|\mathbf{b}_i\|.$$

Ensuite les colonnes de $\widetilde{B}$ sont mises à l'échelle (voir ci-après), afin que les vecteurs de la base $\widehat{B}$ ainsi formée soient de normes $2^{O(d)}$ (on rappelle l'hypothèse $d = \Omega(\log n)$).

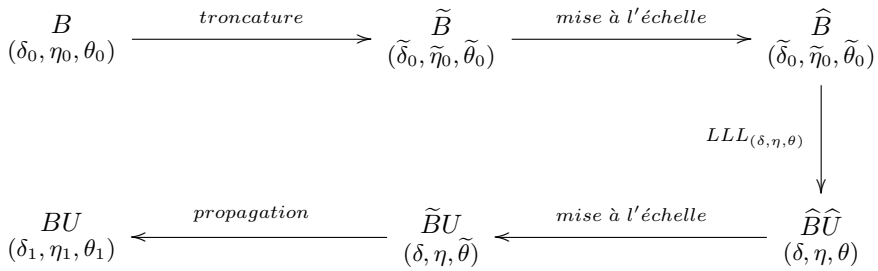

Figure 2. *Les différentes étapes de l'algorithme 3*

L'algorithme LLL est alors appelé sur cette base $\widehat{B}$ avec des paramètres (δ,η,θ) légèrement plus forts que les paramètres cibles $(\delta_1,\eta_1,\theta_1)$. L'utilisation de paramètres « plus forts » est nécessaire ici car la qualité de la réduction va être affaiblie par l'utilisation du théorème 4 pour revenir au réseau initial. Cette dernière étape s'effectue en

rétablissant les différences d'échelle à partir de la transformation unimodulaire $\widehat{U}$ qui correspond à LLL avec $\widehat{B}$ en entrée. Cela fournit une matrice de transformation U qui est finalement appliquée a la base initiale B.

Entrée : Une base $(\delta_0, \eta_0, \theta_0)$-LLL réduite B d'un réseau L
et des paramètres cibles $(\delta_1, \eta_1, \theta_1)$.
Sortie : Une base $(\delta_1, \eta_1, \theta_1)$-LLL réduite du réseau L.

1. Approcher les coefficients de B pour former la base $\widetilde{B}$.
2. Mettre à l'échelle les coefficients de $\widetilde{B}$ pour former la base $\widehat{B}$.
3. $LLL_{(\delta,\eta,\theta)}(\widehat{B})$, dont calcul de la transformation $\widehat{U}$, avec $\delta > \delta_1$, $\eta < \eta_1$ et $\theta < \theta_1$.
4. Mettre à l'échelle les coefficients de $\widehat{U}$ pour obtenir la transformation U.
5. Renvoyer la base BU.

Algorithme 3. *Algorithme améliorant la réduction d'une base réduite*

5.2. *Mise à l'échelle*

Dans le but de diminuer la contribution de la taille des entrées au coût de la réduction, on ne conserve d'abord que les $\Theta(d)$ bits les plus forts des vecteurs de B pour former la matrice $\widetilde{B}$. On remarque que la réduction n'est pas perdue pour $\widetilde{B}$, qui est $(\widetilde{\delta}_0, \widetilde{\eta}_0, \widetilde{\theta}_0)$-LLL-réduite. Comme on souhaite effectuer la réduction de l'étape 3 plus rapidement que dans le cas général de la réduction, on procède ensuite à une mise à l'échelle des coefficients pour obtenir des entiers de longueur $O(d)$. On distingue alors deux cas : dans le *cas dit générique*, où tous les vecteurs sont de tailles similaires, plus précisement

$$\forall i \ : \ \|\mathbf{b}_i\| > 2^{-O(d)} \cdot \max_i \|\mathbf{b}_i\|,$$

cette mise à l'échelle consiste à diviser tous les vecteurs par une puissance de 2 commune pour obtenir $\widehat{B}$ possédant les mêmes propriétés de réduction. Dans le *cas général*, il est nécessaire d'adapter la mise à l'échelle à la taille des différents vecteurs tout en conservant une structure de base similaire à celle de la base d'origine.

5.3. *Cas générique*

Dans le cas générique, tous les vecteurs de $\widetilde{B}$ étant de même magnitude, il est possible d'effectuer une mise à l'échelle globale en divisant tous les vecteurs par une même puissance de 2. Il n'y a ainsi aucune perte d'information entre $\widetilde{B}$ et $\widehat{B}$. Puisque maintenant on a $\log \|\widehat{B}\| = O(d)$, le coût de la réduction de $\widehat{B}$, qui peut s'effectuer en utilisant n'importe quelle variante de l'algorithme LLL, ne dépend que de la dimension d. On peut établir que les coefficients de la transformation $\widehat{U}$ qui en découle sont

de taille $2^{O(d)}$. Ce dernier point se fonde sur les propriétés données au théorème 5 qui permettent de majorer les coefficients de la base avant et après la réduction. On peut aussi remarquer que ces mêmes propriétés permettent de majorer plus efficacement le nombre d'itérations effectuées par l'algorithme LLL sur une base déjà réduite. On relie pour cela les longueurs des orthonogalisés, avant et après réduction, aux minima successifs du réseau.

