KETTINGBREUKEN VAN COMPLEXE GETALLEN

MART KELDER

7 mei 2009

Inhoudsopgave

Hoofdstuk 1

Reële kettingbreuken

1.1 Voorwoord

Ik zal eerst kijken naar de belangrijke eigenschappen van de reële getallen met betrekking tot kettingbreuken. Dat zijn immers de eigenschappen waaraan de complexe getallen ook moeten voldoen. Ook zal ik proberen het een en ander anders te formuleren om de overeenkomst met de kettingbreuken duidelijker te maken. Aan de hand van deze overeenkomsten bekijk ik het complexe-kettingbreuk algoritme.

1.2 Verschillende reële kettingbreuken

Er zijn verschillende reële kettingbreuken. De meest bekende is het naar beneden afronden om het volgende wijzer-getal te krijgen. Dit zorgt ervoor dat $x - \lfloor x \rfloor \in [0, 1)$. Als x geen geheel getal was, en $x - \lfloor x \rfloor$ dus strikt groter dan nul, dan is $\frac{1}{x - \lfloor x \rfloor} \in (1, \infty)$.

Het is ook denkbaar om niet naar beneden af te ronden, maar naar het dichtstbijzijnde gehele getal. Afhankelijk van de precieze definitie geldt dan: $x - [x] \in \left(-\frac{1}{2}, \frac{1}{2}\right]$. Als x geen geheel getal was, dan is $x - [x] \neq 0$, en bovendien: $\frac{1}{x-[x]} \in (-\infty, -2) \cup [2, \infty)$. Het is duidelijk dat dit meer informatie geeft over het getal x.

1.3 Roosters

Er zijn verschillende roosters mogelijk in $\mathbb{C}$. Voor een kettingbreuk is het van belang dat voor een punt in de ruimte en een rooster bepaald kan worden in welk gebied het ligt.

1.3.1 Definities

Eerst bekijk ik een simpel rooster om definities te kunnen verduidelijken.

Definitie 1. *Een rooster R is een verzameling punten uit $\mathbb{R}^n$ die aan de volgende eigenschappen voldoet:*

- $0 \in R$
- *Als $x \in R$ en $y \in R$, dan ook $x + y \in R$ en $x - y \in R$.*
- $\exists \epsilon > 0 \forall r \in R \left[\{x : ||x - r|| < \epsilon\} \cap R = \{r\}\right]$*, oftewel: R is discreet.*

Definitie 2. *Een* verschoven rooster *is een verzameling punten S uit $\mathbb{R}^n$ zodat er een translatie T bestaat zodat $\left\{T^{-1} \circ s | s \in S\right\}$ een rooster vormt. Het rooster $\left\{T^{-1} \circ s | s \in S\right\}$ heet het* teruggeschoven rooster *van S.*

Merk op: een verschoven rooster is dus *niet-noodzakelijk* een rooster. De translatie T die bij het rooster hoort is niet uniek.

Definitie 3. *Een niet-verschoven rooster is een verschoven rooster S waarbij geldt dat $0 \in S$. Een verzameling is een niet-verschoven rooster dan en slechts dan als die verzameling een rooster is.*

Op dezelfde manier definieer ik type roosterpunten.

Definitie 4. *Een element van een rooster of niet-verschoven rooster heet een* niet-verschoven roosterpunt*.*

Definitie 5. *Een element van een verschoven rooster heet een* verschoven roosterpunt

Definitie 6. *Laat R een verschoven rooster zijn. Een* cel *bij R is een verzameling $C \subset \mathbb{R}^n$ zodat de afsluiting $\overline{C}$ convex is. Bovendien moet gelden dat een punt $c \in \overline{C}$ een verschoven roosterpunt uit R is dan en slechts dan als voor alle $l \subset \overline{C}$, waarbij l een lijnstuk uit $\overline{C}$ is zonder eindpunten, $c \notin l$.*

