L'algorithme LLL flottant, théorie et pratique

Damien STEHLÉ

Paris, 13 Janvier 2006

Travail en commun avec Phong NGUYỄN

http://www.loria.fr/~stehle/
stehle@loria.fr

Le plan de l'exposé.

1. Réseaux euclidiens et LLL-réduction.
2. L'algorithme LLL flottant.
3. Implantations de l'algorithme LLL flottant.
4. Remarques expérimentales.

1) Réseaux et LLL-réduction.

Les réseaux, c'est utile.

- Principal outil de cryptanalyse en clé publique (sacs-à-dos, RSA pour certains paramètres).
- Cryptosystèmes : NTRU, GGH, Ajtai-Dwork.
- Calcul formel : factorisation des polynômes rationnels.
- Théorie algorithmique des nombres : idéaux dans les corps de nombres, petites racines de polynômes (Coppersmith), . . .

Quelques définitions.

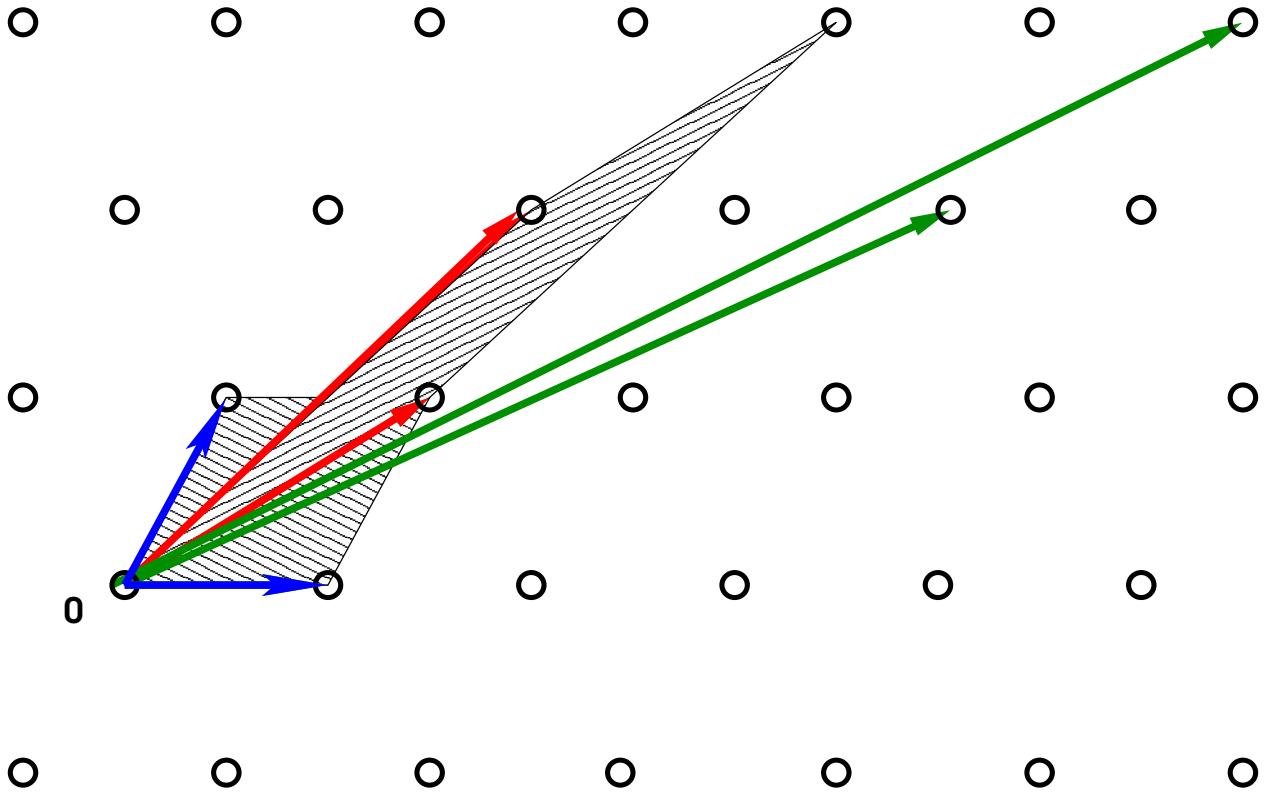
- Réseau == sous-groupe discret d'un $\mathbb{R}^n$.
- Soient $\mathbf{b}_1, \ldots, \mathbf{b}_d \in \mathbb{R}^n$ linéairement indépendants.