Comme tous les vecteurs de $\widetilde{B}$ ont été divisés par une même puissance de 2 pour former $\widehat{B}$, utiliser $U = \widehat{U}$ est suffisant pour que $\widetilde{B}U$ soit (δ, η, θ)-LLL réduite. La transformation U est alors appliquée à la base initiale. La borne sur sa taille permet de prouver que BU et $\widetilde{B}U$ sont suffisamment proches pour que l'analyse de perturbation de la factorisation QR (théorème 4) permette de garantir la réduction de BU à partir de celle de $\widetilde{B}U$.

5.4. *Cas général*

Dans le cas général les magnitudes des différents vecteurs peuvent être très variables. Il peut s'avérer impossible de diviser l'ensemble des vecteurs par une même puissance de 2, pour obtenir des vecteurs de norme $2^{O(d)}$, tout en conservant suffisamment d'information de réduction. Par exemple, il se pourrait que même le rang ne soit pas conservé.

Toutefois, comme la base B fournie en entrée est LLL-réduite, on sait que les vecteurs de la base sont relativement proches en longueur des orthogonalisés de Gram-Schmidt. Or, dans une base LLL-réduite, on sait aussi que la décroissance de la longueur des orthogonalisés est bornée. Ainsi une grande variation dans les magnitudes des vecteurs indique une croissance par paliers des longueurs des orthogonalisés.

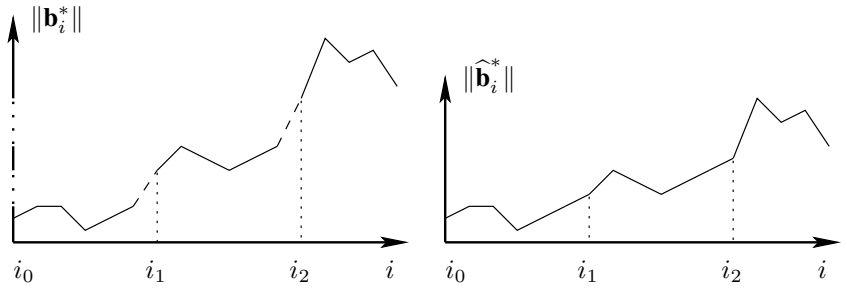

Figure 3. *Mise à l'échelle avec trois paliers*

On peut tirer parti de ce dernier aspect, puisque des $R_{i,i}$ croissants vérifient la condition de Lovász pour toute valeur de δ. On repère les limites de ces paliers de croissance des longueurs : les points à partir desquels les longueurs des orthogonalisés ne redescendent pas jusqu'au niveau qu'elles avaient atteint jusque-là. Il est alors possible d'obtenir $\widehat{B}$, qui conserve les propriétés de réduction de $\widetilde{B}$, en divisant les

vecteurs d'un même palier par une puissance de 2 commune, tout en préservant la croissance des orthogonalisés entre les paliers (voir la figure 3). La base $\widehat{B}$ ainsi formée possède la même structure que la base initiale, tout en étant composée de vecteurs de taille $2^{O(d)}$.

Une fois la transformation $\widehat{U}$ obtenue par réduction de $\widehat{B}$, il est nécessaire de lui appliquer une mise à l'échelle inverse afin de retrouver une transformation U cohérente vis-à-vis du réseau et de la base initiaux. L'étude de la structure et de la taille de cette transformation permet d'assurer, à l'aide de l'analyse de perturbation du théorème 4, les propriétés de réduction sur la base finale BU.

6. Réduire plus efficacement

Dans cette section, nous expliquons comment exploiter la stabilité numérique de la factorisation de Cholesky pour mettre au point un algorithme efficace de LLL-réduction. Pour plus de détails, nous renvoyons le lecteur à (Nguyen *et al.*, 2009).

