WORDT
NIET UITGELEEND

EEN ONDERZOEK NAAR
(DE RELATIE TUSSEN)
HET LLL- EN HET KETTINGBREUK-ALGORITME

DANNY OORBURG. 1990.

$$k+\cfrac{1}{e+\cfrac{1}{t+\cfrac{1}{t+\cfrac{1}{i+\cfrac{1}{n+\cfrac{1}{g+\cfrac{1}{b+\cfrac{1}{r+\cfrac{1}{e+\cfrac{1}{u+\cfrac{1}{k}}}}}}}}}}}$$

LLL ?

Rijksuniversiteit Groningen
Bibliotheek
Wiskunde / Informatica / Rekencentrum
Landleven 5
Postbus 800
9700 AV Groningen

12 MEI 1992

6 NOV. 1992

Inhoud :

0. Inleiding 2

1. Roosters 3

2. Eigenschappen van een (LLL)-gereduceerde basis 5

3. Het LLL-algoritme 7

4. Toepassingen van het LLL-algoritme 12

5. Factoriseren met LLL 14

6. Het ontbindingsalgoritme 18

7. LLL als kettingbreuk 20

8. Meer over kettingbreuken 22

9. De kettingbreuk van een algebraisch getal 24

10. De mathematica-implementatie van het LLL-ontbindingsalgoritme 28

11. De mathematica-implementatie van het kettingbreukalgoritme voor algebraische getallen 31

Literatuurlijst 33

0. Inleiding

Deze afstudeerscriptie is geschreven naar aanleiding van het in 1982 geschreven en alom bejubelde artikel 'Factoring Polynomials with Rational Coefficients' van Lenstra,Lenstra en Lovász [1]. In dat artikel wordt een nieuw algoritme om een basis voor een rooster te reduceren beschreven. Het bijzondere hiervan is dat het algoritme in polynomiale tijd termineert.
In hetzelfde artikel wordt een methode, gebaseerd op het reductie-algoritme, beschreven waarmee een polynoom in polynomiale tijd in irreducibele factoren ontbonden kan worden.
Ik zal in de eerste 5 hoofdstukken de stof uit het bewuste artikel uitleggen en dit leidt in hoofdstuk 6 tot het ontbindingsalgoritme.
Een speciale toepassing van het reductie-algoritme is de mogelijkheid om getallen te benaderen door breuken. Ik zal in hoofdstuk 7 laten zien dat deze methode heel erg op het bekende kettingbreuk-algoritme lijkt, met een klein verschil waardoor het sneller termineert.
Dit verschil zal interessant genoeg blijken om in de resterende hoofdstukken verder te onderzoeken en dat levert ons in hoofdstuk 9 een nieuw en snel algoritme op om algebraïsche getallen door kettingbreuken te benaderen.
Tot slot volgt in de hoofdstukken 10 en 11 de mathematica-implementatie van beide algoritmen.

1. Roosters

Een deelverzameling L van de n-dimensionale vectorruimte R^n heet een **rooster** als er een basis $b_1, b_2, \ldots, b_n$ voor R^n is, zodanig dat

$$(1.1) \quad L = \sum_{i=1}^{n} Zb_i = \{\sum_{i=1}^{n} r_i b_i : r_i \in Z(1 \leq i \leq n)\}$$

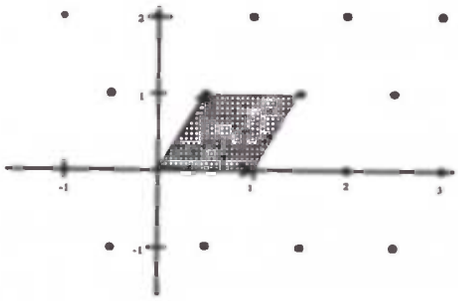
De vectoren $b_1, b_2, \ldots, b_n$ heten een **basis** voor L.

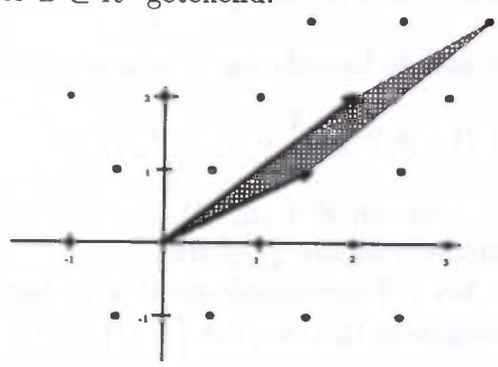
De **determinant** d(L) van L wordt gedefinieerd door

$$(1.2) \quad d(L) = |det(b_1, b_2, \ldots, b_n)|$$

waar de b_i geschreven worden als kolom vectoren. De determinant van een rooster is onafhankelijk van de keuze van de basis.

Voor veel toepassingen willen we dat een rooster gegeven wordt door een zo klein mogelijke basis, d.w.z., een basis bestaande uit zo kort mogelijke vectoren. Ter illustratie is hieronder tweemaal (een deel van) hetzelfde rooster $L \subset R^2$ getekend.





Dit rooster wordt zowel opgespannen door $\left\{\left(\begin{smallmatrix}\frac{1}{2}\\1\end{smallmatrix}\right), \left(\begin{smallmatrix}1\\0\end{smallmatrix}\right)\right\}$ als door $\left\{\left(\begin{smallmatrix}2\\2\end{smallmatrix}\right), \left(\begin{smallmatrix}1\frac{1}{2}\\1\end{smallmatrix}\right)\right\}$. De eerste basis is intuïtief een 'kleinere basis', maar hoe moeten we het begrip kleine basis precies definiëren ?
De determinant van een rooster is precies het volume van het parallelepipedum opgespannen door de vectoren in de basis. In dit voorbeeld is dat het gearceerde oppervlak. Hieraan zie je dat een kleine basis dichter bij een orthogonale basis in R^n moet zijn en in dit 2-dimensionale voorbeeld dus dat de hoek dichter bij 90 graden moet zijn.
Deze opmerking is de basis van het idee van een gereduceerde basis, zoals gedefinieerd door Lenstra, Lenstra en Lovàsz.

Beschouw eerst het Gramm-Schmidt orthogonalisatie proces. De vectoren $b_i^*(1 \leq i \leq n)$ en de reële getallen $\mu_{ij}(1 \leq j < i \leq n)$ worden inductief gedefinieerd door

$$(1.3) \quad b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{ij} b_j^*$$

$$(1.4) \quad \mu_{ij} = \frac{(b_i, b_j^*)}{(b_j^*, b_j^*)}$$

De vectoren $b_1^*, b_2^*, \ldots, b_n^*$ die op deze manier opgebouwd worden vormen een orthogonale basis voor R^n.

Een basis $b_1, b_2, \ldots, b_n$ heet **(LLL-) gereduceerd** als

$$(1.5) \quad |\mu_{ij}| \leq \frac{1}{2} \qquad \textit{voor} \;\; 1 \leq j < i \leq n$$

en

$$(1.6) \quad |b_i^* + \mu_{ii-1} b_{i-1}^*|^2 \geq \frac{3}{4} |b_{i-1}^*|^2 \qquad \textit{voor} \;\; 1 < i \leq n$$

de laatste formulering is equivalent aan

$$(1.7) \quad |b_i^*|^2 \geq (\frac{3}{4} - \mu_{ii-1}^2) |b_{i-1}^*|^2$$

De vectoren $b_i^* + \mu_{ii-1} b_{i-1}^*$ en b_{i-1}^* zijn de projecties van b_i en b_{i-1} op het orthogonale complement van $\sum_{j=1}^{i-2} R b_j$.
In het 2-dimensionale geval is de laatste eis dus gewoon $|b_2|^2 > \frac{3}{4}|b_1|^2$. Een eenvoudige berekening laat zien dat $\left\{ \begin{pmatrix} \frac{1}{2} \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right\}$ een gereduceerde basis voor L is en $\left\{ \begin{pmatrix} 2 \\ 2 \end{pmatrix}, \begin{pmatrix} 1\frac{1}{2} \\ 1 \end{pmatrix} \right\}$ niet.

2. Eigenschappen van een (LLL-) gereduceerde basis.

De stellingen die we nu zullen formuleren en bewijzen, volgen rechtstreeks uit de definitie van een gereduceerde basis en geven aan hoe 'klein' de vectoren in een gereduceerde basis zijn.

Stelling 1 *Zij $b_1, b_2, \ldots, b_n$ een gereduceerde basis voor een rooster L in R^n, en zij $b_1^*, b_2^*, \ldots, b_n^*$ gedefinieerd als eerder. Dan geldt*

(2.1) $|b_j|^2 \leq 2^{i-1}|b_i^*|^2 \qquad voor \;\; 1 \leq j \leq i \leq n$

(2.2) $d(L) \leq \prod_{i=1}^{n} |b_i| \leq 2^{n(n-1)/4} d(L)$

(2.3) $|b_1| \leq 2^{(n-1)/4} d(L)^{1/n}$.

Bewijs.
Uit (1.6) en (1.5) volgt dat

$$|b_i^*|^2 \geq (\frac{3}{4} - \mu_{ii-1}^2) \cdot |b_{i-1}^*|^2 \geq \frac{1}{2} \cdot |b_{i-1}^*|^2$$

voor $1 < i \leq n$, dus met inductie volgt

$$|b_j^*|^2 \leq 2^{i-j} \cdot |b_i^*|^2 \qquad voor \;\; 1 \leq j \leq i \leq n.$$

Uit (1.3) en (1.5) verkrijgen we

$$|b_i|^2 = |b_i^*|^2 + \sum_{j=1}^{i-1} \mu_{ij}^2 \cdot |b_j^*|^2 \leq |b_i^*|^2 + \frac{1}{4} \sum_{j=1}^{i-1} 2^{i-j} \cdot |b_i^*|^2$$

$$= (1 + \frac{1}{4}(2^i - 2)) \cdot |b_i^*|^2 \leq 2^{i-1} \cdot |b_i^*|^2.$$

Deze resultaten samenvoegen geeft

$$|b_j|^2 \leq 2^{j-1} \cdot |b_j^*|^2 \leq 2^{i-1} \cdot |b_i^*|^2 \qquad voor \;\; 1 \leq j \leq i \leq n.$$

Dit bewijst (2.1) . Uit (1.2) en (1.3) volgt dat

$$d(L) = |det(b_1^*, b_2^*, \ldots, b_n^*)|$$

en dus, aangezien de b_i^* paarsgewijs orthogonaal zijn

$$d(L) = \prod_{i=1}^{n} |b_i^*|.$$

Uit $|b_i^*| \leq |b_i|$ en $|b_i| \leq 2^{(i-1)/2} \cdot |b_i^*|$ volgt (2.2). $j = 1$ invullen in (2.1) en het produkt nemen over $i = 2, 3, \ldots, n$ geeft (2.3). Dit bewijst stelling 1.

Stelling 2 *Zij $L \subset R^n$ een rooster met gereduceerde basis $b_1, b_2, \ldots, b_n$. Dan geldt*

(2.4) $|b_1|^2 \leq 2^{n-1}|x|^2$

voor elke $x \in L$, $x \neq 0$.