$$L[\mathbf{b}_1, \ldots, \mathbf{b}_d] = \left\{ \sum_{i=1}^{d} x_i \mathbf{b}_i, x_i \in \mathbb{Z} \right\}.$$

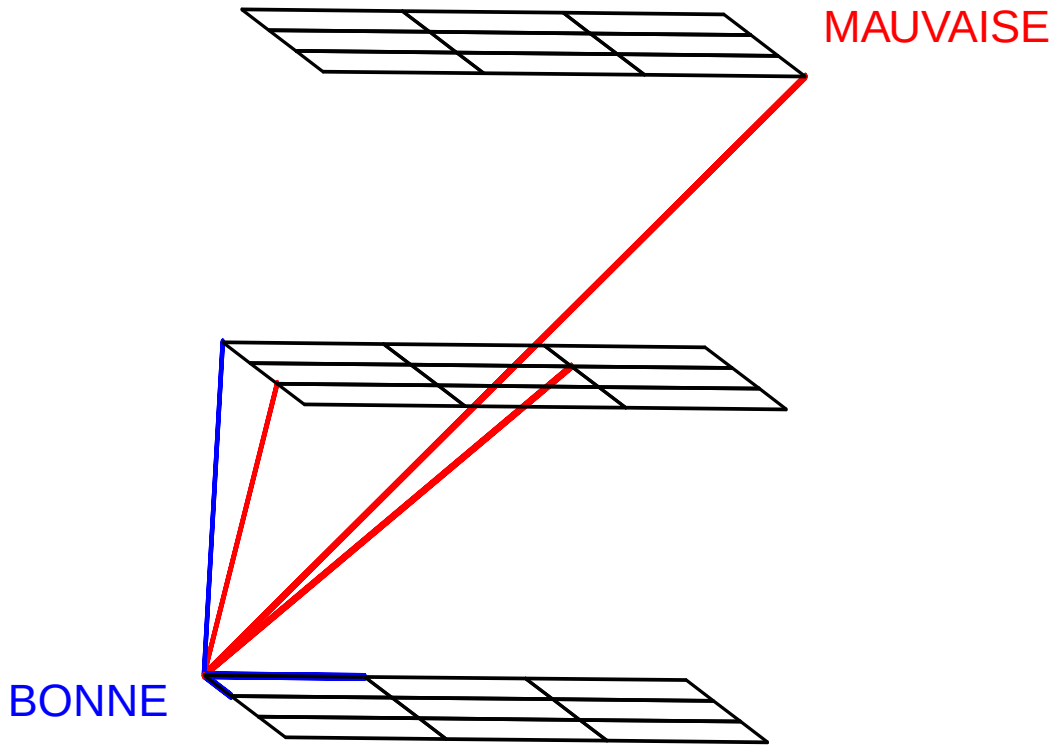
- On se limite aux sous-groupes des $\mathbb{Z}^n$ avec $d = n$.

- Matrice de Gram : $G(\mathbf{b}_1, \ldots, \mathbf{b}_d) = (\langle \mathbf{b}_i, \mathbf{b}_j \rangle)_{i,j \leqslant d}$.
- Infinité de bases, liées par des transformations unimodulaires (matrices carrées à coefficients entiers et déterminants ± 1).
- Le volume $\det L = \sqrt{\det G}$ est indépendant de la base.

Trois bases d'un réseau de dimension deux.



Deux bases d'un réseau de dimension trois.