Definitie 7. *Een collectie $\mathcal{C}$ heet een* collectie cellen *bij R als $\mathcal{C}$ een verzameling cellen bij R is met*

- $C, C' \in \mathcal{C} \Rightarrow C \cap C' = \emptyset$
- $\bigcup_{C \in \mathcal{C}} C = \mathbb{R}^n$

Definitie 8. *Een* verschoven celrooster *is een verschoven rooster R samen met een collectie cellen bij R.*

Definitie 9. *Een* primitief gebied *is de cel bij R uit een verschoven celrooster R waartoe het getal 0 behoort.*

Definitie 10. *Gegeven is een verschoven celrooster R met een collectie cellen $\mathcal{C}$ en een verschoven rooster S. Kies bij elke $C \in \mathcal{C}$ een punt $p_C \in C$ zodat de verzameling $\{p_C | C \in \mathcal{C}\}$ een verschoven rooster vormt. Het teruggeschoven rooster van dit verschoven rooster heet het* verschuivingsrooster *van R. Het punt p_C heet het* basispunt *van cel C.*

Definitie 11. *Een* verschuivingspunt *van R is een element uit het verschuivingsrooster van R.*

Merk op dat een basispunt een afspraak is.

1.4 Voorbeelden van Roosters

In deze sectie kijk ik naar mogelijke verschoven celrooster en het bijbehorende verschuivingsrooster.

1.4.1 Rooster in $\mathbb{R}$

Eerst bekijk ik een simpel rooster uit $\mathbb{R}$. Dit rooster wordt voorgebracht door een getal $v > 0$. Laat t een roosterpunt van dit rooster zijn.

De verzameling $\{t + n \cdot v : n \in \mathbb{Z}\}$ is de verzameling verschoven roosterpunten van dit rooster. Voor de collectie cellen defineer ik eerst een cel C_a, waarbij $a \in \mathbb{Z}$. $C_a = \{t + (a + x) \cdot v : xin\,[0, 1)\}$. De collectie cellen $\mathcal{C}$ is nu:

$$\mathcal{C} = \{C_a : a \in \mathbb{Z}\}$$

Alle andere voorbeelden in deze sectie komen uit $\mathbb{C}$, en daarbij neem ik aan dat 0 een roosterpunt is. Ik gebruik ook dat $\mathbb{C}$ geïdentificeerd kan worden met $\mathbb{R}^2$.

1.4.2 Parallelogram-rooster

Een parallellogramrooster wordt voorgebracht door twee onafhankelijke vectoren. Deze noem ik v_1 en v_2. Een roosterpunt van het verschoven celrooster is een gehele combinatie van die vectoren.

Een cel $C_{(a,b)}$ (met $a, b \in \mathbb{Z}$) van het celrooster is de volgende verzameling:

$$\{(a+x)\cdot v_1 + (b+y)\cdot v_2 | x \in [0,1) \wedge y \in [0,1)\}$$

De collectie cellen $\mathcal{C}$ bij het celrooster is:

$$\left\{C_{(a,b)} | a \in \mathbb{Z}, b \in \mathbb{Z}\right\}$$

Het verschuivingsrooster van een parallellogramrooster wordt ook voortgebracht door v_1 en v_2.

Om een veelvoud $(x, y) \in \mathbb{Z}^2$ te vinden zodat $p - x \cdot v_1 - y \cdot v_2$ in het primitief gebied ligt, kan ik een basistransformatie uitvoeren: $v_1 \mapsto 1$ en $v_2 \mapsto i$. Onder deze basistransformatie is $(x, y) \in \mathbb{Z}^2$ makkelijk te vinden.

1.4.3 Rechthoekrooster

Het rechthoekrooster is een speciaal geval van een parallellogramrooster. Een rechthoekrooster wordt voorgebracht door twee vectoren v_1 en v_2 met $\langle v_1, v_2 \rangle = 0$ (het inproduct door op te vatten als punt in $\mathbb{R}^2$).