6.1. *Une réduction incrémentale*

L'algorithme LLL fait se succéder des étapes de deux types différents. Les proprifications d'un vecteur $\mathbf{b}_i$ de la base courante permettent de rendre les magnitudes des coefficients $\mu_{i,j}$ inférieures à $1/2$ pour $j < i$. Elles conservent les vecteurs $\mathbf{b}_j^*$. Leur rôle est de garantir que les longueurs des vecteurs des bases courantes ne croissent pas trop lors de l'exécution de l'algorithme. Les autres étapes sont les échanges de deux vecteurs consécutifs $\mathbf{b}_i$ et $\mathbf{b}_{i+1}$ de la base courante. Ce sont ces étapes qui permettent des faire progresser la base courante vers une base réduite. Les échanges sont effectués lorsque la condition de Lovász $\delta\|\mathbf{b}_i^*\|^2 \leq \|\mathbf{b}_{i+1}^*\|^2 + \mu_{i+1,i}^2\|\mathbf{b}_i^*\|^2$ est violée. Clairement, si plus aucune telle étape n'est applicable, alors la base courante est LLL-réduite. Par ailleurs, si un échange entre deux vecteurs $\mathbf{b}_i$ et $\mathbf{b}_{i+1}$ qui violaient la condition de Lovász est effectué, alors la quantité $(\|\mathbf{b}_1^*\|, \ldots, \|\mathbf{b}_d^*\|)$ décroît pour l'ordre lexicographique. Cela garantit qu'on ne peut appliquer qu'un nombre fini de fois un échange. Dans (Lenstra *et al.*, 1982), les auteurs observent qu'en fait la quantité $\Delta = \prod_{i \leq d} \|\mathbf{b}_i^*\|^{2(d-i+1)}$ est un entier et décroît d'un facteur $\geq \delta$ lors d'un échange. Avec la maîtrise des tailles provenant des étapes de proprification, cela leur a permis de montrer que l'algorithme LLL finit en temps polynomial en la taille de la donnée.

Dans cette description informelle de l'algorithme LLL, si plusieurs échanges sont possibles à un moment donné, le choix de l'échange à effectuer n'a pas d'importance. Cela est vrai en arithmétique rationnelle, car toutes les quantités sont calculées exactement et la quantité Δ décroît d'un facteur minimum identique. Le choix de l'échange à effectuer prend de l'importance si l'on utilise des approximations, comme par exemple en arithmétique flottante, pour l'orthogonalisation de Gram-Schmidt. Si l'échange choisi $\mathbf{b}_i \leftrightarrow \mathbf{b}_{i+1}$ n'est pas le premier possible, alors les vecteurs précé-

dents $\mathbf{b}_1, \ldots, \mathbf{b}_i$ ne sont pas LLL-réduits. Les longueurs $\|\mathbf{b}_l^*\| = R_{l,l}$ pour $l \leq i$ peuvent donc décroître assez vite, ce qui signifie que la matrice des vecteurs $\mathbf{b}_1, \ldots, \mathbf{b}_i$ peut être mal conditionnée vis-à-vis du calcul des coefficients de Gram-Schmidt (voir la section 3). Dans cette situation, rien ne garantit une quelconque correction des coefficients de Gram-Schmidt si ceux-ci ont été calculés avec une faible précision, typiquement de l'ordre de d bits. Comme vu après l'énoncé du théorème 4, une borne sur le nombre de bits de précision perdus est en effet

$$\approx d + \log \prod_{\substack{j < i \\ R_{j,j} > R_{j+1,j+1}}} \frac{R_{j,j}}{R_{j+1,j+1}},$$

ce qui peut être de l'ordre de $d + \log \|B\|$ si la base n'est pas réduite. Dans les algorithmes LLL flottants (Nguyen *et al.*, 2009; Koy *et al.*, 2001b; Schnorr, 2006), les auteurs choisissent le premier échange possible. Cette stratégie incrémentale permet de garantir que lorsque l'on considère le vecteur $\mathbf{b}_i$ pour une proprification ou un échange, les vecteurs $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$ sont LLL-réduits. L'algorithme LLL incrémental est décrit informellement dans l'algorithme 4.

Entrée : Une base $\mathbf{b}_1, \ldots, \mathbf{b}_d$ d'un réseau L, un facteur $\delta \in]1/4, 1[$.
Sortie : Une base LLL-réduite de L.

1. i:=2. Tant que $i \leq d$,
2. Proprifier $\mathbf{b}_i$ vis-à-vis de $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$.
3. Si $\mathbf{b}_{i-1}$ et $\mathbf{b}_i$ satisfont la condition de Lovász pour le facteur δ, alors i:=$i+1$.
4. Sinon, échanger $\mathbf{b}_{i-1}$ et $\mathbf{b}_i$, et i:=$\max(2, i-1)$.
5. Renvoyer $\mathbf{b}_1, \ldots, \mathbf{b}_d$.

Algorithme 4. *L'algorithme LLL incrémental*

Le correction de l'algorithme avec cette stratégie incrémentale ne peut cependant pas reposer uniquement sur une analyse d'erreur des coefficients de Gram-Schmidt pour une base LLL-réduite. En effet, pour l'indice de boucle i, le vecteur $\mathbf{b}_i$ considéré peut ne pas être LLL-réduit vis-à-vis des précédents. (Nguyen *et al.*, 2009) donnent une généralisation du théorème 6 pour le cas de vecteurs LLL-réduits sauf le dernier.