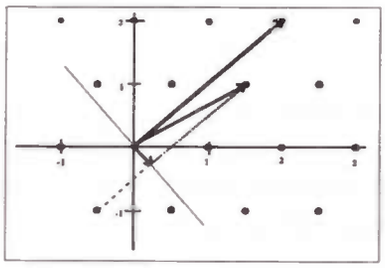
Bewijs.
Schrijf $x = \sum_{i=1}^{n} r_i b_i = \sum_{i=1}^{n} r_i' b_i^*$ met $r_i \in Z, r_i' \in R$ $(1 \leq i \leq n)$. Als i de grootste index met $r_i \neq 0$, dan $r_i' = r_i$, dus

$$|x|^2 \geq r_i' \cdot |b_i^*|^2 \geq |b_i^*|^2$$

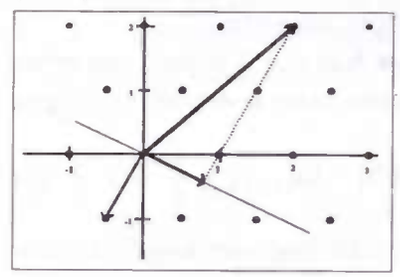
Wegens (2.1) hebben we $|b_1|^2 \leq 2^{i-1} \cdot |b_i^*|^2 \leq 2^{n-1} \cdot |b_i^*|^2$. Dit bewijst stelling 2.

3. Het LLL-algoritme

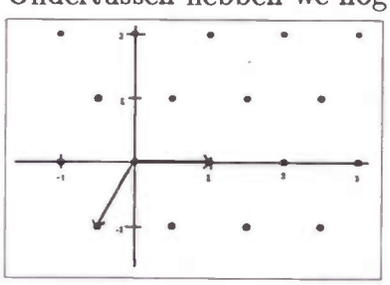
We zullen in dit hoofdstuk een algoritme beschrijven dat voor een gegeven rooster een gereduceerde basis construeert. We beginnen met een voorbeeld dat aangeeft hoe het algoritme er ongeveer uit komt te zien. Bekijk nogmaals het rooster L uit hoofdstuk 1. Stel dat L gegeven wordt door de basisvectoren $b_1 = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$ en $b_2 = \begin{pmatrix} 1\frac{1}{2} \\ 1 \end{pmatrix}$.
Hoe vinden we vanuit deze basis een gereduceerde basis voor L ?



Hiernaast is naast de twee basisvectoren ook de orthogonale projectie b_2^* getekend. Je ziet dat van alle vectoren in L van de vorm $b_2 + zb_1 (z \in Z)$ de vector b_2 niet degene is die het dichtst bij b_2^* is (dit is equivalent met de uitspraak $|\mu_{21}| > \frac{1}{2}$). De vector die dat wel is, is de vector $b_2 + rb_1$, waar r de afronding van μ_{21} naar het dichtstbijzijnde gehele getal is. In dit geval is dat de vector $\begin{pmatrix} -\frac{1}{2} \\ -1 \end{pmatrix} (\mu_{21} = -0.63)$.



Bekijk daarom nu het stelsel $\{b_1, b_2\} = \{\begin{pmatrix} 2 \\ 2 \end{pmatrix}, \begin{pmatrix} -\frac{1}{2} \\ -1 \end{pmatrix}\}$.
Aangezien $|b_2|^2 < \frac{3}{4}|b_1|^2$ is dit geen gereduceerde basis.
We voeren de meest voor de hand liggende oplossing van dit probleem uit en draaien de volgorde om.
Hierdoor hebben we te maken met een nieuwe b_2^* en μ_{21} en zijn we in de situatie zoals hiernaast is getekend.
Ondertussen hebben we nog steeds niet te maken met



een gereduceerde basis, omdat nu $|\mu_{21}|$ opnieuw groter dan $\frac{1}{2}$ is.(Om precies te zijn $\mu_{21} = -2.4$)
We vervangen nu als eerder b_2 door $b_2 - 2b_1$, dat r 2 moet zijn zie je ook duidelijk in bovenstaande tekening.
Op deze wijze komen we uit op het stelsel $\{\begin{pmatrix} -\frac{1}{2} \\ -1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}\}$ en dit is een gereduceerde basis voor L.

Dit is het 2-dimensionale geval van een algemener algoritme dat een n-dimensionale basis $b_1, b_2, \ldots, b_n$ voor een rooster $L \subset R^n$ omzet in een gereduceerde basis. Het complete algoritme volgt nu.

Om het algoritme te initialiseren berekenen we $b_i^* (1 \leq i \leq n)$ en $\mu_{ij} (1 \leq j < i \leq n)$ m.b.v. de Gramm-Schmidt procedure. In de loop van het algoritme zullen de vectoren $b_1, b_2, \ldots, b_n$ verscheidene malen worden aangepast, maar ze zullen altijd een basis voor L vormen. Na elke verandering van de b_i zullen de b_i^* en μ_{ij} aangepast worden, zodat steeds blijft gelden

$$(3.1) \quad b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{ij} b_j^* \qquad (1 \leq i \leq n)$$

$$(3.2) \quad \mu_{ij} = \frac{(b_i, b_j^*)}{(b_j^*, b_j^*)} \qquad (1 \leq j < i \leq n)$$

Tijdens elke stap van het algoritme houden we een 'teller' $k \in \{1, 2, \ldots, n+1\}$ bij. We beginnen met $k = 2$.
Bekijk de volgende voorwaarden

$$(3.3) \quad |\mu_{ij}| \leq \frac{1}{2} \qquad voor \;\; 1 \leq j < i < k$$

en

$$(3.4) \quad |b_i^* + \mu_{ii-1} b_{i-1}^*|^2 \geq \frac{3}{4} |b_{i-1}^*|^2 \qquad voor \;\; 1 < i < k$$

Dit zegt dat de vectoren $b_1, \ldots, b_{k-1}$ voldoen aan de LLL-condities. (Dit klopt triviaal als $k = 2$.)
We zullen in het algoritme steeds een aantal opeenvolgende stappen uitvoeren zodanig dat daarna het bovenstaande (zij het voor een andere k) weer geldig is.

Begin algoritme.
Als $k = n + 1$ is de basis gereduceerd en termineert het algoritme. Stel nu dat $k \leq n$. De eerste stap is ervoor te zorgen dat

$$(3.5) \quad |\mu_{kk-1}| \leq \frac{1}{2} \;\; als \;\; k > 1$$

Als dit nog niet klopt, doen we het volgende.

Zij r de integer het dichtst bij μ_{kk-1} en vervang b_k door $b_k - rb_{k-1}$. De getallen μ_{kj} met $j < k - 1$ worden dan vervangen door $\mu_{kj} - r\mu_{k-1j}$, want

$$\mu_{kj} := \frac{(b_k - rb_{k-1}, b_j^*)}{(b_j^*, b_j^*)} = \frac{(b_k, b_j^*)}{(b_j^*, b_j^*)} - r\frac{(b_{k-1}, b_j^*)}{(b_j^*, b_j^*)} = \mu_{kj} - r\mu_{k-1j}$$

Het belangrijkste is dat we nu μ_{kk-1} kunnen vervangen door $\mu_{kk-1} - r$. Zie de formule hierboven en merk op dat $\frac{(b_{k-1}, b_{k-1}^*)}{(b_{k-1}^*, b_{k-1}^*)} = 1$, na deze veranderingen geldt $|\mu_{kk-1}| \leq \frac{1}{2}$.
Vervolgens onderscheiden we twee gevallen.

1. Stel dat $k \geq 2$ en

$$(3.6) \quad |b_k^* + \mu_{kk-1} b_{k-1}^*|^2 \;\; < \;\; \frac{3}{4} |b_{k-1}^*|^2$$

In dat geval verwisselen we b_{k-1} en b_k en laten de andere b_i onveranderd.
Dit heeft tot gevolg dat we de vectoren b_{k-1}^* en b_k^* en de getallen μ_{kk-1},μ_{k-1j}, μ_{kj},μ_{ik-1},μ_{ik} voor $j < k - 1$ en $i > k$ moeten veranderen. Dit zijn de getallen die anders worden als we de Gramm-Schmidt orthogonalisatie opnieuw uitvoeren met de rij vectoren $b_1, \ldots, b_k, b_{k-1}, \ldots, b_n$.
Door precies de verschillen met de vorige b_i^* en μ_{ij} te bekijken kunnen we formules afleiden voor de veranderingen, i.p.v. de procedure te herhalen. Deze formules volgen later.
Het belangrijkste dat hier nu gebeurt, is dat b_{k-1}^* wordt vervangen door $b_k^* + \mu_{kk-1} b_{k-1}^*$.

Dus de nieuwe waarde van $|b^*_{k-1}|^2$ is kleiner dan $\frac{3}{4}$ keer de oude waarde.
Vervolgens vervangen we k door $k-1$. Nu zijn we weer in de situatie beschreven in (3.3) en (3.4), en we gaan vanaf daar weer verder.

2. Stel dat $k = 1$ of

$$(3.7) \quad |b^*_k + \mu_{kk-1} b^*_{k-1}|^2 \;\geq\; \frac{3}{4}|b^*_{k-1}|^2$$

In dat geval zorgen we eerst dat

$$(3.8) \quad |\mu_{kj}| \leq \frac{1}{2} \qquad \textit{voor} \;\; 1 \leq j < k-1$$

(Voor j=k-1 is dit al waar, daar hebben we eerder al voor gezorgd.)
Als dit nog niet klopt, doen we het volgende.
Zij l de grootste index $< k$ met $|\mu_{kl}| > \frac{1}{2}$, zij r de integer het dichtst bij μ_{kl}, en vervang b_k door $b_k - rb_l$. Als eerder moeten nu de getallen μ_{kj} met $j < l$ vervangen worden door $\mu_{kj} - r\mu_{lj}$, en μ_{kl}, zoals gewenst, door $\mu_{kl} - r$. Alle andere μ_{ij} blijven onveranderd. Ook de b^*_i blijven onveranderd, beschouw bijvoorbeeld het effect op b^*_k :

$$b^*_k := (b_k - rb_l) \;-\; (\sum_{j=1}^{l-1}(\mu_{kj} - r\mu_{lj})b^*_j \;+\; (\mu_{kl} - r)b^*_l \;+\; \sum_{j=l+1}^{k-1} \mu_{kj} b^*_j)$$

$$= b_k \;-\; \sum_{j=1}^{k-1} \mu_{kj} b^*_j \;-\; rb_l \;+\; r\sum_{j=1}^{l-1} \mu_{lj} b^*_j \;+\; rb^*_l = b^*_k - r(b^*_l - b^*_l) = b^*_k$$

Dit herhalen we totdat (3.8) geldt. Vervolgens vervangen we k door $k+1$. Dan zijn we weer in de situatie beschreven in (3.3) en (3.4) en we gaan vanaf daar weer verder. In het geval $k = 1$ hebben we niet meer gedaan dan k vervangen door 2. Dit beëindigt de beschrijving van het algoritme.