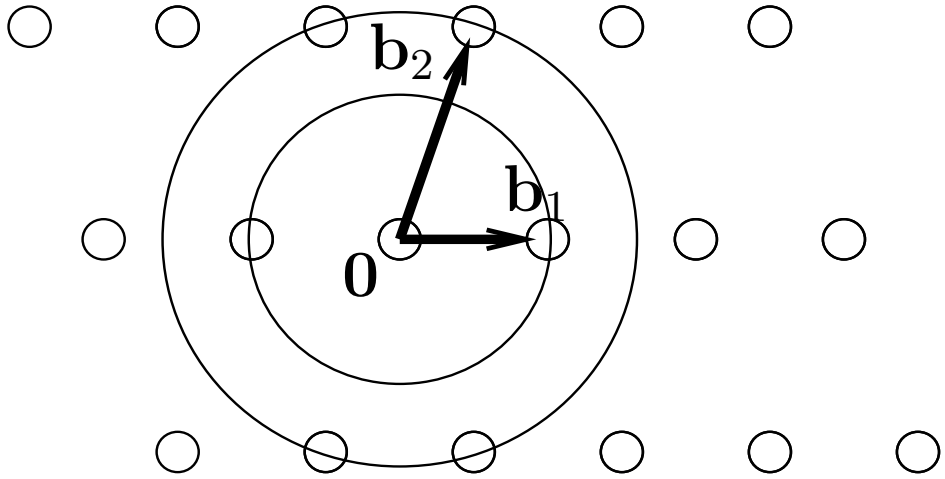


Le problème principal : trouver un vecteur court.

– $\lambda(L) =$ premier minimum = plus petite norme (non nulle).

– Théorème de Minkowski : $\lambda(L) \leqslant \sqrt{d} \cdot (\det L)^{1/d}$.

– Souvent, on se contente d'un $\mathbf{b} \in L$ tel que $\|\mathbf{b}\| \leqslant c_d \cdot \lambda(L)$ ou $\|\mathbf{b}\| \leqslant c'_d \cdot (\det L)^{1/d}$.



Exemple : découverte de petites relations linéaires entières.

– Soient $N_1, \ldots N_d, N \in \mathbb{Z}$.
On cherche $x_1, \ldots, x_d \in [|-C, C|]$ tels que $\sum_i x_i N_i = 0 \, [N]$.

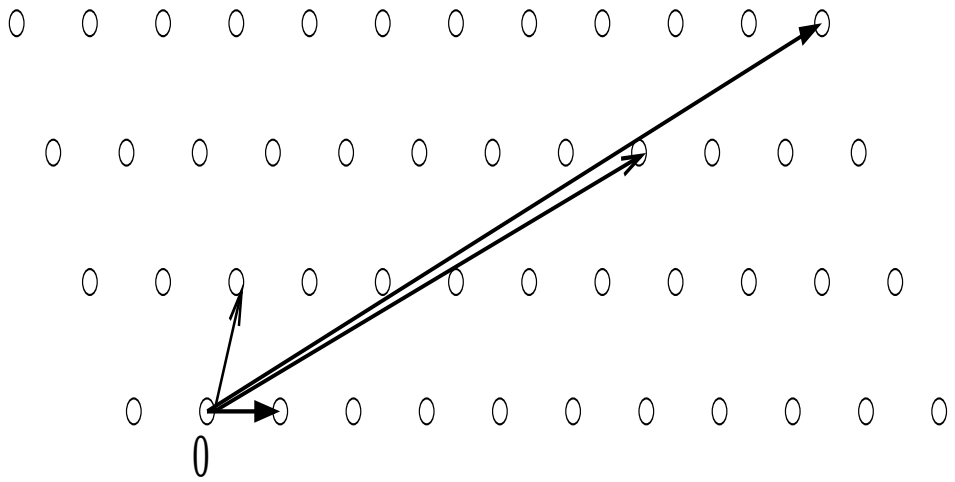
$$B = \begin{bmatrix} C \cdot N & 0 & \ldots & 0 \\ C \cdot N_1 & 1 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C \cdot N_d & 0 & \ldots & 1 \end{bmatrix}.$$

– Un vecteur de $L[B]$ de norme $< C$ donne une solution.

– $\det L = CN, \dim L = d + 1$
$\Rightarrow$ Si $\sqrt{d+1}(CN)^{1/(d+1)} < C$, il y a une solution non nulle.

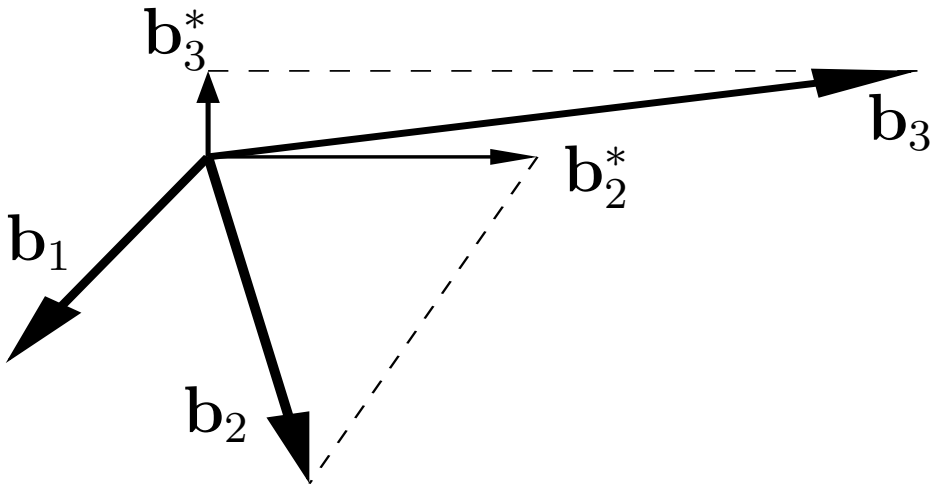
La stratégie générale : réduire le réseau.

