Angriffe auf das GGH-Kryptosystem mittels Gitterreduktion in Blöcken

Henrik Koy

Diplomarbeit

am Fachbereich Informatik
Johann-Wolfgang-Goethe-Universität

Frankfurt am Main

Betreuer: Prof. Dr. C. P. Schnorr

4. Oktober 1999

Zusammenfassung

Wir betrachten das auf der Crypto '97 vorgeschlagene gitterbasierte Kryptosystem von Goldreich, Goldwasser und Halevi (GGH) [11]. Die Autoren veröffentlichten Challenges zu den Sicherheitsparametern 200, 250, 300, 350 und 400 [12]. Jeder Challenge besteht aus dem öffentlichen Schlüssel, sowie einem Ciphertext. Für den Angriff entwickeln wir numerisch stabile Gitterreduktionsalgorithmen, die es ermöglichen, das System in diesen Dimensionen anzugreifen.

Es werden Methoden zur Orthogonalisierung, die sogenannten Householder-Reflexionen und Givens-Rotationen behandelt, und eine praktikable Gleitpunkt-Arithmetik Version des LLL-Algorithmus nach Lenstra, Lenstra und Lovász [16] angegeben. Wir entwickeln und analysieren den LLL-Block-Algorithmus, der die Gitterreduktion in Blöcken organisiert. Die Gleitpunkt-Arithmetik Version des LLL-Block-Algorithmus wird experimentell auf das GGH-Schema angewendet und mit der LLL-Reduktion in den Dimensionen 100 bis 400 verglichen. Neben der besseren numerischen Stabilität ist die LLL-Block-Reduktion um den Faktor 10 bis 18 mal schneller als die gewöhnliche LLL-Reduktion.

Das GGH-Kryptosystem wurde ebenfalls von Nguyen [22] angegriffen, und die ursprünglichen Nachrichten wurden bis in Dimension 350 rekonstruiert. Wir stellen weitere Angriffe auf das Kryptosystem vor. Es zeigt sich, daß die öffentlichen Parameter für erfolgreiche Angriffe benutzt werden können. Der private Schlüssel in der Dimension 200 wird nach ca. 10 Stunden rekonstruiert und Ciphertext-Attaken sind bis in Dimension 300 erfolgreich.

Erklärung

Hiermit versichere ich, die vorliegende Diplomarbeit selbständig verfaßt und keine anderen Hilfsmittel als die angegebenen Quellen verwendet zu haben.

Henrik Koy Frankfurt, den 4. Oktober 1999

Inhaltsverzeichnis

Einführung

Mit zunehmender Verlagerung der Kommunikation auf das Internet wächst der Bedarf an praktikablen Implementierungen der Public-Key-Kryptographie. Es ist so möglich, über den offenen Kanal des Internets geheime Informationen zu senden und Verträge abzuschließen. Die Attraktivität eines Public-Key-Kryptosystems wird neben der Sicherheit, der Schlüsselgröße, auch durch die Berechnungszeiten zur Chiffrierung und Dechiffrierung, sowie durch die Expansionsrate des Chiffres bestimmt. Aufgrund der hohen Anforderungen an die Sicherheit und die Performance haben sich bis jetzt nur wenige Kryptosysteme in der Praxis durchgesetzt. Die Sicherheit dieser Schemata basiert auf der Schwierigkeit der Faktorisierung von ganzen Zahlen [23] sowie der Berechnung diskreter Logarithmen [8].

In letzter Zeit wurden Kryptosysteme vorgeschlagen, deren Sicherheit auf Problemen der Gitterreduktion basiert. Dabei werden das Shortest-Vektor-Problem (SVP) und das Closest-Vector-Problem (CVP) betrachtet. Gitter sind diskrete additive Gruppen, die durch n linear unabhängige Vektoren $b_1, b_2, \ldots, b_n$ (die sogenannte Gitterbasis) erzeugt werden. Zu gegebener Gitterbasis wird beim SVP ein nicht-trivialer kürzester Gittervektor gesucht, während beim CVP zu einem gegebenem Punkt ein nächster Gittervektor zu bestimmen ist. Diese Probleme sind NP-hart [1, 9]. Auch die approximativen Varianten dieser Probleme sind bis auf gewisse Faktoren NP-hart [20, 7]. Für die auf Gitterproblemen basierende Kryptographie fanden das Ajtai-Dwork-Kryptosystem [2] und das Schema von Goldreich, Goldwasser und Halevi [11] die größte Beachtung.

Grundlage für den praktischen Angriff auf diese Kryptosysteme ist der bekannte LLL-Algorithmus von Lenstra, Lenstra und Lovász [16]. Die Implementierung von Schnorr-Euchner [28] verwendet die von der „Rechenmaschine“ zur Verfügung gestellte Gleitpunkt-Arithmetik. Die Bitlänge dieser Gleitpunktzahlen ist konstant, und das Ausführen einer Gleitpunkt-Operation erfolgt in nur einem Rechenschritt. Diese Algorithmen sind jedoch für die Reduktion der in dieser Arbeit auftretenden Gitterbasen numerisch nicht stabil. Wir ersetzen deshalb das Gram-Schmidt-Verfahren durch die numerisch stabilere Orthogonalisierungsmethode der sogenannten Householder-Reflexion und erhalten bessere Stabilitätseigenschaften.

Zur ganzzahligen Eingabebasis $b_1, b_2, \ldots, b_n \in \mathbb{Z}^d$ mit $M := \max_j \|b_j\|$ benötigt die LLL-Reduktion $O(dn^3 \log M)$ arithmetische Schritte. Seien π_ν die orthogonalen Projektionen auf dem zu $\mathrm{span}(b_1, b_2, \ldots, b_{\nu-1})$ senkrechten Raum. Der LLL-Algorithmus für $\nu = 1, 2, \ldots, n-1$ vergleicht die Größen $\|\pi_\nu(b_\nu)\|$ und $\|\pi_\nu(b_{\nu+1})\|$. Die Berechnung dieser Größen erfolgt mit d-stelligen Vektoren. Die Laufzeit der LLL-Reduktion wird demnach von der Dimension des Vektorraumes $\mathbb{R}^d$ bestimmt. Sei $b_{s+1}, \ldots, b_{s+k}$ ein „Block" aus der Gitterbasis. Durch die Orthogonalisierung bestimmen wir die isometrische Darstellung von $\pi_{s+1}(b_{s+1}), \ldots, \pi_{s+1}(b_{s+k})$ als Basis im $\mathbb{R}^k$. Für $s+1 \leq \nu \leq s+k-1$ werden die Größen $\|\pi_\nu(b_\nu)\|$ und $\|\pi_\nu(b_{\nu+1})\|$ im $\mathbb{R}^k$ bestimmt. Wir bezeichnen diese Methode als *Gitterreduktion in Blöcken.*

In Kapitel 3 wird der LLL-Block-Algorithmus zur Gitterreduktion in Blöcken vorgestellt und analysiert. Der Algorithmus wird in Gleitpunkt-Arithmetik implementiert. Anhand von experimentellen Resultaten zeigen wir die bessere numerische Stabilität und schnellere Laufzeit des Verfahrens.

Im letzten Kapitel untersuchen wir das von Goldreich, Goldwasser und Halevi (GGH) vorgeschlagene gitterbasierte Kryptosystem [11]. Die Chiffrierung c einer Nachricht m erfolgt in diesem Schema durch das Addieren eines zufälligen Störvektors e auf den zur Nachricht m korrespondierenden Gittervektor $c(m)$, d.h. $c = c(m) + e$. Mit einer reduzierten Basis, dem geheimen Schlüssel, kann diese Instanz des CVP gelöst werden. Angriffe auf das Schema versuchen den Störvektor e — ohne Kenntnis des geheimen Schlüssels — zu ermitteln, um so die Nachricht zu erhalten. Die Angriffe auf e erfolgen durch die sogenannte Einbettungsmethode: das Gitter wird durch den Ciphertext c erweitert. Aufgrund einer Heuristik ist die Differenz zwischen c und dem nächsten Gittervektor der kürzeste Gittervektor. Nguyen [22] hat das GGH-Schema gebrochen, indem er eine Darstellung e' mit kleiner Länge für den Störvekter e bestimmt. Durch die Einbettungsmethode kann e' leicht als kürzester Basisvektor bestimmt werden, und e damit rekonstruiert werden.

Wir stellen einen alternativen Angriff auf den Fehlervektor vor. Anstatt eine andere Darstellung für e zu bestimmen, betrachten wir die durch c erweiterte Gitterbasis. Durch geeignete Transformationen bestimmen wir ein Untergitter, so daß e als Gittervektor erhalten bleibt. Durch das Entfernen von Vektoren aus dem Gitter erwarten wir heuristisch, daß e für das Untergitter ein kürzester Gittervektor ist, und damit leicht zu bestimmen ist.

Ich bedanke mich bei Herrn Prof. Dr. C.P. Schnorr für die intensive Betreuung meiner Diplomarbeit. Roger Fischlin danke ich für die Hinweise zum Aufschreiben der Diplomarbeit und zahlreichen Verbesserungsvorschlägen.

Kapitel 1

Grundlagen

Wir fassen wichtige Begriffe und elementare Sätze der Gittertheorie zusammen und stellen Methoden zur Orthogonalisierung im $\mathbb{R}^d$ vor. Neben dem bekannten Gram-Schmidt-Verfahren betrachten wir die QR-Zerlegung einer Matrix in die orthogonale Matrix Q und die obere Dreiecksmatrix R. Householder-Reflexionen und Givens-Rotationen sind elementare Transformationen zur praktischen QR-Zerlegung in Gleitpunkt-Arithmetik.

1.1 Notationen

Seien $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}$ die Menge der natürlichen, der ganzen, der rationalen und der reellen Zahlen. Für $a \in \mathbb{R}$ bezeichne $\lceil a \rfloor$ die zu a nächste ganze Zahl, $\lceil a \rceil$ die kleinste ganze Zahl größer oder gleich a und $\lfloor a \rfloor$ die größte ganze Zahl kleiner oder gleich a. Sei weiter $|a|$ der Betrag von a und $\mathrm{sign}(a) \in \{\pm 1\}$ ($\mathrm{sign}(0) := 1$) das Vorzeichen von a.

Wir bezeichnen mit $\mathbb{R}^d$ den euklidischen Vektorraum versehen mit dem Standard-Skalarprodukt $\langle \cdot, \cdot \rangle$ und der Norm $\|x\|^2 := \langle x, x \rangle$. Die Vektoren x und y stehen zueinander *orthogonal* (kurz $x \perp y$), falls $\langle x, y \rangle = 0$. Die Untervektorräume $U, V \subseteq \mathbb{R}^d$ sind zueinander orthogonal ($U \perp V$), wenn für alle $u \in U$, $v \in V$ gilt $u \perp v$. Der Vektorraum

$$V^\perp := \{u \in \mathbb{R}^d \mid \forall v \in V \,:\, u \perp v\}$$

ist das zu V orthogonale Komplement.

Sei $\mathrm{span}(b_1, b_2, \ldots, b_n) \subseteq \mathbb{R}^d$ der von den Vektoren $b_1, b_2, \ldots, b_n$ aufgespannte lineare Raum. Die Vektoren $b_1, b_2, \ldots, b_n$ werden auch als $d \times n$ Matrix $B := [b_1, b_2, \ldots, b_n]$ zusammengefaßt. C^T ist die zu C transponierte Matrix und falls C invertierbar ist, heißt C^{-1} die zu C inverse Matrix. Die Matrix Q heißt *orthogonal*, wenn $Q^T Q$ die Einheitsmatrix I ist.

Für eine Lebesgue-meßbare Menge M sei $\mathrm{Vol}(M)$ deren *Volumen*. Die *Bitlänge* der ganzen Zahl z ist definiert als $l(z) := 1 + \lceil \log_2(|z| + 1) \rceil$. Dement-

sprechend ist die Bitlänge der rationalen Zahl $r = \frac{p}{q}$ die Summe $l(r) := l(p) + l(q)$.

1.2 Einführung in die Gittertheorie

Wir beschäftigen uns mit elementaren Begriffen und grundlegenden Aussagen der Gittertheorie. Die Gitterreduktion wird motiviert und ein algorithmischer Ansatz angegeben.

Gitter sind diskrete additive Untergruppen des $\mathbb{R}^d$.

Definition 1.1 Sei $b_1, b_2, \ldots, b_n \in \mathbb{R}^d$ ein System von n linear unabhängigen Vektoren. Die additive Gruppe

$$L = L(b_1, b_2, \ldots, b_n) := b_1\mathbb{Z} + b_2\mathbb{Z} + \cdots + b_n\mathbb{Z}$$

heißt *Gitter* erzeugt durch die *Gitterbasis* $b_1, b_2, \ldots, b_n$.

Die Gitterbasis $b_1, b_2, \ldots, b_n$ heißt *geordnet*, wenn die Reihenfolge der Basisvektoren durch die Indizierung festgelegt ist. Die geordnete Gitterbasis wird auch in der Matrizenform $B := [b_1, b_2, \ldots, b_n] \in \mathbb{R}^{d \times n}$ dargestellt, man spricht von der *Basismatrix*. Das Gitter L ist die Menge

$$L = L(B) = B \cdot \mathbb{Z}^n.$$

Es bezeichne $Rang(L)$ den Spaltenrang der Basismatrix B. Das Gitter $L \subset \mathbb{R}^d$ hat *vollen* Rang, wenn $Rang(L) = d$ gilt.

Faßt man den Vektorraum $\mathbb{R}^d$ als additive Gruppe auf, so entsprechen die Gitter im $\mathbb{R}^d$ genau den diskreten additiven Untergruppen des $\mathbb{R}^d$ (siehe Gruber, Lekkerkerker [13, Seite 18]). „Diskret“ bedeutet, daß die Gitter $L \subseteq \mathbb{R}^d$ keinen Häufungspunkt besitzen.

Ein Gitter besitzt mehrere Basen.

Sei $GL_n(\mathbb{Z}) = \{ T \in \mathbb{Z}^{n \times n} \mid \det T \in \{\pm 1\} \}$ die Gruppe der *unimodularen* Matrizen. $GL_n(\mathbb{Z})$ wird durch die Elementarmatrizen zur

- Spaltenvertauschung
- Multiplikation einer Spalte mit -1
- Addition einer Spalte zu einer anderen Spalte

erzeugt. Für unimodulare Matrizen T gilt die Identität $\mathbb{Z}^n = T \cdot \mathbb{Z}^n$, da zu jedem $x \in \mathbb{Z}^n$ genau ein $y = T^{-1}x \in \mathbb{Z}^n$ mit $x = Ty$ existiert. Zwei

Basismatrizen $B, C \in \mathbb{R}^{d \times n}$ erzeugen folglich genau dann das gleiche Gitter, wenn es eine unimodulare Matrix T mit $B = CT$ gibt, da

$$L(B) = B \cdot \mathbb{Z}^n = B \cdot (T^{-1} \cdot \mathbb{Z}^n) = (BT^{-1}) \cdot \mathbb{Z}^n = C \cdot \mathbb{Z}^n = L(C).$$

Beispiel: Die Gitterbasen

$$\begin{bmatrix} 4 & -1 \\ 3 & 3 \\ 1 & -4 \end{bmatrix} \quad \text{und} \quad \begin{bmatrix} 616 & -565 \\ 507 & -465 \\ 109 & -100 \end{bmatrix}$$

erzeugen das gleiche Gitter, denn mit der unimodularen Matrix $T = \begin{bmatrix} 157 & -144 \\ 12 & -11 \end{bmatrix}$, gilt:

$$\begin{bmatrix} 616 & -565 \\ 507 & -465 \\ 109 & -100 \end{bmatrix} = \begin{bmatrix} 4 & -1 \\ 3 & 3 \\ 1 & -4 \end{bmatrix} \cdot \begin{bmatrix} 157 & -144 \\ 12 & -11 \end{bmatrix}.$$

Die Komplexität von Gitterproblemen hängt entscheidend von der Wahl der Gitterbasis ab. Betrachte die folgende Aufgabe: Gegeben sei die Gitterbasis

$$B = \begin{bmatrix} 616 & -565 \\ 507 & -465 \\ 109 & -100 \end{bmatrix}.$$

Bestimme einen zu

$$p = \begin{bmatrix} 7.1 \\ 9.2 \\ -1.9 \end{bmatrix}$$

„nächsten“ Gitterpunkt, d.h. bestimme ein $x \in L(B)$ mit

$$\|x - p\| = \min_{x' \in L(B)} \|x' - p\|.$$

Offensichtlich ist diese Aufgabe mit Hilfe der *reduzierten* Basis

$$\begin{bmatrix} 4 & -1 \\ 3 & 3 \\ 1 & -4 \end{bmatrix}$$

einfach zu lösen:

$$\begin{bmatrix} 7 \\ 9 \\ -2 \end{bmatrix} = 2 \cdot \begin{bmatrix} 4 \\ 3 \\ 1 \end{bmatrix} + 1 \cdot \begin{bmatrix} -1 \\ 3 \\ -4 \end{bmatrix}$$

ist der zu p nächste Gitterpunkt.

Ziel der Gitterreduktion ist es eine Basis mit „kurzen“ Vektoren zu finden.

Berechnungsproblem der Gitterreduktion (informell):
Gegeben die Reduktionsbedingung $\mathcal{R}$, finde zu einem Gitter $L = L(B)$ eine Basis B, so daß die Reduktionsbedingung $\mathcal{R}$ erfüllt ist, d.h. $(\mathcal{R}(B) = true)$.

Zur Eingabebasis B wird eine $\mathcal{R}$-reduzierte Basis B' durch die unimodulare Transformation T berechnet ($B' = B \cdot T$). Die Matrix T kann auch als Produkt

$$T = T_1 \cdot T_2 \cdots T_k$$

von unimodularen Matrizen dargestellt werden. Die Basis B' wird schrittweise durch diese Transformationen bestimmt:

Algorithmus zur Gitterreduktion mit Reduktionsbedingung $\mathcal{R}$ (informell)

`INPUT:` Gitterbasis $B = [b_1, b_2, \ldots, b_n]$

`1.` `WHILE` $(\mathcal{R}(B) = false)$ `DO`
  Bestimme unimodulare Transformation T
  $B := B \cdot T$
`END WHILE`

`OUTPUT:` $\mathcal{R}$-reduzierte Basis B.

Die in dieser Arbeit vorgestellten Algorithmen zur Gitterreduktion arbeiten nach diesem Schema.

Beispiel: Die Basis b_1, b_2 heißt *Gauß-reduziert* ($\mathcal{R}_{Gauß}$), wenn

$$\|b_1\| \leq \|b_2\| \quad \text{und} \quad 0 \leq \frac{\langle b_1, b_2 \rangle}{\|b_1\|^2} \leq \frac{1}{2}.$$

Für Gauß-reduzierte Basen b_1, b_2 ist b_1 ein kürzester nicht trivialer Gittervektor [31]. Der folgende Algorithmus bestimmt eine Gauß-reduzierte Basis:

Algorithmus 1.1 Gauß-Reduktion

`INPUT:` $b_1, b_2 \in \mathbb{R}^d$

`1.` `WHILE` $(\mathcal{R}_{Gauß}(b_1, b_2) = false)$ `DO`

`1.1` `IF` $(\|b_1\| > \|b_2\|)$ `THEN` vertausche b_1 mit b_2

`1.2` `IF` $\left(0 \neq \mu := \left\lceil \frac{\langle b_1, b_2 \rangle}{\|b_1\|^2} \right\rfloor\right)$ `THEN` $b_2 := b_2 - \mu \cdot b_1$

`1.3` `IF` $\left(\frac{\langle b_1, b_2 \rangle}{\|b_1\|^2} < 0\right)$ `THEN` $b_2 := -b_2$

`END WHILE`

`OUTPUT:` Gauß-reduzierte Basis b_1, b_2.

Korrektheit. Die Transformation $b_2 := b_2 - \mu \cdot b_1$ in Schritt `1.2` verringert für $0 \neq \mu := \lceil \langle b_2, b_1 \rangle \cdot \|b_1\|^{-2} \rfloor$ die Länge von b_2 (Längenreduktion), denn es gilt

$$\begin{aligned}
\langle b_2 - \mu b_1, b_2 - \mu b_1 \rangle &= \|b_2\|^2 - 2\mu \langle b_2, b_1 \rangle \frac{\|b_1\|^2}{\|b_1\|^2} + \mu^2 \|b_1\|^2 \\
&= \|b_2\|^2 + \mu \|b_1\|^2 \underbrace{(-2\langle b_2, b_1 \rangle \cdot \|b_1\|^{-2} + \mu)}_{< 0, \text{ da } \langle b_2, b_1 \rangle \cdot \|b_1\|^{-2} > \frac{1}{2}} \\
&> \|b_2\|^2 .
\end{aligned}$$

Schritt `1.2` und `1.3` erfüllt die Bedingung $0 \leq \langle b_1, b_2 \rangle \cdot \|b_1\|^{-2} \leq \frac{1}{2}$. Schritt `1.1` sortiert die Vektoren $\|b_1\| \leq \|b_2\|$. Wenn keine Längenreduktion von b_2 erfolgt, ist am Ende von Schritt `1.3` die Reduktionsbedingung $\mathcal{R}_{Gauß}$ erfüllt. Spätestens zu diesem Zeitpunkt stoppt die Reduktion.

Sei $b_1, b_2 \in \mathbb{Z}^d$ mit $M = \max\{\|b_1\|, \|b_2\|\}$ eine Eingabebasis für die obige Gauß-Reduktion. Algorithmus 1.1 wird analysiert, indem man die Anzahl K der Schleifendurchläufe von Schritt `1.` abschätzt. Es gilt die Beziehung (siehe Vallée [31]):

$$K \leq \log_{\sqrt{3}} M + 2.$$

Jede Vektortransformation kostet $O(d)$ arithmetische Operationen. Algorithmus 1.1 benötigt zur Gauß-Reduktion dieser Basis $O(d \cdot \log M)$ arithmetische Operationen.

Die metrischen Eigenschaften eines Gitters sind unabhängig vom euklidischen Vektorraum.

Seien $B = [b_1, b_2, \ldots, b_n] \in \mathbb{R}^{d \times n}$ und $R = [r_1, r_2, \ldots, r_n] \in \mathbb{R}^{d' \times n}$ Gitterbasen mit der Eigenschaft

$$\langle b_i, b_j \rangle = \langle r_i, r_j \rangle.$$

Die von R und B erzeugten Gitter, besitzen die gleichen metrischen Eigenschaften. R heißt *isometrische* Basismatrix zu B (kurz: $R \cong_{\text{iso}} B$).

Bemerkung 1.2 Im Fall des Standard-Skalarprodukts gilt: Die Gitterbasen $b_1, b_2, \ldots, b_n \in \mathbb{R}^d$ und $r_1, r_2, \ldots, r_n \in \mathbb{R}^d$ sind dann isometrisch, wenn es eine orthogonale Matrix Q mit der Eigenschaft $b_i = Q \cdot r_i$ gibt, $i = 1, 2, \ldots, n$. Es gilt dann

$$\langle b_i \, , \, b_j \rangle = \langle Q \cdot r_i \, , \, Q \cdot r_j \rangle = \langle r_i \, , \, Q^T Q \cdot r_j \rangle = \langle r_i \, , \, r_j \rangle.$$

Beispiel:

$$\mathbb{R}^{d \times 2} \ni [b_1, b_2] \cong_{\text{iso}} \begin{bmatrix} \|b_1\| & \frac{\langle b_2, b_1 \rangle}{\|b_1\|} \\ 0 & \left(\|b_2\|^2 - \frac{\langle b_2, b_1 \rangle^2}{\|b_1\|^2} \right)^{1/2} \end{bmatrix} =: [r_1, r_2] \in \mathbb{R}^{2 \times 2}.$$

Beachte: Die Gauß-Reduktion von r_1, r_2 nach Algorithmus 1.1 ist im Vergleich zur Reduktion von b_1, b_2 um den Faktor $d/2$ schneller[1], weil die Berechnungen auf 2-dimensionale Vektoren erfolgen.

Seien $b_1, b_2, \ldots, b_n \in \mathbb{R}^d$ linear unabhängige Vektoren. Die Menge

$$\mathcal{P}(b_1, b_2, \ldots, b_n) = \left\{ \sum_i \tau_i b_i \;\middle|\; \tau_i \in [0,1) \cap \mathbb{R} \right\}$$

ist das von $b_1, b_2, \ldots, b_n$ aufgespannte *Parallelepiped*.