Théorème 9 *(Nguyen* et al.*, 2009) Soit $\eta \in [1/2, 1[$ et $\delta \in]\eta^2, 1[$. Soit d vecteurs linéairement indépendants $\mathbf{b}_1, \ldots, \mathbf{b}_d \in \mathbb{R}^n$ et $\mu_{i,j}$ les coefficients de leur orthogonalisation de Gram-Schmidt. Soit $i \leq d$. Supposons que les vecteurs $(\mathbf{b}_j)_{j<i}$ soient $(\delta, \eta, 0)$-LLL-réduits, et que leur matrice de Gram soit connue à l'ulp près. Supposons également que les calculs soient flottants avec une précision p qui satisfasse $d^2\gamma^d 2^{-p+5} \leq \epsilon$, avec $\gamma = \frac{(1+\eta)^2+\epsilon}{\delta-\eta^2}$ et $\epsilon \in]0, 1/2]$. En notant $\Delta\mu_{i,j}$ les différences entre les quantités exactes et celles renvoyées par l'algorithme 2 en précision p, alors on a pour tout $j < i$:*

$$|\Delta\mu_{i,j}| \leq 32d\gamma^i 2^{-p} \cdot \max_{j<i} |\mu_{i,j}|.$$

6.2. *Une proprification paresseuse*

Le théorème 9 affirme que si l'on effectue des calculs flottants avec une précision $c \cdot d$ pour une constante c suffisamment grande, alors une approximation $(\widetilde{\mu}_{i,1}, \ldots, \widetilde{\mu}_{i,i-1})$ pertinente du vecteur $(\mu_{i,1}, \ldots, \mu_{i,i-1})$ est connue. En connaissant les bits de poids fort des $\mu_{i,j}$, on peut essayer de proprifier le vecteur $\mathbf{b}_i$. L'algorithme idéal de proprification est décrit dans l'algorithme 5.

Entrée : Des vecteurs $\mathbf{b}_1, \ldots, \mathbf{b}_i$ linéairement indépendants.
Sortie : $\mathbf{b}'_i = \mathbf{b}_i - \sum_{j<i} x_j \mathbf{b}_j$ avec des x_j entiers et $\mathbf{b}'_i$ propre par rapport à $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$.

1. Pour j de $i-1$ à 1
2. $\quad x_j := \lfloor \mu_{i,j} \rceil$.
3. $\quad \mathbf{b}_i := \mathbf{b}_i - x_j \mathbf{b}_j$.
4. $\quad$ Pour k de 1 à j, $\mu_{i,k} := \mu_{i,k} - x_j \mu_{j,k}$.
5. Renvoyer le vecteur $\mathbf{b}_i$ courant.

Algorithme 5. *Proprification du vecteur* $\mathbf{b}_i$

Si cet algorithme idéal de proprification est appliqué avec les coefficients de Gram-Schmidt approchés $\widetilde{\mu}_{i,j}$, alors les bits les plus significatifs des entiers x_j sont corrects. Ainsi, au lieu d'« annuler » les parties entières des coefficients $\mu_{i,j}$, l'algorithme flottant a seulement rendu les $\mu_{i,j}$ plus petits. Plus précisément, supposons les vecteurs $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$ LLL-réduits, que la précision des calculs est de $c \cdot d$ bits pour une grande constante c et que les coefficients de Gram-Schmidt ont été calculés avec l'algorithme de factorisation de Cholesky (c'est-à-dire par l'approche de la matrice de Gram exacte). Alors $\Omega(d)$ bits significatifs du vecteur $(\mu_{i,1}, \ldots, \mu_{i,i-1})$ sont connus, et la version flottante (avec une précision de $c \cdot d$ bits) de l'algorithme 5 permet de diminuer la magnitude maximale M des $\mu_{i,j}$ de $\Omega(\min(d, \log(M+1))$ bits. Pour obtenir une proprification complète, il suffit de recalculer des $\widetilde{\mu}_{i,j}$ pour le nouveau vecteur $\mathbf{b}_i$, et de recommencer l'application de la proprification flottante jusqu'à rendre les magnitudes des $\mu_{i,j}$ suffisamment petites. Si M borne les magnitudes initiales, comme l'on gagne $\Omega(d)$ bits à chaque itération, il suffit d'appliquer la proprification flottante $O\left(1 + (\log M)/d\right)$ fois. Il s'agit d'un algorithme paresseux de proprification car on se contente d'approcher les $\mu_{i,j}$ avec d bits de précision, et on proprifie autant que l'on peut avec ces approximations. On peut aussi interpréter cette proprification paresseuse comme un procédé auto-correcteur : les premiers x_i calculés sont faux, alors on les corrige progressivement en réitérant la proprification flottante. L'algorithme de proprification paresseuse est décrit informellement dans l'algorithme 6.

Entrée : Des vecteurs $\mathbf{b}_1, \ldots, \mathbf{b}_i$ linéairement indépendants.
Leur matrice de Gram exacte. Un paramètre $\eta' > 1/2$.
Sortie : $\mathbf{b}'_i = \mathbf{b}_i - \sum_{j<i} x_j \mathbf{b}_j$ avec des x_j entiers et $\mathbf{b}'_i$ propre par rapport à $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$.