Voordat we het bewijs geven dat het algoritme termineert geven we eerst de formules waar we in geval 1 naar verwezen.
We noemen de basis van voor de verandering $b_1, b_2, \ldots, b_n$ en we duiden met c_i, c^*_i en ν_{ij} de vectoren en getallen aan die respectievelijk b_i, b^*_i en μ_{ij} gaan vervangen.
De basis $c_1, c_2, \ldots, c_n$ wordt gegeven door

$$c_{k-1} = b_k, \qquad c_k = b_{k-1}, \qquad c_i = b_i \quad \textit{voor} \quad i \neq k-1, k$$

Als $i \neq k-1, k$ geldt $c^*_i = b^*_i$, omdat c^*_i de projectie is van $c_i = b_i$ op het orthogonale complement van $\sum_{j=1}^{i-1} Rc_j = \sum_{j=1}^{i-1} Rb_j$. Verder geldt $\nu_{ij} = \frac{(c_i, c^*_j)}{(c^*_j, c^*_j)} = \frac{(b_i, b^*_j)}{(b^*_j, b^*_j)} = \mu_{ij}$ als $\{i, j\} \cap \{k-1, k\} = \emptyset$.
Hieruit kunnen we afleiden, zoals we graag willen

$$c^*_{k-1} = c_{k-1} - \sum_{j=1}^{k-2} \nu_{k-1j} c_j = b_k - \sum_{j=1}^{k-2} \mu_{kj} b^*_j = b^*_k + \mu_{kk-1} b^*_{k-1}$$

en vervolgens, nu we c^*_{k-1} weten

$$\nu_{kk-1} = \frac{(c_k, c^*_{k-1})}{(c^*_{k-1}, c^*_{k-1})} = \frac{(b^*_{k-1}, c^*_{k-1})}{|c^*_{k-1}|^2} = \mu_{kk-1}\frac{|b^*_{k-1}|^2}{|c^*_{k-1}|^2}$$

$$c^*_k = b^*_{k-1} - \nu_{kk-1}c^*_{k-1}.$$

Zij nu $i > k$. Om ν_{ik-1} en ν_{ik} te vinden substitueren we de uitdrukkingen

$$b^*_{k-1} = \nu_{kk-1}c^*_{k-1} + c^*_k$$

$$b^*_k = (1 - \mu_{kk-1}\nu_{kk-1})c^*_{k-1} - \mu_{kk-1}c^*_k = \frac{|b^*_k|^2}{|c^*_{k-1}|^2}c^*_{k-1} - \mu_{kk-1}c^*_k$$

in $b_i = b^*_i + \sum_{j=1}^{i-1}\mu_{ij}b^*_j = c^*_i + \sum_{j=1}^{i-1}\nu_{ij}c^*_j$. Dit geeft

$$\nu_{ik-1} = \mu_{ik-1}\nu_{kk-1} + \mu_{ik}\frac{|b^*_k|^2}{|c^*_{k-1}|^2}$$

$$\nu_{ik} = \mu_{ik-1} - \mu_{ik}\mu_{kk-1}$$

Als laatste hebben we nog

$$\nu_{k-1j} = \frac{(c_{k-1}, c^*_j)}{(c^*_j, c^*_j)} = \frac{(b_k, b^*_j)}{(b^*_j, b^*_j)} = \mu_{kj}, \qquad \nu_{kj} = \mu_{k-1j}$$

voor $1 \leq j < k-1$.

Om te bewijzen dat het algoritme termineert definiëren we

$$(3.9) \quad d_i = det\begin{pmatrix} (b_1,b_1) & (b_1,b_2) & \cdots & (b_1,b_i) \\ (b_2,b_1) & (b_2,b_2) & \cdots & (b_2,b_i) \\ \vdots & \vdots & \ddots & \vdots \\ (b_i,b_1) & (b_i,b_2) & \cdots & (b_i,b_i) \end{pmatrix}$$

voor $1 \leq i \leq n$.
We weten dat $(b_j, b_l) = (b^*_j + \sum_{k=1}^{j-1}\mu_{jk}b^*_k, b^*_l + \sum_{k=1}^{l-1}\mu_{lk}b^*_k)$ en aangezien $b^*_1, b^*_2, \ldots, b^*_n$ een orthogonaal stelsel is, geldt dus voor $1 \leq j < l \leq i$:

$$(b_j, b_l) = (b_l, b_j) = \mu_{lj}|b^*_j|^2 + \sum_{k=1}^{j-1}\mu_{jk}\mu_{lk}|b^*_k|^2$$

Op dezelfde manier vinden we op de diagonaal

$$(b_j, b_j) = |b^*_j|^2 + \sum_{k=1}^{j-1}\mu^2_{jk}|b^*_k|^2$$

dus d_i is de determinant van de symmetrische $i \times i$ matrix

$$\begin{pmatrix} |b_1^*|^2 & \mu_{21}|b_1^*|^2 & \cdots & \mu_{i1}|b_1^*|^2 \\ \mu_{21}|b_1^*|^2 & |b_2^*|^2 + \mu_{21}^2|b_1^*|^2 & \cdots & \mu_{i2}|b_2^*|^2 + \mu_{21}\mu_{i1}|b_1^*|^2 \\ \vdots & \vdots & \ddots & \vdots \\ \mu_{i1}|b_1^*|^2 & \mu_{i2}|b_2^*|^2 + \mu_{21}\mu_{i1}|b_1^*|^2 & \cdots & |b_i^*|^2 + \sum_{k=1}^{i-1} \mu_{ik}^2|b_k^*|^2 \end{pmatrix}$$

Als we bovenstaande matrix schrijven als product van een beneden- en een bovendriehoeksmatrix zien we, aangezien $det(A \cdot B) = det(A) \cdot det(B)$, dat

$$(3.10) \quad d_i = det\left(\begin{pmatrix} |b_1^*|^2 & 0 & \cdots & 0 \\ \mu_{21}|b_1^*|^2 & |b_2^*|^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \mu_{i1}|b_1^*|^2 & \mu_{i2}|b_2^*|^2 & \cdots & |b_i^*|^2 \end{pmatrix} \cdot \begin{pmatrix} 1 & \mu_{21} & \cdots & \mu_{i1} \\ 0 & 1 & \cdots & \mu_{i2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}\right) = \prod_{j=1}^{i} |b_j^*|^2$$

voor $1 \leq i \leq n$.
Dus de d_i zijn positieve reële getallen. Definieer nu

$$(3.11) \quad D = \prod_{i=1}^{n-1} d_i$$

Wegens (3.10) verandert het getal D in de loop van het algoritme alleen als er een b_i^* verandert en dat gebeurt alleen in geval 1. Dan wordt het getal d_{k-1} gereduceerd met een factor $< \frac{3}{4}$. De andere d_i blijven hetzelfde, in (3.9) verandert niets als $i < k-1$ en als $i > k-1$ vindt in de matrix zowel een kolom- als een rijverwisseling plaats en blijft de waarde van de determinant onveranderd. De conclusie is dat dan D wordt gereduceerd met een factor $< \frac{3}{4}$. Beneden zullen we bewijzen dat er een positieve benedengrens voor d_i is die alleen afhangt van L, daaruit volgt dat er ook een positieve benedengrens is voor D en dus een bovengrens voor het aantal keren dat we geval 1 passeren.
Omdat het aantal keren dat we geval 2 doorlopen ten hoogste $n-1$ meer is dan het aantal keren dat we geval 1 doorlopen, is het aantal stappen van het algoritme begrensd en termineert het algoritme.
Om te bewijzen dat d_i een benedengrens heeft, definiëren we

$$(3.12) \quad m(L) = min\{|x|^2 : x \in L, x \neq 0\}.$$

We beschouwen d_i als de determinant van het rooster

$$L_i = \sum_{j=1}^{i} Zb_j \subset L.$$

Wegens [2, Hoofdstuk 3, Gevolg 2.12] heeft het rooster $L_i \subset R^n$ een element a met $|a|^2 < n \cdot d(L_i)^{1/n}$, zodat we kunnen concluderen dat

$$m(L) \leq |a|^2 \leq n \cdot d_i^{1/n} \quad \Leftrightarrow \quad d_i \geq \left(\frac{m(L)}{n}\right)^n$$

en we een benedengrens voor d_i verkregen hebben.

4. Toepassingen van het LLL-algoritme

Nu we weten dat elk rooster een gereduceerde basis bezit en we een algoritme hebben om een gereduceerde basis voor een gegeven rooster te construeren, kunnen we de stellingen uit hoofdstuk 2 gaan toepassen.

De eerste toepassing is die van simultane diofantische benadering :
gegeven een positieve n en rationale getallen $\alpha_1, \alpha_2, \ldots, \alpha_n, \varepsilon$ met $0 < \varepsilon < 1$ kunnen we gehele getallen $p_1, p_2, \ldots, p_n, q$ vinden, zodanig dat

$$|p_i - q\alpha_i| \leq \varepsilon \quad \textit{voor} \quad 1 \leq i \leq n,$$

en

$$1 \leq q \leq 2^{n(n+1)/4}\varepsilon^{-n}.$$

Dat gaat als volgt. Zij L het rooster van rang $n+1$ opgespannen door de vectoren $b_1, b_2, \ldots, b_{n+1}$, waar b_i de i-de kolom is van de $(n+1) \times (n+1)$-matrix

$$\begin{pmatrix} 1 & 0 & \cdots & 0 & -\alpha_1 \\ 0 & 1 & \cdots & 0 & -\alpha_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -\alpha_n \\ 0 & 0 & \cdots & 0 & 2^{-n(n+1)/4}\varepsilon^{n+1} \end{pmatrix}$$

Met behulp van het vorige algoritme vinden we een gereduceerde basis $b'_1, b'_2, \cdots, b'_{n+1}$ voor L en stelling 1 vertelt ons dan dat

$$|b'_1| \leq 2^{n/4} \cdot d(L)^{1/(n+1)} = \varepsilon$$

Omdat $b'_1 \in L$ bestaan er $p_1, p_2, \ldots, p_n, q \in Z$ zodanig dat $b'_1 = p_1 b_1 + p_2 b_2 + \ldots + p_n b_n + q b_{n+1}$, oftewel

$$b'_1 = \begin{pmatrix} p_1 - q\alpha_1 \\ p_2 - q\alpha_2 \\ \vdots \\ p_n - q\alpha_n \\ q \cdot 2^{-n(n+1)/4}\varepsilon^{n+1} \end{pmatrix}$$

Er volgt dat

$$|p_i - q\alpha_i| \leq \varepsilon \quad \textit{voor} \quad 1 \leq i \leq n$$

en

$$|q| \leq 2^{n(n+1)/4}\varepsilon^{-n}.$$

Uit $\varepsilon < 1$ en $b_1 \neq 0$ zien we dat $q \neq 0$. Als we $q < 0$ vinden kunnen we, door b'_1 te vervangen door $-b'_1$, bereiken dat $q > 0$.