Definition 1.3 Sei $L \subset \mathbb{R}^d$ ein Gitter mit Basis $b_1, b_2, \ldots, b_n$. Die *Gitterdeterminante* $\det L$ ist das n-dimensionale Volumen des Parallelepipeds $\mathcal{P}(b_1, b_2, \ldots, b_n)$

$$\det L := \mathrm{Vol}_{span(b_1,b_2,\ldots,b_n)}(\mathcal{P}(b_1, b_2, \ldots, b_n)) = \det\left([\langle b_i, b_j\rangle]_{1\le i,j\le n}\right)^{\frac{1}{2}}.$$

Die Gitterdeterminante ist unabhängig von der Gitterbasis.

Definition 1.4 Sei $L \subset \mathbb{R}^d$ ein Gitter. Das zu L *duale Gitter* L^* ist die Menge

$$L^* := \left\{ u \in \mathbb{R}^d \;\middle|\; (\forall b \in L)(\langle u, b\rangle \in \mathbb{Z}) \right\}.$$

Auch das duale Gitter L^* ist eine diskrete additive Gruppe.

Man bestimmt eine Basis $b_1^*, b_2^*, \ldots, b_n^*$ für das zu $L = L(b_1, b_2, \ldots, b_n)$ duale Gitter L^* durch das Gleichungssystem

$$\langle b_i^*, b_j\rangle = \begin{cases} 1 & \text{falls } i = j \\ 0 & \text{sonst} \end{cases} \qquad 1 \le i, j \le n.$$

Wegen der Eigenschaft $[b_1^*, b_2^*, \ldots, b_n^*]^T \cdot [b_1, b_2, \ldots, b_n] = I$ gilt

$$\det L^* = \frac{1}{\det L}.$$

1.3 Orthogonalisierungsverfahren

Mit wachsendem Gitterrang n steigt die Komplexität der Gitterreduktion (vergleiche Kapitel 2.1). In jedem Schritt der Gitterreduktion werden deshalb Gitter kleiner Dimension bearbeitet. Die orthogonale Projektion dient zur Bestimmung dieser Gitter.

[1] Die Transformationsschritte in Algorithmus 1.1 zur Gauß-Reduktion von b_1, b_2 und r_1, r_2 sind – aufgrund der Isometrie zwischen den Basen – identisch.

Orthogonale Projektion. Zur geordneten Basis $b_1, b_2, \ldots, b_n \in \mathbb{R}^d$ sei

$$\pi_i : \mathbb{R}^d \longrightarrow \text{span}(b_1, \ldots, b_{i-1})^\perp \tag{1.1}$$

die i-te orthogonale Projektion. Zu jedem Vektor $b \in \mathbb{R}^d$ ist die Linearkombination $b = u + v$ mit $u = b - \pi_i(b) \in \text{span}(b_1, \ldots, b_{i-1})$ und $v = \pi_i(b) \in \text{span}(b_1, \ldots, b_{i-1})^\perp$ eindeutig bestimmt. Wegen $u \perp v$ gilt $\|b\|^2 = \|u\|^2 + \|v\|^2$.

Es werden k aufeinander folgende Basisvektoren $b_{s+1}, \ldots, b_{s+k}$ als Block zusammengefaßt. Mit Hilfe der orthogonalen Projektion wird im folgenden die *Reduktion in lokalen Koordinaten* und die *Längenreduktion für diesen Block* erklärt.

Reduktion in lokalen Koordinaten. Die orthogonale Projektion π_{s+1} bestimmt das Rang-k Gitter

$$L_{s+1}(b_{s+1}, \ldots, b_{s+k}) := L\left(\pi_{s+1}(b_{s+1}), \ldots, \pi_{s+1}(b_{s+k})\right). \tag{1.2}$$

Dieses Gitter steht senkrecht auf $\text{span}(b_1, \ldots, b_s)$. Die Reduktion von $\pi_{s+1}(b_{s+1}), \ldots, \pi_{s+1}(b_{s+k})$ bezeichnen wir als Reduktion in lokalen Koordinaten. Im Anschluß wird die unimodulare Transformation T der Reduktion in lokalen Koordinaten auf den Block übertragen:

$$[b_{s+1}, \ldots, b_{s+k}] := [b_{s+1}, \ldots, b_{s+k}] \cdot T. \tag{1.3}$$

Reduktion der Längen $\|b_{s+i}\|^2$. Die Punkte $p_{s+i} := (b_{s+i} - \pi_{s+1}(b_{s+i}))$ sind linear abhängig von den Vektoren $b_1, \ldots, b_s$. Betrachte die folgende Aufgabe: Gegeben der Punkt p_{s+i}, bestimme ein Gitterpunkt $x \in L(b_1, \ldots, b_s)$ mit kleinen Abstand $\|x - p_{s+i}\|^2$. Da der Vektor $\pi_{s+1}(b_{s+i})$ senkrecht auf x und p_{s+i} steht, gilt die Gleichung

$$\|b_{s+i} - x\|^2 = \|p_{s+i} - x\|^2 + \|\pi_{s+1}(b_{s+i})\|^2. \tag{1.4}$$

Für $\| p_{s+i} - x \|^2 < \| p_{s+i} \|^2$ reduziert die Transformation $b_{s+i} := b_{s+i} - x$ die Länge von b_{s+i}. Der Gitterpunkt x sollte den Vektor p_{s+i} gut approximieren.

Wir erhalten den folgenden Ansatz zur Gitterreduktion in Blöcken:

Algorithmus zur Gitterreduktion in Blöcken (informell)

INPUT: Gitterbasis $B = [b_1, b_2, \ldots, b_n]$
1. WHILE $(\mathcal{R}(B) = false)$ DO /* $\mathcal{R}$ ist die Reduktionsbedingung */
1.1 Wähle s und k.
1.2 Berechne die orthogonale Projektion $\pi_{s+1}(\cdot)$.
1.3 Reduziere $L_{s+1}(b_{s+1}, \ldots, b_{s+k})$ in lokalen Koordinaten und übertrage die reduzierende Transformation T auf den k-Block $[b_{s+1}, \ldots, b_{s+k}] := [b_{s+1}, \ldots, b_{s+k}] \cdot T$.
1.4 Reduktion der Längen von $b_{s+1}, \ldots, b_{s+k}$.
END WHILE
OUTPUT: $\mathcal{R}$-reduzierte Basis B.

Wir berechnen die orthogonale Projekion π_{s+1} mit Hilfe der Orthogonalisierung von Gitterbasen.

1.3.1 Orthogonalisierung

Zur geordneten Basis $b_1, b_2, \ldots, b_n \in \mathbb{R}^d$ seien $\pi_i : \mathbb{R}^d \longrightarrow \text{span}(b_1, \ldots, b_{i-1})^\perp$ die Projektionen in dem zu $\text{span}(b_1, \ldots, b_{i-1})$ orthogonalen Raum. Definiere $\hat{b}_1 := \pi_1(b_1), \ldots, \hat{b}_n := \pi_n(b_n)$. Ein konstruktives Verfahren zur Bestimmung der Vektoren $\hat{b}_1, \hat{b}_2, \ldots, \hat{b}_n$, ist das bekannte *Gram-Schmidt*-Verfahren:

$$\hat{b}_1 := \pi_1(b_1) = b_1, \tag{1.5}$$

$$\hat{b}_i := \pi_i(b_i) = b_i - \sum_{j=1}^{i-1} \mu_{i,j} \hat{b}_j \quad \text{für } i = 2, 3, \ldots, n. \tag{1.6}$$

Dabei heißen $\mu_{i,j} := \langle b_i, \hat{b}_j \rangle \cdot \|\hat{b}_j\|^{-2}$ die *Gram-Schmidt*-Koeffizienten. Mit $\mu_{i,i} := 1$ und $\mu_{i,j} := 0$, für $i < j$, gilt das Gleichungssystem

$$[b_1, b_2, \ldots, b_n] = [\hat{b}_1, \hat{b}_2, \ldots, \hat{b}_n] \cdot [\mu_{i,j}]^T. \tag{1.7}$$

Die Matrix $[\mu_{i,j}]^T$ ist eine obere Dreicksmatrix, deshalb gilt

$$\text{span}(b_1, \ldots, b_i) = \text{span}(\hat{b}_1, \ldots, \hat{b}_i) \quad \text{für } 1 \leq i \leq n. \tag{1.8}$$

Die Vektoren $\hat{b}_1, \hat{b}_2, \ldots, \hat{b}_n$ bilden eine Orthogonalbasis.

Bemerkung 1.5 Betrachte die Gitterreduktion im Block $b_{s+1}, \ldots, b_{s+k}$. Die Basis $\pi_{s+1}(b_{s+1}), \ldots, \pi_{s+1}(b_{s+k})$ wird berechnet durch

$$\pi_{s+1}(b_{s+i}) = b_{s+i} - \sum_{j=1}^{s} \mu_{s+i,j} \hat{b}_j \in \mathbb{R}^d, \quad \text{für } 1 \leq i \leq k. \tag{1.9}$$

1.3.2 QR-Zerlegung

Aus Gleichungssystem (1.7) erhalten wir zur Basismatrix $B = [b_1, b_2, \ldots, b_n]$ die Zerlegung

$$[b_1, b_2, \ldots, b_n] = \underbrace{\left[\frac{\hat{b}_1}{\|\hat{b}_1\|}, \ldots, \frac{\hat{b}_n}{\|\hat{b}_n\|}\right]}_{Q} \cdot \underbrace{\begin{bmatrix} \|\hat{b}_1\| & & 0 \\ & \ddots & \\ 0 & & \|\hat{b}_n\| \end{bmatrix} \cdot [\mu_{i,j}]^T}_{R}$$

in die orthogonale Matrix Q und in die obere Dreiecksmatrix $R = [r_{i,j}]$ mit $r_{i,j} = \mu_{j,i} \cdot \|\hat{b}_i\|$. Diese Zerlegung ist eindeutig bestimmt (siehe Golub und Van Loan [14, Seite 218]):

Satz 1.6 *Sei $B = [b_1, b_2, \ldots, b_n] \in \mathbb{R}^{d \times n}$ eine Matrix mit vollen Spaltenrang. Dann ist für $\nu = 1, 2, \ldots, n$ die Zerlegung*

$$B^{(\nu)} = [b_1, b_2, \ldots, b_\nu] = Q^{(\nu)} R^{(\nu)}$$

eindeutig bestimmt wenn $Q^{(\nu)}$ orthogonal und $R^{(\nu)}$ eine obere Dreicksmatrix mit positiven Diagonaleinträgen ist.

Bemerkung 1.7 Sei B eine Matrix mit vollen Spaltenrang. Für die Zerlegung $B = QR$ in die orthogonale Matrix Q und in die obere Dreiecksmatrix $R = [r_{i,j}]$ mit Diagonaleinträgen ungleich Null gilt der Zusammenhang

$$r_{i,j} = \text{sign}(r_{j,j}) \cdot \mu_{j,i} \|\hat{b}_i\|. \tag{1.10}$$

Aufgrund der Orthogonalität von $Q^{(\nu)}$ gilt nach Bemerkung 1.2 die Isometrie zwischen den Basen $B^{(\nu)} = [b_1, \ldots, b_\nu]$ und $R^{(\nu)} = [r_1, \ldots, r_\nu]$. Die Matrix $R^{(\nu)}$ ist eine obere Dreicksmatrix, also

$$(R^{(\nu)})^T \cdot R^{(\nu)} = \begin{bmatrix} \|\hat{b}_1\|^2 & & 0 \\ & \ddots & \\ 0 & & \|\hat{b}_\nu\|^2 \end{bmatrix}.$$

Bemerkung 1.8 Mit der Isometrie zwischen $B^{(\nu)}$ und $R^{(\nu)}$ folgt

$$\begin{aligned} \det L(B^{(\nu)}) &= \det \left((B^{(\nu)})^T \cdot B^{(\nu)}\right)^{1/2} \\ &= \det \left((R^{(\nu)})^T \cdot R^{(\nu)}\right)^{1/2} = \prod_{l=1}^{\nu} \|\hat{b}_l\|. \end{aligned} \tag{1.11}$$

Mit Hilfe der QR-Zerlegung kann die Gitterreduktion in lokalen Koordinaten beschleunigt werden. Dazu benötigen wir folgendes Lemma:

Lemma 1.9 *Die Basen* $[\pi_{s+1}(b_{s+1}), \ldots, \pi_{s+1}(b_{s+k})]$ *und*

$$R^{(s,k)} := \begin{bmatrix} r_{s+1,s+1} & \cdots & r_{s+1,s+k} \\ & \ddots & \vdots \\ 0 & & r_{s+k,s+k} \end{bmatrix}.$$

sind isometrisch.

Beweis. Mit Bemerkung 1.5 und der Isometrie zwischen den Basen B und R folgt

$$\begin{aligned} \langle \pi_{s+1}(b_i), \pi_{s+1}(b_j) \rangle &= \langle b_i, b_j \rangle - \sum_{l=1}^{s} \mu_{i,l} \cdot \mu_{j,l} \|\hat{b}_l\|^2 \\ &= \langle r_i, r_j \rangle - \sum_{l=1}^{s} r_{l,i} \cdot r_{l,j} = \sum_{l=s+1}^{s+k} r_{l,i} \cdot r_{l,j}. \end{aligned}$$

Damit gilt die Isometrie zwischen $[\pi_{s+1}(b_{s+1}), \ldots, \pi_{s+1}(b_{s+k})]$ und $R^{(s,k)}$. ■

Die Reduktion in lokalen Koordinaten wird in der Basis $R^{(s,k)}$ durchgeführt. Der Vorteil ist, daß mit Gittervektoren im $\mathbb{R}^k$ gerechnet wird, während die Basisvektoren $\pi_{s+1}(b_{s+1}), \ldots, \pi_{s+1}(b_{s+k})$ d Einträge haben.

In den folgenden Abschnitten werden praktische Verfahren zur Orthogonalisierung vorgestellt.

1.3.3 Gram-Schmidt-Orthogonalisierung

Wir stellen das Verfahren zur Orthogonalisierung ganzzahliger Basen nach Gram-Schmidt vor. Die Koeffizienten $\mu_{i,j}$ und $\|\hat{b}_j\|^2$ werden in exakter Arithmetik berechnet. Diese Zahlen sind rational (siehe Schnorr [29, Seite 39.]):

Lemma 1.10 *Für ganzzahlige Gitterbasen* $b_1, b_2, \ldots, b_n \in \mathbb{Z}^d$ *gilt*

1.) $D_{j-1} \cdot \hat{b}_j \in \mathbb{Z}^d$, 2.) $D_j \cdot \mu_{\nu,j} \in \mathbb{Z}$.

Dabei ist $D_j := \det^2 L(b_1, b_2, \ldots, b_j) = \prod_{i=1}^{j} \|\hat{b}_i\|^2$ *ganzzahlig.*

Die Determinante ganzzahliger Gitter ist größer oder gleich 1. Für die Basis $b_1, b_2, \ldots, b_n \in \mathbb{Z}^d$ gilt

$$\|\hat{b}_j\| = \frac{\prod_{i=1}^{j} \|\hat{b}_i\|}{\prod_{i=1}^{j-1} \|\hat{b}_i\|} = \frac{\det L(b_1, b_2, \ldots, b_j)}{\prod_{i=1}^{j-1} \|\hat{b}_i\|} \geq \frac{1}{\prod_{i=1}^{j-1} \|\hat{b}_i\|}, \tag{1.12}$$

also mit Hilfe der Abschätzungen $\|\hat{b}_i\| \leq \|b_i\|$

$$\|\hat{b}_j\| \geq \left(\max_{i=1}^{j-1} \|b_i\| \right)^{1-j}. \tag{1.13}$$

Durch die Cauchy-Schwarz'sche Ungleichung und die Ungleichung (1.13) ergibt sich für den Betrag der Gram-Schmidt-Koeffizienten $\mu_{\nu,j}$:

$$|\mu_{\nu,j}| = \frac{|\langle b_\nu, \hat{b}_j \rangle|}{\|\hat{b}_j\|^2} \leq \frac{\|b_\nu\| \cdot \|\hat{b}_j\|}{\|\hat{b}_j\|^2} = \frac{\|b_\nu\|}{\|\hat{b}_j\|} \tag{1.14}$$

$$\leq \|b_\nu\| \left(\max_{i=1}^{j-1} \|b_i\| \right)^{j-1} \leq \left(\max_{i=1}^{\nu} \|b_i\| \right)^{j}. \tag{1.15}$$

Die ganzen Zahlen aus Lemma 1.10 können nicht beliebig groß werden, denn es gilt:

Lemma 1.11 *Sei $b_1, b_2, \ldots, b_n \in \mathbb{Z}^d$ eine ganzzahlige Gitterbasis mit $M := \max_j \|b_j\|^2$. Die ganzen Zahlen $D_{j-1}\|\hat{b}_j\|^2$ und $D_j \mu_{\nu,j}$ sind im Betrag kleiner als $M^{3j/2}$.*

Beweis. Nach Ungleichung (1.15) gilt $|\mu_{\nu,j}| \leq (\max_j \|b_j\|)^j \leq M^{j/2}$. Aufgrund der Abschätzung $D_j \leq M^j$ folgt die Behauptung. ■

Algorithmus 1.2 ORT_GS(ν): Gram-Schmidt-Orthogonalisierung von b_ν

`INPUT`: Teilbasis $b_1, b_2, \ldots, b_{\nu-1} \in \mathbb{Z}^d$, $b_\nu \in \mathbb{Z}^d$,
Koeffizienten $c_j := \|\hat{b}_j\|^2$ und $\mu_{i,j}$ für $1 \leq j < i < \nu$

`1.` `FOR` $(j = 1, \ldots, \nu - 1)$ `DO`
$\mu_{\nu,j} := \left(\langle b_\nu, b_j \rangle - \sum_{i=1}^{j-1} \mu_{\nu,i} \cdot \mu_{j,i} \cdot c_i \right) \cdot c_j^{-1}$
`END DO`
`2.` $c_\nu := \langle b_\nu, b_\nu \rangle - \sum_{j=1}^{\nu-1} \mu_{\nu,j}^2 c_j$

`OUTPUT`: Koeffizienten $c_\nu = \|\hat{b}_\nu\|^2$ und $\mu_{\nu,j}$ für $1 \leq j < \nu$.

Die Orthogonalisierung der Eingabebasis $b_1, b_2, \ldots, b_n \in \mathbb{Z}^d$ erfolgt durch ORT_GS(1), ORT_GS(2), ..., ORT_GS(n). Jeder Aufruf ORT_GS(ν) kostet höchstens $2\nu d + 3\nu^2$ arithmetische Operationen, die Orthogonalisierng der Gitterbasis kostet $\sum_\nu 2\nu d + 3\nu^2 \approx dn^2 + n^3$ arithmetische Operationen. Mit Lemma 1.11 gilt der folgende Satz:

Satz 1.12 *Zur Eingabebasis $b_1, b_2, \ldots, b_n \in \mathbb{Z}^d$ mit $M := \max_j \|b_j\|^2$ benötigt die Orthogonalisierung mit Algorithmus 1.2 höchstens $dn^2 + n^3$ arithmetische Operationen. Die Bitlängen der rationalen Zahlen $\|\hat{b}_j\|^2$ und $\mu_{\nu,j}$ sind kleiner als $\frac{3}{2} \cdot n \cdot \log M$.*

Die Orthogonalisierung in exakter Arithmetik rechnet nach Satz 1.12 mit Zahlen der Bitlänge $O(n \cdot \log M)$. Durch das Rechnen mit Gleitpunktzahlen

kleiner Bitlänge wird die Orthogonalisierung beschleunigt. Nachteilig für die Praxis ist allerdings die fehlende numerische Stabilität der Gram-Schmidt-Orthogonalisierung. Durch das Arbeiten in Gleitpunkt-Arithmetik, bei der der Computer versucht mit der beschränkten Bitzahl reelle Zahlen approximativ darzustellen, können Rundungsfehler auftreten. Dadurch wird das Ergebnis unbrauchbar. Diese Problematik versucht man durch numerisch stabilere Verfahren wie Givens-Rotation oder Householder-Reflexion zu lösen. In den folgenden Abschnitten werden diese Methoden beschrieben und in Kapitel 1.3.6 wird die Rundungsfehleranalyse der Orthogonalisierung in Gleitpunkt-Arithmetik vorgestellt.

1.3.4 Givens-Rotation

Die Zerlegung $B = QR$ einer Basismatrix ist nach Satz 1.5 eindeutig bestimmt wenn Q orthogonal und R eine obere Dreicksmatrix mit positiven Diagonaleinträgen ist. In diesem Abschnitt stellen wir die orthogonalen Matrizen der Givens-Rotation vor. Mit Hilfe von Givens-Rotationen können selektiv Matrixeinträge nach Null rotiert werden. Wir berechnen die QR-Zerlegung einer Basismatrix B, indem alle Matrixeinträge unterhalb der Diagonalen nach Null rotiert werden.

Sei $x = (x_1, \ldots, x_d)^T \in \mathbb{R}^d$. Die *Givens-Rotation* $G_{i,j}(x) : \mathbb{R}^d \longrightarrow \mathbb{R}^d$

$$\begin{bmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_j \\ \vdots \\ x_d \end{bmatrix} \longmapsto G_{i,j}(x) \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_j \\ \vdots \\ x_d \end{bmatrix} = \begin{bmatrix} x_1 \\ \vdots \\ \sqrt{x_i^2 + x_j^2} \\ \vdots \\ 0 \\ \vdots \\ x_d \end{bmatrix},$$

rotiert x_j in die Koordinate i. Die Matrix $G_{i,j}(x)$ sieht wie folgt aus:

$$G_{i,j}(x) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \begin{matrix} \\ \\ i \\ \\ j \\ \\ \\ \end{matrix} \quad \text{mit} \quad \begin{aligned} c &= \frac{x_i}{\sqrt{x_i^2 + x_j^2}}, \\ s &= \frac{x_j}{\sqrt{x_i^2 + x_j^2}}. \end{aligned}$$

Alle Einträge außerhalb der Diagonalen sind – mit Ausnahme in den Positionen (i, j) und (j, i) – Null. Offensichtlich ist die Matrix $G_{i,j}(x)$ orthogonal.

Die Matrixmultiplikation $B \mapsto G_{i,j}(b_k) \cdot B$ rotiert den Eintrag $b_{j,k}$ zu 0. Diese Givens-Rotation operiert in der Matrix B nur in den Zeilen i und j.

Beispiel. Wir beschreiben die QR-Zerlegung anhand der 3×2-Matrix $B = [b_1, b_2]$. Durch drei Givens-Rotationen werden die Matrixeinträge $b_{3,1}, b_{2,1}$ und $b_{3,2}$ nach Null rotiert.

$$\underbrace{\begin{bmatrix} \times & \times \\ \times & \times \\ \times & \times \end{bmatrix}}_{B} \xrightarrow{G_{2,3}(b_1)} \underbrace{\begin{bmatrix} \times & \times \\ \times & \times \\ 0 & \times \end{bmatrix}}_{[b'_1, b'_2]} \xrightarrow{G_{1,2}(b'_1)} \underbrace{\begin{bmatrix} \times & \times \\ 0 & \times \\ 0 & \times \end{bmatrix}}_{[r_1, b''_2]} \xrightarrow{G_{2,3}(b''_2)} \underbrace{\begin{bmatrix} \times & \times \\ 0 & \times \\ 0 & 0 \end{bmatrix}}_{R}$$

Die orthogonale Matrix Q ist das Produkt

$$Q = G_{2,3}(b_1) \cdot G_{2,1}(b'_1) \cdot G_{3,2}(b''_2).$$

Das numerische Verfahren zur QR-Zerlegung durch Givens-Rotationen ist in dem Buch von Golub und Van Loan [14] beschrieben. Die Basis $B = [b_1, b_2, \ldots, b_n] \in \mathbb{R}^{d \times n}$ wird durch eine Folge von elementaren Rotationen $G_{i,j}(\cdot)$ mit $1 \leq j < i \leq d$ in die obere Dreiecksmatrix R transformiert. Insgesamt sind dies $\sum_{k=1}^{n}(d-k) = nd - n(n+1)/2$ Rotationen. Das Produkt $Q^T = \prod G_{i,j}(\cdot)$ der elementaren Givens-Rotationen ist orthogonal, da orthogonale Matrizen mit der Multiplikation eine Gruppe bilden. Die Givensmethode benötigt zur Berechnung der oberen Dreiecksmatrix R maximal $(3n^2(d - n/3))$ Gleitpunkt-Operationen (siehe Golub und Van Loan [14]).

1.3.5 Householder-Reflexion

Wir beschreiben die Methode zur QR-Zerlegung durch die sogenannten Householder-Reflexionen. Die obere Dreicksmatrix R wird durch eine Folge von Householder-Reflexionen aus der Eingabematrix B berechnet.