1.4.4 Vierkantrooster

Het vierkantrooster is een speciaal geval van een rechthoekrooster. Een vierkantrooster wordt voorgebracht door één vector v_1. De v_2 benodigd voor het rechthoekrooster kan berekend worden met $v_2 = v_1 \cdot i$.

1.4.5 Driehoekrooster

Het driehoekrooster kan ook opgebouwd worden uit het parallellogramrooster. Het driehoekrooster wordt voorgebracht door twee vectoren v_1 en v_2. Een cel uit het parallelogramrooster wordt opgedeeld in twee cellen. De cellen zijn als volgt gedefineerd:

$$\begin{cases} C_{(a,b,0)} &= \{(a+x)\cdot v_1 + (b+y)\cdot v_2 | x \in [0,1), y \in [0,1), x+y \leq 1\} \\ C_{(a,b,1)} &= \{(a+x)\cdot v_1 + (b+y)\cdot v_2 | x \in [0,1), y \in [0,1), x+y > 1\} \end{cases}$$

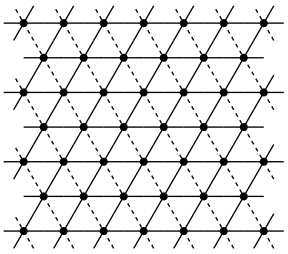
De collectie cellen voor het celrooster is nu $\left\{C_{(a,b,n)} | a \in \mathbb{Z}, b \in \mathbb{Z}, n \in \{0,1\}\right\}$ De verschoven roosterpunten zijn hetzelfde als in het parallelogramrooster waaruit dit rooster is opgebouwd.

Laat $p \in \mathbb{C}$ een punt zijn waarvan ik de cel wil bepalen. Hiervoor genereer ik eerst een parallellogramrooster voorgebracht door deze twee vectoren. Vervolgens breng ik p terug naar het primitief gebied van het parallellogramrooster.

Het primitief gebied van het parallellogramrooster bestaat uit twee driehoeken. Om te bepalen tot welke driehoek het punt p behoort kan de afstand van p tot 0 en van p tot $v_1 + v_2$ berekend worden. Als $||p|| \leq ||p - (v_1 + v_2)||$ dan behoort p tot de driehoek bij 0, anders tot de driehoek bij $v_1 + v_2$.

Ik merk ook op dat ook gedefinieerd moet worden waar de randen naar toe gaan.

Voor basispunt van cel is er nog een keuze. Laat $c \in \left[d, \frac{1}{2}\right)$. Gegeven deze keuze voor d liggen de basispunten op de volgende manier vast: $C_{(a,b,0)}$ is $(d+a)\cdot v_1 + (d+b)\cdot v_2$ en het basispunt van de cel $C_{(a,b,1)}$ is $\left(d+\frac{1}{2}+a\right)\cdot v_1 + \left(d+\frac{1}{2}+b\right)\cdot v_2$.

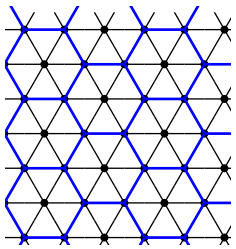


Figuur 1.1: Een driehoekrooster kan gemaakt worden uit een parallellogramrooster

1.4.6 Hexagonaalrooster

Tenslotte bestaat er nog een hexagonale rooster. Het hexagonaal rooster is een speciaal geval van een driehoekrooster. Dus maak een driehoekrooster voorgebracht door $c \cdot \zeta_6$ en c, waarbij c de straal van de omschreven cirkel van het primitief gebied is. Met behulp van het algoritme voor driehoekrooster is het hiermee mogelijk een driehoek te vinden in het hexagonaal rooster. Nu liggen twee van de hoeken van het gevonden driehoek op een zeshoek, en het derde punt is het middelpunt van een zeshoek. Aan de hand van het middelpunt ligt het gebied vast.