Een andere toepassing van het LLL-algoritme is het vinden van $Q-$lineaire relaties tussen gegeven reële getallen $\alpha_1, \alpha_2, \ldots, \alpha_n$. Om die te vinden nemen we voor L het rooster Z^n, ingebed in R^{n+1} door

$$(m_1, m_2, \ldots, m_n) \mapsto (m_1, m_2, \ldots, m_n, c \sum_{i=1}^{n} m_i \alpha_i')$$

hier is c een groot getal en α_i' is een goede rationale benadering voor α_i.
Het LLL-algoritme zal in de eerste basisvector relatief kleine gehele getallen $m_1, m_2, \ldots, m_n$ vinden zodanig dat $\sum_{i=1}^{n} m_i \alpha_i'$ heel klein is.
Als we hetzelfde doen met $\alpha_i = \alpha^{i-1}$ zien we dat het algoritme gebruikt kan worden om voor een gegeven reëel getal α te kijken of het een algebraïsch getal is en om zijn minimumpolynoom te bepalen.
We willen bijvoorbeeld het minimumpolynoom van $1 + \sqrt{5} \approx 3.236068$ weten, we construeren de basis voor L als volgt

$$b_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 10000 \end{pmatrix}, \quad b_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 32360 \end{pmatrix}, \quad b_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 104721 \end{pmatrix}.$$

Als we het LLL-algoritme vervolgens uitvoeren, krijgen we als eerste vector in de gereduceerde basis

$$b_1' = \begin{pmatrix} -4 \\ -2 \\ 1 \\ 1 \end{pmatrix}$$

en we hebben als minimumpolynoom voor $1 + \sqrt{5}$ het polynoom $f(X) = X^2 - 2X - 4$ gevonden.

5. Factoriseren met LLL

De toepassing van het LLL-algoritme waar het de heren Lenstra,Lenstra en Lovász eigenlijk om te doen was, is de mogelijkheid om het LLL-algoritme te gebruiken om polynomen te ontbinden in irreducibele factoren. De stellingen die hieronder zijn gegeven zullen ons in staat stellen dit te realiseren.

In het volgende is p een priemgetal en k een positief geheel getal.
Kies een vast polynoom $f \in Z[X]$ van graad n en een polynoom $h \in Z[X]$ met de volgende eigenschappen:

(5.1) *h is monisch*

(5.2) *$(h \bmod p^k)$ is een deler van $(f \bmod p^k)$ in $(Z/p^kZ)[X]$*

(5.3) *$(h \bmod p)$ is irreducibel in $F_p[X]$*

(5.4) *$(h \bmod p)^2$ is geen deler van $(f \bmod p)$ in $F_p[X]$*

Definieer $l = gr(h)$, dus $0 < l \leq n$.

Deze voorwaarden geven de volgende voorstellingen van f :

1.

$$f = h_0 \cdot f_1 \cdots f_j$$

2.

$$(f \bmod p^k) = (h \bmod p^k) \cdot (r \bmod p^k) \cdot (f_1 \bmod p^k) \cdots (f_j \bmod p^k)$$

3.

$$(f \bmod p) = (h \bmod p) \cdot (r_1 \bmod p) \cdots (r_k \bmod p) \cdot (f_{11} \bmod p) \cdots (f_{1l} \bmod p) \cdots (f_{j1} \bmod p) \cdots (f_{jm} \bmod p)$$

Toelichting:
1. f is het product van een aantal irreducibele factoren. Wegens (5.2) en (5.3) is er een irreducibele factor van f, die modulo p genomen een veelvoud van $(h \bmod p)$ in $F_p[X]$ is. Laten we deze factor h_0 noemen. Wegens (5.4) is h_0 uniek tot op het teken. Als f zelf irreducibel is geldt natuurlijk $f = h_0$.
2. $(f \bmod p^k)$ is het product van de irreducibele factoren uit 1, maar dan modulo p^k genomen. In $(Z/p^kZ)[X]$ hoeven deze niet meer irreducibel te zijn. Dat $(h \bmod p^k)$ een deler van $(h_0 \bmod p^k)$ in $(Z/p^kZ)[X]$ is, zie je als volgt:
Wegens (5.3) zijn de polynomen $(h \bmod p)$ en $((f/h_0) \bmod p)$ relatief priem in $F_p[X]$, dus geldt

$$(\lambda_1 \bmod p) \cdot (h \bmod p) + (\mu_1 \bmod p) \cdot ((f/h_0) \bmod p) = 1$$

voor zekere $\lambda_1, \mu_1 \in Z[X]$. Dus $\lambda_1 h + \mu_1(f/h_0) = 1 - p\nu_1$ voor een zekere $\nu_1 \in Z[X]$.

Beide kanten vermenigvuldigen met $h_0 \cdot \sum_{j=0}^{k-1}(p\nu_1)^j$ geeft

$$\lambda_2 h + \mu_2 f = h_0 \ + \ p^k(\nu_1^k \ h_0)$$

voor zekere $\lambda_2, \mu_2 \in Z[X]$. Bekijk deze vergelijking modulo p^k. Het linkerlid is wegens (5.2) deelbaar door $(h \ mod \ p^k)$ in $(Z/p^kZ)[X]$ en in het rechterlid staat alleen nog h_0, dus daar moet hetzelfde voor gelden.
3. Als we $(f \ mod \ p)$ in $F_p[X]$ ontbinden in irreducibele factoren komt de uitdrukking $(h \ mod \ p)$ hier wegens (5.4) precies één keer in voor. Hieruit volgt dat hetzelfde moet gelden voor $(h \ mod \ p^k)$ in de ontbinding van $(f \ mod \ p^k)$ in $(Z/p^kZ)[X]$.

Bovenstaande stelt ons in staat de volgende stelling te formuleren:

Stelling 3 *Als g een deler van f in $Z[X]$ is, zijn de volgende drie beweringen equivalent :*

- *$(h \ mod \ p)$ is een deler van $(g \ mod \ p)$ in $F_p[X]$*
- *$(h \ mod \ p^k)$ is een deler van $(g \ mod \ p^k)$ in $(Z/p^kZ)[X]$*
- *h_0 is een deler van g in $Z[X]$*

Immers, als g een deler van f is, dan is g het product van een aantal irreducibele factoren van f. De eerste twee uitspraken kloppen d.e.s.d.a. h_0 daar één van is.

In het vervolg van dit hoofdstuk kiezen we een vast geheel getal m, met $m \geq l$ en we definiëren L als de verzameling van alle polynomen in $Z[X]$ van graad $\leq m$ die, modulo p^k genomen, deelbaar door $(h \ mod \ p^k)$ in $(Z/p^kZ)[X]$ zijn.
Door het polynoom $a_0 + a_1X + \ldots + a_nX^n$ te identificeren met de vector $(a_0, a_1, \ldots, a_n)$ zien we dat L het rooster in R^{m+1} is, gegeven door de basis

$$\{p^kX^i : 0 \leq i < l\} \ \cup \ \{hX^j : 0 \leq j \leq m-l\}$$

Er geldt, omdat h monisch is, dat $d(L) = p^{kl}$. Verder definiëren we de norm van een polynoom als de norm van de corresponderende vector. Nu we de koppeling naar de roosters gelegd hebben, kunnen we de volgende stelling bewijzen (en later gebruiken).

Stelling 4 *Als een polynoom $b \in L$ voldoet aan*

(5.5) $\quad p^{kl} \ > \ |f|^m \cdot |b|^n$

dan is b deelbaar door h_0 in $Z[X]$.

Bewijs.
We stellen $g = ggd(f, b)$, $m' = graad(b)$ en $e = graad(g)$, (er geldt dus $0 \leq e \leq m' \leq m$) en definiëren vervolgens

$$M = \{\lambda f + \mu b;\ \lambda, \mu \in Z[X], graad(\lambda) < m' - e, graad(\mu) < n - e\}$$

$$\subset Z + Z \cdot X + \ldots + Z \cdot X^{n+m'-e-1}$$

en M' is de projectie van M op

$$Z \cdot X^e + Z \cdot X^{e+1} + \ldots + Z \cdot X^{n+m'-e-1}$$

Onthoudt dat we zulke verzamelingen identificeren met deelverzamelingen van vectorruimtes. Stel bijvoorbeeld dat $f = f_0 + f_1 \cdot X + \ldots + f_4 \cdot X^4$, $b = b_0 + b_1 \cdot X + \ldots + b_3 \cdot X^3$ en $e = 1$, dan is M de vectorruimte opgespannen door de kolommen van de matrix

$$\left(\begin{array}{ccccc} f_0 & 0 & b_0 & 0 & 0 \\ \hline f_1 & f_0 & b_1 & b_0 & 0 \\ f_2 & f_1 & b_2 & b_1 & b_0 \\ f_3 & f_2 & b_3 & b_2 & b_1 \\ f_4 & f_3 & 0 & b_3 & b_2 \\ 0 & f_4 & 0 & 0 & b_3 \end{array}\right)$$

en M' is de vectorruimte opgespannen door de kolommen van de matrix onder de streep.

Stel dat $\lambda f + \mu b$ naar 0 projecteert in M', dan is $graad(\lambda f + \mu b) < e$, maar g deelt $\lambda f + \mu b$, dus $\lambda f + \mu b = 0$. Uit $\lambda \frac{f}{g} = -\mu \frac{b}{g}$ en $ggd(\frac{f}{g}, \frac{b}{g}) = 1$ volgt dat $\frac{f}{g}$ een deler is van μ, maar $graad(\mu) < n - e = graad(\frac{f}{g})$, dus $\mu = 0$ en dus geldt ook $\lambda = 0$.
Dit bewijst dat de projecties van

$$\{X^i f : i = 0, 1, \ldots, m' - e - 1\} \cup \{X^j b : j = 0, 1, \ldots, n - e - 1\}$$

op M' lineair onafhankelijk zijn. Aangezien deze projecties M' opspannen, volgt dat M' een rooster van rang $n + m' - 2e$ is. Hadamard's ongelijkheid en de aanname dat $p^{kl} > |f|^m \cdot |b|^n$ vertellen ons dat

$$d(M') \leq |f|^{m'-e} \cdot |b|^{n-e} \leq |f|^m \cdot |b|^n < p^{kl}.$$

We zullen hieronder bewijzen dat uit de aanname dat h_0 geen deler van b is, volgt dat $d(M') \geq p^{kl}$. Deze tegenspraak bewijst de stelling.

Stel dat h_0 geen deler van b is, dan is h_0 geen deler van g en stelling 3 vertelt ons dan dat $(h \ mod\ p)$ geen deler van $(g \ mod\ p)$ is. Omdat $(h \ mod\ p)$ irreducibel in $F_p[X]$ is, bestaan er dan zekere λ_1, μ_1 en $\nu_1 \in Z[X]$ met

(5.6) $\lambda_1 h + \mu_1 g = 1 - p\nu_1$

Stel $\phi \in M$ en $graad(\phi) < e + l$. Dan is g een deler van ϕ.

Als we nu (5.6) vermenigvuldigen met $\frac{\phi}{g} \cdot \sum_{j=0}^{k-1}(p\nu_1)^j$ krijgen we

(5.7) $\lambda_2 h + \mu_2 \phi = \phi/g + p^k \nu_2$

voor zekere $\lambda_2, \mu_2, \nu_2 \in Z[X]$. Modulo p^k gezien is het linkerlid deelbaar door $(h \ mod \ p^k)$, dus hetzelfde moet gelden voor $\frac{\phi}{g}$. h heeft graad l en $graad(\frac{\phi}{g}) < e + l - e = l$, dus $\phi \equiv 0 \ mod \ p^k Z[X]$.
Hieruit volgt dat

$$\{\phi \in M : graad(\phi) < e + l\} \subset p^k Z[X].$$

Dus als we een basis $b_e, b_{e+1}, \ldots, b_{n+m'-e-1}$ voor M' kiezen met $graad(b_j) = j$, dan zijn de kopcoëfficiënten van $b_e, b_{e+1}, \ldots, b_{e+l-1}$ veelvouden van p^k. Aangezien $d(M')$ de absolute waarde van het product van alle kopcoëfficiënten van $b_e, b_{e+1}, \ldots, b_{n+m'-e-1}$ is, zien we dat $d(M') \geq p^{kl}$. Dit bewijst stelling 4.
Het idee achter het factorisatie-algoritme is om k zo groot te kiezen dat een mogelijke factor h_0, waarvan de norm begrensd wordt door de Landau-Mignotti grens, een relatief kleine vector in L is.