Sei $v \in \mathbb{R}^d \setminus \{0\}$ ein Vektor. Die orthogonale $d \times d$ Matrix H der Form

$$H := I - 2 \cdot \frac{v \cdot v^T}{\|v\|^2}$$

heißt *Householder-Reflexion* zum *Householder-Vektor* v. Die Abbildung

$$x \longmapsto Hx = x - \left(2\frac{\langle v, x \rangle}{\|v\|^2}\right) v$$

ist die senkrechte Spiegelung von x an der Hyperebene $\text{span}(v)^{\perp}$. Im speziellen benutzen wir zur QR-Zerlegung die folgenden Transformationen:

Sei $x = (x_1, \ldots, x_d)^T \in \mathbb{R}^d$ und für $1 \leq i \leq d$ setze $x_{[i:d]} := (x_i, \ldots, x_d)^T$. Die Matrix $H_i(x)$ ist die Householder-Reflexion mit dem Householder-Vektor

$$v := (0, \ldots, 0, x_i \pm \|x_{[i:d]}\|, x_{i+1}, \ldots, x_d)^T.$$

Wegen $2v^T x = v^T v = \|v\|^2$ folgt

$$H_i(x) \cdot x = \left(I - 2\frac{vv^T}{\|v\|^2} \right) x = x - \frac{2v^T x}{\|v\|^2} \cdot v = x - v$$

also

$$H_i(x) \cdot x = (x_1, \ldots, x_{i-1}, \mp\|x_{[i:d]}\|, 0, \ldots, 0)^T.$$

Beispiel. Für die 3×2 Matrix $B = [b_1, b_2]$, berechnen wir die obere Dreieicksmatrix R durch 2 Householder-Reflexionen: Zunächst werden die Einträge $b_{3,1}$ und $b_{3,2}$ in $b_{1,1}$ gespiegelt. Danach betrachten wir den zu $(b_{1,1}, 0, 0)^T$ orthogonalen Anteil $(0, b_{2,2}, b_{3,2})$ vom zweiten Spaltenvektor in B. Wir spiegeln $b_{3,2}$ in $b_{2,2}$.

$$\underbrace{\begin{bmatrix} \times & \times \\ \times & \times \\ \times & \times \end{bmatrix}}_{B} \xrightarrow{H_1(b_1)} \underbrace{\begin{bmatrix} \times & \times \\ 0 & \times \\ 0 & \times \end{bmatrix}}_{[r_1, b_2']} \xrightarrow{H_2(b_2')} \underbrace{\begin{bmatrix} \times & \times \\ 0 & \times \\ 0 & 0 \end{bmatrix}}_{R}.$$

Wir leiten die QR-Zerlegung durch Householder-Reflexionen ab. Die Orthogonalisierung betrifft immer nur den Vektor b_ν der die Basis erweitert. Die Teilbasis $b_1, b_2, \ldots, b_{\nu-1}$ sei bereits mit den Householder-Reflexionen

$$(Q^{(\nu-1)})^T = H_{\nu-1}(b_{\nu-1}^{(\nu-1)}) \cdots H_1(b_1^{(1)})$$

orthogonalisiert. Es gilt $R^{(\nu-1)} = (Q^{(\nu-1)})^T [b_1, \ldots, b_{\nu-1}]$. Wir erweitern die Teilbasis um b_ν. Zur QR-Zerlegung berechne zunächst

$$C^{(\nu)} := (Q^{(\nu-1)})^T [b_1, \ldots, b_\nu] = \left[R^{(\nu-1)} \,\middle|\, H_{\nu-1}(b_{\nu-1}') \cdots H_1(b_1) b_\nu \right]$$

$$C^{(\nu)} = \begin{bmatrix} r_{1,1} & \cdots & r_{1,\nu-1} & r_{1,\nu} \\ 0 & \ddots & \vdots & \vdots \\ \vdots & \ddots & r_{\nu-1,\nu-1} & r_{\nu-1,\nu} \\ \vdots & & 0 & | \\ \vdots & & \vdots & b_\nu' \\ 0 & \cdots & 0 & | \end{bmatrix},$$

wobei $b_\nu^{(\nu)} = (r_{1,\nu}, \ldots, r_{1,\nu-1}, -b_\nu'-)^T$ den letzten Spaltenvektor in $C^{(\nu)}$ bezeichne. Die Berechnung wird durch die Multiplikation

$$R^{(\nu)} = H_\nu(b_\nu^{(\nu)}) \cdot C^{(\nu)} = \left[R^{(\nu-1)} \,\middle|\, H_\nu^{(\nu)} b_\nu^{(\nu)} \right]$$

abgeschlossen. Algorithmus 1.3 beschreibt die Orthogonalisierung des ν-ten Basisvektors:

Algorithmus 1.3 ORT_HS(ν): Householder-Verfahren

INPUT: Householder-Vektoren $v_1, v_2, \ldots, v_{\nu-1}$, Vektor $b_\nu \in \mathbb{R}^d$

1. $r_\nu := b_\nu$
2. FOR $(j = 1, 2, \ldots, \nu - 1)$ DO /* $b_\nu^{(\nu)} := H_{\nu-1}(b'_{\nu-1}) \cdots H_1(b_1) b_\nu$ */
   $r_\nu := r_\nu - (2\langle r_\nu, v_j \rangle \cdot \|v_j\|^{-2}) \cdot v_j$
   END DO
/* Householder-Vektor v_ν berechnen */
3. IF $(\|r_{[j:d],\nu}\| > 0)$ THEN
3.1 $v_\nu := (0, \ldots, 0, r_{j,\nu} + \mathrm{sign}(r_{j,\nu}) \|r_{[j:d],\nu}\|, r_{j+1,\nu}, \ldots, r_{d,\nu})^T$
3.2 $r_\nu := r_\nu - (\langle r_\nu, v_\nu \rangle \cdot \|v_\nu\|^{-2}) \cdot v_\nu$
   END IF

OUTPUT: Householder-Vektor v_ν, Vektor r_ν.

Bemerkungen.

1. Die Wahl $v_{\nu,j} = r_{j,\nu} + \mathrm{sign}(r_{j,\nu}) \|r_{[j:d],\nu}\|$ in Schritt 3.1 verhindert das Auslöschen von Präzisionsbits.
2. Die Diagonaleinträge $r_{j,j}$ der Matrix $R^{(\nu)}$ sind nicht alle positiv. Es gilt der Zusammenhang $r_{i,j} = \mathrm{sign}(r_{i,i}) \mu_{j,i} \|\hat{b}_i\|$.

Wir benutzen das Householder-Verfahren zur Orthogonalisierung in Gleitpunkt-Arithmetik. Die Einträge des Eingabevektors b_ν werden zunächst als τ-stellige Gleitpunktzahlen in r_ν gespeichert. Jeder Aufruf von ORT_HS(ν) kostet höchstens $\sum_{j=1}^{\nu} 4(d - j + 1) = 4d\nu - 2\nu^2$ Gleitpunkt-Operationen. Die Orthogonalisierung von $b_1, b_2, \ldots, b_n \in \mathbb{R}^d$ benötigt $\sum_{\nu=1}^{n} (4d\nu - 2\nu^2) \approx 2n^2(d - n/3)$ Gleitpunkt-Operationen.

1.3.6 Rundungsfehleranalyse bei der Orthogonalisierung

Sei $B = [b_1, b_2, \ldots, b_n]$ eine Basismatrix. Durch die QR-Zerlegung erhalten wir die zu B isometrische Basismatrix

$$R = [r_1, r_2, \ldots, r_n] = Q^T [b_1, b_2, \ldots, b_n].$$

In Kapitel 3 stellen wir Algorithmen vor, die mit Einträgen aus R rechnen, wobei die Matrix R durch das Rechnen in Gleitpunkt-Arithmetik approximiert wird. Wir benötigen eine Fehlerabschätzung der Gleitpunktzahlen in R. Dazu definiere die τ-stellige nächste *Gleitpunktzahl* a' zu $a \in \mathbb{R}$ durch

$$a' := \mathrm{sign}(a) \left(\sum_{j=1}^{\tau} b_{j-1} 2^{-j} \right) \cdot 2^\alpha, \quad b_j \in \{0, 1\} \ : \quad |a' - a| < 2^{\alpha - \tau}.$$

Die Binärzahl „$0.b_0 b_1 \cdots b_{\tau-1}$“ ist die *Mantisse* und α ist der *Exponent* der Zahl a'. Im Modell der numerischen Fehleranalyse nach Wilkinson [32] wird nach jeder elementaren arithmetischen Berechnung (Maschinenbefehl im Computer) das Ergebnis — gerundet als τ-stellige Gleitpunktzahl — gespeichert. Die Hintereinanderausführung von elementaren Operationen führt zu Fehlern, da Zwischenergebnisse der Berechnung gerundet werden. Sei im folgenden t' die in τ-stelliger Gleitpunkt-Arithmetik numerisch gewonnene Lösung für t. Der Betrag $|t' - t|$ ist der *absolute* Rundungsfehler der Gleitpunktzahl t'.

Die Givens-Rotationen $G_{i_1,j_1}(\cdot), \ldots, G_{i_k,j_k}(\cdot)$ bezeichnen wir als *disjunkt*, wenn alle Indizes der Givens-Rotationen paarweise verschieden sind. Für die Rundungsfehleranalyse gilt nach Gentleman [10]:

Lemma 1.13 *Sei $T = G_{i_1,j_1}(\cdot) \cdots G_{i_k,j_k}(\cdot)$ das Produkt von k disjunkten Givens-Rotationen und b ein Vektor. Dann gilt für die in τ-stelliger Gleitpunkt-Arithmetik numerisch gewonnene Lösung $(Tb)'$ die Fehlerabschätzung*

$$\|(Tb)' - Tb\| \leq 7 \cdot 2^{-\tau} \|b\|.$$

Für die Rundungsfehleranalyse einer elementaren Householder-Reflexion gilt nach Mennicken und Wagenführer [21, Seite 119]:

Lemma 1.14 *Sei $H_i(\cdot) \in \mathbb{R}^{d \times d}$ eine Householder-Reflexion in den Koordinaten $i, i+1, \ldots, d$. Dann gilt für die in τ-stelliger Gleitpunkt-Arithmetik numerisch gewonnene Lösung $(H_i(\cdot)b)'$ die Fehlerabschätzung*

$$\|(H_i(\cdot)b)' - H_i(\cdot)b\| \leq (5(d - i + 1) + 15)2^{-\tau} \|b\|.$$

Wir vervollständigen die numerische Fehleranalyse der Orthogonalisierung:

Lemma 1.15 *Seien $T_1, \ldots, T_\nu$ orthogonale Matrizen. Erfüllen die numerisch gewonnenen Lösungen $(T_j b)'$ die Fehlerabschätzungen*

$$\|(T_j b)' - T_j b\| \leq \eta_j \|b\|, \quad 1 \leq j \leq \nu,$$

so gilt für die numerisch gewonnene Lösung $(Q_\nu b)' := (T_\nu T_{\nu-1} \cdots T_1 b)'$:

1.) $\|(Q_\nu b)'\| \leq \prod_{j=1}^{\nu} (1 + \eta_j) \|b\|.$

2.) $\|(Q_\nu b)' - Q_\nu b\| \leq \sum_{j=1}^{\nu} \eta_j \prod_{j=1}^{\nu-1} (1 + \eta_j) \|b\|.$

Beweis. Aufgrund der Orthogonalität von T_j gilt $\|T_j b\| = \|b\|$.

$$\begin{aligned}\|(Q_\nu b)'\| &= \|(T_\nu Q_{\nu-1} b)' - T_\nu (Q_{\nu-1} b)' + T_\nu (Q_{\nu-1} b)'\| \\ &\leq \|(T_\nu Q_{\nu-1} b)' - T_\nu (Q_{\nu-1} b)'\| + \|T_\nu (Q_{\nu-1} b)'\| \\ &\leq \eta_\nu \|(Q_{\nu-1} b)'\| + \|(Q_{\nu-1} b)'\| = (1+\eta_\nu)\|(Q_{\nu-1} b)'\|.\end{aligned}$$

Die Behauptung 1. folgt durch Induktion nach ν.

Wir wenden wieder die Dreiecksungleichung an und erhalten Behauptung 2:

$$\begin{aligned}\|(Q_\nu b)' - Q_\nu b\| &\leq \|(T_\nu Q_{\nu-1} b)' - T_\nu (Q_{\nu-1} b)'\| + \|T_\nu \Big((Q_{\nu-1} b)' - Q_{\nu-1} b \Big)\| \\ &\leq \eta_\nu \|(Q_{\nu-1} b)'\| + \|(Q_{\nu-1} b)' - Q_{\nu-1} b)\| \\ &\vdots \\ &\leq \eta_\nu \|(Q_{\nu-1} b)'\| + \cdots + \eta_2 \|(Q_1 b)'\| + \eta_1 \|b\| \\ &\leq \sum_{j=1}^{\nu} \eta_j \prod_{j=1}^{\nu-1} (1+\eta_j) \|b\| \quad \text{(nach Behauptung 1.).}\end{aligned}$$

■

Sei $B = [b_1, b_2, \ldots, b_n] \in \mathbb{R}^{d \times n}$ eine Basismatrix. Gentleman [10] gibt eine Reihenfolge von $2d-3$ disjunkten Givens-Rotationen zur QR-Zerlegung an. Wir folgern aus Lemma 1.13 und Lemma 1.15:

Satz 1.16 *Zur QR-Zerlegung der Eingabematrix $B = [b_1, b_2, \ldots, b_n] \in \mathbb{R}^d$ nach Gentleman gilt für die in τ-stelliger Gleitpunkt-Arithmetik numerisch gewonnene Lösung $R' = [r'_1, r'_2, \ldots, r'_n] = (Q^T B)'$ die Fehlerabschätzung*

$$\|r'_\nu - r_\nu\| \leq (14d-3)(1 + 7 \cdot 2^{-\tau})^{(2d-3)} \cdot 2^{-\tau} \|b_\nu\|. \tag{1.16}$$

Wir approximieren den Faktor $(1 + 7 \cdot 2^{-\tau})^{(2d-3)}$ durch 1. Für die Einträge der Matrix $R' = [r'_{i,\nu}]$ gilt nach Satz 1.16 die Fehlerabschätzung

$$|r'_{i,\nu} - r_{i,\nu}| \leq \|r'_\nu - r_\nu\| \leq 14d \cdot 2^{-\tau} \cdot \|b_\nu\|. \tag{1.17}$$

Die Orthogonalisierug nach Householder wird durch eine Folge von orthogonalen Matrizen bestimmt. Die Fehleranalyse folgt mit Lemma 1.15.

Satz 1.17 *Zur QR-Zerlegung der Eingabematrix $B = [b_1, b_2, \ldots, b_n] \in \mathbb{R}^{d \times n}$ nach der Householder-Methode gilt für die in τ-stelliger Gleitpunkt-Arithmetik numerisch gewonnene Lösung $R' = [r'_1, r'_2, \ldots, r'_n] = (Q^T B)'$ die Fehlerabschätzung*

$$\|r'_\nu - r_\nu\| \leq 5\Big((d+3)\nu - \nu(\nu-1)/2 \Big) \cdot 2^{-\tau} \cdot \sigma \|b_\nu\|,$$

mit

$$\sigma := \prod_{j=1}^{\nu-1} \Big(1 + (5(d-j+1) + 15) \cdot 2^{-\tau} \Big).$$

Beweis. Nach Lemma 1.14 gilt für $j = 1, \ldots, n$ die Fehlerabschätzung

$$\|(H_j(\cdot)b_\nu)' - H_j(\cdot)b_\nu\| \leq (5(d-j+1)+15) \cdot 2^{-\tau}\|b_\nu\|.$$

Wir folgern mit Lemma 1.15

$$\begin{aligned}
\|r'_\nu - r_\nu\| &= \|\Big(H_\nu(b'_\nu)\cdots H_1(b_1)b_\nu\Big)' - H_\nu(b'_\nu)\cdots H_1(b_1)b_\nu\| \\
&\leq \sum_{j=1}^{\nu}\Big(5(d-j+1)+15\Big)\cdot 2^{-\tau}\prod_{j=1}^{\nu-1}\Big(1+(5d-j+1)\cdot 2^{-\tau}\Big)\cdot\|b_\nu\| \\
&\leq 5\Big((d+3)\nu - \nu(\nu-1)/2\Big)\cdot 2^{-\tau}\cdot\sigma\|b_\nu\|. \qquad \blacksquare
\end{aligned}$$

Für $\sigma \approx 1$ und $d \geq n$ gilt die Fehlerabschätzung

$$|r'_{i,\nu} - r_{i,\nu}| \leq \|r'_\nu - r_\nu\| \leq 2.5d^2 \cdot 2^{-\tau} \cdot \|b_\nu\|. \tag{1.18}$$

Wir fassen die Eigenschaften der hier vorgestellten Orthogonalisierungsverfahren zusammen. Sei $B = [b_1, b_2, \ldots, b_n] \in \mathbb{Z}^{d\times n}$ eine ganzzahlige Basismatrix. Bestimme durch das Rechnen in τ-stelliger Gleitpunkt-Arithmetik die Matrix $R = [r_1, r_2, \ldots, r_n] = Q^T B$.

Methode	Operationen	Fehler $\lvert r'_{j,\nu} - r_{j,\nu}\rvert$	Exakte Arithmetik
Gram-Schmidt	$dn^2 + n^3$	—–	ja
Givens	$3n^2\,(d - \frac{n}{3})$	$\leq 14d \cdot 2^{-\tau}\|b_\nu\|$	nein
Householder	$2n^2\,(d - \frac{n}{3})$	$\leq 2.5d^2 \cdot 2^{-\tau}\|b_\nu\|$	nein

Tabelle 1.1: Gegenüberstellung der Methoden zur Orthogonalisierung von ganzzahligen Gitterbasen.

Das Gram-Schmidt-Verfahren eignet sich für die Orthogonalisierung in exakter Arithmetik. Für das Rechnen mit Gleitpunk-Arithmetik bevorzugen wir aufgrund der schnelleren Berechnungszeit die Householder-Methode. Die Givens-Methode ist vorzuziehen, wenn in B nur wenige Einträge unterhalb der Diagonalen ungleich Null sind.

Kapitel 2

Gitterreduktion

Die Gitterreduktion beschäftigt sich mit der folgenden Aufgabe: Gegeben sei das Gitter $L \subset \mathbb{R}^d$. Finde eine Gitterbasis, in der die Normen der Basisvektoren möglichst klein sind, bzw. die Basisvektoren möglichst orthogonal zueinander stehen. Wir formulieren in Abschnitt 2.1 die Probleme kürzester Gittervektor, nächster Gittervektor und das Gitterreduktions-Problem. Diese Probleme sind NP-hart. Unter der Komplexitätsannahme P $\neq$ NP gibt es keinen effizienten Algorithmus der diese Probleme löst.

Wir beschreiben in Abschnitt 2.2 die Längenreduktion von Gitterbasen. In Abschnitt 2.3 beschäftigen wir uns mit dem bekannten LLL-Algorithmus von Lenstra, Lenstra und Lovász [16]. Bei Eingabebasis $b_1, b_2, \ldots, b_n \in \mathbb{Z}^d$ und $M := \max_j \|b_j\|^2$ benötigt das Verfahren $O(n^3 d \cdot \log M)$ arithmetische Schritte. Die vorgeschlagene Variante der *minimalen* LLL-Reduktion benötigt nur $O(n^2 d \cdot \log M)$ arithmetische Schritte. Als weitere Variante beschreiben wir den LLL-Algorithmus mit iterativer Orthogonalisierung. Das Verfahren eignet sich zur LLL-Reduktion in Gleitpunkt-Arithmetik.

Die Erweiterung des LLL-Algorithmus zum Block-Korkin-Zolotarev-Algorithmus von Schnorr [25, 27] wird in Abschnitt 2.4 beschrieben.

2.1 Probleme der Gitterreduktion

Als Maß für die Reduziertheit einer Gitterbasis betrachtet man die Quotienten zwischen den sukzessiven Minima und der Länge der einzelnen Vektoren.

Definition 2.1 Sei $L \subset \mathbb{R}^d$ ein Gitter vom Rang n. Das *i-te sukzessive Minimum* von L ist für $i = 1, 2, \ldots, n$ definiert durch die Zahl

$$\lambda_i = \lambda_i(L) := \min \left\{ \max_{1 \le j \le i} \|c_j\| \;\middle|\; c_1, c_2, \ldots, c_i \in L \text{ linear unabhängig} \right\}.$$

Es gilt allgemein $\lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$, und insbesondere ist $\lambda_1(L)$ die Länge eines kürzesten vom Nullvektor verschiedenen Gittervektors. Für die

geordnete Gitterbasis $b_1, b_2, \ldots, b_n$ gilt $\max\{\|b_1\|, \ldots, \|b_i\|\} \geq \lambda_i$. Ziel der Gitterreduktion ist es, die Quotienten $\|b_i\|/\lambda_i$, $1 \leq i \leq n$, zu minimieren. Reduzierte Basen $b_1, b_2, \ldots, b_n$ sind nahezu orthogonal.

Lemma 2.2 *Sei $L \subset \mathbb{R}^d$ ein Gitter mit geordneter Basis $b_1, b_2, \ldots, b_n \in \mathbb{R}^d$ und $\hat{b}_1, \hat{b}_2, \ldots, \hat{b}_n$ deren Gram-Schmidt-Orthogonalisierung, dann gilt*

$$\min_{i=1}^{n} \|\hat{b}_i\| \leq \lambda_1.$$

Beweis. Sei $b = \sum \alpha_i b_i$ ein kürzester Gittervektor, $k := \max\{j \mid \alpha_j \neq 0\}$ und seien $\mu_{i,j}$ die Gram-Schmidt-Koeffizienten. Es gilt

$$\lambda_1 = \|b\| = \left\| \sum_{i=1}^{k} \alpha_i \sum_{j=1}^{i} \mu_{i,j} \hat{b}_j \right\| = \sum_{j=1}^{k} \left\| \sum_{i=j}^{k} \alpha_i \mu_{i,j} \hat{b}_j \right\| \geq \|\hat{b}_k\| \geq \min_i \|\hat{b}_i\|.$$

■

Ein weiteres Maß für die Reduziertheit einer Gitterbasis ist ihr Orthogonalitätsdefekt:

Definition 2.3 Der *Orthogonalitätsdefekt* der Gitterbasis $b_1, b_2, \ldots, b_n$ ist definiert als

$$\mathrm{OrthDefect}(b_1, b_2, \ldots, b_n) := \frac{\prod_{i=1}^{n} \|b_i\|}{\det L(b_1, b_2, \ldots, b_n)}.$$

Sei $[b_1^*, b_2^*, \ldots, b_n^*] = (B^{-1})^T$ die durch $B = [b_1, b_2, \ldots, b_n]$ definierte Basismatrix des dualen Gitters. Der *duale Orthogonalitätsdefekt* von B ist erklärt als

$$\mathrm{OrthDefect}^*(b_1, b_2, \ldots, b_n) := \mathrm{OrthDefect}(b_1^*, b_2^*, \ldots, b_n^*).$$

Optimierungsprobleme der Gitterreduktion. Wir beschreiben Probleme der Gitterreduktion für die euklidische Norm. Gegeben sei ein Gitter $L \subset \mathbb{R}^d$ mit Basis $b_1, b_2, \ldots, b_n$.

- **Shortest-Vector-Problem (SVP)** Bestimme einen Gittervektor $b \in L$, mit $\|b\| = \lambda_1(L)$.
- **Closest-Vector-Problem (CVP)** Bestimme zu einem gegebenem Punkt $p \in \mathbb{R}^d$ einen Gittervektor b, der den Abstand $\|b - p\|$ minimiert.
- **Basis-Reduction-Problem (BRP)** Bestimme eine Gitterbasis $b_1, b_2, \ldots, b_n$ für L, mit

$$\prod_{j=1}^{n} \|b_j\| = \min_{L = L(b_1', b_2', \ldots, b_n')} \left(\prod_{j=1}^{n} \|b_j'\| \right).$$

Für das lange Zeit offene Problem der Komplexität des SVP zeigte Ajtai 1997 [1]: das SVP-Problem ist unter randomisierten Reduktionen NP-hart. Micciancio [20] verschärfte die Aussage: Es ist NP-hart, einen Gittervektor der Länge kleiner als $\sqrt{2} \cdot \lambda_1(L)$ zu bestimmen.