- Principe : trouver une base avec de petits vecteurs.
- Pas vraiment de définition optimale/canonique.
- Moralement, une bonne base est plutôt orthogonale.



L'orthogonalisation de Gram-Schmidt.

- Processus itératif d'orthogonalisation de $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$.
- $\mathbf{b}_i^*$ est $\mathbf{b}_i$ projeté sur l'orthogonal de $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$.
- $\mathbf{b}_1^* = \mathbf{b}_1,\ \mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b}_j^*,\ \mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\|\mathbf{b}_j^*\|^2}$.



La proprification.

– Version « entière » de Gram-Schmidt.

– Étant donnés $(\mathbf{b}_1, \ldots, \mathbf{b}_k)$, on enlève à $\mathbf{b}_k$ une combinaison entière des précédents pour que : $\forall i < k,\ |\mu_{k,i}| \leqslant 1/2$.

1. Calculer les $\mu_{k,i}$ pour $i < k$.
2. Pour $i = (k-1)..1$,
3. $\quad x_i := \lceil \mu_{\kappa,i} \rfloor$, $\mathbf{b}_k := \mathbf{b}_\kappa - x_i \mathbf{b}_i$,
4. $\quad$ Pour $j = 1..i$, $\mu_{\kappa,j} := \mu_{\kappa,j} - x_i \mu_{i,j}$.

La LLL-réduction (1982), une réduction locale.

– $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ est LLL-réduite ssi :

1) $\forall i > j, |\mu_{i,j}| \leqslant 1/2$ [Propre].

2) $\forall i, 1^- \|\mathbf{b}_{i-1}^*\| \leqslant \|\mathbf{b}_i^* + \mu_{i,i-1}\mathbf{b}_{i-1}^*\|$ [Lovász].

– 2) signifie : dans $(\mathbf{b}_1, \ldots, \mathbf{b}_{i-2})^{orth}$, $\mathbf{b}_{i-1}$ est plus court que $\mathbf{b}_i$.

– 1) et 2) impliquent $\|\mathbf{b}_i^*\| \geqslant \sqrt{\frac{3}{4}^-} \cdot \|\mathbf{b}_{i-1}^*\|$.

Les orthogonalisés de Gram-Schmidt ne chutent pas trop vite.

Pourquoi la LLL-réduction est-elle intéressante ?

– On peut l'obtenir en temps polynomial.

– Le premier vecteur renvoyé n'est pas « trop » long :

$$\begin{aligned} \|\mathbf{b}_1\| &\leqslant (4/3+\varepsilon)^{d/2} \cdot \lambda(L), \\ \|\mathbf{b}_1\| &\leqslant (4/3+\varepsilon)^{d/4} \cdot (\det L)^{1/d}. \end{aligned}$$

L'algorithme LLL.

Entrée : $[\mathbf{b}_1, \ldots, \mathbf{b}_d]$ linéairement indépendants.

Sortie : Une base LLL-réduite de $L[\mathbf{b}_1, \ldots, \mathbf{b}_d]$.

1. [GS] Calculer les $\mu_{i,j}$ et les $\|\mathbf{b}_i^*\|^2$.
2. $\kappa := 2$. Tant que $\kappa \leq d$,
3. [Propre ?] Proprifi er $\mathbf{b}_\kappa$ par rapport à $\mathbf{b}_1, \ldots, \mathbf{b}_{\kappa-1}$.
4. [Lovász ?] Si $(1^- - \mu_{\kappa,\kappa-1}^2)\|\mathbf{b}_{\kappa-1}^*\|^2 \leq \|\mathbf{b}_\kappa^*\|^2$, alors $\kappa := \kappa + 1$.
5. Sinon échanger $\mathbf{b}_{\kappa-1}$ et $\mathbf{b}_\kappa$, $\kappa := \max(\kappa - 1, 2)$.