De cellen van het hexagonaalrooster is een vereniging van zes cellen uit het onderliggende driehoekrooster. De basispunten zijn alle verschoven roosterpunten uit het onderliggende driehoeksrooster die geen verschoven roosterpunt is in de hexagonaalrooster (als gevolg van de definitie van cel).



Figuur 1.2: Een hexagonaal is een speciaal geval van een driehoekrooster

Hoofdstuk 2

Algoritme

2.1 Stap

Een stap van het kettingbreuk-algoritme (in een verschoven celrooster R) gaat nu als volgt (voor zowel $\mathbb{R}$ als $\mathbb{C}$):

1. Gegeven is een punt p waarvan de kettingbreuk bepaald moet worden. Bepaal de cel waar p inligt.
2. Definieer b_p als het basispunt waar de cel waar p inligt.
3. Definieer b_0 als het basispunt van de cel waarin het punt 0 ligt.
4. Het wijzergetal is nu: $b_p - b_0$. Bereken een nieuw punt p' door:

$$p' := \begin{cases} \frac{1}{p+b_0-b_p} & \text{als} \quad p+b_0-b_p \neq 0 \\ 0 & \text{als} \quad p+b_0-b_p = 0 \end{cases}$$

5. Indien $p' \neq 0$, herhaal dit algoritme.

De kettingbreuk-rij is nu de lijst met wijzergetallen die tijdens het algoritme gevonden wordt. De wijzergetallen komen uit het verschuivingsrooster behorende bij het verschoven celrooster R.

Het punt b_0 is in elke stap van het algoritme hetzelfde. b_0 hoeft dus maar één keer berekend te worden, en niet in elke stap opnieuw.

2.2 Verband met reële kettingbreuk

Het is wenselijk als de reële kettingbreuk is speciaal geval is van de complexe kettingbreuk. Dit is het geval indien geldt:

$$\forall x \in \mathbb{R}\,[\mathrm{cel}\,(x) = [\lfloor x \rfloor, \lfloor x \rfloor + 1)]$$

Een voorbeeld van zo'n verschoven celrooster is een vierkantrooster voortgebracht door $\frac{1}{2} + \frac{1}{2} \cdot i$ en $\frac{1}{2} - \frac{1}{2} \cdot i$. Het basispunt van een cel is het punt uit de cel met het kleinste reële deel.

Ik merk op dat niet alle verschoven celroosters werken. Het verschoven vierkantrooster voortgebracht door 1 en i waarbij $-\frac{1}{2} \cdot i$ een verschoven roosterpunt is werkt niet, omdat dan de convergenten van $\frac{9}{10} + \frac{9}{20} \cdot i$ niet convergeert naar dat getal.

2.3 Eigenschappen

2.3.1 Zuiver repeterende kettingbreuken

Stelling 12. *Als a zuiver repeteert met periode 1, dan is a van de vorm $\frac{1}{2} \cdot b \pm \frac{1}{2} \cdot \sqrt{b^2+4}$, waarbij b een element uit het verschuivingsrooster is.*

Bewijs. a repeteert zuiver met periode 1 betekent dat $a = \frac{1}{a-b}$ waarbij b een element uit het verschuivingsrooster is. Deze vergelijking is op te lossen:

$$\begin{aligned} a &= \frac{1}{a-b} && (2.1)\\ a \cdot (a-b) &= 1 && (2.2)\\ a^2 - a \cdot b - 1 &= 0 && (2.3)\\ a &= \frac{b \pm \sqrt{b^2+4}}{2} && (2.4) \end{aligned}$$

□

Ik merk op dat een a van deze vorm niet hoeft te repeteren: alleen als het eerste wijzer-getal b dezelfde is als de b waaruit a gemaakt is, repeteert a zuiver met periode 1.