Auch das CVP ist NP-hart (van Emde Boas '81 [9]). Dinur, Kindler und Safra [7] zeigten, daß die Approximation des CVP bis auf einen Faktor kleiner $n^{O(1/\log\log n)}$ NP-hart ist. Das BRP ist ebenfalls NP-hart (siehe Lovász [17]).

2.2 Längenreduktion

Die Längenreduktion von Gitterbasen ist ein erster Ansatz zur Gitterreduktion im euklidischen Raum. Sei $R = [r_1, r_2, \ldots, r_n] = Q^T B$ die zu $B = [b_1, b_2, \ldots, b_n]$ isometrische Basismatrix der QR-Zerlegung. R ist die obere Dreiecksmatrix

$$R = \begin{bmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,n} \\ 0 & r_{2,2} & \cdots & r_{2,n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & r_{n,n} \end{bmatrix} = \begin{bmatrix} \|\hat{b}_1\| & & 0 \\ & \ddots & \\ 0 & & \|\hat{b}_n\| \end{bmatrix} \cdot [\mu_{i,j}]^T$$

Durch unimodulare Transformationen rückwärts in den Spalten $j = \nu - 1, \ldots, 1$

$$r_\nu := r_\nu - \lceil \mu_{\nu,j} \rfloor r_j$$

erreichen wir, daß $|r_{j,\nu}/r_{j,j}| = |\mu_{\nu,j}| \leq 1/2$ gilt. Die Orthogonalbasis bleibt bei der Transformation unverändert. Wir übertragen diese Transformation auf die Gitterbasis B:

Definition 2.4 Eine geordnete Basis $b_1, b_2, \ldots, b_n$ heißt *längenreduziert*, wenn:

$$|\mu_{\nu,j}| \leq 1/2, \quad 1 \leq j < \nu \leq n.$$

Jede Gitterbasis läßt sich effizient in eine längenreduzierte Basis des gleichen Gitters transformieren. Der Algorithmus LRED(ν) führt die Längenreduktion auf dem Basisvektor b_ν aus.

Algorithmus 2.1 LRED(ν) Längenreduktion

`INPUT:` Gitterbasis $b_1, b_2, \ldots, b_\nu$, Gram-Schmidt-Koeffizienten $[\mu_{i,j}]_{1 \le i,j \le \nu}$

```
FOR (j = ν − 1, ..., 1) DO
   IF (0 ≠ μ := ⌈μ_{ν,j}⌋) THEN
      FOR (i = 1, ..., j) DO μ_{ν,i} := μ_{ν,i} − μ · μ_{j,i}
      b_ν := b_ν − μ · b_j
   END IF
```

`OUTPUT:` Längenreduzierter Vektor b_ν.

Bemerkungen.

1. Der Transformationsschritt $b_\nu := b_\nu - \lceil \mu_{\nu,j} \rfloor b_j$ verändert nur die Gram-Schmidt-Koeffizienten $\mu_{\nu,1}, \ldots, \mu_{\nu,j}$. Nach der Transformation gilt $|\mu_{\nu,j}^{neu}| \le 1/2$. Hieraus folgt die Korrektheit von LRED($\cdot$).

2. Die Längenreduktion von b_ν benötigt höchstens $2\nu d + \nu^2$ arithmetische Operationen. Zur Längenreduktion der Basis $b_1, b_2, \ldots, b_n$ werden höchstens $\sum_{\nu=2}^{n} 2\nu d + \nu^2 \approx n^2 d + \frac{1}{3} n^2$ arithmetische Operationen benötigt.

3. Für längenreduzierte Basisvektoren gilt die Abschätzung

$$\|b_\nu\|^2 = \sum_{j=1}^{\nu} \mu_{\nu,j}^2 \|\hat{b}_j\|^2 \le \frac{1}{4} \sum_{j=1}^{\nu-1} \|\hat{b}_j\|^2 + \|\hat{b}_\nu\|^2. \tag{2.1}$$

Die Längenreduktion verändert die Orthogonalbasis $\hat{b}_1, \hat{b}_2, \ldots, \hat{b}_n$ nicht, d.h. die Ordnung der Basisvektoren bleibt erhalten. Deshalb ist die Längenreduktion nur ein schwacher Reduktionsbegriff. Zum Beispiel ist für $0 < \epsilon < 1$ die Basis $\begin{bmatrix} \epsilon & 0 \\ 2 & 1 \end{bmatrix}$ längenreduziert. Der kürzeste Basisvektor hat die Länge ϵ.

2.3 LLL-Reduktion

Zur Gitterreduktion in der euklidischen Norm wurde 1982 von Lenstra, Lenstra und Lovász der LLL-Algorithmus vorgestellt [16].

2.3.1 Einführung

Sei $b_1, b_2, \ldots, b_n \in \mathbb{R}^d$ eine geordnete Gitterbasis. Das Gram-Schmidt-Verfahren bestimmt die Orthogonalbasis $\hat{b}_1, \hat{b}_2, \ldots, \hat{b}_n$ und die Gram-Schmidt-Koeffizienten $\mu_{i,j}$. Wir führen den Begriff der LLL-Reduktion ein:

Definition 2.5 (Lenstra, Lenstra, Lovász 1982) Eine längenreduzierte Gitterbasis $b_1, b_2, \ldots, b_n \in \mathbb{R}^d$ ist *LLL-reduziert* zum Reduktionsparameter $\delta \in (\frac{1}{4}, 1)$, wenn für $\nu = 2, 3, \ldots, n$ gilt:

$$\delta \cdot \|\hat{b}_{\nu-1}\|^2 \leq \mu_{\nu,\nu-1}^2 \cdot \|\hat{b}_{\nu-1}\|^2 + \|\hat{b}_\nu\|^2. \qquad \text{(Lovász}-\text{Bedingung)}$$

Die orthogonale Projektion $\pi_{\nu-1} : \mathbb{R}^d \longrightarrow \text{span}(b_1, \ldots, b_{\nu-2})^\perp$ beschreibt das Rang-2 Gitter $L(\pi_{\nu-1}(b_{\nu-1}), \pi_{\nu-1}(b_\nu))$. Die Lovász-Bedingung ist äquivalent zu

$$\delta \cdot \|\pi_{\nu-1}(b_{\nu-1})\|^2 \leq \|\pi_{\nu-1}(b_\nu)\|^2.$$

LLL-reduzierte Basen approximieren die Größen λ_j bis auf einen exponentiellen Faktor:

Satz 2.6 (Lenstra, Lenstra, Lovász [16]) *Sei $b_1, b_2, \ldots, b_n \in \mathbb{R}^d$ eine mit $\delta \in (\frac{1}{4}, 1)$ LLL-reduzierte Gitterbasis. Dann gilt für $\alpha = (\delta - \frac{1}{4})^{-1}$:*

1. $\alpha^{1-j} \leq \|\hat{b}_j\|^2 \lambda_j^{-2}$ *für* $j = 1, \ldots, n$.
2. $\|b_j\|^2 \lambda_j^{-2} \leq \alpha^{n-1}$ *für* $j = 1, \ldots, n$.
3. $\|b_k\|^2 \leq \|\hat{b}_j\|^2 \alpha^{j-1}$ *für* $k \leq j$.

2.3.2 Der LLL-Reduktionsalgorithmus von Lovász

Der Algorithmus von Lovász transformiert eine ganzzahlige Gitterbasis in eine LLL-reduzierte Basis.

Algorithmus 2.2 Lovász-Verfahren zur LLL-Reduktion

INPUT: Gitterbasis $b_1, b_2, \ldots, b_n \in \mathbb{Z}^d$, Reduktionsparameter δ

1. $\nu := 2$ /* ν ist die Stufe der Reduktion */
2. WHILE $(\nu \leq n)$ DO
2.1 LRED(ν) /* Längenreduktion */
2.2 IF $(\delta \cdot \|\hat{b}_{\nu-1}\|^2 > \mu_{\nu,\nu-1}^2 \cdot \|\hat{b}_{\nu-1}\|^2 + \|\hat{b}_\nu\|^2)$ THEN
vertausche b_ν mit $b_{\nu-1}$ /* LLL-Austausch */
$\nu := \max(\nu - 1, 2)$
ELSE
$\nu := \nu + 1$
END IF
END WHILE

OUTPUT: LLL-reduzierte Basis $b_1, b_2, \ldots, b_n$.

Korrektheit. Durch Induktion nach $\nu = 2, 3, \ldots, n+1$ zeigt man, daß zu Beginn von Stufe ν die Teilbasis $b_1, b_2, \ldots, b_{\nu-1}$ LLL-reduziert ist. Die Reduktion hält in Stufe $n+1$.

Bemerkung 2.7 Der LLL-Austausch $b_{\nu-1} \leftrightarrow b_\nu$ in Schritt 2.2 verändert die Längen $\|\hat{b}_{\nu-1}\|^2, \|\hat{b}_\nu\|^2$, sowie die Koeffizienten $\mu_{\nu,\nu-1}$ und $\mu_{j,\nu-1}, \mu_{j,\nu}$ für $j = \nu+1, \ldots, n$. Betrachte die zu $B = [b_1, b_2, \ldots, b_n]$ isometrische Basismatrix $R = [r_1, r_2, \ldots, r_n]$ der QR-Zerlegung. Nach dem Vektortausch gilt

$$R' := [r_1, \ldots, r_\nu, r_{\nu-1}, \ldots, r_n] \cong_{\text{iso}} [b_1, \ldots, b_\nu, b_{\nu-1}, \ldots, b_n] =: B'$$

mit

$$R' = \begin{bmatrix} r_{1,1} & \cdots & r_{1,\nu} & r_{1,\nu-1} & \cdots & r_{1,n} \\ 0 & \ddots & \vdots & \vdots & & \vdots \\ \vdots & & r_{\nu-1,\nu} & r_{\nu-1,\nu-1} & \cdots & r_{\nu-1,n} \\ \vdots & & r_{\nu,\nu} & 0 & \cdots & r_{\nu,1} \\ \vdots & & & & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & 0 & r_{n,n} \end{bmatrix}.$$

Die einzelne Givens-Rotation $R^{\text{neu}} := G_{\nu-1,\nu}(r_\nu) \cdot R'$ berechnet die obere Dreicksmatrix $R^{\text{neu}} \cong_{\text{iso}} B'$. Also

$$\text{sign}(r_{i,i}^{\text{neu}}) \cdot r_{i,j}^{\text{neu}} = \mu_{j,i}^{\text{neu}} \cdot \|\hat{b}_i^{\text{neu}}\| \quad \text{für } i = \nu-1, \nu \text{ und } j = \nu, \ldots, n.$$

Diese Givens-Rotation bestimmt die Berechnungen

$$\|\hat{b}_{\nu-1}^{\text{neu}}\|^2 = \mu_{\nu,\nu-1}^2\|\hat{b}_{\nu-1}\|^2 + \|\hat{b}_\nu\|^2, \tag{2.2}$$

$$\|\hat{b}_\nu^{\text{neu}}\|^2 = \frac{\|\hat{b}_{\nu-1}\|^2\|\hat{b}_\nu\|^2}{\|\hat{b}_{\nu-1}^{\text{neu}}\|^2}, \tag{2.3}$$

$$\mu_{\nu,\nu-1}^{\text{neu}} = \mu_{\nu,\nu-1} \cdot \frac{\|\hat{b}_{\nu-1}\|^2}{\|\hat{b}_{\nu-1}^{\text{neu}}\|^2} \tag{2.4}$$

und für $j = \nu+1, \nu+2, \ldots, n$

$$\mu_{j,\nu-1}^{\text{neu}} = \mu_{\nu,\nu-1}\mu_{j,\nu-1} \cdot \frac{\|\hat{b}_{\nu-1}\|^2}{\|\hat{b}_{\nu-1}^{\text{neu}}\|^2} + \mu_{j,\nu} \cdot \frac{\|\hat{b}_\nu\|^2}{\|\hat{b}_{\nu-1}^{\text{neu}}\|^2}, \tag{2.5}$$

$$\mu_{j,\nu}^{\text{neu}} = -\mu_{j,\nu-1} + \mu_{\nu,\nu-1}\mu_{j,\nu}. \tag{2.6}$$

Die Neuberechnung der Längen $\|\hat{b}_{\nu-1}\|^2$, $\|\hat{b}_{\nu-1}\|^2$ und der Gram-Schmidt-Koeffizienten nach einem LLL-Austausch benötigt demnach $O(n)$ arithmetische Operationen.

Analyse. Sei K die Anzahl der Schleifendurchläufe in Schritt 2. Wir sagen der LLL-Test in Schritt 2.2 ist *positiv*, wenn

$$\delta \cdot \|\hat{b}_{\nu-1}\|^2 > \mu_{\nu,\nu-1}^2\|\hat{b}_{\nu-1}\|^2 + \|\hat{b}_\nu\|^2.$$

Sei K^+ die Anzahl der positiven LLL-Tests und K^- die Anzahl der negativen LLL-Tests während der Reduktion. Es gilt $K = K^+ + K^-$. Jeder negative LLL-Test erhöht die Stufe der Reduktion ν um 1 und jeder positive LLL-Test verringert ν maximal um 1. Es folgt $K^- \leq K^+ + \nu - 1$. Am Ende der LLL-Reduktion gilt:

$$K \leq K^+ + K^- \leq 2K^+ + n. \tag{2.7}$$

Lemma 2.8 *Zur Eingabebasis $b_1, b_2, \ldots, b_n$ mit $M := \max_j \|b_j\|^2$ und Reduktionsparameter δ ist die Anzahl K^+ der positiven LLL-Tests beschränkt durch*

$$K^+ \leq \frac{1}{2}\, n(n-1) \log_{1/\delta} M.$$

Beweis. Siehe Daudé und Vallée [6]. ∎

Satz 2.9 *Sei $b_1, b_2, \ldots, b_n \in \mathbb{Z}^d$ eine Eingabebasis mit $M \geq \max_j \|b_j\|^2$. Das Lovász-Verfahren (Algorithmus 2.2) benötigt $O(dn^3 \log M)$ arithmetische Operationen.*

Beweis. Für die Anzahl $K \leq 2K^+ + n$ der Schleifendurchläufe in Schritt `2` gilt mit Lemma 2.8 $K \leq n^2 \log M + n$. Jeder Schleifendurchlauf in Schritt `2` längenreduziert einen Basisvektor (Schritt `2.1`) und führt einen LLL-Test aus (Schritt `2.2`). Die Längenreduktion kostet $O(nd)$ Schritte, und nach Bemerkung 2.7 kostet ein LLL-Test $O(n)$ arithmetische Schritte. Es folgt die Behauptung. ∎

Für das Lovász-Verfahren gelten die folgenden Ergebnisse (s. Schnorr [29]):

Lemma 2.10 *Sei $b_1, b_2, \ldots, b_n \in \mathbb{Z}^d$ mit $M := \max_j \|b_j\|^2$ eine Eingabebasis für das Lovász-Verfahren (Algorithmus 2.2) zum Reduktionsparameter $\delta \in (\frac{1}{4}, 1)$. Dann gilt mit $\alpha := (\delta - \frac{1}{4})^{-1}$:*

1. *$\max_j \|\hat{b}_j\|$ fällt monoton und $\min_j \|\hat{b}_j\|$ wächst monoton.*
2. *$\|b_i\|^2 \leq \frac{i+3}{4} M$.*
3. *In Stufe ν der Reduktion gilt für $j < \nu$:*

$$\|\hat{b}_j\|^2 \geq \|b_j\|^2 \cdot \alpha^{1-j} \quad \text{und} \quad |\mu_{i,j}|^2 \leq \frac{i+3}{4} M \cdot \alpha^{j-1}.$$

4. *Während der Längenreduktion von b_ν gilt $|\mu_{\nu,j}|^2 \leq \frac{\nu+3}{4} M \left(\frac{9\alpha}{4}\right)^{\nu-1}$.*
5. *In Stufe ν der Reduktion gilt für $j \geq \nu$: $|\mu_{i,j}|^2 \leq \frac{i+3}{4} M^j$.*

Da das Lovász-Verfahren in der Stufe ν den Basisvektor b_ν längenreduziert, folgen die Abschätzungen 2 bis 5.

Korollar 2.11 (Lenstra, Lenstra, Lovász [16]) *Sei $b_1, b_2, \ldots, b_n \in \mathbb{Z}^d$ mit $M = \max_j \|b_j\|^2$ eine Eingabebasis für das Lovász-Verfahren. Die Bitlänge der rationalen Zahlen $\|\hat{b}_j\|^2$ und $\mu_{i,j}$ ist beschränkt durch $O(n \cdot \log M)$.*

2.3.3 Minimale LLL-Reduktion

Die Längenreduktion in Schritt `2.1` des Lovász-Verfahrens kostet in Stufe ν $O(d\nu)$ arithmetische Schritte, während im Austauschschritt `2.2` $O(n)$ arithmetische Schritte zur Korrektur der Gram-Schmidt-Koeffizienten notwendig sind. Wir beschleunigen das Verfahren, indem wir die Längenreduktion in Schritt `2.1` ersetzen:

`2.1'` `IF` $(0 \neq \mu = \lceil \mu_{\nu,\nu-1} \rfloor)$ `THEN`
$b_\nu := b_\nu - \mu \cdot b_{\nu-1}$
$\mu_{\nu,j} := \mu_{\nu,j} - \mu \cdot \mu_{\nu-1,j}, \quad j = 1, \ldots, \nu - 1$
`END IF`

Ein Schleifendurchlauf in Schritt 2 des modifizierten Lovász-Verfahren kostet $O(n)$ arithmetische Operationen. Wir führen den Begriff der minimalen LLL-Reduktion ein:

Definition 2.12 Die Gitterbasis $b_1, b_2, \ldots, b_n \in \mathbb{Z}^d$ ist *minimal LLL-reduziert* mit $\delta \in (\frac{1}{4}, 1)$, falls für $\nu = 2, 3, \ldots, n$ gilt:

(1) $|\mu_{\nu,\nu-1}| \leq \frac{1}{2}$.

(2) $\delta \cdot \|\hat{b}_{\nu-1}\|^2 \leq \mu_{\nu,\nu-1}^2 \|\hat{b}_{\nu-1}\|^2 + \|\hat{b}_\nu\|^2$.

Offensichtlich ist die Ausgabebasis des modifizierten Lovász-Algorithmus minimal LLL-reduziert. Durch die anschließende Längenreduktion erhalten wir eine LLL-reduzierte Gitterbasis. Bei einer Eingabebasis $b_1, b_2, \ldots, b_n \in \mathbb{Z}^d$ mit $M = \max_j \|b_j\|^2$ beträgt die Laufzeit jetzt $O(n^2 d \cdot \log M)$.

Bemerkungen 2.13

1. Die Schranken 2 bis 5 aus Lemma 2.10 gelten für den Verlauf der minimalen LLL-Reduktion nicht. Die Bitlängen der Zahlen $\|\hat{b}_j\|^2$ und $\mu_{i,j}$ sind deshalb während der minimalen LLL-Reduktion nicht durch $O(n \cdot \log M)$ beschränkt.

2. Interessant ist die minimale LLL-Reduktion, wenn die Eingabebasis längenreduziert ist (d.h. $|\mu_{i,j}| \leq 1/2$). Wir erwarten, daß während der minimalen LLL-Reduktion die Beträge der Gram-Schmidt-Koeffizienten $\mu_{i,j}$ zunächst nur langsam wachsen (vergleiche auch Bemerkung 2.7).

2.3.4 Lovász-Verfahren mit iterativer Orthogonalisierung

Wir geben eine weitere Variante des Lovász-Verfahrens an: Zu Beginn der Reduktionsstufe ν erfolgt die Orthogonalisierung für den Vektor b_ν.

Algorithmus 2.3 Lovász-Verfahren mit iterativer Orthogonalisierung

INPUT: Gitterbasis $b_1, b_2, \ldots, b_n \in \mathbb{Z}^d$, Reduktionsparameter δ

```
1.  ν := 1        /* ν ist die Stufe der Reduktion    */
2.  WHILE (ν ≤ n) DO
2.1    ORT_GS(ν)                          /* siehe Algorithmus 1.2 */
2.2    LRED(ν)                            /* Längenreduktion */
2.3    IF (ν > 1) AND (δ · ||b̂_{ν-1}||^2 > μ^2_{ν,ν-1} · ||b̂_{ν-1}||^2 + ||b̂_ν||^2) THEN
          vertausche b_ν mit b_{ν-1}      /* LLL-Austausch */
          ν := ν - 1
       ELSE
          ν := ν + 1
       END IF
    END WHILE
```

OUTPUT: LLL-reduzierte Basis $b_1, b_2, \ldots, b_n$.

Bemerkungen

1. Die Gram-Schmidt-Orthogonalisierung in Schritt 2.1 benötigt $O(nd)$ arithmetische Schritte (vergleiche Algorithmus 1.2). Die Korrektur der Gram-Schmidt-Koeffizienten nach einen LLL-Austauschschritt (siehe Bemerkung 2.7) ist nicht notwendig.

2. Für eine Eingabebasis $b_1, b_2, \ldots, b_n$ mit $M = \max_j \|b_j\|^2$ beträgt die Laufzeit des Lovász-Algorithmus mit iterativer Orthogonalisierung weiter $O(n^3 d \cdot \log M)$ (vergleiche Satz 2.9).

3. Für die iterative Orthogonalisierung gelten in Lemma 2.10 die Aussagen 1 bis 4. Die Aussage 5: $|\mu_{i,j}| \leq \frac{i+3}{4} M^j$ für $j > \nu$ ist für die iterative LLL-Reduktion nicht bedeutsam, weil diese Zahlen in Stufe ν nicht gespeichert werden.

2.3.5 LLL-Reduktion in Gleitpunkt-Arithmetik

Das Lovász-Verfahren mit iterativer Orthogonalisierung eignet sich gut zur Reduktion in Gleitpunkt-Arithmetik. Dazu wird die exakte Arithmetik auf den rationalen Zahlen $\mu_{i,j}$ und $\|\hat{b}_j\|^2$ durch die τ-stellige Gleitpunkt-Arithmetik auf den reellwertigen Zahlen $r_{j,i} = \mu_{i,j}\|\hat{b}_j\|$ ersetzt. Die Transformationen auf der ganzzahligen Gitterbasis $b_1, b_2, \ldots, b_n \in \mathbb{Z}^d$ werden weiter in exakter Arithmetik ausgeführt. Zur Reduktion ersetzen wir Schritt 2.1 und 2.2 in Algorithmus 2.3 durch

```
2.1'   ORT_HS(ν)            /* Householder Transformation */
2.2'   IF (∃j)(|μ_{ν,j}| > 1/2) THEN
         LRED(ν)            /* Längenreduktion */
         ORT_HS(ν)
       END IF
```

Bemerkungen 2.14

1. Nach Tabelle 1.1 gilt für die in τ-stelliger Gleitpunkt-Arithmetik berechneten Koeffizienten $(r_{i,j})' = (\mu_{j,i}\|\hat{b}_i\|)'$ die Fehlerabschätzung $|r'_{i,j} - r_{i,j}| \leq 2.5d^2 2^{-\tau}\|b_j\|$. Aus Lemma 2.10 folgt, daß für die numerisch stabile Durchführung der LLL-Reduktion in Gleitpunkt-Arithmetik $\tau = O(n + \log M)$ hinreichend ist (vergleiche auch Schnorr [26]).

2. Die Orthogonalisierung in Schritt `2.2` wird nach der Längenreduktion wiederholt, um die Gram-Schmidt-Koeffizienten zu *korrigieren.* Wir verdeutlichen die Korrektur anhand eines Beispiels:

   Sei $\mu_{\nu,\nu-1} = 1245.2343$ auf 8 Dezimalstellen gerundet. Die Längenreduktion $\mu_{\nu,\nu-1} := \mu_{\nu,\nu-1} - \lceil \mu_{\nu,\nu-1} \rfloor = 0.2343$ löscht die ersten 4 Dezimalstellen. Damit verringert sich die Präzision der Gleitpunktzahl $\mu_{\nu,\nu-1}$. Am Ende der Längenreduktion werden die Gram-Schmidt-Koeffizienten $\mu_{\nu,j}$, $j = 1, \ldots, \nu - 1$ durch die erneute Orthogonalisierung korrigiert.

LLL-Reduktion mit Gleitpunktzahlen kleiner Bitlänge Wir implementieren das Lovász-Verfahren mit Gleitpunkt-Arithmetik bei fester Bitlänge τ. Die Bitlänge τ erreicht nicht den Wert $n + \log M$ (Vergleiche auch Bemerkung 2.14). Wir bemerken:

1. Die Koeffizienten $r_{i,j} = \mu_{j,i}\|\hat{b}_i\|$ für $j = i, \ldots, n$ sind *skaliert* durch die Länge $\|\hat{b}_i\|$. Ist der Basisvektor b_j längenreduziert, so gilt $|r_{i,j}| \leq \|\hat{b}_j\|$.