1. Calculer des approximations $\widetilde{\mu}_{j,k}$ avec l'algorithme de Cholesky (pour $k < j \leq i$), à partir de la matrice de Gram.
2. Si $\max_{j<i} |\widetilde{\mu}_{i,j}| > \eta'$, faire
3. Pour j de $i-1$ à 1
4. $x_j := \lfloor \widetilde{\mu}_{i,j} \rceil$.
5. $\mathbf{b}_i := \mathbf{b}_i - x_j \mathbf{b}_j$.
6. Pour k de 1 à j, $\widetilde{\mu}_{i,k} := \diamond(\widetilde{\mu}_{i,k} - \diamond(x_j \widetilde{\mu}_{j,k}))$.
7. Mettre à jour la matrice de Gram, et retourner à l'étape 2.
8. Tant que l'un des x_j calculés est non-nul.
9. Renvoyer le vecteur $\mathbf{b}_i$ courant.

Algorithme 6. *Proprification paresseuse du vecteur* $\mathbf{b}_i$

Il est important d'introduire un paramètre $\eta' > 1/2$ dans la proprification paresseuse, pour se prémunir de boucles infinies. En effet, outre le fait que cela n'aurait pas de sens numériquement, si l'on prend $\eta' = 1/2$, alors on peut alterner indéfiniment entre deux itérations où un $|\mu_{i,j}|$ est proche de $1/2$ et est successivement surestimé et sous-estimé. Si l'on veut obtenir une η-proprification (c'est-à-dire $|\mu_{i,j}| \leq \eta$), alors il convient de fixer $\eta' < \eta$ pour prendre en compte l'incertitude portant sur les coefficients $\mu_{i,j}$.

6.3. *La proprification paresseuse est plus efficace !*

À première vue, il pourrait sembler qu'il soit plus cher d'effectuer une proprification paresseuse en plusieurs étapes que d'effectuer la proprification en une seule fois avec une précision suffisante. En fait, le gain provient du fait que la proprification paresseuse permet de n'exiger que $O(d)$ bits de précision sur les coefficients de Gram-Schmidt des vecteurs précédents, alors que sinon il pourrait falloir $\Omega(d + \log \|B\|)$ bits de pécision. On peut montrer que la proprification paresseuse coûte $O\left(n\mathcal{M}(d)\left(1 + (\log M)/d\right)(d + \log \|B\|)\right)$ opérations élémentaires. Dans le cas le pire, on a $\log M \approx \log \|B\|$, mais cette situation ne peut se reproduire pour un nombre arbitraire d'itérations de boucle successives. Pour tirer parti de cela, il convient d'effectuer une analyse amortie, en regroupant des itérations de boucle.

L'algorithme L^2 de (Nguyen *et al.*, 2009) consiste essentiellement à remplacer les proprifications rationnelles de l'algorithme LLL par des proprifications paresseuses. Par rapport à la description de l'algorithme 4, il convient aussi de remplacer le facteur δ de l'étape 3 par un facteur $\delta' \in]\delta, 1[$ plus fort que le facteur δ exigé pour la base de sortie, pour prendre en compte l'inexactitude des coefficients de Gram-Schmidt approchés utilisés pour tester la condition de Lovász.

Finalement, pour obtenir la borne de complexité $O(d^2 n\mathcal{M}(d) \log \|B\|(d + \log \|B\|))$, il convient de sommer les coûts des proprifications paresseuses de toutes les itérations de la boucle principale de l'algorithme. Soit τ le nombre d'itérations de l'algorithme, et $M(t)$ la quantité $\max_{i<k(t)} |\mu_{k(t),i}|$ à l'itération t de la boucle. Alors le coût de l'algorithme L^2 est borné par :

$$Cn\mathcal{M}(d)(d + \log \|B\|) \cdot \sum_{t \leq \tau} \left(1 + \frac{\log M(t)}{d}\right),$$

pour une certaine constante C. Le nombre d'itérations de boucle est classiquement borné par $O(d^2 \log \|B\|)$ (voir (Lenstra *et al.*, 1982) par exemple), ce qui donne la borne suivante pour le coût de L^2 :

$$Cn\mathcal{M}(d)(d + \log \|B\|) \left(d^2 \log B + \frac{1}{d} \sum_{t \leq \tau} \log M(t)\right).$$

Borner la quantité $\sum_{t \leq \tau} \log M(t)$ est plus compliqué. La preuve de (Nguyen *et al.*, 2009) est une généralisation de l'analyse de complexité de l'algorithme d'Euclide pour calculer le pgcd.