Stelling 13. *Als a zuiver repeteert met periode 2, dan is a van de vorm $\frac{1}{2} \cdot b \pm \frac{1}{2 \cdot c} \cdot \sqrt{b^2 \cdot c^2 + 4 \cdot b \cdot c}$, waarbij b en c elementen uit het verschuivingsrooster zijn.*

Bewijs. a repeteert zuiver met periode 2 betekent dat $a = \frac{1}{\frac{1}{a-b}-c}$. Hieruit is uit te rekenen:

$$\begin{aligned} a &= \frac{1}{\frac{1}{a-b} - c} && (2.5)\\ a \cdot \left(\frac{1}{a-b} - c\right) &= 1 && (2.6)\\ \frac{a}{a-b} - a \cdot c &= 1 && (2.7)\\ \frac{a}{a-b} &= a \cdot c + 1 && (2.8)\\ a &= (a \cdot c + 1) \cdot (a-b) && (2.9)\\ a &= a^2 \cdot c + a - a \cdot b \cdot c - b && (2.10)\\ 0 &= a^2 \cdot c - a \cdot (b \cdot c) - b && (2.11)\\ a &= \frac{b \cdot c \pm \sqrt{b^2 \cdot c^2 + 4 \cdot c \cdot b}}{2 \cdot c} && (2.12)\\ a &= \frac{1}{2} \cdot b \pm \frac{1}{2 \cdot c} \cdot \sqrt{b^2 \cdot c^2 + 4 \cdot c \cdot b} && (2.13) \end{aligned}$$

□

Hoofdstuk 3

Euclides

Verwant aan het kettingbreukalgoritme is het algoritme van Euclides. Het algoritme van Euclides kan de eindigheid van sommige kettingbreuken aantonen. In het reële geval is het mogelijk met het algoritme van Euclides te bewijzen dat $\frac{p}{q}$ een eindige kettingbreuk heeft.

3.1 Algoritme van Euclides in roosters

Eerst moet ik natuurlijk zeggen wat ik van het algoritme verwacht. Daarvoor bekijk ik eerst een voorbeeld in het reële geval.
Voorbeeld 14.

$$\begin{aligned} \underline{59} &= 1 \cdot \underline{37} + \underline{22} \\ \underline{37} &= 1 \cdot \underline{22} + \underline{15} \\ \underline{22} &= 1 \cdot \underline{15} + \underline{7} \\ \underline{15} &= 2 \cdot \underline{7} + \underline{1} \\ \underline{7} &= 7 \cdot \underline{1} \end{aligned}$$

De bovenste regel zegt iets over $\frac{59}{37}$. Nadat er 1 vanaf gehaald is, blijft er $\frac{22}{37}$ over. $\frac{1}{\frac{22}{37}} = \frac{37}{22}$, dus daarom gaat het algoritme verder met $\frac{37}{22}$. Dit illustreert ook het verband tussen het Euclidisch algoritme en het kettingbreuk-algoritme.

Nu probeer ik voorbeeld 14 op te vatten als een verschoven celrooster. Het is een (niet-verschoven) rooster in $\mathbb{R}$ voorgebracht door de vector 1. De breuk waarmee het algoritme begint is een breuk waarbij zowel teller als noemer een verschuivingspunt is. Hierop is dus delen met rest toe te passen. Het getal waarmee ik de deler vermenigvuldig is ook een verschuivingspunt (dit moet zo zijn wegens het verband met het kettingbreuk-algoritme). In dit voorbeeld is $59 = 1 \cdot 37 + 22$. Het getal wat overblijft (als ik de roosterpunten opvat als getallen in $\mathbb{R}$ en ik het een en ander opschrijf als breuk) is $\frac{22}{37}$. Dit is een punt uit het primitief gebied.
Voorbeeld 15. *Ik ga nu nog een voorbeeld bekijken. In dit geval gebruik ik een (niet-verschoven) vierkantrooster voortgebracht door $\frac{1}{2}$ en $\frac{1}{2} \cdot i$. De getallen volgen uit het algoritme voor kettingbreuken. Ik begin met $\frac{9}{10} + \frac{9}{10} \cdot i$.*