   Nach der Fehlerabschätzung zur Orthogonalisierung nach Householder (siehe Tabelle 1.1) gilt für den in τ-stelliger Gleitpunkt-Arithmetk berechneten Wert $r'_{i,j}$ die Fehlerabschätzung $|r'_{i,j} - r_{i,j}| < 2.5d^2\|b_j\| \cdot 2^{-\tau}$. Wir können heuristisch annehmen: Je kleiner im Verhältnis zu $\|b_j\|$ die Länge $\|\hat{b}_i\|$ ist, desto größer ist der relative Rundungsfehler von $r'_{i,j}$.

   Für die numerische Stabilität sind hier starke Reduktionsparameter vorzuziehen.

2. Für die LLL-Reduktion mit Gleitpunktzahlen wird angenommen, daß die zunächst ungenau berechneten Koeffizienten $r'_{i,j}$ durch die im iterativen Lovász-Verfahren wiederholte Orthogonalisierung im Verlauf der Reduktion an Präzision gewinnen.

2.4 Block-Korkin-Zolotarev-Reduktion

Sei $L = L(B) \subset \mathbb{R}^d$ ein Gitter mit Basismatrix $B = [b_1, b_2, \ldots, b_n] \in \mathbb{R}^{d \times n}$. Seien $\pi_i : \mathbb{R}^n \longrightarrow \text{span}(b_1, \ldots, b_{i-1})^\perp$ die orthogonalen Projektionen. Wir bezeichnen mit L_i das Gitter $\pi_i(L)$ mit Rang $n - i + 1$.

Die längenreduzierte Gitterbasis $b_1, b_2, \ldots, b_n$ heißt *Korkin-Zolotarev*-reduziert, wenn $\|\hat{b}_i\| = \lambda_1(L_i)$ für $i = 1, \ldots, n$ gilt. Für Korkin-Zolotarev-reduzierte Basen gilt die Abschätzung

$$\frac{4}{i+3} \leq \frac{\|b_i\|^2}{\lambda_i^2} \leq \frac{i+3}{4}.$$

Die Berechnung einer Korkin-Zolotarev-Basis ist nach Ajtai [1] NP-hart. Für LLL-reduzierte Gitterbasen $b_1, b_2, \ldots, b_\beta$ mit konstantem Rang kann der kürzeste Gittervektor $b = \sum_{i=1}^{\beta} \bar{u}_i b_i$ durch die Enumeration ganzzahliger Linearkombinationen aus $b_1, b_2, \ldots, b_\beta$ in $\beta^{\beta/2+O(\beta)}$ arithmetischen Schritten berechnet werden [25]. Dazu wird das Längenquadrat der folgenden Gleichung über die Menge der Koeffizientenvektoren $u \in \mathbb{Z}^\beta \setminus 0$ minimiert:

$$\|[b_1, \ldots, b_\beta] \cdot u\|^2 = \sum_{s=1}^{\beta} \left(\sum_{i=s}^{\beta} u_i \mu_{i,s} \right)^2 \|\hat{b}_s\|^2$$

Definition 2.15 (Schnorr [28]) Die längenreduzierte Gitterbasis $b_1, b_2, \ldots, b_n$ heißt *Block-Korkin-Zolotarev*-reduziert (BKZ-reduziert) zur Blocklänge $\beta \in \{2, 3, \ldots, n-1\}$, und dem Reduktionsparameter $\delta \in (\frac{1}{4}, 1)$, wenn

$$\delta \cdot \|\hat{b}_i\|^2 \leq \lambda_1 \left(L_i(b_1, b_2, \ldots, b_{\min(i+\beta-1,n)}) \right)^2.$$

Für $\beta = 2$ ist die Definition äquivalent mit dem Reduktionsbegriff der LLL-Reduktion. In [28] wird ein Algorithmus zur BKZ-Reduktion vorgestellt. Es zeigt sich in der Praxis, daß für kleine Blockweiten der BKZ-Algorithmus, im Vergleich zur LLL-Reduktion, bessere Reduktionsergebnisse liefert. Es ist allerdings offen, ob dieser Algorithmus polynomielle Laufzeit hat.

Kapitel 3

Gitterreduktion in Blöcken

Wir unterteilen die Gitterbasis $b_1, b_2, \ldots, b_n$ in Blöcke der Länge k. Innerhalb eines Blockes $b_{s+1}, b_{s+2}, \ldots, b_{s+k}$ führen wir die Reduktion in *lokalen* Koordinaten $\mu_{s+i,s+j}\|\hat{b}_{s+j}\|$, $1 \leq i, j \leq k$, durch. Der Vorteil ist, daß lokal mit Gittervektoren in $\mathbb{R}^k$ gerechnet wird, während die globalen Gittervektoren d Einträge haben. Im Anschluß an die Reduktion in lokalen Koordinaten wird die reduzierende Transformation auf den Block übertragen. Durch die Längenreduktion von $b_{s+1}, b_{s+2}, \ldots, b_{s+k}$ wird die Reduktion vervollständigt. Die Reduktion in lokalen Koordinaten wurde von Schönhage [30] im Rahmen der *Semi-Reduktion* von Gitterbasen eingeführt.

3.1 Reduktion in lokalen Koordinaten

Wir beschreiben die Reduktion von $b_{s+1}, b_{s+2}, \ldots, b_{s+k}$ in lokalen Koordinaten. Dazu setzen wir voraus, daß diese Basisvektoren längenreduziert sind.

Sei $B = [b_1, b_2, \ldots, b_n] \in \mathbb{Z}^{d \times n}$ die Basismatrix für das Gitter L und $\pi_i : \mathbb{R}^n \longrightarrow \text{span}(b_1, \ldots, b_{i-1})^\perp$ die i-te orthogonale Projektion auf den zu $\text{span}(b_1, \ldots, b_{i-1})$ senkrechten Teilraum. Die QR-Zerlegung von B bestimmt die isometrische Basismatrix $R = Q^T B = [r_{i,j}]_{1 \leq i,j \leq n}$ mit $r_{i,j} = \mu_{j,i}\|\hat{b}_i\|$. Wir bezeichnen mit $R^{(s,k)} := [r_{i,j}]_{s+1 \leq i,j \leq s+k}$ die $k \times k$-Matrix aus der Diagonalen von

$$R = \begin{bmatrix} r_{1,1} & & & \cdots & & & r_{1,n} \\ & \ddots & & & & & \\ & & \underbrace{\begin{bmatrix} r_{s+1,s+1} & \cdots & r_{s+1,s+k} \\ & \ddots & \vdots \\ 0 & & r_{s+k,s+k} \end{bmatrix}}_{=:R^{(s,k)}} & & & \vdots & \\ & & & & & & \\ & & & & & \ddots & \\ 0 & & & & & & r_{n,n} \end{bmatrix}.$$

Die Basen $\pi_{s+1}(b_{s+1}), \ldots, \pi_{s+1}(b_{s+k})$ und $R^{(s,k)}$ sind nach Lemma 1.9 isometrisch. Die Reduktion in lokalen Koordinaten ist hier die minimale LLL-Reduktion der Spalten in $R^{(s,k)}$ (vergleiche Kapitel 2.3). Bei der Reduktion von $R^{(s,k)}$ werden alle Transformationen in der unimodularen $k \times k$-Matrix T mitgeführt (zu Beginn der Reduktion ist T die Einheitsmatrix). Wir übertragen die Reduktion in lokalen Koordinaten anschließend durch die Transformation $[b_{s+1}, \ldots, b_{s+k}] := [b_{s+1}, \ldots, b_{s+k}] \cdot T$ auf das Gitter.

Algorithmus 3.1 $LLL_\delta(s,k)$ lokale, minimale LLL-Reduktion

---

`EINGABE:` $k > 1$, δ, $b_{s+1}, \ldots, b_{s+k} \in \mathbb{Z}^d$ längenreduziert, $R^{(s,k)}$

`1.` $T = [t_1, t_2, \ldots, t_k] := I_k$, $R' = [r'_1, r'_2, \ldots, r'_k] := R^{(s,k)}$, $\nu := 2$
`2.` `WHILE` $(\nu \leq k)$ `DO`
`2.1` `IF` $(0 \neq \mu := \lceil r'_{j,\nu} / r'_{j,j} \rfloor)$ `THEN`
$t_\nu := t_\nu - \mu \cdot t_{\nu-1}$, `/*` minimale Längenredunktion `*/`
$r'_\nu := r'_\nu - \mu \cdot r'_{\nu-1}$
`2.2` `IF` $(\delta \cdot r'^2_{\nu-1,\nu-1} \geq r'^2_{\nu-1,\nu} + r'^2_{\nu,\nu})$ `THEN`
Vertausche t_ν mit $t_{\nu-1}$ und r'_ν mit $r'_{\nu-1}$
$R' := GI_{\nu-1,\nu}(r'_{\nu-1}) \cdot R'$ `/*` Givens-Rotation `*/`
$\nu := \max(2, \nu - 1)$
`ELSE`
$\nu := \nu + 1$
`END IF`
`END WHILE`
`3.` $[b_{s+1}, b_{s+2}, \ldots, b_{s+k}] := [b_{s+1}, b_{s+2}, \ldots, b_{s+k}] \cdot T$ `/*` Update `*/`

`AUSGABE:` Lokal, minimal LLL-reduzierter Block $b_{s+1}, \ldots, b_{s+k}$.

Bemerkungen.

1. Nach einen Basisvektortausch erfolgt in Schritt `2.2` die Aktualisierung der Koeffizienten $r'_{\nu-1,j}$ und $r'_{\nu,j}$ für $j = \nu - 1, \ldots k$ mit Hilfe einer elementaren Givens-Rotation in den Koordinaten $\nu - 1, \nu$ (vergleiche Bemerkung 2.7). Dazu wird der Eintrag $r'_{\nu-1,\nu}$ nach Null rotiert.

2. Die Koeffizienten $r_{i,j}$ sind im allgemeinen nicht rational. Eine Reduktion in exakter Arithmetik kann unter Verwendung der rationalen Koeffizienten $\|\hat{b}_j\|^2$ und $\mu_{i,j}$ durchgeführt werden.

Bemerkung 3.1 Jeder Schleifendurchlauf in Schritt 2 kostet $O(k)$ arithmetische Schritte. Die Ausführung von Schritt `1` und Schritt `3` benötigt $O(k^2 d)$ arithmetische Schritte.

Lemma 3.2 *Sei $\mathcal{A}$ ein Reduktionsalgorithmus, der zur Eingabebasis $b_1, b_2, \ldots, b_n \in \mathbb{Z}^d$ mit $M = \max_j \|b_j\|^2$ intern nur Transformationen zur Längenreduktion ausführt. Wenn $\mathcal{A}$ C-mal Algorithmus 3.1 $LLL_\delta(\cdot, k)$ mit Blockweite $k < n$ aufruft, dann beträgt die Gesamtlaufzeit der Ausführungen von $LLL_\delta(\cdot, k)$*

$$O\Big(C \cdot (k^2 d) + kn^2 \cdot \log M\Big)$$

arithmetische Schritte.

Beweis. Zum l-ten Aufruf von $LLL_\delta(\cdot, k)$ sei K_l die Anzahl der Schleifendurchläufe in Schritt 2 und K_l^+ die Anzahl der positiven LLL-Tests in Schritt 2.2 (Vertauschen der Basisvektoren $r'_{\nu-1}$ mit r'_ν in Stufe ν). Nach der Analyse für den Lovász-Algorithmus (siehe Kapitel 2.3) gilt mit Ungleichung (2.7): $K_l \leq 2K_l^+ + k$.

Nach Lemma 2.8 ist die Anzahl K^+ der positiven LLL-Tests beschränkt durch $n^2/2 \cdot \log_{1/\delta} M$. Es folgt mit $\sum_{l=1}^{C} K_l^+ \leq K^+$ für die Gesamtzahl K der Schleifen in Schritt 2:

$$K = \sum_{l=1}^{C} K_l \leq \sum_{l=1}^{C} \Big(2K_l^+ + k\Big) \leq C \cdot k + n^2 \cdot \log_{1/\delta} M.$$

Nach Bemerkung 3.1 kostet Schritt 1 und 3 für jeden Aufruf von $LLL_\delta(\cdot, k)$ $O(k^2 d)$ arithmetische Schritte und jede Schleife in Schritt 2 kostet $O(k)$. Es folgt die Behauptung. ∎

3.2 LLL-Block-reduzierte Gitterbasen

Bei der Reduktion von $b_{s+1}, b_{s+2}, \ldots, b_{s+k}$ in lokalen Koordinaten wird lokal mit Vektoren im $\mathbb{R}^k$ gerechnet. Für die Initialisierung und der anschließenden Transformation und Längenreduktion entstehen zusätzliche Kosten. Wir teilen die geordnete Gitterbasis in Blöcke von jeweils k aufeinander folgenden Basisvektoren auf. Auf diese Unterteilung der Gitterbasis bezieht sich die LLL-Block-Reduktion.

Durch die Verwendung der minimalen LLL-Reduktion in lokalen Koordinaten erhalten wir einen schnellen Reduktionsalgorithmus zur LLL-Block-Reduktion.

3.2.1 Grundlagen

Wir führen die *LLL-Block-Reduktion* ein. LLL-Block-reduzierte Gitterbasen und LLL-reduzierte Gitterbasen haben vergleichbare Eigenschaften.

Definition 3.3 Eine längenreduzierte Gitterbasis $b_1, b_2, \ldots, b_n \in \mathbb{R}^d$ heißt *LLL-Block-reduziert* mit Reduktionsparameter $\delta \in (\frac{1}{4}, 1)$ und Blockweite $k \leq n$, wenn mit $\alpha := (\delta - \frac{1}{4})^{-1}$ gilt:

$$R_1 : \quad \delta \cdot \|\hat{b}_\nu\|^2 \leq \mu_{\nu+1,\nu}^2 \|\hat{b}_\nu\|^2 + \|\hat{b}_{\nu+1}\|^2 \qquad \text{für } \nu \in \{1, 2, \ldots, n-1\} \setminus k\mathbb{N}$$

$$R_2 : \quad \|\hat{b}_{kl}\|^2 \leq \alpha\delta^{-k} \cdot \|\hat{b}_{kl+1}\|^2 \qquad \text{für } l \in \{1, 2, \ldots, \lceil \tfrac{n}{k} \rceil - 1\}.$$

Innerhalb der Blöcke gilt also die LLL-Reduktionsbedingung, während zwischen zwei Blöcken die Reduktionsbedingung abgeschwächt wird.

Lemma 3.4 *Sei $b_1, b_2, \ldots, b_n$ LLL-Block-reduziert mit $\delta \in (\frac{1}{4}, 1)$ und Blockweite k. Dann gilt mit $\alpha := (\delta - \frac{1}{4})^{-1}$ für $1 \leq i \leq j \leq n$, daß*

$$\|\hat{b}_i\|^2 \leq \alpha^{j-i} \cdot \delta^{-kt} \cdot \|\hat{b}_j\|^2, \qquad \begin{array}{l} i \in \{1, 2, \ldots, k\} + kl \\ j \in \{1, 2, \ldots, k\} + k(l+t), \end{array}$$

$t \in \mathbb{N}_0$ beschreibt hierbei die Anzahl der k-Blöcke zwischen i und j.

Beweis. Sei $i = kl + r_i \leq j = k(l+t) + r_j$, und $r_i, r_j \in \{1, 2, \ldots, k\}$. Falls $t = 0$, folgt mit R_1 die Behauptung $\|\hat{b}_i\|^2 \leq \alpha^{j-i}\|\hat{b}_j\|^2$. Sei $t > 0$. Durch die sukzessive Anwendung der Bedingungen R_1 und R_2 erhält man:

$$\begin{array}{rcll} \|\hat{b}_i\|^2 & \leq & \alpha^{k-r_i} \cdot \|\hat{b}_{k(l+1)}\|^2 & (R_1) \\ & \leq & \alpha^{k-r_i+1} \cdot \delta^{-k} \cdot \|\hat{b}_{k(l+1)+1}\|^2 & (R_2) \\ & \leq & \alpha^{k-r_i+1+k(t-1)} \cdot \delta^{-kt} \cdot \|\hat{b}_{k(l+t)+1}\|^2 & (R_1,\, R_2) \\ & \leq & \alpha^{kt-r_i+1+r_j-1} \cdot \delta^{-kt} \cdot \|\hat{b}_{k(l+t)+r_j}\|^2 & (R_1) \\ & \leq & \alpha^{j-i} \cdot \delta^{-kt} \cdot \|\hat{b}_j\|^2. & \end{array}$$

■

Bemerkung. Die Anzahl t der k-Blöcke zwischen 1 und j ist stets $t = \lfloor \frac{j-1}{k} \rfloor$. Hieraus folgt für $j = 1, 2, \ldots, n$:

$$\|b_1\|^2 = \|\hat{b}_1\|^2 \leq \alpha^{j-1} \cdot \delta^{-kt} \cdot \|\hat{b}_j\|^2 \leq \left(\frac{\alpha}{\delta}\right)^{j-1} \|\hat{b}_j\|^2 \tag{3.1}$$

und für $i \leq j$

$$\|\hat{b}_i\|^2 \leq \alpha^{j-i} \cdot \delta^{-\lfloor (j-1)/k \rfloor k} \cdot \|\hat{b}_j\|^2. \tag{3.2}$$

Der folgende Satz zeigt, daß eine LLL-Block-reduzierte Gitterbasis im wesentlichen mit einer LLL-reduzierten Basis vergleichbar ist.

Satz 3.5 *Sei $b_1, b_2, \ldots, b_n$ LLL-Block-reduziert mit δ und Blockweite k. Dann gilt mit $\alpha = (\delta - \frac{1}{4})^{-1}$:*

1. $(\alpha/\delta)^{1-j} \leq \|\hat{b}_j\|^2 \lambda_j^{-2}$ *für* $j = 1, \ldots, n$.
2. $\|b_j\|^2 \lambda_j^{-2} \leq (\alpha/\delta)^{n-1}$ *für* $j = 1, \ldots, n$.
3. $\|b_\nu\| \leq \|\hat{b}_j\|^2 \, (\alpha/\delta)^{j-1}$ *für* $\nu \leq j$.

Beweis.
3.: Sei $\nu \leq j$. Da b_ν längenreduziert ist, gilt

$$\begin{aligned}
\|b_\nu\|^2 &\leq \|\hat{b}_\nu\|^2 + \frac{1}{4}\sum_{i=1}^{\nu-1} \|\hat{b}_i\|^2 && \text{(nach Ungleichung (2.1))} \\
&\leq \|\hat{b}_j\|^2 \cdot \delta^{-\lfloor (j-1)/k \rfloor k} \left(\alpha^{j-\nu} + \frac{1}{4}\sum_{i=1}^{\nu-1} \alpha^{j-i} \right) && \text{(n. Ungl. (3.1))} \\
&\leq \|\hat{b}_j\|^2 \cdot \alpha^{j-1} \delta^{-\lfloor (j-1)/k \rfloor k} \underbrace{\left(\alpha^{1-\nu} + \frac{1}{4}\sum_{i=1}^{\nu-1} \alpha^{1-i} \right)}_{\leq 1, \text{ da } \alpha \geq 4/3} \\
&\leq \|\hat{b}_j\|^2 \cdot \alpha^{j-1} \cdot \delta^{-\lfloor (j-1)/k \rfloor k} && (3.3) \\
&\leq \|\hat{b}_j\|^2 \left(\frac{\alpha}{\delta} \right)^{j-1}.
\end{aligned}$$

1.: Folgt aus 3., da $\lambda_j^2 \leq \max_{1 \leq \nu \leq j} \|b_\nu\|^2$ gilt.
2.: Aufgrund der Definition des j-ten sukzessiven Minimas λ_j gibt es ein $\nu \geq j$, mit $\lambda_j \geq \|\hat{b}_\nu\|^2$. Mit Lemma 3.4 erhalten wir:

$$\begin{aligned}
\lambda_j^2 &\geq \|\hat{b}_\nu\|^2 \geq \|\hat{b}_j\|^2 \alpha^{-\nu+j} \delta^{kt}, \\
&\geq \left(\|b_j\|^2 \cdot \alpha^{-(j-1)} \cdot \delta^{\lfloor (j-1)/k \rfloor k} \right) \alpha^{-\nu+j} \delta^{kt} \text{ (n. Ungleichung (3.3))} \\
&\geq \|b_j\|^2 \cdot \alpha^{-\nu+1} \delta^{\lfloor (j-1)/k \rfloor k + kt} \\
&\geq \|b_j\|^2 \cdot \left(\frac{\alpha}{\delta} \right)^{-n+1}.
\end{aligned}$$

■

Bemerkung 3.6 Wegen

$$\lim_{\delta \to 1} \left(\frac{\alpha}{\delta} \right) = \lim_{\delta \to 1} \alpha = \frac{4}{3}$$

sind die Reduktionsschranken in Satz 3.5 und Satz 2.6 (LLL-Reduktion) äquivalent.

3.2.2 Verfahren zur LLL-Block-Reduktion

Wir verwenden Algorithmus 3.1 $LLL_\delta(s,k)$ zur lokalen, minimalen LLL-Reduktion des Blockes $b_{s+1}, b_{s+2}, \ldots, b_{s+k}$. Der folgende Algorithmus transformiert eine Gitterbasis zum Reduktionsparamater $\delta \in (\frac{1}{4}, 1)$ in eine LLL-Block-reduzierte Basis.

Algorithmus 3.2 LLL-Block-Reduktion

`INPUT:` Gitterbasis $b_1, b_2, \ldots, b_n$, Blockgröße $k \leq n$, $n = km$, $\delta \in (\frac{1}{4}, 1)$

`1.` $l := 0$, $\alpha := (\delta - \frac{1}{4})^{-1}$

`/*` l ist die Stufe der Reduktion `*/`

`2.` `WHILE` $(l < m)$ `DO`
`2.1` Längenreduziere $b_{kl+1}, b_{kl+2}, \ldots, b_{kl+k}$
`2.2` `IF` $(\exists \nu)(1 < \nu \leq k)(\delta \cdot \|\hat{b}_{\nu-1}\|^2 > \mu^2_{\nu,\nu-1}\|\hat{b}_{\nu-1}\|^2 + \|\hat{b}_\nu\|^2)$ `THEN`
$LLL_\delta(kl, k)$
Längenreduziere $b_{kl+1}, b_{kl+2}, \ldots, b_{kl+k}$
`END IF`
`2.3` `IF` $(l > 0)$ `AND` $(\|\hat{b}_{kl}\|^2 > \alpha \cdot \delta^{-k} \cdot \|\hat{b}_{kl+1}\|^2)$ `THEN`
$l := l - 1$, $LLL_\delta(kl, 2k)$ `/*` Doppelblock-Reduktion `*/`
`ELSE`
$l := l + 1$
`END IF`
`END WHILE`

`OUTPUT:` LLL-Block-reduzierte Basis $b_1, b_2, \ldots, b_n$.

Korrektheit. BEHAUPTUNG: Zu Beginn von Stufe $l \in \{0, 1, \ldots, m\}$ ist die Teilbasis $b_1, b_2, \ldots, b_{kl}$ LLL-Block-reduziert.

Beweis der Behauptung. $l = 1$: Schritt `2.1` und `2.2` LLL-reduziert für $l = 0$ die Teilbasis $b_1, b_2, \ldots, b_k$, danach wird l erhöht, d.h die Behauptung ist für $l = 1$ erfüllt.

Sei die Behauptung für $l > 0$ bewiesen. Schritt `2.1` und `2.2` LLL-reduziert den nächsten Block $b_{kl+1}, b_{kl+2}, \ldots, b_{kl+k}$. Eine Erhöhung von $l \to l + 1$ erfolgt nur dann, wenn $\|\hat{b}_{kl}\|^2 \leq \alpha \cdot \delta^{-k} \cdot \|\hat{b}_{kl+1}\|^2$ gilt. Damit ist $b_1, b_2, \ldots, b_{kl+k}$ LLL-Block-reduziert. Da $LLL_\delta(kl, 2k)$ die Basisvektoren $b_1, b_2, \ldots, b_{kl}$ nicht verändert, gilt die Behauptung auch nach Erniedrigung von $l \to l - 1$.