Analyse rapide de LLL.

- Soit $B = \max_i \|\mathbf{b}_i^{init}\|$.
- $O(d^2 \log B)$ itérations de boucle.
- Dans chaque itération, la proprification domine : $O(d^2)$ *op. ar.*
- Les entiers (ou rationnels) sont de taille $O(d \log B)$.

- Coût total : $O((d^2 \log B) \cdot d^2 \cdot (d \log B)^2) = O(d^6 \log^3 B)$.

- L'arithmétique rationnelle de l'orthogonalisation domine.

C’est beaucoup trop cher !

– Cas de la factorisation d’un module RSA $N = p \cdot q$ quand la moitié des bits de p sont connus [Coppersmith, Eurocrypt, 1996] :

$$\log B \approx 1000,\ d \approx 64, \text{« } d^6 \log^3 B = 2^{66} \text{ ».}$$

– L’arithmétique liée à l’orthogonalisation est beaucoup trop coûteuse.

– LLL n’est pas utilisé pour de grands réseaux.

2) L’algorithme LLL flottant.

L'arithmétique flottante ?

- $x = \pm m_x \cdot 2^{e_x}$, avec $m_x \in [1/2, 1[$ sur ℓ bits.
- On veut que $\diamond(a \text{ op } b)$ soit un flottant le plus proche de $(a \text{ op } b)$, pour op $\in \{+, -, *, /\}$.

- Double précision : $1 + (52 + 1) + 11$ bits, LLL_{FP, XD} de NTL.
- Précision arbitraire : MPFR, classe RR de NTL.

LLL flottant, le principe général.

– Dans LLL, les calculs liés à Gram-Schmidt dominent.

– Idée : faire les calculs de Gram-Schmidt en arithmétique flottante.
Ça concerne les $\mu_{i,j}$ et les $\|\mathbf{b}_i^*\|^2$ uniquement.

– Calculs sur la base : toujours avec des entiers.

– L'idée date du début des années '80, avec la cryptanalyse des cryptosystèmes « sacs-à-dos ».

État de l'art.

– [Schnorr, J. of A., 1988] : calcul de $(\mu_{i,j})_{i,j}^{-1}$ et itération de Schultz.
$\Rightarrow$ Précision des calculs $\approx 12d + 7\log B$.

– [Koy-Schnorr, CALC, 2001] : Givens-Householder plutôt que Gram-Schmidt. Preuve incorrecte (corrigée dans Schnorr'05).

– Schnorr'05 (non publié) : même complexité que Schnorr'88, avec une précision $\approx 5d + 2\log B$.

État de l'art, suite.

L^2 : un LLL flottant prouvé dans un modèle précis.

	LLL'82	Schnorr'88	NS'05 : L^2
Précision	$O(d \log B)$	$> 12d + 7 \log_2 B$	$d \log_2 3 \approx 1.58d$
Complexité	$O(d^6 \log^3 B)$	$O(d^4(d + \log B)^2 \log B)$	$O(d^5(d + \log B) \log B)$

Les trois idées principales de l'algorithme L^2.

– Utilisation exclusive de la matrice de Gram
(pour éviter les pertes de précision liées aux produits scalaires).
– Version paresseuse de l'algorithme de proprification
(pour éviter d'utiliser plus de bits de précision qu'il n'en faut).
– Généralisation en dimension d de la « cascade » de l'analyse de l'algorithme d'Euclide.

La proprification paresseuse.

– Proprification flottante :

1. Calculer des $\bar{\mu}_{k,i}$ pour $i < k$.
2. Pour $i = (k-1)..1$,
3. Si $|\bar{\mu}_{\kappa,i}| > 0.5^+$, $x_i := \lceil \bar{\mu}_{\kappa,i} \rfloor$, $\mathbf{b}_k := \mathbf{b}_\kappa - x_i \mathbf{b}_i$,
4. Pour $j = 1..i$, $\bar{\mu}_{\kappa,j} := \bar{\mu}_{\kappa,j} - x_i \bar{\mu}_{i,j}$.

– Prendre une précision de $(\log_2 3 + \varepsilon) \cdot d$ bits.

– Proprification paresseuse : répéter la proprification flottante tant que les x_i calculés ne sont pas nuls.

Analyse heuristique de L^2.