$$\begin{aligned} \underline{\left(\frac{9}{2} + \frac{9}{2} \cdot i\right)} &= \left(\frac{1}{2} + \frac{1}{2} \cdot i\right) \cdot \underline{(5)} + \underline{(2 + 2 \cdot i)} \\ \underline{5} &= \left(1 - \frac{3}{2} \cdot i\right) \cdot \underline{(2 + 2 \cdot i)} + \underline{(i)} \end{aligned}$$

$$\underline{2+2\cdot i} \quad = \quad (2-2\cdot i)\cdot \underline{(i)}$$

Dit voorbeeld levert de volgende rij op waarop een ϕ gedefinieerd moet worden om de eindigheid te kunnen bewijzen:

- $\frac{9}{2}+\frac{9}{2}\cdot i$
- 5
- $2+2\cdot i$
- $2\cdot i$

Dus als functie ϕ ligt voor de hand: $\phi\left(x_1\cdot\frac{1}{2}+x_2\cdot\frac{1}{2}\cdot i\right)=|x_1|+|x_2|$. In dit geval wordt ϕ elke stap steeds kleiner.

In het algemeen hoeft het algoritme helemaal niet eindig te zijn. Ik begin nu met hetzelfde voorbeeld, alleen in een ander rooster, namelijk het verschoven vierkantrooster voortgebracht door 1 en i met $-\frac{1}{2}\cdot i$ als verschoven roosterpunt.

Voorbeeld 16.

$$\begin{aligned}
\underline{(18+9\cdot i)} &= (0)\cdot\underline{(20)}+\underline{(18+9\cdot i)}\\
\underline{(20)} &= (0)\cdot\underline{(18+9\cdot i)}+\underline{(20)}\\
\vdots &= \vdots
\end{aligned}$$

In dit geval is het dus zeker niet mogelijk een ϕ te definieren die steeds kleiner wordt.

3.2 Wanneer niet mogelijk

Er is een geval waarin het makkelijk te zien is dat zo'n ϕ niet bestaat. Dit is als er een element x in het primitief gebied is waarvan ook $\frac{1}{x}$ ook in het primitief gebied zit. Dit was ook het geval in het voorbeeld 16.

Een ander geval waarin een ϕ niet mogelijk is, komt voor als er tijdens het kettingbreuk-algoritme een niet-verschuivingspunt als rest optreed. De getallen kunnen daardoor steeds kleiner worden en convergeren naar 0, maar 0 nooit bereiken. Voorbeeld 17 is hier een voorbeeld van. In dit voorbeeld heb ik een hexagonaalrooster gebruikt voortgebracht door $\frac{1}{2}$ en $\frac{1}{2}\zeta_6$. Het verschuivingsrooster van dit rooster wordt voortgebracht door $\frac{3}{2}$ en $\frac{1}{2}+\frac{1}{2}\cdot\zeta_6$.

Voorbeeld 17.

$$\begin{aligned}
\underline{(3\cdot\zeta_6)} &= \left(\frac{3}{2}\cdot\zeta_6\right)\cdot\underline{(3)}+\underline{\left(-\frac{3}{2}\cdot\zeta_6\right)}\\
\underline{(3)} &= (-4+5\cdot\zeta_6)\cdot\underline{\left(-\frac{3}{2}\cdot\zeta_6\right)}+\underline{\left(\frac{3\cdot\zeta_6-3}{4}\right)}\\
\underline{\left(-\frac{3}{2}\cdot\zeta_6\right)} &= (-4+5\cdot\zeta_6)\cdot\underline{\left(\frac{3\cdot\zeta_6-3}{4}\right)}+\underline{\left(\frac{3}{8}\right)}\\
\underline{\left(\frac{3\cdot\zeta_6-3}{4}\right)} &= (-4+5\cdot\zeta_6)\cdot\underline{\left(\frac{3}{8}\right)}+\underline{\left(-\frac{3}{16}\cdot\zeta_6\right)}\\
\underline{\left(\frac{3}{8}\right)} &= (-4+5\cdot\zeta_6)\cdot\underline{\left(-\frac{3}{16}\cdot\zeta_6\right)}+\underline{\left(\frac{3\cdot\zeta_6-3}{32}\right)}
\end{aligned}$$