Der Algorithmus endet mit Stufe $l = m$. Nach Beendigung der `WHILE` - Schleife ist die Basis $b_1, b_2, \ldots, b_n$ LLL-Block-reduziert. ■

Analyse. Betrachte Algorithmus 3.2. Sei C die Anzahl der Schleifendurchläufe in Schritt 2. Wir sagen, der Test in Schritt 2.3 ist *positiv*, wenn die Doppelblock-Reduktion $LLL_\delta(\cdot, 2k)$ aufgerufen wird.

Sei C^+ die Anzahl der positiven Tests und C^- die Anzahl der negativen Tests während der Reduktion. Jeder positive Test erniedrigt die Stufe l der Reduktion, während der negative Test l um 1 erhöht. Es folgt $C^- \leq C^+ + l$. Am Ende der Reduktion ($l = m$) gilt

$$C = C^+ + C^- \leq 2C^+ + m. \tag{3.4}$$

Die lokale, minimale LLL-Reduktion $LLL_\delta(\cdot, \cdot)$ wird C^+-mal als Doppelblock-Reduktion und m-mal in Schritt 2.2 aufgerufen. Nach Lemma 3.2 gilt für die Gesamtlaufzeit der lokalen, minimalen LLL-Reduktion die Abschätzung

$$O\left((C^+ + m)(2k)^2 d + 2kn^2 \log M\right). \tag{3.5}$$

In jedem der C Schleifendurchläufe werden k Vektoren längenreduziert. Zusätzlich erfolgt noch m-mal die Längenreduktion von k Basisvektoren in Schritt 2.2. Die Längenreduktion von k Basisvektoren benötigt $O(knd)$ arithmetische Schritte. Algorithmus 3.2 benötigt

$$(C + m)O(knd) = O(C^+ knd) + O(n^2 d) \tag{3.6}$$

arithmetische Operationen zur Längenreduktion. Die Anzahl C^+ der Doppelblock-Reduktionen können wir wie folgt abschätzen.

Lemma 3.7 *Sei $b_1, b_2, \ldots, b_n \in \mathbb{Z}^d$, $n = km$ eine Eingabebasis mit $M := \max_j \|b_j\|^2$. Sei $\delta \in (\frac{1}{4}, 1)$. Die Anzahl der Doppelblock-Reduktionen C^+ ist beschränkt durch*

$$C^+ \leq \frac{n}{k}\left(1 + \frac{n}{k}\right) \log_{1/\delta} M.$$

Beweis: Wir definieren für $l = 1, \ldots, m$

$$D_l := \prod_{j=1}^{kl} \|\hat{b}_j\|^2, \qquad D := \prod_{l=1}^{m} D_l.$$

Da das Gitter ganzzahlig ist, gilt $D_l = (\det L(b_1, b_2, \ldots, b_{kl}))^2 \in \mathbb{Z}$. Es folgt die Abschätzung

$$1 \leq D = \prod_{l=1}^{m} \prod_{j=1}^{kl} \|\hat{b}_j\|^2 \leq \prod_{l=1}^{m} \prod_{j=1}^{kl} M = \prod_{l=1}^{m} M^{kl} = M^{k\frac{1}{2}m(m+1)} = M^{\frac{1}{2}n(m+1)}.$$

Jede Doppelblock-Reduktion von $b_{kl+1}, b_{kl+2}, \ldots, b_{kl+2k}$ verringert D_{l+1} um den Faktor $\delta^{-k/2}$, denn es gilt vor der Reduktion

$$\|\hat{b}_{kl+k}\|^2 > \alpha\delta^{-k}\|\hat{b}_{kl+k+1}\|^2,$$

und nach der Reduktion

$$\|\hat{b}_{kl+k}\|^2 \leq \alpha \|\hat{b}_{kl+k+1}\|^2.$$

Die D_j für $j \neq l$ bleiben dabei unverändert. Insgesamt gilt: Jede Doppelblock-Reduktion vermindert D um den Faktor $\delta^{-k/2}$. Aus

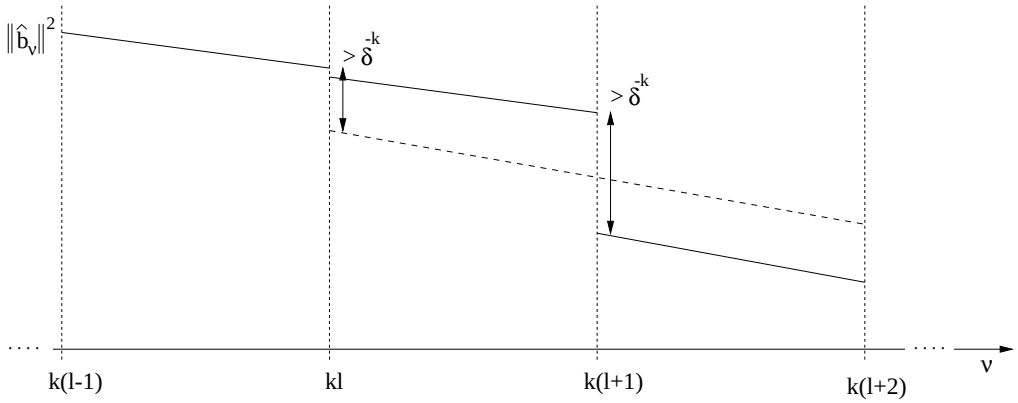
$$1 \leq \delta^{k/2 \cdot C^+} \leq M^{\frac{1}{2}n(m+1)}$$

folgt mit $m = \frac{n}{k}$

$$C^+ \leq \frac{2}{k} \log_{1/\delta} M^{\frac{1}{2}n(m+1)} = \frac{n}{k}\left(1 + \frac{n}{k}\right) \log_{1/\delta} M.$$

■

Bemerkung. Daß jede Doppelblock-Reduktion $D = \prod_{l=1}^{m} \prod_{j=1}^{kl} \|\hat{b}_j\|^2$ nur um den Faktor $\delta^{-k/2}$ verringert, ist pessimistisch. Wir können annehmen, daß viele der Doppelblock-Reduktionen D um den Faktor δ^{-k^2} verringern. Wir motivieren die Aussage durch die folgende Heuristik. Betrachte die Höhen $\|\hat{b}_\nu\|^2$ in Abhängigkeit der Ordnung ν der Basisvektoren. Es ergibt sich im allgemeinen das folgende Schaubild:



Die Höhen $\|\hat{b}_\nu\|^2$ fallen mit dem Index. Wir können annehmen, daß zwischen dem Aufruf $LLL_\delta(kl, 2k)$ und $LLL_\delta(kl - k, 2k)$ jede der Höhen $\|\hat{b}_{kl+1}\|^2, \ldots, \|\hat{b}_{kl+k}\|^2$ um den Faktor δ^{-k} kleiner geworden ist (Vergleiche die gestrichelte Linie). Es folgt die Aussage.

Bemerkung 3.8 Wenn für hinreichend viele der Doppelblock-Reduktionen $D = \prod_{l=1}^{m} \prod_{j=1}^{kl} \|\hat{b}_j\|^2$ um den Faktor δ^{-k^2} fällt, ist die Anzahl C^+ der Doppelblock-Reduktionen beschränkt durch

$$C^+ = O\left(\frac{n^2}{k^3} \log M\right).$$

Satz 3.9 *Zur Eingabebasis $b_1, b_2, \ldots, b_n \in \mathbb{Z}^d$ mit $M := \max_j \|b_j\|^2$ benötigt die LLL-Block-Reduktion*

$$O\left(\left(n^2 d + kn^2 + \frac{n^3 d}{k}\right) \cdot \log M\right) + O(n^2 d)$$

arithmetische Schritte.

Beweis. Die Behauptung folgt aus der Addition der Laufzeiten für die lokalen Reduktionen (Abschätzung 3.5) und der Längenreduktionen (Abschätzung 3.6), sowie aus Lemma 3.7. ■

Bemerkung 3.10 Unter der Voraussetzung, daß die Bemerkung 3.8 gültig ist, benötigt die LLL-Block-Reduktion

$$O\left(\left(\frac{n^2 d}{k} + kn^2 + \frac{n^3 d}{k^2}\right) \cdot \log M\right) + O(n^2 d)$$

arithmetische Schritte.

Für $d = n$ ist die Laufzeit nach Satz 3.9 für die Blockweite $k = n$ optimal. Wir erhalten dann die minimale LLL-Reduktion (Vergleiche Kapitel 2.3). Die minimale LLL-Reduktion beschränkt die Größen $|\mu_{i,j}|$ nicht, und ist deshalb in der Praxis schwierig zu realisieren.

Bemerkung 3.11 Gilt Bemerkung 3.10, dann ist die Laufzeit der LLL-Block-Reduktion optimal für die Blockweite $k = n^{2/3}$. Wir erhalten die Laufzeit

$$O(n^{2+\frac{2}{3}} \cdot \log M) + O(n^3).$$

Für die Blockweite $k = n^{2/3}$ erfolgt die lokale, minimale LLL-Reduktion in kleinen Blöcken, wobei die Eingabebasis längenreduziert ist. Man kann heuristisch erwarten, daß nur wenige LLL-Austauschschritte in einen kleinen Block erfolgen, und deshalb die Größen $|\mu_{i,j}|$ während der minimalen LLL-Reduktion in lokalen Koordinaten nicht zu stark anwachsen. Die LLL-Block-Reduktion ist dann in Gleitpunkt-Arithmetik praktisch realisierbar.

3.2.3 Implementierung mit Gleitpunkt-Arithmetik

Wie in der LLL-Reduktion in Gleitpunkt-Arithmetik werden die Transformationen auf der Gitterbasis in exakter Arithmetik ausgeführt, während die Zahlen $r_{i,j}$ mit Hilfe der Householder-Methode in Gleitpunkt-Arithmetik berechnet werden. Bei der lokalen, minimalen LLL-Reduktion wird auch die Basis durch Gleitpunktzahlen approximiert.

Gegeben sei die Eingabebasis $b_1, b_2, \ldots, b_n \in \mathbb{Z}^d$ mit $M := \max_j \|b_j\|$. Im Gegensatz zur LLL-Reduktion gelten während der LLL-Block-Reduktion keine Schranken für die Längen $\|b_j\|$ und die Größen $|\mu_{j,i}|$ (vergleiche Lemma 2.10). Wir müssen durch geeignete Eingriffe in der Reduktion dafür sorgen, daß diese Größen nicht „zu stark“ ansteigen.

Lokale, minimale LLL-Reduktion. Zur numerischen Stabilität diskutieren wir die lokale, minimale LLL-Reduktion (Algorithmus 3.1). Für den längenreduzierten Block $b_{s+1}, \ldots, b_{s+k}$ sei $R^{(s,k)} = [r_{s+i,s+j}]_{1 \le i,j \le k}$ die Eingabebasis zur lokalen, minimalen LLL-Reduktion.

1. Die minimale LLL-Reduktion beschränkt die Größen $|r_{s+i,s+j}|$ nicht.

2. Die Koeffizienten in $R^{(s,k)}$ sind τ-stellige Gleitpunktzahlen. Die Transformationen während der lokalen Reduktion vergrößern die Rundungsfehler dieser Gleitpunktzahlen.

3. Gilt die Ungleichung $\|\hat{b}_s\| > 2^c \|\hat{b}_{s+1}\|$, so können wir annehmen, daß bei der numerischen Orthogonalisierung und Längenreduktion nur für den Übergang $s \to s+1$ mehr als c Bits an Genauigkeit zur numerischen Berechnung der Koeffizienten $r_{s+1,j} = \mu_{j,s+1} \|\hat{b}_{s+1}\|$ verloren gehen:

   Für die Berechnung der Größe $r_{s+1,j}$ in τ-stelliger Gleitpunkt-Arithmetik erwarten wir für eine geeignete Konstante κ den Fehler

   $$|r'_{s+1,j} - r_{s+1,j}| \approx \kappa \cdot \|\hat{b}_{s+1}\| 2^{-\tau} \|b_{s+1}\|.$$

   (Vergleiche auch die Fehlerabschätzung der Householder QR-Zerlegung in Kapitel 1.3).

Wir geben für die lokale, minimale LLL-Reduktion einen *Korrekturparameter* c_1 und einen *Abbruchparameter* c_2 an.

Korrektur. Für die Punkte 1. und 2. überwachen wir mit Hilfe von c_1 die Transformationsmatrix T zur lokalen, minimalen LLL-Reduktion:

- Falls $\|T\|_\infty < 2^{c_1}$, reduziere die Größen $|r_{s+i,s+j}|$ durch die Längenreduktion der Basis.

- Wenn für die längenreduzierte Basis $R^{(s,k)}$ $\|T\|_\infty > 2^{c_1}$ gilt, dann korrigieren wir die Einträge im Gitter $R^{(s,k)}$ *global*:

  Die bisherige lokale Reduktion in $R^{(s,k)}$ wird auf das Gitter durch die Transformation $[b_{s+1}, \ldots, b_{s+k}] := [b_{s+1}, \ldots, b_{s+k}] \cdot T$ übertragen. Durch die anschließende Längenreduktion und Orthogonalisierung von $b_{s+1}, \ldots, b_{s+k}$ erfolgt die Korrektur der Koeffizienten in $R^{(s,k)}$.

  Die lokale Reduktion wird nach der Korrektur und der Initialisierung der Transformationsmatrix T fortgesetzt.

Abbruch. Sei δ der Reduktionsparameter der Reduktion und $\alpha = (\delta - \frac{1}{4})^{-1}$. Gilt die Bedingung $\|\hat{b}_s\|^2 > 2^{c_1}\alpha\delta^{-k}\|\hat{b}_{s+1}\|^2$, so wird die lokale Reduktion abgebrochen. Wegen der Bedingung $\|\hat{b}_s\|^2 > \alpha\delta^{-k}\|\hat{b}_{s+1}\|^2$ verringert der LLL-Block-Algorithmus die Stufe l der Reduktion.

Bemerkung. Durch die Darstellung 2^{c_1} und 2^{c_2} für Korrktur und Abbruch geben wir an, welchen zusätzlichen Präzisionsbit-Verlust wir in der lokalen, minimalen LLL-Reduktion zulassen.

Der folgende Algorithmus führt die minimale, lokale LLL-Reduktion auf $R^{(s,k)}$ in Gleitpunkt-Arithmetik aus.

Algorithmus 3.3 $LLL_\delta FP(s,k)$: lokale, minimale LLL-Reduktion

---

`INPUT:` k-Block $b_{s+1}, \ldots, b_{s+k}$, $R^{(s,k)}$, $\delta \in (\frac{1}{4}, 1)$

`1.` $T := I_k$, $R' = [r'_1, \ldots, r'_k] := R^{(s,k)}$, $\nu = \nu_{max} := 2$

`2.` `IF` $(0 \neq \mu := \lceil r'_{\nu-1,\nu} / r'_{\nu-1,\nu-1} \rfloor)$ `THEN` `/*` minimale Längenreduktion `*/`
$t_\nu := t_\nu - \mu t_{\nu-1}$, $r'_\nu := r'_\nu - \mu r'_{\nu-1}$
`END IF`

`3.` `IF` $(\delta \cdot r'^2_{\nu-1,\nu-1} \geq r'^2_{\nu-1,\nu} + r'^2_{\nu,\nu})$ `THEN` `/*` LLL-Tausch `*/`
Vertausche $t_\nu, t_{\nu-1}$ und $r'_\nu, r'_{\nu-1}$
$R' := GI_{\nu-1,\nu}(r'_{\nu-1}) \cdot R'$
`IF` $(\nu = 2)$ `AND` $(r'^2_{1,1} < \alpha \cdot \delta \cdot 2^{c_2} \cdot \|\hat{b}_s\|^2)$ `THEN` `/*` Abbruch ! `*/`
`RETURN`
$\nu := \max(2, \nu - 1)$
`ELSE`
$\nu := \nu + 1$, $\nu_{max} := \max(\nu, \nu_{max})$
`END IF`

`4.` `IF` $(\|T\|_{sup} \geq 2^{10})$ `THEN` `/*` lokale Korrektur von $r_{s+i,s+j}$ `*/`
Längenreduziere $r'_1, \ldots, r'_{\nu_{\max}}$.
(Transformationen in T speichern)
`END IF`

`5.` `IF` $(\nu \leq k)$ `AND` $(\|T\|_{sup} \geq 2^{c_2})$ `THEN` `/*` Korrektur `*/`
$[b_{s+1}, \ldots, b_{s+k}] := [b_{s+1}, \ldots, b_{s+k}] \cdot T$,
Längenreduziere $b_{s+1}, \ldots, b_{s+k}$, `GOTO 1.`
`END IF`

`6.` `IF` $(\nu \leq k)$ `THEN GOTO 2.`

`7.` $[b_{s+1}, \ldots, b_{s+k}] := [b_{s+1}, \ldots b_{s+k}] \cdot T$ `/*` Update `*/`

`OUTPUT:` Lokal, minimal LLL-reduzierter Block $b_{s+1}, \ldots, b_{s+k}$.

Bemerkung. Die Schleife `2`, `3`, `6` beschreibt die minimale LLL-Reduktion. Schritt `4` und `5` dient zur Korrektur der Koeffizienten in $R' = R^{(s,k)}$.

Bemerkung 3.12 Sei K die Anzahl der Korrekturschritte in Verlauf der lokalen, minimalen LLL-Reduktionen. Die Laufzeit zur numerischen LLL-Block-Reduktion erhöht sich um

$$K \cdot O(k^2 d + knd)$$

Rechenschritte, denn jede Korrektur bedingt eine Transformation von $b_{s+1}, \ldots, b_{s+k}$ und die Längenreduktion der Vektoren.

Bemerkung 3.13 Die Abbrüche der lokalen Reduktion erhöhen die Anzahl C^+ der Doppelblock-Reduktionen, da die lokale Reduktion im Block nicht vervollständigt wird.

Längenreduktion. Wir beschreiben die blockweise Längenreduktion der Basisvektoren $b_{kl+1}, b_{kl+2}, \ldots b_{kl+k}$ in Gleitpunkt-Arithmetik. Die Orthogonalisierung erfolgt durch die Householder-Methode. Nach der Längenreduktion wird die Orthogonalisierung zur Korrektur der Koeffizienten $r_{i,j} = \mu_{j,i}\|\hat{b}_i\|$ wiederholt (vergleiche auch Bemerkung 2.14).

Algorithmus 3.4: RED(l, κ) Längenreduktion eines k-Blocks

`INPUT:` Gitterbasis $b_1, b_2, \ldots, b_n$, Blockweite k

`1.` `FOR` $(\nu := kl + 1, \ldots, kl + k)$ `DO`
`1.1` ORT_HS(ν)
`1.2` `IF` $(\exists j)(1 \leq j < \nu)(|\mu_{\nu,j}| > \frac{1}{2})$ `THEN`
LRED(ν)
ORT_HS(ν)
`END IF`
`END FOR`

`OUTPUT:` Längenreduzierter Block $b_{s+1}, \ldots, b_{s+k}$.

LLL-Block-Reduktion. Die LLL-Block-Reduktion in Gleitpunkt-Arithmetik ist identisch mit Algorithmus 3.2. Bedingt durch die notwendigen Korrekturschritte (vergl. Bemerkungen 3.11 und 3.12) erhöht sich die Anzahl der Gleitpunkt-Operationen. Demgegenüber steht der Vorteil der kleineren Bitlänge der Gleitpunktzahlen.

Algorithmus 3.5 LLL-Block-Reduktion in Gleitpunkt-Arithmetik

INPUT: Gitterbasis $b_1, b_2, \ldots, b_n$ $n = km$, $\delta \in (\frac{1}{4}, 1)$

1. $l := 0$, $\alpha := (\delta - \frac{1}{4})^{-1}$ /* l ist die Stufe der Reduktion */
2. WHILE $(l < m)$ DO
2.1 RED(kl, k)
2.2 IF $(\exists \nu)(1 < \nu \leq k)(\delta \cdot \|\hat{b}_{\nu-1}\|^2 > \mu_{\nu,\nu-1}^2 \|\hat{b}_{\nu-1}\|^2 + \|\hat{b}_\nu\|^2)$ THEN
$LLL_\delta FP(kl, k)$
RED(kl, k)
END IF
2.3 IF $(l > 0)$ AND $(\|\hat{b}_{kl}\|^2 > \alpha \cdot \delta^{-k} \cdot \|\hat{b}_{kl+1}\|^2)$ THEN
$l := l - 1$, $LLL_\delta FP(kl, 2k)$ /* Doppelblock-Reduktion */
ELSE
$l := l + 1$
END IF
END WHILE

OUTPUT: LLL-Block-reduzierte Basis $b_1, b_2, \ldots, b_n$.

3.2.4 Experimentelle Ergebnisse

Die entwickelten Algorithmen wurden an den GGH-Gitterbasen getestet, auf die wir in Kapitel 4 genauer eingehen. Die Programme wurden mit Hilfe der C-Programmiersprache unter Verwendung der GMP-Integer-Bibliothek [15] implementiert. Die Bitlänge der Gleitpunktzahlen beträgt 53 Bit. Alle Reduktionen erfolgten auf Workstations des Typs c160 der Firma HP mit der Prozessortaktfrequenz 160 MHz. Die Korrektur- und Abbruchparameter für die lokale, minimale LLL-Reduktion wurden mit $c_1 = c_2 = 10$ (10-Bit Präzisionsverlust) gewählt.

Erzeugen der GGH-Gitterbasis. Wir beschreiben zunächst die Erzeugung von Gitterbasen nach Goldreich, Goldwasser und Halevi [11]. Der folgende Algorithmus erzeugt die *geheime* Basis R und die *öffentliche* Basis B, die aus R durch eine randomisierte, unimodulare Transformation hervorgeht.

Algorithmus 3.6 Erzeugen einer GGH-Gitterbasis

INPUT: Dimension n, Anzahl der *Mix-Runden* r

1. RAND $:= [r_{i,j} \in_R \{-4, -3, \ldots, 4\}]_{1 \le i,j \le n}$
2. D $:= 4\lceil\sqrt{n} + 1\rceil \cdot I_n$
3. B $:=$ R $:= D + RAND$
4. FOR $(i = 0, \ldots, r)$ DO

4.1 FOR $(j = 0, \ldots, n)$ DO
(i) Erzeuge *Mix-Matrix* $M_j \in GL_n(\mathbb{Z})$:
(ii) $B := B \cdot M_j$
END DO
END DO

OUTPUT: GGH-Gitterbasen R, B.

Die Mix-Matrix M_j beschreibt die Addition $b_j = b_j + \sum_{k \neq j} x_{k,j} b_k$

$$M_j := \begin{bmatrix} 1 & & & x_{1,j} & & & \\ & \ddots & & \vdots & & & \\ & & 1 & x_{j-1,j} & & & \\ & & & 1 & & & \\ & & & x_{j+1,j} & 1 & & \\ & & & \vdots & & \ddots & \\ & & & x_{n,j} & & & 1 \end{bmatrix},$$

wobei die Koeffizienten $x_{k,j} \in \{\pm 1, 0\}$ zufällig gewählt sind. Die Wahrscheinlichkeitsverteilung ist hierbei $Pr[X = 1] = Pr[X = -1] = \frac{1}{7}$.

Die Gitterdeterminante $\det L(R)$ wird durch die Orthogonalisierung der geheimen Basis R nach der Householder-Methode numerisch als

$$\log_2(\det L(R)) = \sum_{i}^{n} \log_2 \|\hat{b}_i\|$$

berechnet. Die Gitterreduktion der öffentlichen GGH-Basis B hat sich als „numerisch stabil“ erwiesen, wenn für die Ausgabebasis B' die berechneten Zahlen $\log_2(\det L(R))$ und $\log_2(\det L(B'))$ bis auf zwei Dezimalstellen nach dem Komma übereinstimmen (andernfalls sind die Rundungsfehler der Längen $\|\hat{b}_\nu\|^2$ für den LLL-Test zu groß).

Es wurden GGH-Gitterbasen der Dimension 100 bis 400 mit 2 bzw. 3 Mix-Runden erzeugt. Wir vergleichen im folgenden die LLL-Reduktion mit der LLL-Block-Reduktion. Die LLL-Reduktion erfolgte zum Reduktionsparameter $\delta = 0.96$ und der Reduktionsparameter δ zur LLL-Block-Reduktion betrug 0.99. Die Blockweite k wurde mit $k := \lfloor n^{\frac{2}{3}} \rfloor$ gewählt. In Tabelle 3.1 sind

die Reduktionsergebnisse der LLL-Reduktion und der LLL-Block-Reduktion gegenübergestellt. Dabei wurden 2 Mix-Runden zur Erzeugung der Basis ausgeführt.