Stelling 5 *Zij $p, k, f, n, h, l, h_0, m, L$ als eerder. Stel dat $b_1, b_2, \ldots, b_{m+1}$ een gereduceerde basis voor L is en dat*

(5.8) $p^{kl} > 2^{mn/2} \binom{2m}{m}^{n/2} |f|^{m+n}.$

Dan geldt dat de graad van $h_0 \leq m$ d.e.s.d.a.

(5.9) $|b_1| < (p^{kl}/|f|^m)^{1/n}.$

Bewijs.
'$\Leftarrow$' volgt uit de vorige stelling, b_1 heeft graad $\leq m$ en h_0 is een deler van b_1.
Het bewijs van '$\Rightarrow$' gaat als volgt; $graad(h_0) \leq m$. Dan geldt dat $h_0 \in L$, h_0 is immers (modulo p^k genomen) een veelvoud van $(h \ mod \ p^k)$. Stelling 2 toegepast op $x = h_0$ geeft $|b_1| \leq 2^{m/2}|h_0|$. Wegens de Landau-Mignotti grens geldt $|h_0| \leq \binom{2m}{m}^{\frac{1}{2}} |f|$, dus $|b_1| \leq 2^{m/2} \binom{2m}{m}^{\frac{1}{2}} |f|$.
(5.8) omgeschreven zegt $2^{m/2} \binom{2m}{m}^{\frac{1}{2}} |f| < (p^{kl}/|f|^m)^{1/n}$.
De laatste twee ongelijkheden combineren geeft (5.9).

Opmerking.
Stel $m = graad(h_0)$ en (5.9) geldt. Dan is b_1 een veelvoud van h_0 van dezelfde graad en bovendien primitief omdat b_1 een vector uit een basis is. Conclusie, $b_1 = h_0$.

6. Het ontbindingsalgoritme

We zullen hieronder in een aantal stappen een algoritme beschrijven dat een primitief polynoom $f \in Z[X]$ van graad n factoriseert.
Als we de schematische voorstelling van f in het oog houden die we in het begin van het vorige hoofdstuk gegeven hebben, volgt het ontbindingsalgoritme op een natuurlijke wijze uit de vorige stellingen.

Begin algoritme.
1. Bereken de resultante $R(f, f')$ [3, Sect. 4.6.1]. Als $R(f, f') = 0$ dan hebben f en f' een gemeenschappelijke factor, dit geval behandelen we later. Stel eerst $R(f, f') \neq 0$.
2. Bepaal het kleinste priemgetal p dat geen deler is van $R(f, f')$. Dat betekent namelijk dat $(f \ mod \ p)$ ook geen meervoudige factoren heeft, zodat (5.4) geldt voor alle factoren h van $(f \ mod \ p)$, en bovendien heeft $(f \ mod \ p)$ dan ook graad n. Vervolgens factoriseren we $(f \ mod \ p)$ met behulp van Berlekamp's algoritme [3, Sect. 4.6.2].
3. Neem aan dat we een decompositie $f = f_1 \cdot f_2$ kennen, zodanig dat de factorisatie van f_1 in $Z[X]$ en die van $(f_2 \ mod \ p)$ in $F_p[X]$ bekend zijn. Als initialisatie nemen we $f_1 = 1$ en $f_2 = f$. Vanuit deze situatie werken we als volgt:
Kies een irreducibele factor $(h \ mod \ p)$ van $(f_2 \ mod \ p)$ in $F_p[X]$. We mogen aannemen dat de coëfficiënten van h gereduceerd modulo p zijn en dat h monisch is (F_p is een lichaam). Met behulp van deelalgoritme 1, hieronder gegeven, vinden we een irreducibele factor h_0 van f_2 in $Z[X]$. We vervangen nu f_1 door $f_1 \cdot h_0$ en f_2 door f_2/h_0 en we verwijderen uit de ontbinding van $(f_2 \ mod \ p)$ alle delers van $(h_0 \ mod \ p)$.
Deze procedure zetten we voort totdat $f_2 = \pm 1$, $f_1 = \pm f$ is dan volledig gefactoriseerd. Dit beëindigt het algoritme in het geval dat $R(f, f') \neq 0$.

Deelalgoritme 1.
Gegeven zijn f_2, $n = graad(f_2)$, een priemgetal p en een polynoom $h \in Z[X]$, zodanig dat (5.1) t/m (5.4) gelden met $k = 1$ en $f = f_2$. Stel dat de coëfficiënten van h gereduceerd modulo p zijn, het volgende algoritme bepaalt de irreducibele factor h_0 van f waarvoor $(h \ mod \ p)$ een deler is van $(h_0 \ mod \ p)$.
1. Neem $l = graad(h)$. Als $l = n$ dan is $h_0 = f$ en stopt het algoritme. Stel dus dat $l < n$. Als eerste berekenen we de kleinste positieve k waarvoor (5.8) klopt met m vervangen door $n - 1$, oftewel

$$p^{kl} > 2^{(n-1)n/2} \binom{2(n-1)}{n-1}^{n/2} |f|^{2n-1}.$$

2. Pas h aan met behulp van *Hensel's Lemma* [4], zodanig dat (5.2) geldt voor de zojuist berekende waarde van k. Voer daarna Stap 3 uit voor achtereenvolgens $m = l, l+1, \ldots, n-1$, maar stop zodra h_0 gevonden is, als dit niet gebeurt dan moet gelden $graad(h_0) > n - 1$, dus $h_0 = f_2$.

3. Definieer L als het rooster met basis

$$\{p^k X^i : 0 \leq i < l\} \cup \{hX^j : 0 \leq j \leq m - l\}.$$

We passen het LLL-reductie algoritme toe op deze basis en vinden een gereduceerde basis $b_1, b_2, \ldots, b_{m+1}$ voor L. Als $|b_1| \geq (p^{kl}/|f|^m)^{1/n}$ dan is $graad(h_0) > m$ en zoeken we het hogerop. Als $|b_1| < (p^{kl}/|f|^m)^{1/n}$ hebben we $graad(h_0) \leq m$ en vanwege de opmerking na stelling 5 geldt $h_0 = b_1$, in de vorige stap bleek immers $graad(h_0) > m-1$.

Als $R(f, f') = 0$, neem dan $g = ggd(f, f')$ en schrijf $f_0 = f/g$. f_0 heeft dan geen meervoudige factoren meer, dus $R(f_0, f_0') \neq 0$ en we kunnen f_0 factoriseren zoals net is beschreven. Aangezien elke factor van g ook een factor van f_0 moet zijn, kunnen we met een paar testen gemakkelijk de factorisatie van $f = f_0 \cdot g$ bepalen.

7. LLL ALS KETTINGBREUK

We bekijken in dit hoofdstuk een speciaal geval van de diofantische benadering uit hoofdstuk 4.
Gegeven de rationale getallen α en ε, $0 < \varepsilon < 1$, willen we gehele getallen p en q vinden, zodanig dat

$$|p - q\alpha| \leq \varepsilon \qquad en \qquad 1 \leq q \leq \frac{\sqrt{2}}{\varepsilon}.$$

Hiertoe beschouwen we het rooster $L \subset R^2$ opgespannen door de vectoren

$$b_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad en \quad b_2 = \begin{pmatrix} \alpha \\ \frac{\varepsilon}{\sqrt{2}} \end{pmatrix}$$

en de eerste vector van een gereduceerde basis voor L geeft de oplossing.
We zullen laten zien dat het LLL-algoritme, geïnitialiseerd met deze b_1 en b_2, eigenlijk niets anders is dan het bekende kettingbreukalgoritme. Voor een introductie in de theorie van de kettingbreuken verwijzen we naar [5, Sect. 4] en [6].

Het is duidelijk dat de keus van ε het aantal stappen van het algoritme bepaalt. ε heel klein kiezen geeft een goede benadering van α door $\frac{p}{q}$. Om te laten zien wat er in de loop van het algoritme gebeurt, is het het eenvoudigst om alle invloed van ε te verwaarlozen, je ziet dan duidelijk de contouren van het kettingbreukalgoritme verschijnen.
Op deze manier werken we als het ware met de twee '1-dimensionale vectoren' $b_1 = 1, b_2 = \alpha$. De Gramm-Schmidt orthogonalisatie geeft natuurlijk altijd $b_1^* = b_1$, $\mu_{21} = \frac{b_1 b_2}{b_1^2} = \frac{b_2}{b_1}$ en $b_2^* = 0$. Merk op dat in de beginsituatie $\mu_{21} = \alpha$ geldt.
Het LLL-algoritme is vanuit deze beginsituatie niets meer dan steeds de herhaling van twee stappen.

1. Schrijf $\mu_{21} = \frac{b_1}{b_2} = q_0 + r_0$, met $q_0 \in Z$ en $-\frac{1}{2} < r_0 \leq \frac{1}{2}$.
Vervolgens passen we b_2 aan.

$$b_2 := b_2 - q_0 b_1$$

Maar $\mu_{21} = \frac{b_2}{b_1} = q_0 + r_0$ omgeschreven zegt $b_2 = q_0 b_1 + r_0 b_1$, dus bovenstaande mutatie is gelijk aan:

$$b_2 := r_0 b_1$$

2. Omdat $b_2^* = 0$ geldt $|b_2^* + \mu_{21} b_1^*|^2 \leq \frac{1}{4}|b_1|^2$ en belanden we in het algoritme altijd in situatie 1, dus komt nu de verwisseling:

$$\begin{pmatrix} b_1 \\ b_2 \end{pmatrix} := \begin{pmatrix} b_2 \\ b_1 \end{pmatrix}$$

$b_2 = r_0 b_1$ hier invullen geeft :

$$\begin{pmatrix} b_1 \\ b_2 \end{pmatrix} := \begin{pmatrix} r_0 b_1 \\ b_1 \end{pmatrix}$$

Nu gaan we weer terug naar stap 1 met deze nieuwe b_1 en b_2.
In de volgende stap zie je:

$$\mu_{21} = \frac{b_2}{b_1} = \frac{1}{r_0} = q_1 + r_1$$

en we zien dat het proces zich herhaalt, met q_1 en r_1 in de rol van resp. q_0 en r_0.
Het algoritme verder uitvoeren geeft de volgende rij:

$$\alpha = q_0 + r_0$$

$$\frac{1}{r_0} = q_1 + r_1$$

$$\frac{1}{r_1} = q_2 + r_2$$

etc.

en dit uitwerken geeft

$$\alpha = q_0 + \cfrac{1}{q_1 + \cfrac{1}{q_2 + \cfrac{1}{\ddots}}}$$

Afhankelijk van de keuze van ε zullen we een n vinden zodat

$$\alpha \approx q_0 + \cfrac{1}{q_1 + \cfrac{1}{q_2 + \cfrac{1}{\ddots + \frac{1}{q_n}}}}$$

Deze formule uitschrijven geeft p en q.