$$\begin{aligned}
\underline{\left(-\frac{3}{16}\cdot\zeta_6\right)} &= (-4+5\cdot\zeta_6)\cdot\underline{\left(\frac{3\cdot\zeta_6-3}{32}\right)}+\underline{\left(\frac{3}{64}\right)}\\
\underline{\left(\frac{3\cdot\zeta_6-3}{32}\right)} &= (-4+5\cdot\zeta_6)\cdot\underline{\left(\frac{3}{64}\right)}+\underline{\left(-\frac{3}{128}\cdot\zeta_6\right)}\\
\vdots &\quad \vdots
\end{aligned}$$

3.3 Euclides in een verschoven celrooster

Ik ga nu naar een voorbeeld kijken van een vierkantrooster voortgebracht door 1 en i. Als $-\frac{1}{2}\cdot i$ een verschoven roosterpunt is, dan bestaat er een punt x uit het primitieve gebied zodat ook $\frac{1}{x}$ punt uit het primitieve gebied (voorbeeld 16). Dit is dus geen goed verschoven celrooster, dus probeer ik een ander punt. In dit geval zorg ik dat $-\frac{1}{2}-\frac{1}{2}\cdot i$ een verschoven roosterpunt is.
Voorbeeld 18.

$$\begin{aligned}
\underline{(9+9\cdot i)} &= (i+i)\cdot\underline{(10)}+\underline{(-1-i)}\\
\underline{(10)} &= (-5+5\cdot i)\cdot\underline{(-1-I)}
\end{aligned}$$

Omdat dit voorbeeld zo kort is een iets langer voorbeeld:
Voorbeeld 19.

$$\begin{aligned}
\underline{(37+31\cdot i)} &= (4+2\cdot i)\cdot\underline{(11+i)}+\underline{(-5+5\cdot i)}\\
\underline{(11+i)} &= (-1-i)\cdot\underline{(-5+5\cdot i)}+\underline{(i+i)}\\
\underline{(-5+5\cdot i)} &= (5\cdot i)\cdot\underline{(1+i)}
\end{aligned}$$

Als het algoritme begint met twee punten uit het verschuivingsrooster, dan komen alle andere punten die ook in dit voorbeeld voorkomen elementen uit het verschuivingsrooster. Alleen de rest bij deling hoeft niet uit het verschuivingsrooster te komen.

3.4 Euclidische functie

Het is dus niet mogelijk een algemene Euclidische functie te vinden, omdat die functie niet altijd werkt. Voor het verschoven celrooster voorgebracht door 1 en i, waarbij $-\frac{1}{2}-\frac{1}{2}\cdot i$ een verschoven roosterpunt is.

Hierbij maak ik ϕ:

$$\phi(x_1+x_2\cdot i)=|x_1|+|x_2| \tag{3.1}$$

Stelling 20. *ϕ is een euclidische functie voor het bovenstaande rooster.*

Bewijs. Te bewijzen: Als $x=y\cdot q+r$ (volgens Euclidisch algoritme), dan $\phi(q)>\phi(r)$. Uit het verwante kettingbreuk-algoritme volgt dat $\frac{r}{q}$ in het primitief gebied zit. Dus $\frac{r}{q}=x_1+x_2\cdot i$, waarbij $x_1,x_2\in\left[-\frac{1}{2},\frac{1}{2}\right)$. q schrijf ik ook op zo'n manier: $q=q_1+q_2\cdot i$.