- $O(d^2 \log B)$ itérations de boucle.
- Dans chaque itération, la proprification domine.
- Une proprification comporte $O(d^2)$ opérations arithmétiques.
- Celles-ci sont du type $(d$ bits$) \times (\log B$ bits$)$.
- Coût total : $O(d^5 \log^2 B)$.

- L'arithmétique entière sur la base domine.

3) Implantations de l’algorithme LLL flottant.

Où trouver un LLL flottant efficace ?

– [Schnorr, Euchner, Math. of Prog., 1994].

– NTL, Magma, Pari GP (?), Lidia (?) ont des LLL flottants.

– fplll-1.2 codé en C, sous GPL (sur ma page web).

	d	$\log B$	NTL	variante rapide	variante prouvée
entrées aléatoires	750	1000	243	15	2287
sac-à-dos	100	20000	2031	368	1599

(temps en secondes, sur un Opteron 2.4 GHz)

fplll-1.2.

- Arithmétique entière : GNU MP.
- Quatre arithmétiques flottantes : double, long double, DPE, MPFR.
- Trois variantes : proved, heuristic, fast.

- NTL et fplll % Magma : libres, plus efficaces, moins conviviaux.
- fplll % NTL : plus efficace, formes quadratiques.
- NTL % fplll : autres orthogonalisations, tout ce qu'il y a autour.

Les trois variantes.

– Variante prouvée : exactement l'algorithme.

– Variante heuristique : sans maintenir la matrice de Gram (problèmes de détection des annulations de produits scalaires).

– Variante rapide : pour une ligne de d entrées, d mantisses mais un seul exposant. Problèmes plus importants avec les pdts scalaires.

4) Remarques expérimentales.

a- Temps d’exécution.

b- Qualité de la sortie.

c- « Stabilité numérique ».

Réseaux aléatoires, bases aléatoires.

– Les réseaux de déterminant 1 correspondent à $\mathrm{SL}_n(\mathbb{R})/\mathrm{SL}_n(\mathbb{Z})$.
Mesure de Haar associée de masse finie.

– [Goldstein-Mayer, F.M., 2003] Choisir un grand p premier et prendre le réseau généré par les lignes de la matrice :

$$\frac{1}{p} \cdot \begin{bmatrix} p & 0 & \dots & 0 \\ \mathrm{rdm}(0..p-1) & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{rdm}(0..p-1) & 0 & \dots & 1 \end{bmatrix}.$$

Réseaux aléatoires, bases aléatoires.

– Pour un réseau donné, comment choisir une base aléatoire ?
– [Goldreich, Goldwasser, Halevi, Crypto, 1997] : additionner des lignes aléatoires jusqu'à obtenir de grosses entrées.

– « Conjecture » : pour presque tous les réseaux, si l'on choisit une base aléatoire, le comportement pratique de LLL sera le même.
– C'est impliqué par une conjecture d'Ajtai [Ajtai, FOCS, 2002].

Temps d'exécution.

- Réseaux « d'Ajtai » : triangulaire inférieur, $B[i,i] = 2^{(d-i)^\alpha}$, $B[i,j] = \text{rdm}(-0.5, 0.5) \cdot B[j,j]$ si $i > j$.
- La borne $O(d^5 \log^2 B)$ devrait être atteinte pour ces réseaux.
- En pratique, on observe plutôt $O(d^4 \log^2 B)$.
- Dans la proprification, les $\mu_{i,j}$ croissent exponentiellement mais très lentement...

- Pour les bases « sacs-à-dos », on gagne encore un facteur d.

Qualité des bases renvoyées.

– Que valent les facteurs $\frac{\|\mathbf{b}_1\|}{(\det L)^{1/d}}$ et $\frac{\|\mathbf{b}_1\|}{\lambda(L)}$?