Het verschil dat opvalt met het kettingbreukalgoritme zoals het beschreven staat in [5] en [6], is dat hier de getallen q_i zo gekozen worden dat $-\frac{1}{2} < r_i \leq \frac{1}{2}$, terwijl normaal $0 \leq r_i < 1$, of met andere woorden q_i de entier van $\frac{1}{r_{i-1}}$ zou zijn. De relatie tussen deze twee methodes zullen we in het volgende hoofdstuk bekijken.

8. Meer over kettingbreuken

Stel we willen een getal $\alpha \in Q$ schrijven als een kettingbreuk. We zullen twee methodes met elkaar vergelijken. Bekijk de rijen

$$\begin{array}{ll} \alpha = q_0 + r_0 & \alpha = s_0 + t_0 \\ \frac{1}{r_0} = q_1 + r_1 & \frac{1}{t_0} = s_1 + t_1 \\ \vdots & \vdots \\ \frac{1}{r_{n-1}} = q_n & \frac{1}{t_{m-1}} = s_m \end{array}$$

waar we in de eerste rij $q_0 \in Z, q_i \in Z_{>0}$ als $i \geq 1$ en $0 < r_i < 1$ en in de tweede $s_i \in Z$ en $-\frac{1}{2} < t_i \leq \frac{1}{2}$ kiezen. Beide rijen uitwerken geeft de twee kettingbreuken

$$\alpha = q_0 + \cfrac{1}{q_1 + \cfrac{1}{\ddots + \frac{1}{q_n}}} = s_0 + \cfrac{1}{s_1 + \cfrac{1}{\ddots + \frac{1}{s_m}}}.$$

De eerste uitdrukking noemen we de kettingbreuk volgens de standaard- en de tweede volgens de LLL-methode. We schrijven een kettingbreuk

$$q_0 + \cfrac{1}{q_1 + \cfrac{1}{\ddots + \frac{1}{q_n}}}$$

ook wel als $[q_0, q_1, \ldots, q_n]$. We krijgen bijvoorbeeld $\frac{345}{98} = [3, 1, 1, 11, 1, 3]$ als we met de standaard- en $\frac{345}{98} = [4, -2, -12, 4]$ als we met de LLL-methode werken. Wat intrigeert is dat de tweede uitdrukking korter is, we zullen hieronder de precieze samenhang tussen beide methodes onderzoeken.

Stel we hebben met de standaard methode $\alpha = [q_0, q_1, \ldots, q_n]$ en met de LLL-methode $\alpha = [s_0, s_1, \ldots, s_m]$ gevonden.

We definiëren i als de kleinste index waarvoor $s_i \neq q_i$. Dan geldt $r_{i-1} = t_{i-1}$ (als $i = 0$ is dit $\frac{1}{\alpha}$) en

$$\frac{1}{r_{i-1}} = q_i + r_i \qquad \frac{1}{2} < r_i < 1.$$

Omdat $\frac{1}{2} < r_i < 1$ zijn de volgende twee stappen in de standaard-methode :

$$\frac{1}{r_i} = 1 + r_{i+1} \tag{8.1}$$

$$\frac{1}{r_{i+1}} = q_{i+2} + r_{i+2} \tag{8.2}$$

De LLL-methode vindt

$$\frac{1}{t_{i-1}} = s_i + t_i$$

maar we weten dat $t_{i-1} = r_{i-1}$, dus

$$s_i + t_i = \frac{1}{t_{i-1}} = \frac{1}{r_{i-1}} = (q_i + 1) + (r_i - 1)$$

en omdat $\frac{1}{2} < r_i < 1$ en $-\frac{1}{2} < t_i \leq \frac{1}{2}$ moet gelden $s_i = q_i + 1$ en $t_i = r_i - 1$.
De volgende stap is

$$\frac{1}{t_i} = s_{i+1} + t_{i+1}$$

We weten dat $t_i = r_i - 1$. Verder vullen we (8.1) en (8.2) in en verkrijgen zo

$$s_{i+1} + t_{i+1} = \frac{1}{t_i} = \frac{1}{r_i - 1} = \frac{1}{\frac{1}{1+r_{i+1}} - 1} = -(\frac{1}{r_{i+1}} + 1) = -(q_{i+2} + 1) - r_{i+2}.$$

Als het linkerlid opschrijven als een kettingbreuk via de LLL-methode en het rechterlid als een standaardkettingbreuk zien we

$$[s_{i+1}, s_{i+2}, \ldots, s_m] = [-(q_{i+2} + 1), -q_{i+3}, \ldots, -q_n].$$

Nu zijn we weer in een identieke situatie als in het begin beland.

Vanwege het rechtstreekse verband tussen de beide methodes kunnen we betrekkelijk eenvoudig een kettingbreuk die met de ene methode berekend is, omschrijven naar het resultaat van de andere.
We zullen hieronder het algoritme formuleren dat een kettingbreuk $[s_0, s_1, \ldots, s_n]$, berekend met de LLL-methode, omschrijft naar een standaard kettingbreuk $[q_0, q_1, \ldots, q_n]$.

1. $i := 1$, $j := 1$, $q_0 := s_0$, $teken := 1$
2. Als $teken \cdot s_i < 0$, dan $q_{j-1} := q_{j-1} - 1$, $q_j := 1$, $q_{j+1} := |s_i| - 1$, $teken := -teken$, $j := j + 2$.
   Als $teken \cdot s_i > 0$, dan $q_j := |s_i|, j := j + 1$.
3. Als $i < m$, dan $i := i + 1$ en ga naar 2.
   Als $i = m$ zijn we klaar en geldt $[s_0, s_1, \ldots, s_m] = [q_0, q_1, \ldots, q_n]$.

Nu zien we ook waarom de LLL-methode een kortere kettingbreuk geeft, de stappen waar de standaardmethode een 1 geeft verdwijnen. We zullen met behulp van deze ideeën in het volgende hoofdstuk in zo weinig mogelijk stappen de kettingbreuk van een algebraïsch getal proberen te construeren.

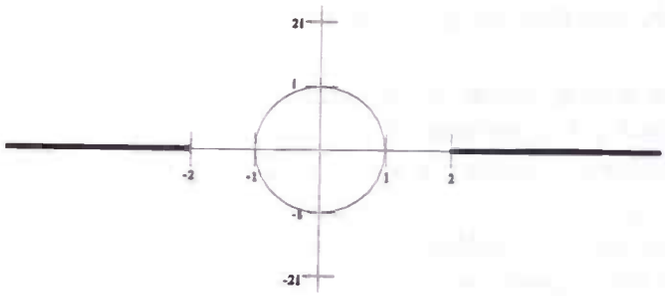
9. De kettingbreuk van een algebraïsch getal

Stel we willen de kettingbreuk van een algebraïsch getal $u \in R$ construeren, een eerste gedachte is om u numeriek te benaderen door $u' \in Q$ en met u' het algoritme uit het vorige hoofdstuk uit te voeren. Maar het doel is eigenlijk om via de kettingbreuk van u een goede benadering $u' \in Q$ te vinden.
Hoe vinden we de 'echte' kettingbreuk ? We veronderstellen dat u wordt gegeven als nulpunt van een polynoom $f(X) = a_r X^r + a_{r-1} X^{r-1} + \ldots + a_0 \in Z[X]$. Verder kunnen we veronderstellen dat u een enkelvoudig nulpunt van f is, als dat niet het geval is, zouden we dat eenvoudig kunnen bereiken door in plaats van f te werken met $ggd(f, f')$. We kunnen u dus opsporen doordat f van teken verandert in een interval dat u bevat, maar hoe weten we, als we een tekenwisseling in een interval vinden, dat u daar in ligt en niet een ander nulpunt van f ? Daarvoor zullen we eerst een idee moeten hebben over de locatie van de andere nulpunten. Een antwoord op bovenstaande vragen geeft de volgende definitie.

We noemen een polynoom $f(X) = a_r X^r + a_{r-1} X^{r-1} + \ldots + a_0$ een **gereduceerd polynoom** als:

1. $a_0, \ldots, a_r \in Z$.
2. f is irreducibel.
3. f heeft een reëel nulpunt u met $|u| > 2$ en alle andere nulpunten γ van f voldoen aan $|\gamma| < 1$.

Dit geeft in het complexe vlak de volgende tekening, het nulpunt u ligt op de reële rechte in $(-\infty, -2) \cup (2, \infty)$ en de andere $r-1$ nulpunten liggen binnen de eenheidscircel S_1.



Stelling 6 *Stel f_i is een gereduceerd polynoom van graad r, u_i is het reële nulpunt van f_i met $|u_i| > 2$ en q_i is de afronding van u_i naar het dichtstbijzijnde gehele getal. Dan is het polynoom f_{i+1}, gedefinieerd door*

$$f_{i+1} = X^r f_i(q_i + \frac{1}{X}) \tag{9.1}$$

een gereduceerd polynoom met reëel nulpunt $u_{i+1} = \frac{1}{u_i - q_i}$ $(|u_{i+1}| > 2)$.

Bewijs.
We controleren de eisen voor een gereduceerd polynoom.

- *f_{i+1} is irreducibel.*
  Omdat f_i irreducibel is in $Z[X]$ is het polynoom $f_i(q_i+Y)$ irreducibel in $Z[Y]$. Als we $Y=\frac{1}{X}$ nemen volgt dat f_{i+1} irreducibel is in $Z[X]$, omdat 0 geen nulpunt van f_i kan zijn.

- *f_{i+1} heeft een reëel nulpunt u met $|u|>2$.*
  Het getal $u_{i+1}=\frac{1}{u_i-q_i}$ is een nulpunt van f_{i+1}, $|u_i-q_i|<\frac{1}{2}$, dus $|u_{i+1}|>2$.

- *Voor elk ander nulpunt γ van f_{i+1} geldt $|\gamma|<1$.*
  Zij $\gamma\neq 0$ een nulpunt van f_{i+1}, dan geldt :

$$f_{i+1}(\gamma)=0 \;\Leftrightarrow\; \gamma^r f_i(q_i+\frac{1}{\gamma})=0 \;\Leftrightarrow\; \gamma=\frac{1}{\beta-q_i}$$

  waar β een nulpunt van f_i is. Dus de nulpunten van f_{i+1} zijn precies de nulpunten van f_i na de transformatie $z\mapsto\frac{1}{z-q_i}$. We zagen al dat het nulpunt u_i van f_i door deze transformatie afgebeeld wordt op u_{i+1}. Elk ander nulpunt γ van f_i ligt binnen S_1, dus $\gamma-q_i$ ligt buiten S_1 ($q_i>2$) en dat betekent dat $\frac{1}{\gamma-q_i}$ weer in S_1 ligt.

Dit bewijst stelling 6.

De volgende stelling helpt ons het gebied waarin we naar u_i moeten zoeken te verkleinen.