7. Conclusion

Pour certifier et améliorer une LLL-réduction, ainsi que pour accélérer l'algorithme LLL, il est utile de remplacer les valeurs rationnelles exactes de l'orthogonalisation de Gram-Schmidt par des approximations flottantes. L'analyse de la correction des algorithmes reposant sur de telles approximations se ramène à des analyses de perturbation et de stabilité liées aux factorisations de Cholesky et QR de matrices. Ces études sont classiques en analyse numérique, mais il est difficile d'utiliser directement les résultats généraux, aussi est-il nécessaire de les améliorer pour le cas spécifique de la réduction LLL.

De nombreuses questions restent ouvertes dans la continuité des travaux que nous avons décrits dans cet article. Tout d'abord, comme suggéré par Schnorr (2006; 2009), il serait intéressant de faire reposer l'algorithme L^2 sur la factorisation QR plutôt que sur la factorisation de Cholesky. En effet, les théorèmes 4 et 6 invitent à penser que la précision requise pourrait être diminuée d'un facteur 2. Cela permettrait de réduire des réseaux de dimension double pour toute précision fixée. Des premiers éléments de réponse ont été apportés récemment par (Morel *et al.*, 2009).

Par ailleurs, les techniques développées pour la réduction LLL pourraient être adaptées aux autres algorithmes de réduction de réseaux : les algorithmes accélérant LLL (comme par exemple (Schönhage, 1984; Storjohann, 1996; Koy *et al.*, 2001a)), ou les algorithmes renvoyant des bases de meilleure qualité que celles renvoyées par LLL (par exemple ceux décrits dans (Schnorr *et al.*, 1994; Schnorr,

2006; Gama *et al.*, 2006; Gama *et al.*, 2008)). Dans cette dernière direction, nous mentionnons le récent résultat de (Pujol *et al.*, 2008), sur l'énumération flottante des vecteurs les plus courts d'un réseau euclidien, qui est au cœur des algorithmes réduisant plus fortement que LLL.

Un autre problème lié aux techniques que nous avons décrites est la stabilité de l'algorithme LLL lui-même : si deux bases sont proches, à quel point l'algorithme LLL se comporte-t-il de manière similaire ? Cette question soulevée pour la première fois par (Buchmann, 1994) est importante quand le réseau à réduire ne peut être connu qu'approximativement. Cela est le cas par exemple en théorie des communications (Mow, 1994; Hassibi *et al.*, 1998), mais aussi pour certains problèmes en théorie algorithmique des nombres, dont le calcul du groupe des unités de l'anneau des entiers d'un corps de nombre (Fieker *et al.*, 2006).

Remerciements

Le travail de Ivan Morel et Damien Stehlé a été financé en partie par le projet ANR Lareda, celui de Gilles Villard par le projet ANR Gecko.

8. Bibliographie

Buchmann J., « Reducing Lattice Bases by Means of Approximations », *Actes de la conférence ANTS I*, vol. 877 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 160-168, 1994.

Cannon J. J., Bosma W., « Handbook of Magma Functions, Edition 2.15 », 2008, disponible à l'url `http://magma.maths.usyd.edu.au`.

Chang X.-W., Stehlé D., Villard G., « Perturbation Analysis of the QR-factor R in the context of LLL lattice basis reduction », 2009, soumis, disponible à l'url `http://perso.ens-lyon.fr/QRPERTURB.html`.

Cohen H., *A Course in Computational Algebraic Number Theory, 2ème édition*, Springer-Verlag, 1995.

Erlingsson Ú., Kaltofen E., Musser D., « Generic Gram-Schmidt Orthogonalization by Exact Division », *Actes de la conférence ISSAC'96, Zurich, Suisse*, ACM Press, p. 275-282, 1996.

Fieker C., Pohst M., « Dependency of units in number fields », *Mathematics of Computation*, vol. 75, n° 255, p. 1507-1518, 2006.

Fousse L., Hanrot G., Lefèvre V., Pélissier P., Zimmermann P., « MPFR, a multiple-precision binary floating-point library with correct rounding », *ACM Transactions on Mathematical Software*, 2007. Disponible à l'url `http://www.mpfr.org/`.

Gama N., Howgrave-Graham N., Koy H., Nguyen P., « Rankin's Constant and Blockwise Lattice Reduction », *Actes de la conférence Crypto 2006*, n° 4117 in *Lecture Notes in Computer Science*, Springer-Verlag, p. 112-130, 2006.

Gama N., Nguyen P., « Finding short lattice vectors within Mordell's inequality », *Actes de la conférence STOC 2008*, ACM Press, p. 207-216, 2008.

Hanrot G., « LLL: A tool for effective Diophantine approximation », 2009, *Actes de la conférence LLL+25, Caen, France*, à paraître.

Hassibi A., Boyd S., « Integer Parameter Estimation in Linear Models with Applications to GPS », *IEEE Transactions on signal processing*, vol. 46, n° 11, p. 2938-2952, 1998.

Higham N., *Accuracy and Stability of Numerical Algorithms*, Society for Industrial and Applied Mathematics, 2002.