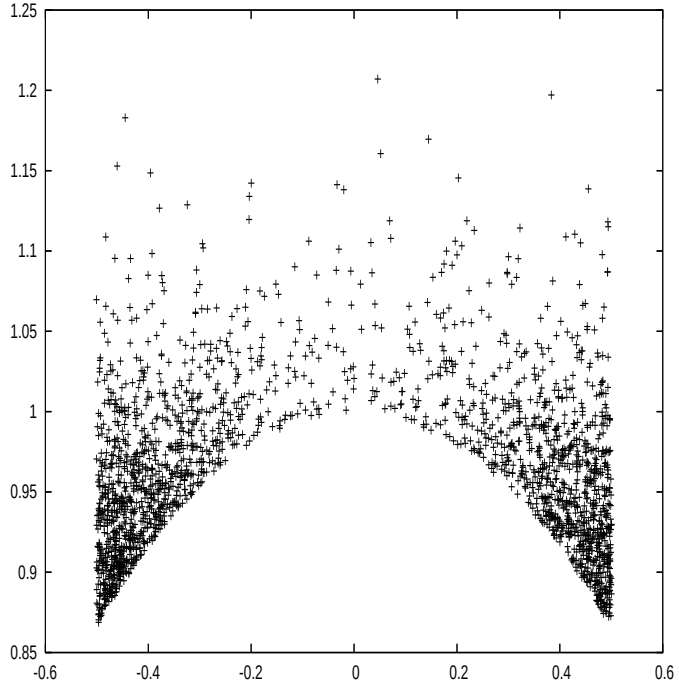
– Cas le pire : $(4/3)^{d/4}$ et $(4/3)^{d/2}$.

– Cas moyen : $(1.02)^d$ et $(1.04)^d$.

– Ces bornes proviennent des valeurs des $\frac{\|\mathbf{b}_i^*\|}{\|\mathbf{b}_{i-1}^*\|}$...

Bases locales.

Géométrie des paires de vecteurs $(\mathbf{b}_{i-1}^*, \mathbf{b}_i^* + \mu_{i,i-1}\mathbf{b}_{i-1}^*)$, avec $\mathbf{b}_1, \ldots, \mathbf{b}_d$ une base renvoyée par LLL.



Insertion profonde de Schnorr-Euchner.

– Au lieu de remplir la condition de Lovász pour les paires $(\mathbf{b}_{i-1}, \mathbf{b}_i)$, la remplir pour les paires $(\mathbf{b}_j, \mathbf{b}_i)$ pour tous $j < i$.
– La complexité n'est peut-être plus polynomiale.
– Les bases renvoyées sont un peu meilleures : $1.04 \rightarrow 1.025$.

L'arithmétique flottante dans LLL : Cholesky.

Un réseau de dimension 55 fait boucler le LLL_{FP,XD} de NTL, fplll-1.2 en double précision, et le LLL de Magma en enlevant les bonnes options.

$$\begin{bmatrix} 28915\ldots036 & 0 & 0 & \ldots & 0 & 0 \\ 14457\ldots940 & 25167\ldots693 & 0 & \ldots & 0 & 0 \\ -14457\ldots783 & 12583\ldots053 & 21905\ldots751 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 14457\ldots631 & -12583\ldots224 & 10952\ldots511 & \ldots & 18\ldots616 & 0 \\ 15264\ldots310 & -35953\ldots625 & 19879\ldots178 & \ldots & -37590\ldots437 & 83\ldots884 \end{bmatrix}.$$

L'arithmétique flottante dans LLL : cas « moyen ».

– Motivation : garder la double précision.

– Jusqu'à $d \approx 180$, il ne semble pas y avoir de problème.

– Analyse heuristique : dim-max $\approx$ précision / $\log_2(\delta^{-1}(1+\mu_2))$, avec :

$$\begin{aligned} \delta^{-1} &= \mathbb{E}[\|\mathbf{b}_i^*\|^2 / \|\mathbf{b}_{i+1}^*\|^2] \\ \mu_2 &= \mathbb{E}[\mu_{i,j}^2] \end{aligned}$$

– Du point de vue de l'arithmétique flottante, le coût diminue quand on devient plus exigeant sur la qualité de la sortie !

L'arithmétique flottante dans LLL : Givens/Householder.

- Faut-il remplacer Gram-Schmidt par Givens/Householder ?
- Un réseau de dimension 3 fait boucler le G_LLL_FP de NTL.

$$\begin{bmatrix} 1 & 1 & 0 \\ 2^{54}+1 & -2^{54}+1 & 0 \\ 2^{54} & 2^{54}-1 & 1 \end{bmatrix}.$$

- Dans le cas moyen, rien de clair.

Problèmes ouverts et travaux en cours.

- Obtenir un LLL flottant prouvé n'utilisant pas la matrice de Gram.
- Complexité quasi-linéaire, comme pour le pgcd.
- Diminution du coût « algèbre linéaire ».

- Mieux comprendre les phénomènes observés en pratique, et essayer de les exploiter.