Stelling 7 *Zij $f=a_rx^r+a_{r-1}x^{r-1}+\ldots+a_0$ een gereduceerd polynoom met een nulpunt u waarvoor geldt $|u|>2$ en schrijf $R(u)$ voor het gehele getal het dichtst bij u, dan geldt*

$$R(\frac{-a_{r-1}}{a_r})-r+1\leq R(u)\leq R(\frac{-a_{r-1}}{a_r})+r-1. \tag{9.2}$$

Bewijs
Er geldt

$$a_rx^r+a_{r-1}x^{r-1}+\ldots+a_0 \;=\; a_r(x-u)\prod_{i=1}^{r-1}(x-\gamma_i),$$

waar $\gamma_1,\ldots\gamma_{r-1}$ de andere $r-1$ nulpunten van f zijn. Als we nu in beide uitdrukkingen de coëfficiënt van x^{r-1} vergelijken, krijgen we

$$\frac{-a_{r-1}}{a_r}=u+\sum_{i=1}^{r-1}\gamma_i.$$

Aangezien $\frac{-a_{r-1}}{a_r}$ en u beiden reëel zijn, moet de som dat ook zijn en geldt

$$\sum_{i=1}^{r-1}\gamma_i \;=\; Re(\sum_{i=1}^{r-1}\gamma_i) \;=\; \sum_{i=1}^{r-1}Re(\gamma_i).$$

Omdat $|\gamma_i| < 1$ hebben we nu

$$-(r-1) \;<\; \frac{-a_{r-1}}{a_r} - u \;<\; r-1$$

en hier afronden naar het dichtstbijzijnde gehele getal geeft (9.2). Dit bewijst stelling 7.

Zij $u_0 \in (2, \infty)$ gegeven als nulpunt van een gereduceerd polynoom f_0. Het volgende algoritme bepaalt de eerste $n+1$ getallen uit de kettingbreukontwikkeling $u = [q_0, q_1, q_2 \ldots]$.

1. $i = 0$.
2. $k = \mathrm{Max}(2, R(\frac{-a_{r-1}}{a_r}) + r - 1)$.
3. Bereken het teken van $f_i(j)$ $j = k, k+1, k+2, \ldots$ totdat je een tekenwisseling vindt, je weet dan dat $m < u_i < m+1$ voor een zekere $m \in Z$.
   Bekijk vervolgens of $m < u_i < m + \frac{1}{2}$, of dat $m + \frac{1}{2} < u_i < m+1$. In het eerste geval geldt $q_i = m$ en $u_{i+1} \in (2, \infty)$. In het tweede geval geldt $q_i = m+1$ en $u_{i+1} \in (-\infty, -2)$. Ga hierna naar stap 6.
4. $k = \mathrm{Min}(-2, R(\frac{-a_{r-1}}{a_r}) - r + 1)$.
5. Bereken het teken van $f_i(j)$ $j = k, k-1, k-2, \ldots$ totdat je een tekenwisseling vindt, je weet dan dat $m - 1 < u_i <$ voor een zekere $m \in Z$.
   Bekijk vervolgens of $m - 1 < u_i < m - \frac{1}{2}$, of dat $m - \frac{1}{2} < u_i < m$. In het eerste geval geldt $q_i = m - 1$ en $u_{i+1} \in (2, \infty)$. In het tweede geval geldt $q_i = m$ en $u_{i+1} \in (-\infty, -2)$.
6. Stop als $i = n$.
   Als $i < n$, stel dan $i := i + 1$. Bereken de nieuwe f_i met behulp van (9.1), ga vervolgens naar 2 als $u_i \in (2, \infty)$ en naar 4 als $u_i \in (-\infty, -2)$.

We kunnen natuurlijk ook beginnen met een $u_0 \in (-\infty, -2)$, we stellen dan $i = 0$ en beginnen in 4. Het belangrijkste is dat je steeds weet waar je moet zoeken.

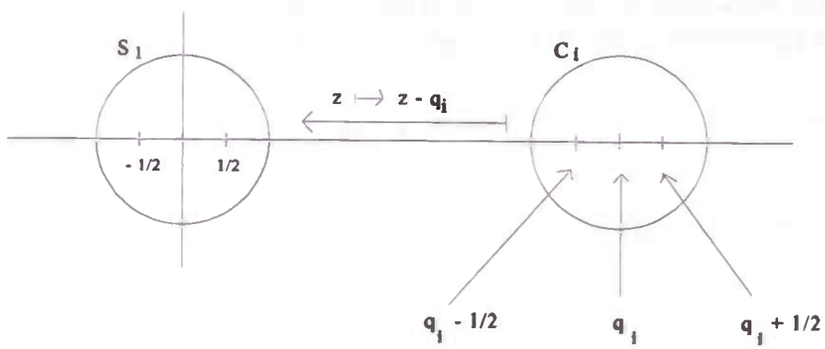
Maar ondertussen kunnen we nog steeds alleen maar werken met gereduceerde polynomen, en die zijn wel heel specifiek gedefinieerd. De volgende stelling redt ons. We gaan ervan uit dat we de kettingbreuk van u_0 al kennen (in de praktijk betekent dit de kettingbreuk van een goede benadering voor u_0 berekenen.) Dus $u_0 = [q_0, q_1, q_2 \ldots]$ en we construeren de rij $f_0, f_1, f_2 \ldots$ als eerder door

$$f_{i+1} = X^r f_i(q_i + \frac{1}{X})$$

waarbij f_0 een irreducibel polynoom met $f_0(u_0) = 0$ en $f_0'(u_0) \neq 0$ is. Merk op dat $u_i = [q_i, q_{i+1}, q_{i+2}, \ldots]$ een nulpunt van f_i is.

Stelling 8 *Zij $f_0, f_1, f_2 \ldots$ gedefinieerd als boven, dan bestaat er een $n \in Z$ zodanig dat voor alle $i \geq n$ het polynoom f_i een gereduceerd polynoom is.*

Bewijs.
Als f_n een gereduceerd polynoom is, dan zijn alle f_i met $i > n$ dat wegens stelling 6 ook. We moeten dus bewijzen dat er zo'n n is.
Als we alle eigenschappen van een gereduceerd polynoom langsgaan, zien we dat $f_1, f_2, \ldots$ allemaal aan alle eigenschappen voldoen (als $i > 0$ is $|u_i| > 2$), behalve aan de laatste eis, dat alle nulpunten van f_i, behalve u_i, in S_1 liggen.
We hebben al gezien dat als γ_i een nulpunt van f_i is, dat $\gamma_{i+1} = \frac{1}{\gamma_i - q_i}$ dan een nulpunt van f_{i+1} is en we weten ook dat als γ_i in S_1 ligt, dat γ_{i+1} dan ook in S_1 ligt. We zullen nu laten zien dat voor elk nulpunt $\gamma_0 \neq u_0$ van f_0 geldt, dat als we de rij $\gamma_0, \gamma_1, \gamma_2, \ldots$ definiëren door $\gamma_{i+1} = \frac{1}{\gamma_i - q_i}$, dat er dan een $n \in Z$ is waarvoor uit $i > n$ volgt dat γ_i in S_1 ligt. Bekijk de volgende tekening.



Als γ_i buiten C_i, de circel met straal 1 om q_i, ligt, dan ligt $\gamma_i - q_i$ buiten S_1 en $\gamma_{i+1} = \frac{1}{\gamma_i - q_i}$ in S_1. Stel nu dat γ_i in C_i ligt. Definieer $\varepsilon_i = |Re(\gamma_i) - u_i|$. Dan geldt $0 < \varepsilon_i < 1\frac{1}{2}$. Hieruit volgt

$$\varepsilon_{i+1} = |Re(\gamma_{i+1}) - u_{i+1}| = |\frac{1}{Re(\gamma_i) - q_i} - \frac{1}{u_i - q_i}| = \frac{|u_i - Re(\gamma_i)|}{|(Re(\gamma_i) - q_i)(u_i - q_i)|}$$

en aangezien $|Re(\gamma_i) - q_i| < 1$ en $|u_i - q_i| < \frac{1}{2}$ volgt nu dat $\varepsilon_{i+1} > 2\varepsilon_i$.
Dus er moet een n zijn waarvoor $\varepsilon_n > 1\frac{1}{2}$ en dan ligt γ_{n+1} in S_1. Dit bewijst stelling 8.

Het laatste gedeelte kunnen we ook als volgt bewijzen :
We definiëren de getallen r_i, s_i door

$$\begin{pmatrix} q_0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} q_1 & 1 \\ 1 & 0 \end{pmatrix} \cdots \begin{pmatrix} q_i & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} r_i & r_{i-1} \\ s_i & s_{i-1} \end{pmatrix}$$

Dan geldt $\frac{r_i}{s_i} = [q_0, q_1, \ldots, q_i]$. Verder geldt

$$(9.3) \quad |u_0 - \frac{r_i}{s_i}| < \frac{1}{|q_{i+1}| s_i^2}.$$

Met inductie zie je dat

$$f_{i+1} = (s_i\, x + s_{i-1})^r f_0(\frac{r_i\, x + r_{i-1}}{s_i\, x + s_{i-1}}).$$

Zij $u_{i+1}, \gamma_{i+1}^{(1)}, \ldots, \gamma_{i+1}^{(r-1)}$ de nulpunten van f_{i+1}, dan geldt

$$\frac{r_i\ u_{i+1} + r_{i-1}}{s_i\ u_{i+1} + s_{i-1}} = u_0$$

en

$$\frac{r_i\ \gamma_{i+1}^{(j)} + r_{i-1}}{s_i\ \gamma_{i+1}^{(j)} + s_{i-1}} = \gamma_0^{(j)} \quad \textit{voor} \quad j = 1, \ldots, r-1$$

Als we dit anders opschrijven krijgen we

$$\gamma_{i+1}^{(j)} = \frac{s_{i-1}\ \gamma_0^{(j)} - r_{i-1}}{-s_i\ \gamma_0^{(j)} + r_i}$$

en hiermee kunnen we laten zien dat $|\gamma_i^{(j)}| < 1$ als i groot genoeg is, dit volgt uit (9.3) en de opmerking dat $|q_i| > 2$ voor $i > 0$, als volgt :

$$|s_{i-1}\ \gamma_0^{(j)} - r_{i-1}| = |s_{i-1}\ (\gamma_0^{(j)} - u_0) + s_{i-1}\ (u_0 - \frac{r_{i-1}}{s_{i-1}})| \leq |s_{n-1}|\ |\gamma_0^{(j)} - u_0| + \frac{1}{2|s_{i-1}|}$$

$$|-s_i\ \gamma_0^{(j)} + r_i| = |s_i\ (\gamma_0^{(j)} - u_0) + s_i\ (u_0 - \frac{r_i}{s_i})| \geq |s_i|\ |\gamma_0^{(j)} - u_0| - \frac{1}{2|s_i|}$$

Dus

$$|-s_i\ \gamma_0^{(j)} + r_i| \ - \ |s_{i-1}\ \gamma_0^{(j)} - r_{i-1}| \geq (|s_i| - |s_{i-1}|)\ |\gamma_0^{(j)} - u_0| - \frac{1}{2|s_i|} - \frac{1}{2|s_{i-1}|}.$$

Dit is groter dan 0 als i groot genoeg is, omdat $|s_i| - |s_{i-1}| \geq 1$ en $\lim_{i\to\infty} \frac{1}{|s_i|} = 0$ en in dat geval geldt $|\gamma_{i+1}^{(j)}| < 1$. Hetzelfde argument geldt voor alle $j = 1, \ldots, r-1$. Dit bewijst stelling 8.