Eingabebasis			*LLL-Reduktion*		*LLL-Block-Reduktion*		
Dim.	$\log_2(\det(L))$	$l^{av}(b_{i,j})$	$l^{av}_{L^3}(b_{i,j})$	t_{L^3}	$l^{av}_{Bl}(b_{i,j})$	t_{Bl}	t_{Korr}
100	545.9	34.1	5.8	164	5.5	17	1
	545.9	32.9	5.9	160	5.7	16	1
121	674.9	39.2	6.5	492	6.5	35	4
	676.2	38.5	6.5	485	6.4	36	3
144	823.2	44.5	7.6	1025	7.3	76	9
	822.0	47.7	7.5	1136	7.3	86	6
169	984.9	58.1	8.5	2270	8.0	164	16
	984.2	55.5	8.4	2226	8.2	152	20
196	1161.9	66.6	9.1	4617	9.1	253	52
	1161.3	63.6	9.1	4688	9.0	270	53
225	1355.5	68.4	10.3	8221	10.1	515	94
	1355.4	66.9	10.3	8486	10.3	524	95
256	1564.8	76.3	11.2	14824	11.2	856	213
	1564.9	78.5	11.0	15665	10.9	903	221
289	1791.7	87.8	* 12.7	23621	11.9	1601	527
	1790.1	95.7	* 13.4	18235	12.0	1609	521
324	2034.6	104.6	* 15.6	15102	13.0	2826	1580
	2033.3	103.0	* 15.9	13482	13.2	2872	1417
361	2294.1	116.4	* 17.5	13764	14.2	4544	2151
	2292.3	117.8	* 17.7	13567	14.3	4654	2300
400	2569.1	124.9	* 18.7	20001	* 17.5	4728	2898
	2568.6	127.4	* 19.1	17447	* 16.9	4685	2780

Tabelle 3.1: Reduktion von GGH-Basen der Dimension 100 bis 400, 2 Mix-Runden (Die mit * gekennzeichneten Reduktionen waren nicht stabil).

Die ersten zwei Spalten beschreiben die Gitterdimension und die Gitterdeterminante der GGH-Gitter. In der dritten Spalte ist die durchschnittliche Bitlänge l^{av} der Zahlen der nicht reduzierten Basis eingetragen. In den Spalten 4 und 6 steht die durchschnittliche Bitlänge der Zahlen nach der LLL-Reduktion $l^{av}_{L^3}$ bzw. der LLL-Block-Reduktion l^{av}_{Bl}, während in den Spalten 5 und 7 die Reduktionszeiten t_{L^3} bzw. t_{Bl} in Sekunden eingetragen sind. In Spalte 8 steht der Zeitaufwand t_{Korr} der Korrekturschritte während der LLL-Block-Reduktion.

Ab Dimension 289 waren die nach der LLL-Methode ausgegebenen Basen nicht mehr korrekt reduziert. Dagegen war die LLL-Block-Reduktion bis Dimension 361 stabil. Es zeigt sich, daß beide Methoden bei stabilem Verlauf

vergleichbare Reduktionsergebnisse liefern. Die Rechenzeit der LLL-Block-Reduktion ist aber im Vergleich zur LLL-Reduktion um den Faktor 10 bis 18 schneller. Bemerkenswert ist, daß in Dimension 100 die Korrekturschritte die Laufzeit der LLL-Block-Reduktion kaum beinflussen, während in Dimension 361 etwa 50 % der Rechenzeit für Korrekturschritte benötigt werden.

Tabelle 3.2 zeigt die Reduktionsergebnisse für GGH-Basen, die mit 3 Mix-Runden erzeugt wurden. Im Vergleich zu den vorherigen Basen erhöhte sich die Bitlänge der nicht reduzierten Basen etwa um den Faktor 1.5, während die Bitlängen der reduzierten Basen vergleichbar sind.

Eingabebasis			*LLL-Reduktion*		*LLL-Block-Reduktion*		
Dim.	$\log_2(\det(L))$	$l^{av}(b_{i,j})$	$l^{av}_{L^3}(b_{i,j})$	t_{L^3}	$l^{av}_{Bl}(b_{i,j})$	t_{Bl}	t_{Korr}
100	546.2	50.0	6.0	276	5.8	26	3
	541.4	46.2	5.8	284	5.5	29	5
121	676.7	55.7	6.5	737	6.6	60	13
	677.0	55.2	6.7	725	6.5	58	10
144	823.5	69.3	7.4	1762	7.2	143	24
	823.3	72.3	7.5	1840	7.3	145	25
169	985.1	82.1	8.5	4000	8.2	273	63
	985.7	83.9	8.3	4032	8.1	266	57
196	1163.0	93.5	9.4	7954	9.0	523	163
	1162.8	98.2	9.1	8245	8.9	570	159
225	1354.5	105.6	10.3	15604	9.9	1043	291
	1354.3	100.9	10.2	15495	9.9	1003	307
256	1564.5	117.9	11.2	28452	10.8	1885	683
	1564.6	120.8	11.3	27585	11.3	1828	689
289	1789.7	133.9	* 15.2	27678	12.2	3161	1350
	1791.0	135.4	* 15.3	25483	12.1	3187	1335
324	2035.2	143.2	* 18.3	24795	13.1	5762	3324
	2034.3	153.1	* 19.0	20003	13.3	5380	3001
361	2292.8	174.7	* 20.5	27055	14.7	9419	5399
	2291.4	177.2	* 20.5	27011	14.5	9902	5430
400	2568.4	185.5	* 22.4	30233	* 19.5	9860	7287
	2569.3	184.8	* 22.6	29706	* 19.8	8711	6186

Tabelle 3.2: Reduktion von GGH-Basen der Dimension 100 bis 400, 3 Mix-Runden (Die mit * gekennzeichneten Reduktionen waren nicht stabil).

Die LLL-Reduktion verlief bis Dimension 256 stabil, während bei der LLL-Block-Reduktion Stabilitätsprobleme ab Dimension 361 auftraten. Im Vergleich zur LLL-Reduktion ergeben sich durch die LLL-Block-Reduktion etwa um den Faktor 10 bis 15 kürzere Rechenzeiten. Die Laufzeit der LLL-Block-Reduktion wurde durch die Korrekturschritte noch stärker beeinflußt. Es

zeigte sich, daß die Basistransformationen die Laufzeiten dominieren. Diese Berechnungen wurden in exakter Arithmetik ausgeführt, wobei die Rechenzeit der Multiplikation quadratisch in der Bitlänge der Zahlen ist. Beim Vergleich von Tabelle 3.1 und 3.2 zeigt sich, daß ein Anstieg der Bitlänge um den Faktor 1.5 die Reduktionszeiten verdoppelt.

Blockweite. Nach Bemerkung 3.10 ist die Laufzeit der LLL-Block-Reduktion bei einer Blockweite von $k \approx n^{\frac{2}{3}}$ optimal. Bei der LLL-Block-Reduktion mit Gleitpunktzahlen wird dieses Resultat durch die notwendigen Korrekturschritte in der lokalen LLL-Reduktion beeinflußt. Weiterhin ist die Rechenzeit der Basistransformationen abhängig von der Bitlänge der Zahlen, während für Gleitpunkt-Operationen diese Abhängigkeit nicht gegeben ist. Wir untersuchen deshalb die Rechenzeit der LLL-Block-Reduktion in Abhängigkeit von der Blockweite.

In Abbildung 3.1 und 3.2 ist die Reduktionszeit, abhängig von der Blockweite und der Gitterdimension für GGH-Basen der Dimension 100 bis 350 dargestellt. Die Basen wurden durch 2 bzw. 3 Mix-Runden generiert. Die vermutete optimale Blockweite ist auf den Kurven markiert.

Für kleine Gitterdimensionen ist die zur Reduktionsdauer optimale Blockweite etwas größer als $n^{2/3}$, während sich mit Zunahme der Gitterdimension die optimale Blockweite im Vergleich zur $n^{2/3}$-Kurve nach links verschiebt.

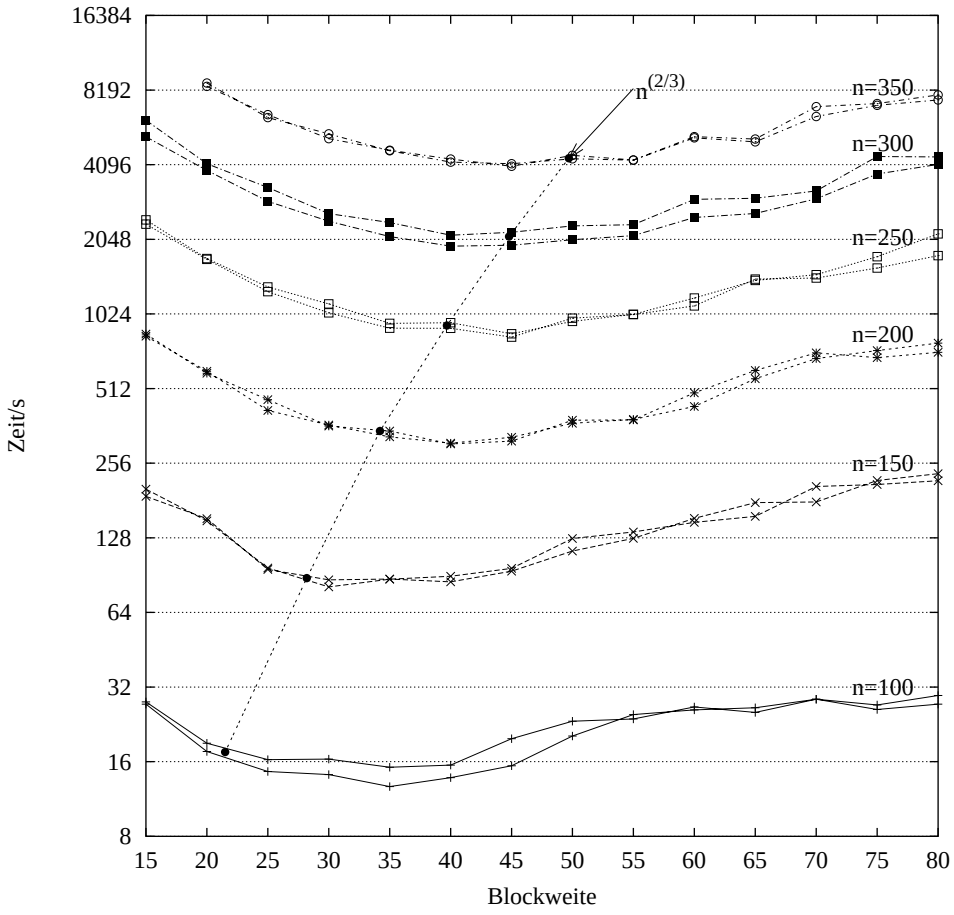


Abbildung 3.1: LLL-Block-Reduktion mit $\delta = 0.99$. Reduktionszeit in Abhängigkeit der Blockweite. Die GGH-Basen der Dimension 100 bis 350 wurden durch 2 Mix-Runden erzeugt.

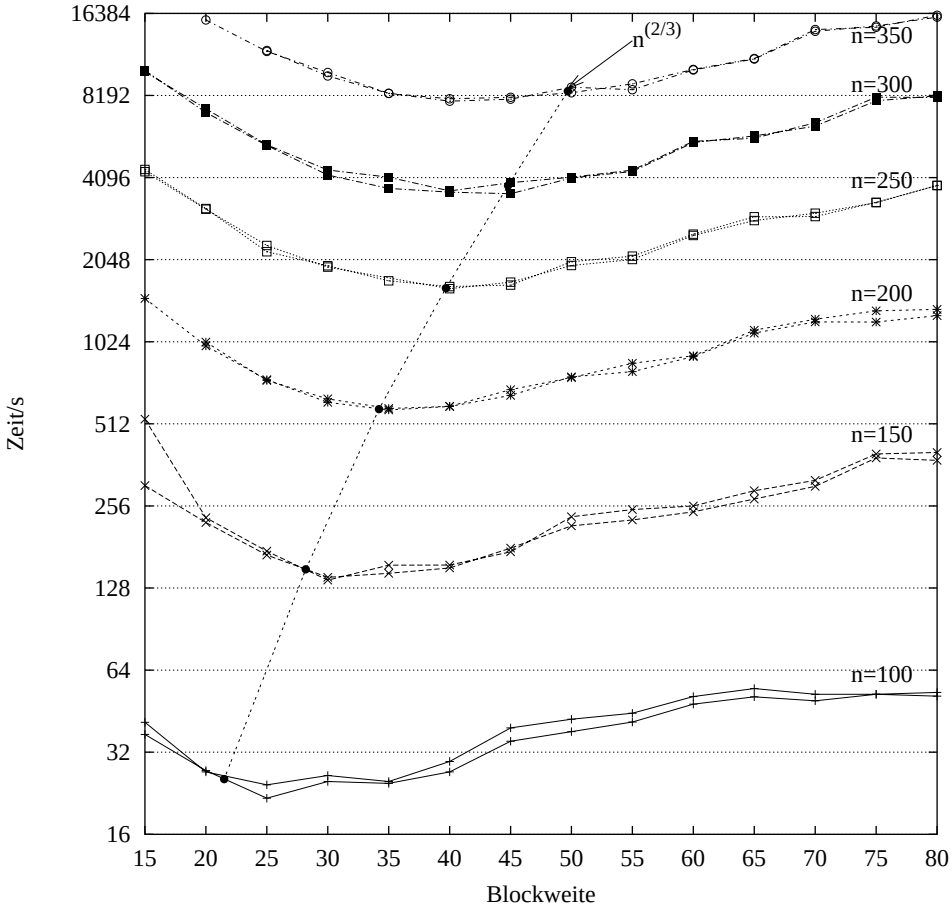


Abbildung 3.2: LLL-Block-Reduktion mit $\delta = 0.99$. Reduktionszeit in Abhängigkeit der Blockweite. Die GGH-Basen der Dimension 100 bis 350 wurden durch 3 Mix-Runden erzeugt.

Kapitel 4

Angriffe auf das GGH-Kryptosystem

Wir analysieren das von Goldreich, Goldwasser und Halevi (GGH) [11] vorgeschlagene gitterbasierte Kryptosystem. Es zeigt sich, daß die öffentlichen Parameter Angriffe auf das Kryptosystem ermöglichen.

4.1 Das GGH-Kryptosystem

Der öffentliche Schlüssel B und der geheime Schlüssel R sind Gitterbasen des gleichen Gitters. Die kodierte Nachricht c ist der zur Nachricht m eindeutig zugeordnete Gittervektor $x := B \cdot m$, zu der ein zufällig gewählter Störvektor e addiert wird. Die spezielle Struktur von R ermöglicht es, x als nächsten Gittervektor zu c effizient zu bestimmen.

Das GGH-Kryptosystem ist mit dem *McEliece*-Kryptosystem [18] vergleichbar. In diesem Schema sind der öffentliche Schlüssel und der geheime Schlüssel Basen des gleichen linearen Codes. Die Chiffrierung einer Nachricht erfolgt durch die Addition eines zufälligen Vektors der zur Nachricht korrespondierenden Kodierung. Die spezielle Eigenschaft des geheimen Schlüssels erlaubt diesen Fehlervektor bei der Dechiffrierung zu eliminieren. Das McEliece-Schema und das GGH-Schema unterscheiden sich jedoch wesentlich in der Domäne wo diese Operationen ausgeführt werden.

4.1.1 Öffentliche Parameter

Die Parameter dienen zur Erzeugung von Schlüssel-Instanzen und für die Chiffrierung von Nachrichten.

- Der **Sicherheitsparameter** $n \in \mathbb{N}$ bestimmt den Rang des zugrunde liegenden Gitters. Der öffentliche Schlüssel B und der geheime Schlüssel R sind Gitterbasen des gleichen Gitters.

- **Störparameter** σ. Der zur Nachricht m korrespondierende Gittervektor $x \in \mathbb{Z}^n$ wird durch einen zufälligen Vektor e mit Einträgen $\pm\sigma$ gestört.
- **Störparameter** l **und Skalierungsfaktor** k_n. Diese Parameter werden für die Erzeugung des geheimen Schlüssels verwendet.

Im Hinblick auf die in der Sicherheitsanalyse betrachteten Angriffe und die experimentellen Ergebnisse schlugen die Autoren die Parameter $\sigma = 3$, $l = 4$, sowie als Sicherheitsparameter $n, 250 \leq n \leq 300$ vor. Als Skalierungsfaktor k_n wählten die Autoren $k_n := l \cdot \lceil \sqrt{n} + 1 \rceil$.

4.1.2 Geheimer und öffentlicher Schlüssel

Betrachte das durch k_n skalierte Gitter $k_n \cdot \mathbb{Z}^n$ mit der Basismatrix $k_n I = [d_1, d_2, \ldots, d_n]$. Der duale Orthogonalitätsdefekt dieser Basis ist 1 (siehe Definition 2.3). Das Stören der Vektoren d_i durch Vektoren s_i mit kleiner Länge verändert die Eigenschaft der Reduziertheit nur wenig. Zum Sicherheitsparameter n wird die Basis

$$R = k_n I + [s_1, \ldots, s_n] = \begin{bmatrix} k_n + s_{1,1} & s_{1,2} & \cdots & s_{1,n} \\ s_{2,1} & k_n + s_{2,2} & \cdots & s_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ s_{n,1} & s_{n,2} & \cdots & k_n + s_{n,n} \end{bmatrix}$$

erzeugt. R ist der geheime Schlüssel bzw. die geheime Basis. Die Einträge $s_{i,j}$ sind Zufallswerte aus dem Wertebereich $\{-l, \ldots, +l\}$. Der öffentliche Schlüssel — die Gitterbasis B — geht aus R (wie in Kapitel 3 Abschnitt 2.4 beschrieben) durch 2 Mix-Runden hervor.

Die Einträge in der Basis R sind genau in der Diagonalen „groß“. Der duale Orthogonalitätsdefekt ist klein. Die Längen der Vektoren in der öffentlichen Basis B sind im Vergleich zu R sehr groß (vergleiche Tabelle 3.1). Zu gegebener Basis B die Basis R zu berechnen, ist eine Instanz des Basis-Reduction-Problems. Die Autoren vermuten, daß dieses Problem für die aus R randomisiert erzeugten Basen B schwierig ist.

4.1.3 Chiffrieren und Dechiffrieren

Sei $B \in \mathbb{Z}^{n \times n}$ eine zum Sicherheitsparameter n öffentliche Basis. Der zur ganzzahligen Nachricht m korrespondierende Gittervektor $x := B \cdot m$ wird durch einen zufälligen Vektor $e \in \{\pm\sigma\}^n$ gestört. Zu dem Ciphertext $c = x + e$ ist x der nächste Gittervektor.

Mit der geheimen Basis R erfolgt die Dechiffrierung von c mit Hilfe der *Round-Off*-Methode nach Babai [3]: Die Koeffizienten im rellen Vektor $R^{-1}c$ werden gerundet (wir schreiben kurz $\lceil R^{-1}c \rfloor$). Es gilt

$$\lceil R^{-1}c \rfloor = \lceil R^{-1}(x + e) \rfloor = \lceil (R^{-1}B)B^{-1}x + R^{-1}e \rfloor.$$

Da die Basen R und B das gleiche Gitter erzeugen ist die Matrix $R^{-1}B$ unimodular. Der Vektor $B^{-1}x$ ist nach Konstruktion die ganzzahlige Nachricht m. Es folgt $\lceil R^{-1}x \rfloor = R^{-1}x$. Die Dechiffrierung m des Ciphertextes c wird bestimmt als:

$$m = B^{-1}R\lfloor R^{-1}c \rceil = B^{-1}R\left(R^{-1}x + \lceil R^{-1}e \rfloor\right).$$

Die Berechnung von $R\lfloor R^{-1}c \rceil$ bestimmt den zu c nächsten Gittervektor x Die Dechiffrierung ist genau dann korrekt, wenn $\lfloor R^{-1}e \rceil = 0$. Die Autoren zeigen dazu das folgende Resultat [11]:

Satz 4.1 *Sei R die geheime Basis und der Absolutbetrag der Einträge in R^{-1} durch $\frac{\gamma}{\sqrt{n}}$ beschränkt. Für den Störvektor e gelte $\|e\|_\infty \leq \sigma$. Dann ist die Wahrscheinlichkeit für das Auftreten eines Dechiffrierfehlers nach oben beschränkt durch $2n \cdot \exp\left(-\frac{1}{8\sigma^2\gamma^2}\right)$.*

Die „Sicherheit“ [1] des GGH-Kryptosystems basiert auf der Annahme, daß nur mit Kenntnis der öffentlichen Basis B eine Dechiffrierung in der Praxis unmöglich ist:

Annahme 4.2 *Für hinreichend große n gilt: Ist B eine zum Sicherheitsparameter n randomisiert erzeugte öffentliche GGH-Gitterbasis und c eine durch B verschlüsselte zufällige Nachricht, so ist es praktisch unmöglich, den zu c nächsten Gittervektor nur mit Kenntnis von B zu bestimmen.*

Im folgenden Abschnitt zeigen wir, daß diese Annahme für den vorgeschlagenen Sicherheitsparameter n, $250 \leq n \leq 300$, nicht richtig ist.

4.2 Angriffe auf das GGH-Kryptosystem

Goldreich, Halevi und Goldwasser [12] veröffentlichten Challenges der Dimension 200, 250, 300, 350 und 400, jeweils bestehend aus einer öffentlichen GGH-Basis B und einen durch B erzeugten Ciphertext c. Aufgabe war es, diese Ciphertexte zu dechiffrieren.

Ein Ansatz für Angriffe auf das GGH-Schema ist die sogenannte Einbettungsmethode [11]. Es handelt sich hierbei um die heuristische Reduktion des Closest-Vector-Problems auf das Shortest-Vector-Problem. Sei $c = B \cdot m + e$ die chiffrierte Nachricht m. Ziel der Angriffe ist es, den Fehlervektor $e \in \{\pm\sigma\}^n$ als kürzesten Gittervektor zu bestimmen. Durch $m = B^{-1}(c - e)$ kann die Nachricht effizient dechiffriert werden.

[1] Sicherheit gegen sogenannte Chosen-Plaintext-Angriffe, der schwächsten Sicherheitsanforderung an Public-Key-Schemata (vergleiche [19]).

4.2.1 Die Einbettungsmethode

Betrachte das Closest-Vector-Problem: Gegeben die ganzzahlige Gitterbasis $B = [b_1, b_2, \ldots, b_n]$ für $L \subset \mathbb{Z}^n$ und ein Vektor $c \in \mathbb{Z}^n$. Bestimme den zu c nächsten Gittervektor x. Wir reduzieren unter Verwendung einer Heuristik dieses Closest-Vektor-Problem auf das Shortest-Vector-Problem. Betrachte die Basis

$$B_c = \begin{bmatrix} | & | & & | & | \\ b_1 & b_2 & \cdots & b_n & c \\ | & | & & | & | \\ 0 & 0 & & 0 & 1 \end{bmatrix}. \tag{4.1}$$

Für den zu c nächsten Gittervektor x gilt heuristisch, daß der Vektor $(c - x, \pm 1)^T$ ein kürzester nichttrivialer Vektor im Gitter $L(B_c)$ ist (beachte, daß die Basiserweiterung die Gitterdeterminante nicht erhöht). Das so gewonnene Shortest-Vector-Problem ist in der Praxis leichter zu lösen. Betrachte dazu das folgende Gitterproblem:

Unique-Shortest-Vector-Problem (u-SVP): Gegeben das Gitter $L \subset \mathbb{R}^n$ mit der Eigenschaft $C \cdot \lambda_1(L) \leq \lambda_2(L)$ für $C > 1$. Bestimme den bis auf das Vorzeichen eindeutig bestimmten kürzesten Gittervektor.

Alle Gittervektoren der Länge kleiner als $C \cdot \lambda_1(L)$ zeigen in die gleiche Richtung. Der Faktor C ist die *Lücke* zwischen den Längen des kürzesten Gittervektors b und den von b linear unabhängigen Gittervektoren.

Das u-SVP wird durch die LLL-Reduktion gelöst, falls $C > (4/3)^{(n-1)/2}$ (vergleiche Satz 2.6). In der Praxis löst die LLL-Reduktion dieses Problem auch für wesentlich kleinere Lücken C. Für die Angriffe auf das GGH-Schema konstruieren wir (mit Hilfe der Einbettungsmethode) Gitter für das u-SVP mit möglichst großer Lücke C.

4.2.2 Angriff durch Basiserweiterung

Die öffentlichen Parameter $\sigma = 3$ und $l = 4$ des GGH-Schemas geben dem Angreifer Hinweise auf die Länge von Zielvektoren. Durch die Einbettungsmethode können diese als kürzeste Gittervektoren bestimmt werden.