IEEE, « IEEE Standard 754-2008 for Floating-Point Arithmetic », 2008, IEEE Society Press.

Kaltofen E., « On the complexity of finding short vectors in integer lattices », *Actes de la conférence EUROCAL'83*, vol. 162 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 236-244, 1983.

Kannan R., Lenstra A. K., Lovász L., « Polynomial Factorization and Nonrandomness of Bits of Algebraic and Some Transcendental Numbers », *Proceedings of STOC 1984*, ACM Press, p. 191-200, 1984.

Khot S., « Hardness of approximating the shortest vector problem in lattices », *Actes de la conférence FOCS 2004*, IEEE Computer Society Press, p. 126-135, 2004.

Koy H., Schnorr C. P., « Segment LLL-reduction of lattice bases », *Actes de la conférence CALC'01*, vol. 2146 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 67-80, 2001a.

Koy H., Schnorr C. P., « Segment LLL-reduction of lattice bases with floating-point orthogonalization », *Actes de la conférence CALC'01*, vol. 2146 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 81-96, 2001b.

LaMacchia B. A., « Basis Reduction Algorithms and Subset Sum Problems », 1991, SM Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA.

Lenstra A. K., Lenstra Jr. H. W., Lovász L., « Factoring polynomials with rational coefficients », *Mathematische Annalen*, vol. 261, p. 513-534, 1982.

May A., « Using LLL-reduction for solving RSA and factorization problems: a survey », 2009, *Actes de la conférence LLL+25, Caen, France*, à paraître.

Merkle R., Hellman M., « Hiding Information and Signatures in Trapdoor Knapsacks », *IEEE Transactions on Information Theory*, vol. 24, n° 5, p. 525-530, 1978.

Micciancio D., Goldwasser S., *Complexity of lattice problems: a cryptographic perspective*, Kluwer Academic Press, 2002.

Minkowski H., *Geometrie der Zahlen*, Teubner-Verlag, 1896.

Morel I., Stehlé D., Villard G., « From an LLL-reduced basis to another », 2009, en préparation (*Int. Symp. Symb. Alg. Comp. 2008 Poster presentation*, Hagenberg, Autriche).

Mow W. H., « Maximum Likelihood Sequence Estimation from the Lattice Viewpoint », *IEEE Transactions on Information Theory*, vol. 40, p. 1591-1600, 1994.

Nguyen P., Stehlé D., « Floating-Point LLL Revisited », *Actes de la conférence Eurocrypt 2005*, vol. 3494 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 215-233, 2005.

Nguyen P., Stehlé D., « LLL on the average », *Actes de la conférence ANTS VII*, vol. 4076 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 238-256, 2006.

Nguyen P., Stehlé D., « An LLL algorithm with quadratic complexity », *SIAM Journal on Computing*, vol. 39, n° 3, p. 874-903, 2009.

Novocin A., Factoring Univariate Polynomials over the rationals, PhD thesis, Florida State University, 2008.

Odlyzko A. M., « Cryptanalytic Attacks on the Multiplicative Knapsack Cryptosystem and on Shamir's Fast Signature Scheme », *IEEE Transactions on Information Theory*, vol. IT-30, n° 4, p. 594-601, 1984.

Oishi S., Rump M., « Fast verification of solutions of matrix equations », *Numerische Mathematik*, vol. 90, n° 4, p. 755-773, 2002.

Pujol X., Stehlé D., « Rigorous and efficient short lattice vectors enumeration », *Actes de la conférence Asiacrypt 2008*, n° 5350 in *Lecture Notes in Computer Science*, Springer-Verlag, p. 390-405, 2008.

Rump S., « Computer-Assisted Proofs and Self-Validating Methods », *in* B. Einarsson (ed.), *Handbook of Accuracy and Reliability in Scientific Computation*, SIAM, p. 195-240, 2005.

Rump S., « Verification of positive definiteness », *BIT Numerical Mathematics*, vol. 46, p. 433-452, 2006.

Schnorr C. P., « A Hierarchy of Polynomial Lattice Basis Reduction Algorithms », *Theoretical Computer Science*, vol. 53, p. 201-224, 1987.

Schnorr C. P., « A more efficient algorithm for lattice basis reduction », *Journal of Algorithms*, vol. 9, n° 1, p. 47-62, 1988.

Schnorr C. P., « Fast LLL-type lattice reduction », *Information and Computation*, vol. 204, p. 1-25, 2006.

Schnorr C. P., « Hot Topics of LLL and lattice reduction », 2009, *Actes de la conférence LLL+25, Caen, France*, à paraître.

Schnorr C. P., Euchner M., « Lattice basis reduction : improved practical algorithms and solving subset sum problems », *Mathematics of Programming*, vol. 66, p. 181-199, 1994.