Dit beëindigt de beschrijving van het algoritme. We maken de rij $f_0, f_1, f_2, \ldots$ met behulp van (9.1) en de kettingbreuk van een benadering van u_0, totdat we een gereduceerd polynoom f_i tegenkomen en gaan dan verder zoals eerder beschreven. Het voorgaande berekent de kettingbreuk via de LLL-methode, als we liever de klassieke kettingbreuk willen bekijken kunnen we simpelweg achteraf het omschrijfalgoritme uit het vorige hoofdstuk gebruiken.

10. De mathematica-implementatie van het LLL-ontbindingsalgoritme

```
BeginPackage["LLLfactorisation`"]

(* This package uses the lattice-reduction algorithm programmed by
   Wilberd van der Kallen. *)
<< LLLexp

(* And it uses the package "extended Gcd" which is an additional package
   by mathematica. *)
<< ext.m

LLLfactor::usage:=
"LLLfactor[f,x] factors a given polynomial f[x] with integer
\ncoefficients into irreducible parts, provided that f is primitive,
\nusing the factorisation method discussed in a paper written by
\nLenstra, Lenstra and Lovasz.(Mathematische Annalen 261, 515-534 (1982))"

Unprotect[LLLfactor] ;

Begin["LLLfactorisation`Private`"]

DeltaPk[i_,j_,pk_]:=If[i==j,pk,0];

Norm[polylist_]:=N[Sqrt[Apply[Plus,polylist^2]]] ;

Lattice[hlist_,pk_,m_]:=
 Module[{ l=Length[hlist]-1,
          i,j,g=hlist,Lat},
          lat=Table[DeltaPk[i,j,pk],{i,l},{j,m+1}] ;
          Do[AppendTo[g,0],{m-l}];
          For[i=0,i<=m-l,i++,
              AppendTo[lat,RotateRight[g,i]]];
          Return[lat]] ;

FindHO[flist_,hlist_,pk_,m_]:=
 Module[{n=Length[flist]-1,
 l=Length[hlist]-1,
         b=ExtendedLatticeReduce[Lattice[hlist,pk,m]][[1]],
        },
         If[Norm[b[[1]]] < N[(pk^l/(Norm[flist]^m))^(1/n)] ,
            Return[b[[1]]],Return[{}]]
        ] ;


PolynomialQuotientMod[f_, g_, x_, p_]:=
Block[{fp = PolynomialMod[f, p], gp = PolynomialMod[g, p]},
       {fp, gp} =
       PolynomialMod[
        PowerMod[Coefficient[gp, x, Exponent[gp, x]], -1, p] {fp, gp},
        p];
       PolynomialMod[PolynomialQuotient[fp, gp, x], p]
```

```
      ] ;

 ExtPolGcd[a_,b_,x_,p_]:=
   PolynomialMod[
     Expand[
       PolynomialExtendedGCD[a,b,Modulus->p][[2]] ], p ] ;

 Poly[hlist_,x_]:=
   Module[{pol=First[hlist],graad=Length[hlist]-1},
           For[i=1,i<=graad,i++,pol=pol+hlist[[i+1]] x^i];
           Return[pol]] ;

 PolynomialModPlus[f_,x_,p_]:=
   Module[{g=CoefficientList[f,x],i},
           If[g=={},Return[0] ] ;
           For[ i=1,i<=Length[g],i++,
                g[[i]]=Mod[g[[i]],p] ;
                If[g[[i]]>p/2,g[[i]]=g[[i]]-p ]
              ] ;
           Return[Poly[g,x]] ] ;


 Hensel[f_,h_,p_,k_,x_]:=
   Module[{a=h,b=PolynomialQuotientMod[f,h,x,p],
           i,d,s,t,u,v},
          {u,v}=ExtPolGcd[a,b,x,p] ;
          For[i=1,i<k,i++,
              d=Simplify[(f - a b)/(p^i)] ;
              If[d==0,Return[a] ] ;
              s=PolynomialModPlus[
                 PolynomialRemainder[Expand[v d],a,x],x,p] ;
              t=PolynomialModPlus[
                 PolynomialRemainder[Expand[u d],b,x],x,p] ;
              a=Expand[a+p^i s] ;
              b=Expand[b+p^i t] ;
            ] ;
          Return[a] ] ;

FindIrredFactor[f_,h_,p_,x_]:=
  Module[{ n=Exponent[f,x],
           l=Exponent[h,x],
           flist=CoefficientList[f,x],
           hlist,m,k,H0list },
         k=Floor[N[Log[2^((n-1)n/2) Binomial[2(n-1),n-1]^(n/2)
                   * Norm[flist]^(2n-1)]/Log[p^l]]]+1 ;
         hlist=CoefficientList[Hensel[f,h,p,k,x],x];
         For[m=l,m<n,m++,
             H0list=FindH0[flist,hlist,p^k,m] ;
             If[H0list!={},Return[Poly[H0list,x]]]] ;
         Return[f] ] ;

SmallestPrimeNotDividing[n_] :=
  Module[{i=1},
```

```
        While[Mod[n,Prime[i]]==0,i++] ;
        Return[Prime[i]] ] ;

ModFactors[f_,p_]:=
  Transpose[FactorList[f,Modulus->p]][[1]] ;

LLLfactor[f_,x_]:=
  Module[{ fr=f,p,i,k,
           r=Resultant[f,D[f,x],x],
           Berlekamp,
           IrreducibeleFactors={},
           h,H0,g=1
         },
         If[r==0,g=PolynomialGCD[f,D[f,x]] ;
            fr=PolynomialQuotient[f,g,x] ;
            r=Resultant[fr,D[fr,x],x] ] ;
         p=SmallestPrimeNotDividing[r];
         Berlekamp=ModFactors[fr,p];
         While[Not[Berlekamp==={}],
           h=Berlekamp[[1]];
           H0=FindIrredFactor[fr,h,p,x];
           Berlekamp=Complement[Berlekamp,ModFactors[H0,p]];
           AppendTo[IrreducibeleFactors,H0] ] ;
         If[Not[TrueQ[g==1]],
            i=1 ;
            k=1 ;
            While[i<=Length[IrreducibeleFactors],
                  If[PolynomialRemainder
                      [g,IrreducibeleFactors[[i]],x]==0,
                     g=PolynomialQuotient[g,IrreducibeleFactors[[i]],x] ;
                     k=k+1 ,
                     IrreducibeleFactors[[i]] =
                      IrreducibeleFactors[[i]]^k ;
                     i=i+1 ;
                     k=1 ]
                  ]
            ] ;
          Return[IrreducibeleFactors]
          ] ;

End[]
Protect[LLL_Factor]
EndPackage[]
```

11. De mathematica-implementatie van het kettingbreukalgoritme voor algebraïsche getallen

```
BeginPackage["AlgebraicContinuedFraction`"]

ContinuedFraction::usage:=
"ContinuedFraction[u,f,x,n] calculates the first n numbers in
\nthe continued fraction of an algebraic number u with minimum
\npolynomial f(x), using the LLL method, which means that negative
\nnumbers are allowed in the continued fraction. For example,
\nthe continued fraction of the number 5^(1/3) up to a 1000 numbers
\nis given by ContinuedFraction[5^(1/3),x^3-5,x,1000]. "

LLLToStandard::usage:=
"LLLToStandard[cf] transforms the continued fraction cf, calculated
\nusing the LLL method, which allows negative numbers, to the method
\nin which only positive numbers are allowed in a continued fraction."

Unprotect[ContinuedFraction,LLLToStandard]

Begin["AlgebraicContinuedFraction`Private`"] ;

ReducedQ[f_,x_]:=
 Module[{roots=x/.NSolve[f==0,x],u},
        u=Select[roots,If[Im[#]==0,#>1,False] &];
        roots=Complement[roots,u] ;
        If[Length[u]==1,
           Return[
            Apply[And,
             Map[ Abs[#]<1 &, roots] ] ],
           Return[False]]
       ] ;

ComputeCPos[f_,x_] :=
 Module[{r=Exponent[f,x],a0,a1,c},
         a0=Coefficient[f,x^r] ;
         a1=Coefficient[f,x^(r-1)] ;
         c= Max[Floor[-a1/a0-r+1],1];
         While[ Sign[ f/.x->c]==Sign[f/.x->c+1],c++];
         If[Sign[f/.x->c]==Sign[f/.x->c+.5],
         Return[{c+1,-1}],Return[{c,1}] ]
       ] ;

ComputeCNeg[f_,x_] :=
 Module[{r=Exponent[f,x],a0,a1,c},
         a0=Coefficient[f,x^r] ;
         a1=Coefficient[f,x^(r-1)] ;
         c= Min[-1,Floor[-a1/a0+r]];
         While[ Sign[ f/.x->c]==Sign[f/.x->c-1],c--];
         If[Sign[f/.x->c]==Sign[f/.x->c-.5],
         Return[{c-1,-1}],Return[{c,1}] ]
       ] ;
```

```
Transform[f_,x_,c_]:=
 Module[{fr,r=Exponent[f,x]},
        fr=f/.x->c+(1/x) ; Return[-Expand[x^r fr] ]
      ] ;

CalculateReduced[f_,x_,n_,S_]:=
 Module[{ket={},i,c,ver,teken=S,fr=f},
        For[i=1,i<n+1,i++,
            If [teken==1,ver=ComputeCPos[fr,x],
                         ver=ComputeCNeg[fr,x] ] ;
            teken=ver[[2]] teken ;
            c=ver[[1]] ;
            fr=Transform[fr,x,c] ;
            AppendTo[ket,c] ] ;
        Return[ket]
       ] ;

ContinuedFraction[u_,f_,x_,n_]:=
 Module[{k={},fr=f,ru,bu=N[u,40],i=1},
         While[And[Not[ReducedQ[fr,x]],i<=n],
                ru=Round[bu] ;
                bu=1/(bu-ru) ;
                AppendTo[k,ru] ;
                i=i+1 ;
                fr=Transform[fr,x,ru] ] ;
         k=Join[k,CalculateReduced[fr,x,n-i,Sign[bu]]] ;
         Return[k]
       ] ;

LLLToStandard[lk_]:=
 Module[{k={},i,n=Length[lk],teken=1},
        AppendTo[k,lk[[1]]] ;
        For[i=2,i<=n,i++,
            If[teken lk[[i]] > 0 ,
               AppendTo[k,teken lk[[i]]] ,
               k[[Length[k]]]=k[[Length[k]]]-1 ;
               AppendTo[k,1] ;
               AppendTo[k,-teken lk[[i]] - 1] ;
               teken = -teken ] ] ;
        Return[k] ] ;

End[]
Protect[ContinuedFraction,LLLToStandard]
EndPackage[]
```

Literatuurlijst

1. A.K. Lenstra, H.W. Lenstra, Jr. en L. Lovasz : Factoring Polynomials with Rational Coefficients. Mathematische Annalen 261, 515-534 (1982).

2. H.W. Lenstra, Jr. : Syllabus Elementaire Algebraische Getaltheorie.

3. D.E. Knuth : The art of computer programming, Vol 2, seminumerical algorithms.

4. H. Zassenhaus : On Hensel factorization. I. J. Number Theory 1, 291-311

5. H. Davenport : The Higher Arithmetic. Sixth edition.

6. E. Bombieri en A. van der Poorten : Continued Fractions Of Algebraic Numbers. (W. Bosma en A. van der Poorten (eds.) : Computational Algebra and Number Theory, 10.)