Angriff auf den Fehlervektor.
Dieser Angriff wurde in der Sicherheitsanlyse der Autoren vorgestellt [11]. Sei $e = c - B \cdot m \in \{\pm 3\}^n$ der zum Ciphertext c gehörende Fehlervektor. Durch die Basiserweiterung

$$B_c = \begin{bmatrix} | & | & & | & | \\ b_1 & b_2 & \cdots & b_n & c \\ | & | & & | & | \\ 0 & 0 & & 0 & 1 \end{bmatrix} \tag{4.2}$$

erhalten wir die Darstellung für e als Gittervektor

$$(e,1)^T = (c,1)^T - (B \cdot m, 0)^T \in L(B_c).$$

Sei $R = [r_1, r_2, \ldots, r_n]$ die geheime Basis. Nach Konstruktion von R erwarten wir für die Basisvektoren die Längen

$$\|r_i\|^2 \approx \sum_{i=-4}^{4} \frac{(k_n + i)^2}{9} + (n-1) \sum_{i=-4}^{4} \frac{i^2}{9} \approx 22.7n.$$

Für den Fehlervektor e gilt $\|e\|^2 = 9n$.

Bemerkung 4.3 Für das durch die Basis B_c definierte u-SVP erwartet man heuristisch die Lücke

$$C \approx \frac{\min_i \|r_i\|}{\|e\|} \approx 1.5\,.$$

Angriff auf die geheime Gitterbasis.
Sei $R = [r_1, r_2, \ldots, r_n]$ die private GGH-Basismatrix. Dabei ist r_i die Summe des zufälligen Störvektor $s_i \in_R \{-4, \ldots, 4\}^n$ und dem Diagonaleintrag $d_i = (0, \ldots, 0, k_n, 0, \ldots, 0)^T$. Wenn die öffentilche Basis B durch den Vektor d_i erweitert wird, erhält man eine Darstellung von s_i als Gittervektor:

$$B_{d_i} := \begin{bmatrix} | & | & | & & | \\ d_i & b_1 & b_2 & \cdots & b_n \\ | & | & | & & | \\ 1 & 0 & 0 & & 0 \end{bmatrix}, \tag{4.3}$$

denn mit $r_i \in L(B)$ folgt

$$(r_i, 0)^T - (d_i, 1)^T = (s_i, 1)^T \in L(B_{d_i}).$$

Wir erwarten für s_i die Länge

$$\|s_i\|^2 = n \cdot \sum_{i=-4}^{4} \frac{i^2}{9} \approx 6.3n.$$

Durch heuristische Annahmen gestützt, können wir voraussetzen, daß s_i der kürzeste Vektor im Gitter $L(B_{d_i})$ ist. Die Gitterreduktion von B_{d_i} bestimmt den Zielvektor s_i. Nur mit Kenntnis von s_i erhält man den geheime Basisvektor $r_i = s_i + d_i$.

Bemerkung 4.4 Für das durch die Basis B_{d_i} definierte u-SVP erwartet man heuristisch die Lücke

$$C \approx \frac{\min_j \|r_j\|}{\|d_i\|} \approx 1.9\,.$$

Um die geheime Basis R zu bestimmen, ermittelt man nach dem obigen Verfahren alle Vektoren r_i (der Angriff kann parallelisiert werden). Der Angriff auf die geheime Gitterbasis ist aufgrund der größeren Lücke C erfolgreicher als der obige Ciphertextangriff.

4.2.3 Der Angriff von Nguyen [22]

Sei $B \in \mathbb{Z}^{n \times n}$ eine öffenliche GGH-Basis zum Sicherheitsparameter n. Die Nachricht $m \in \mathbb{Z}^n$ wird in einem Ciphertext

$$c = Bm + e, \quad e \in_R \{\pm\sigma\}$$

kodiert. Der Fehlervektor $e \in \{\pm\sigma\}^n$ wird nach Reduktion der Gleichung modulo σ eliminiert:

$$c \equiv Bm + e \equiv Bm \pmod{\sigma}.$$

Durch die Modifikation $c' := c + \sigma^n$ gilt $c' = Bm + e'$ mit $e' \in \{0, 2\sigma\}^n$. Wir rechnen so mit dem *besseren* Modul 2σ:

$$c' \equiv Bm + e' \equiv Bm \pmod{2\sigma}.$$

Der Angriff auf die Ciphertextnachricht c erfolgt in 2 Schritten:

1. **Rekonstruiere die Nachricht modulo 2σ:** Bestimme

   $$m_{2\sigma} \equiv m \pmod{2\sigma} \in Loes_{B,2\sigma}(c') := \{\, m' \mid c' \equiv Bm' \pmod{2\sigma} \,\}.$$

   Für die meisten Fälle gilt $\dim(Loes_{B,2\sigma}(c')) \leq 2$. Diese Lösungen können laut Nguyen [22] schnell aufgezählt werden.

2. **Vereinfache das Problem: bestimme den zu c nächsten Gittervektor.** Mit $m_{2\sigma} = Bm \pmod{2\sigma}$ und $c = Bm + e$ folgt

   $$c - Bm_{2\sigma} = B(m - m_{2\sigma}) + e.$$

   Da 2σ die Differenz $(m - m_{2\sigma})$ teilt, erhalten wir mit der Nachricht $m' := (m - m_{2\sigma})/2\sigma \in \mathbb{Z}^n$ den Ciphertext c'' durch

   $$c'' = \frac{c' - m_{2\sigma}B}{2\sigma} = Bm' + \frac{e}{2\sigma}.$$

   Erweitere die Basis B um den modifizierten Ciphertext c'':

   $$B_{c''} = \begin{bmatrix} | & | & & | & | \\ b_1 & b_2 & \cdots & b_n & c'' \\ | & | & & | & | \\ 0 & 0 & & 0 & 1 \end{bmatrix}$$

   Durch die Gitterreduktion von $B_{c''}$ wird der Vektor $\frac{e}{2\sigma}$ als kürzester Vektor bestimmt. Die Länge des Fehlervektors $\frac{e}{2\sigma}$ ist

   $$\|e/2\sigma\| = 1/2 \cdot \sqrt{n},$$

   und damit als Zielvektor für die SVP-Instanz $L(B_{c''})$ „leicht“ durch Gitterreduktion zu bestimmen.

Bemerkung 4.5 Für das durch die Basis $B_{c''}$ definierte u-SVP erwartet man heuristisch die Lücke

$$C \approx \frac{\min_i \|r_i\|}{\|e/(2\sigma)\|} \approx 9.5\,.$$

Nguyen konnte mit diesem Angriff die GGH-Challenges bis zur Dimension 350 lösen.

4.2.4 Angriff auf die Gitterdeterminante

In diesem Abschnitt stellen wir einen alternativen Angriff auf das GGH-Kryptosystem vor. Wie bei der Analyse von Nguyen wird ausgenutzt, daß die Einträge im Fehlervektor aus dem Wertebereich $+\sigma$ und $-\sigma$ gezogen sind. Betrachte die Basiserweiterung

$$B_c = \begin{bmatrix} | & | & & | & | \\ b_1 & b_2 & \cdots & b_n & c \\ | & | & & | & | \\ 0 & 0 & & 0 & 1 \end{bmatrix}, \quad c = B \cdot m + e.$$

Dieses Gitter wird in das Untergitter $L(\bar{B}_c) \subset L(B_c)$ transformiert. Wir bestimmen mit Kenntnis von $\sigma > 1$ die Transformation so, daß der Zielvektor e im Gitter $L(\bar{B}_c)$ enthalten bleibt, erhöhen aber die Gitterdeterminate. Es gilt das heuristische Argument, daß die Gitterreduktion den Zielvektor e in $\bar{B}_c$ leichter bestimmt als in der ursprünglichen Basis B_c, denn der nötige Approximationsfaktor $C = \frac{\lambda_2}{\lambda_1}$ wird größer (d.h. die verlangte Approximationslücke nimmt zu). Unter der plausiblen Annahme $\lambda_2(L(B_c)) = \lambda_1(L(b_1, b_2, \ldots, b_n))$ erreichen wir durch Skalieren von $L(b_1, b_2, \ldots, b_n)$ mit dem Faktor $\sigma > 1$, daß für das neue Gitter $\bar{B}_c$ gilt:

$$\frac{\lambda_2(L(\bar{B}_c))}{\lambda_1(L(\bar{B}_c))} \approx \frac{\lambda_2(\sigma \cdot L(b_1, b_2, \ldots, b_n))}{\lambda_1(L(B_c))} = \sigma \cdot \frac{\lambda_2(L(B_c))}{\lambda_1(L(B_c))}.$$

Sei $b_{i,1}, b_{i,2}, \ldots, b_{i,n}, c_i$ die i-te Koordinate in der Basis B_c. Mit großer Wahrscheinlichkeit gilt, daß der größte gemeinsame Teiler (ggT) der Zahlen $b_{i,1}, b_{i,2}, \ldots, b_{i,n}$ gleich 1 ist. Sei $K = 2^{n/2}$. Die LLL-Reduktion der Basis

$$B'_c := \begin{bmatrix} b_{1,1} & \cdots & b_{1,n-1} & b_{1,n} & c_1 \\ \vdots & & \vdots & \vdots & \vdots \\ b_{i-1,1} & \cdots & b_{i-1,n-1} & b_{i-1,n} & c_{i-1} \\ Kb_{i,1} & \cdots & Kb_{i,n-1} & Kb_{i,n} & Kc_i \\ b_{i+1,1} & \cdots & b_{i+1,n-1} & b_{i+1,n} & c_{i+1} \\ \vdots & & \vdots & \vdots & \vdots \\ b_{n,1} & \cdots & b_{n,n-1} & b_{n,n} & c_n \\ 0 & \cdots & 0 & 0 & K^2 \end{bmatrix}$$

bestimmt unter der Vorraussetzug, daß der ggT der Zahlen $b_{i,1}, b_{i,2}, \ldots, b_{i,n}$ gleich 1 ist, die LLL-reduzierte Basis

$$B'_c := \begin{bmatrix} b_{1,1} & \cdots & b_{1,n-1} & b_{1,n} & c_1 \\ \vdots & & \vdots & \vdots & \vdots \\ b_{i-1,1} & \cdots & b_{i-1,n-1} & b_{i-1,n} & c_{i-1} \\ 0 & \cdots & 0 & K & 0 \\ b_{i+1,1} & \cdots & b_{i+1,n-1} & b_{i+1,n} & c_{i+1} \\ \vdots & & \vdots & \vdots & \vdots \\ b_{n,1} & \cdots & b_{n,n-1} & b_{n,n} & c_n \\ 0 & \cdots & 0 & 0 & K^2 \end{bmatrix}.$$

Nach der Division der Zeile i durch K und der Zeile $n+1$ durch K^2 bestimmt man die Basis

$$B'_c := \begin{bmatrix} | & | & & | & | \\ b'_1 & b'_2 & \cdots & b'_n & c' \\ | & | & & | & | \\ 0 & 0 & & 0 & 1 \end{bmatrix}.$$

In der i-ten Koordinate gilt

$$b'_{i,1} = b'_{i,2} = \cdots = b'_{i,n-1} = 0, \quad b'_{i,n} = 1, \quad c'_i = 0.$$

Der Fehlervektor e besitzt nur Einträge aus dem Wertebereich $\{0, \pm 3\}$. Es folgt daher

$$e \in c' + \sum_{i=1}^{n-1} \mathbb{Z} b'_i + \{0, \pm 3\} b'_n \quad \subset \quad c' + \sum_{i=1}^{n-1} \mathbb{Z} b'_i + \mathbb{Z} \cdot 3 b'_n.$$

Wir definieren die Basismatrix

$$\bar{B}_c := \begin{bmatrix} | & & | & | & | \\ \bar{b}_1 & \cdots & \bar{b}_{n-1} & 3 \cdot \bar{b}_n & \bar{c} \\ | & & | & | & | \\ 0 & & 0 & 0 & 1 \end{bmatrix}$$

mit den Eigenschaften $e \in L(\bar{B}_c)$ und $\det(L(\bar{B}_c)) = 3 \cdot \det(L(B_c))$. Damit erhalten wir:

Satz 4.6 *Sei $B = [b_1, b_2, \ldots, b_n]$ eine ganzzahlige Basismatrix und $c = x + e \in \mathbb{Z}^d$ ein Punkt, so daß $x \in L(B)$ ein Gittervektor ist. Ist e aus dem Wertebereich $\{0, \pm\sigma\}^n$ mit ganzzahligem $\sigma > 1$, dann existiert ein $k \in \{0, 1, \ldots, n\}$, so daß die durch c erweiterte Basismatrix*

$$B_c = \begin{bmatrix} | & | & & | & | \\ b_1 & b_2 & \cdots & b_n & c \\ | & | & & | & | \\ 0 & 0 & & 0 & 1 \end{bmatrix}, \quad c = B \cdot m + e$$

effizient in die Basismatrix $\bar{B}_c$ mit den Eigenschaften

$$e \in L(\bar{B}_c) \subseteq L(B_c) \quad \text{und} \quad \det(L(\bar{B}_c)) = \sigma^k \det(L(B_c))$$

transformiert werden kann.

Beweis. Wende die oben beschriebene Methode nacheinander für alle Koordinaten j an, für die

$$\mathrm{ggT}(b_{j,1}, b_{j,2}, \ldots, b_{j,n}) = 1$$

gilt. Es folgt die Behauptung. ■

Der Angriff auf das GGH-Kryptosystem erfolgt durch die Reduktion der Basis $\bar{B}_c$ wobei e als kürzster Gittervektor bestimmt wird. Analog zum Angriff von Nguyen betrachten wir den modifizierten Ciphertext

$$c' = c + \sigma^n = x + e', \quad x \in L(B) \text{ und } e' \in \{0, 2\sigma\}^n.$$

Nach Satz 4.6 bestimmt man für ein $k \leq n$ zur Basis $B_{c'}$ die Basis $\bar{B}_{c'}$ mit den Eigenschaften

$$e' \in L(\bar{B}_{c'}) \quad \text{und} \quad \det L(\bar{B}_{c'}) = (2\sigma)^k \cdot \det L(B_{c'}).$$

Die Gitterreduktion der Basis $\bar{B}_{c'}$ bestimmt den Zielvektor e' mit der erwarteten Länge $\sqrt{2} \cdot \|e\|$. Durch die Basiserweiterung

$$\bar{B}_{\sigma^n, c'} := \begin{bmatrix} | & | & & | & | \\ \sigma^n & \bar{b}_1 & \cdots & \bar{b}_n & \bar{c}' \\ | & | & & | & | \\ 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & 0 & 0 \end{bmatrix}$$

wird mit Hilfe der Gitterreduktion e in der Darstellung $(e, 1, -1)^T$ als Zielvektor bestimmt. In der folgenden Tabelle werden die wesentlichen Eigenschaften der für den Angriff konstruierten Gitterbasen dargestellt:

Gitterbasis	$\bar{B}_c$	$\bar{B}_{c'}$	$\bar{B}_{\sigma^n, c'}$
Länge d. Zielvektors	$\|e\|$	$\approx \sqrt{2} \cdot \|e\|$	$\|e\|$
Gitterdeterminante	$\sigma^k \det L(B_c)$	$(2\sigma)^k \det L(B_c)$	$(2\sigma)^k \det L(B_c)$
Gitterrang	$n+1$	$n+1$	$n+2$

Tabelle 4.1: Gegenüberstellung der konstruierten Basen für den Angriff auf das GGH-Schema.

Im Gegensatz zum Angriff von Nguyen ist ein Aufzählen der Menge $Loes_{B,2\sigma}(c')$ nicht notwendig (Vergleiche den ersten Schritt im Angriff von Nguyen). Auch für den Fall das die Menge $Loes_{B,2\sigma}(c')$ zu groß ist, können diese Angriffe ausgeführt werden.

4.2.5 Experimentelle Resultate auf die Internet Challenges

Für Experimente auf die GGH-Challenges wurde neben der LLL-Block-Reduktion die in Kapitel 2.4 beschriebene BKZ-Reduktion [28] implementiert. Die Programme sind in der Programmiersprache C unter Verwendung der GMP-Integer-Bibliothek [15] geschrieben. Als Option benutzt der BKZ-Algorithmus, die von Schnorr und Hörner [24] eingeführte, geschnittene Enumeration von Basisvektoren. Unter heuristischen Annahmen bestimmt die geschnittene Enumeration mit positiver Wahrscheinlichkeit den kürzesten Gittervektor einer gegebenen Basis. Die Experimente wurden auf Workstations des Typs c160 der Firma HP, mit der Prozessortaktfrequenz 160 MHz, durchgeführt.

Die Reduktion der Gitterbasen erfolgte zum LLL-Reduktionsparameter $\delta = 0.99$ (siehe Abschnitt 2.3) in drei Stufen:

1. LLL-Block-Reduktion mit Blockweite $k = \lfloor n^{\frac{2}{3}} \rfloor$ (Siehe Kapitel 3.2).
2. BKZ-Reduktion mit Blockweite $\beta = 20$.
3. Geschnittene BKZ-Reduktion mit Blockweite $\beta = 50$.

Angriff auf den Fehlervektor nach Kapitel 4.2.2
Die Reduktion erfolgte auf das in Gleichung (4.2) beschriebene Gitter B_c. Der Fehlervektor e wurde für die Gitterdimension 200 durch die geschnittene BKZ-Reduktion nach drei Tagen bestimmt. Für den Challenge der Dimension 250 wurde die Reduktion nach zwei Wochen Berechnungszeit erfolglos abgebrochen.

Angriff auf die geheime Gitterbasis R nach Kapitel 4.2.2
Die Reduktionen erfolgten in n Stufen auf das in Gleichung (4.3) beschriebene Gitter. Für die Dimension 200 konnten alle Vektoren der geheime Basis nach 12 Stunden rekonstruiert werden.

Angriff auf die Gitterdeterminante nach Kapitel 4.2.4
Für die Challenges der Dimension $n = 200, 250, 300$ wurde wie in Satz 4.6 beschrieben das Untergitter $L(\bar{B}_c) \subset L(B_c)$ mit der Eigenschaft

$$\det L(\bar{B}_c) = \sigma^n \cdot L(B_c)$$

konstruiert. Mit anderen Worten: alle Koordinatentransformationen zur Erhöhung der Gitterdeterminante konnten durchgeführt werden. Die anschließende Reduktion des Gitters $L(\bar{B}_c)$ führte zu den folgenden Ergebnissen:

- In den Dimensionen 200 und 250 wurde der Fehlervektor nach 5 bzw. 40 Minuten bestimmt (BKZ-Reduktion $\beta = 20$).

- In Dimension 300 wurde der Fehlervektor mit Hilfe der geschnittenen BKZ-Reduktion mit Blockweite $\beta = 70$ nach 6 Tagen bestimmt.

Die Angriffe mit Hilfe der Basen $\bar{B}_{c'}$ und $\bar{B}_{\sigma^n,c'}$ konnten aus Zeitgründen nicht mehr experimentell durchgeführt werden. Wir vermuten, daß die Determinanten der konstruierten Untergitter im Vergleich zu $L(\bar{B}_c)$ nochmals um den Faktor 2^n wächst, und das diese Angriffe ähnlich erfolgreich sind wie der in Abschnitt 4.2.3 vorgestellte Angriff von Nguyen.

Literaturverzeichnis

[1] M. Ajtai,
The Shortest Vector Problem is NP-Hard for Randomized Reductions, Proc. 30th Symposium on Theory of Computing, (1998) 10-19.

[2] M. Ajtai, C. Dwork,
A public-key cryptosystem with worst-case/average-case equivalence, Proc. 29th Symposium on Theory of Computing, (1997) 284-293.

[3] L. Babai,
On Lovász lattice reduction and the nearest lattice point problem, Combinatorica, 6 (1986) 1-13.

[4] J. Blömer, J. P. Seifert,
On the complexity of computing short linearly independent vectors and short bases in a lattice, Proc. 31st Symposium on Theory of Computing, (1999) 711-720.

[5] J. Y. Cai,
Some Recent Progress on the Complexity of Lattice Problems, Electronic Colloqium on Computational Complexity, Report No. 6 (1999).

[6] Hervé Daudé, Brigitte Vallée,
An upper bound on the average number of iterations of the LLL algorithm, Theoretical Computer Science 123 (1994) 95-115.

[7] Dinur, Kindler, Safra,
Approximating CVP to within almost polynomial factors is NP-hard, Proc. 39 Symp. on Foundations of Computer Science, (1998) 99-109.

[8] T. ElGamal,
A public key cryptosystem and a signature sheme based on discrete logarithms, IEEE Transactions on Information Theory IT-31, (1985), 473-481.

[9] P. van Emde Boas,
Another NP-complete partition problem and the complexity of computing short vectors in lattices, Technischer Report 81-04, Mathematisches Department, Universität Amsterdam, (1981).

[10] W. M. Gentleman,
Error Analysis of QR Decompositions by Givens Transformations, Linear Algebra and its Applications 10, (1975) 189-197.

[11] Oded Goldreich, Shafi Goldwasser, Shai Halevi,
Public-Key Cryptosystems from Lattice Reduction Problems, Proc. of Crypto'97, Lecture Notes in Computer Science, Vol. 1294, Springer, (1997) 112-131.

[12] The Goldreich-Goldwasser-Halevi Cryptosystem - Challenges in dimension 200 - 400:
"`http://theory.lcs.mit.edu/~shaih/challenge.html`".

[13] P.N. Gruber, C.G. Lekkerkerker,
Geometry of Numbers - 2. ed., North Holland, Amsterdam (1987).

[14] G. H. Golub und C. F. Van Loan,
Matrix Computations, The Johns Hopkins University Press, London (1989).

[15] The GNU Multiple Precision Arithmetic Library (GMP), Edition 2.0.2, Free Software Foundation, Bosten, USA, (June 1997).

[16] A. K. Lenstra, H. W. Lenstra, L. Lovász,
Factoring polynomials with rational coefficients, Math. Ann. 261 (1982), 515-534.

[17] L. Lovász,
An Algorithmic Theory of Numbers, Graphs and Convexity, Society for Industrial and Applied Mathemathics, Philadelphia, Pennsylvania (1986).

[18] R. J. McEliece,
A public-key cryptosystem based on algebraic coding theory, JPL Pasadena, DSN Progress Reports 42-44, (1978) 114-116.

[19] A. J. Menezes, P. C. van Oorschot, S. A. Vanstone,
Handbook of Applied Cryptography - 4. ed., CRC Press (1996).

[20] D. Micciancio,
The shortest vector problem is NP-hard to approximate within some constant. Proc. 39th Symposium on Foundations of Computer Science, (1998), 92-98.

[21] R. Mennicken, E. Wagenführer,
Numerische Mathematik 1, Rowohlt Taschenbuch Verlag GmbH, Hamburg, (1977).

[22] P. Nguyen,
Cryptoanalysis of the Goldreich-Goldwasser-Halevi Cryptosystem from Crypto '97, Advances in Cryptology - Crypto '99, (1999).

[23] R.L. Rivest, A. Shamir, L. Adleman,
A method for obtaining digital signatures and public key cryptosystems, Communications of the ACM 21, (1978) 120-126.

[24] C.P. Schnorr, H.H. Hörner,
Attacking the Chor-Rivest Cryptosystem by Improved Lattice Reduction, Advances in Cryptology - Eurocrypt '95, Lecture Notes in Computer Science, Vol. 921, Springer Verlag, (1995) 1-12.

[25] C.P. Schnorr,
A Hierarchy of Polynomial Time Lattice Basis Reduction Algorithms, Theoretical Computer Science 53 (1987) 201-224.

[26] C.P. Schnorr,
A more Efficient Algorithm for Lattice Basis Reduction, Journal of Algorithms 9, 47-62 (1988).

[27] C.P. Schnorr,
Block Reduced Lattice Basis and Succesive Minima, Combinatorics, Probability and Computing (1994) 3, 507 - 522.

[28] C.P. Schnorr, M. Euchner,
Lattice Basis Reduction: Improved Practical Algorithms and Solving Subset Sum Problems, (1993).

[29] C.P. Schnorr,
Vorlesungsscript Ganzzahlige Optimierung und Gitterreduktion, 4. September (1996).

[30] A. Schönhage,
Factorization of univariante integer polynomials by diophantine approximation and an improved basis reduction algorithm, Lecture Notes in Computer Science, Vol. 172, Springer, (1985) 436-447.

[31] B. Vallée,
Gauss' Algorithm Revisited, Journal of Algorithms 12, (1991) 556-572.

[32] J.H. Wilkinson,
Rundungsfehler, Springer, (1969).