Schönhage A., « Factorization of univariate integer polynomials by Diophantine approximation and improved basis reduction algorithm », *Actes de la conférence ICALP 1984*, vol. 172 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 436-447, 1984.

Schönhage A., Strassen V., « Schnelle Multiplikation grosser Zahlen », *Computing*, vol. 7, p. 281-292, 1971.

Stehlé D., Algorithmique de la réduction de réseaux et application à la recherche de pires cas pour l'arrondi de fonctions mathématiques, PhD thesis, Université Henri Poincaré Nancy 1, 2005.

Stehlé D., « Floating-point LLL : theoretical and practical aspects », 2009, *Actes de la conférence LLL+25, Caen, France*, à paraître.

Stein W., « SAGE : open source mathematics software », 2009, disponible à l'url `http://www.sagemath.org/`.

Storjohann A., Faster Algorithms for Integer Lattice Basis Reduction, Technical Report n° 249, ETH-Zurich, Dpt. Comp. Sc., Zurich, Switzerland, 1996.

Sun J.-G., « Componentwise perturbation bounds for some matrix decompositions », *BIT*, vol. 32, p. 702-714, 1992.

van Hoeij M., « Factoring polynomials and 0-1 vectors », *Actes de la conférence CALC'01*, vol. 2146 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 45-50, 2001.

Villard G., « Certification of the QR factor R and of lattice basis reducedness », *Actes de la conférence ISSAC'07, Waterloo, Canada*, ACM Press, p. 361-368, 2007a.

Villard G., Certification of the QR Factor R, and of lattice basis reducedness, RR n° 2007-03, ENS Lyon, France, 2007b. http://hal.archives-ouvertes.fr/hal-00127059/en.

Article reçu le 14 janvier 2008
Accepté le 19 février 2009

Ivan Morel *effectue son doctorat en cotutelle entre l'École Normale Supérieure de Lyon et l'Université de Sydney. Ses travaux portent sur la réduction des réseaux euclidiens, et plus particulierement sur l'amélioration, en théorie comme en pratique, de l'algorithme LLL.*

Damien Stehlé *est chargé de recherche au CNRS, en mise à disposition auprès des Universités de Sydney et Macquarie (Australie). Ses travaux portent sur l'algorithmique des réseaux euclidiens, ainsi que sur ses applications en cryptologie, en arithmétique des ordinateurs et en théorie algorithmique des nombres.*

Gilles Villard *est directeur de recherche au CNRS, directeur du Laboratoire de l'Informatique du Parallélisme à Lyon (LIP). Ses recherches en calcul algébrique portent sur la complexité algorithmique, les méthodes certifiées et les environnements de programmation sûre et haute performance.*

ANNEXE POUR LE SERVICE FABRICATION

A FOURNIR PAR LES AUTEURS AVEC UN EXEMPLAIRE PAPIER
DE LEUR ARTICLE ET LE COPYRIGHT SIGNE PAR COURRIER
LE FICHIER PDF CORRESPONDANT SERA ENVOYE PAR E-MAIL

1. ARTICLE POUR LA REVUE :
RSTI - TSI. Volume 29 – n° 1/2010

2. AUTEURS :
Ivan Morel[*,**] *— Damien Stehlé*[***] *— Gilles Villard*[****,**]

3. TITRE DE L'ARTICLE :
Analyse numérique et réduction de réseaux

4. TITRE ABRÉGÉ POUR LE HAUT DE PAGE MOINS DE 40 SIGNES :
Analyse numérique et réduction de réseaux

5. DATE DE CETTE VERSION :
25 janvier 2010

6. COORDONNÉES DES AUTEURS :

– adresse postale :

[*] Université de Lyon, ÉNS de Lyon, Université de Sydney
[**] INRIA, Laboratoire LIP, 46 Allée d'Italie 69364 Lyon Cedex 07, France
ivan.morel@ens-lyon.fr

[***] CNRS, Universités de Sydney et Macquarie
Département de Mathématiques et de Statistiques (F07)
Université de Sydney NSW 2006, Australie
damien.stehle@gmail.com

[****] CNRS, Université de Lyon, ÉNS de Lyon
gilles.villard@ens-lyon.fr

– téléphone :

– télécopie :

– e-mail : ivan.morel@ens-lyon.fr, gilles.villard@ens-lyon.fr, damien.stehle@gmail.com

7. LOGICIEL UTILISÉ POUR LA PRÉPARATION DE CET ARTICLE :
LaTeX, avec le fichier de style `article-hermes2.cls`,
version 1.23 du 17/11/2005.

8. FORMULAIRE DE COPYRIGHT :
Retourner le formulaire de copyright signé par les auteurs, téléchargé sur :
`http://www.revuesonline.com`

SERVICE ÉDITORIAL – HERMES-LAVOISIER
14 rue de Provigny, F-94236 Cachan cedex
Tél. : 01-47-40-67-67
E-mail : revues@lavoisier.fr
Serveur web : http://www.revuesonline.com