Nu kan ik het te bewijzen omschrijven als

$$\phi(q_1+q_2\cdot i)>\phi((q_1\cdot x_1-q_2\cdot x_2)+(q_1\cdot x_2+q_2\cdot x_1)\cdot i) \tag{3.2}$$

De definitie van ϕ schrijf ik nu uit:

$$|q_1| + |q_2| > |q_1 \cdot x_1 - q_2 \cdot x_2| + |q_1 \cdot x_2 + q_2 \cdot x_1| \tag{3.3}$$

Ik onderscheid dit in vier gevallen:

- $|q_1 \cdot x_1 - q_2 \cdot x_2| \geq 0 \wedge |q_1 \cdot x_2 + q_2 \cdot x_1| \geq 0$

$$\begin{aligned} q_1 \cdot x_1 - q_2 \cdot x_2 + q_1 \cdot x_2 + q_2 \cdot x_1 &= q_1 \cdot (x_1 + x_2) + q_2 \cdot (x_1 - x_2) \\ &< |q_1| \cdot |x_1 + x_2| + |q_2| \cdot |x_1 - x_2| \\ &< |q_1| + |q_2| \end{aligned}$$

  Stap 3.4 volgt uit de grenzen van x_1 en x_2. $|x_1 - x_2| < 1$: daaruit volgt dat de laatste stap een ongelijkheid is.

- $|q_1 \cdot x_1 - q_2 \cdot x_2| \geq 0 \wedge |q_1 \cdot x_2 + q_2 \cdot x_1| < 0$

$$\begin{aligned} q_1 \cdot x_1 - q_2 \cdot x_2 - q_1 \cdot x_2 - q_2 \cdot x_1 &= q_1 \cdot (x_1 - x_2) + q_2 \cdot (x_1 + x_2) \\ &< |q_1| \cdot |x_1 - x_2| + |q_2| \cdot |x_1 + x_2| \\ &< |q_1| + |q_2| \end{aligned}$$

  Stap 3.4 volgt uit de grenen van x_1 en x_2. $|x_1 - x_2| < 1$: daaruit volgt dat de laatste stap een ongelijkheid is.

- $|q_1 \cdot x_1 - q_2 \cdot x_2| < 0 \wedge |q_1 \cdot x_2 + q_2 \cdot x_1| \geq 0$

$$\begin{aligned} -q_1 \cdot x_1 + q_2 \cdot x_2 + q_1 \cdot x_2 + q_2 \cdot x_1 &= q_1 \cdot (x_2 - x_1) + q_2 \cdot (-x_1 - x_2) \\ &< |q_1| \cdot |x_2 - x_1| + |q_2| \cdot |-x_1 - x_2| \\ &< |q_1| + |q_2| \end{aligned}$$

  Stap 3.4 volgt uit de grensden van x_1 en x_2. $|x_2 - x_1| < 1$: daaruit volgt dat de laatste stap een ongelijkheid is.

- $|q_1 \cdot x_1 - q_2 \cdot x_2| < 0 \wedge |q_1 \cdot x_2 + q_2 \cdot x_1| < 0$

$$\begin{aligned} -q_1 \cdot x_1 + q_2 \cdot x_2 - q_1 \cdot x_2 - q_2 \cdot x_1 &= q_1 \cdot (-x_1 - x_2) + q_2 \cdot (x_2 - x_1) \\ &< |q_1| \cdot |-x_1 - x_2| + |q_2| \cdot |x_2 - x_1| \\ &< |q_1| + |q_2| \end{aligned}$$

  Stap 3.4 volgt uit de grenzen van x_1 en x_2. $|x_2 - x_1| < 1$: daaruit volgt dat de laatste stap een ongelijkheid is.

In alle gevallen wordt het Te Bewijzen bewezen, en deze gevallen zijn alle mogelijke gevallen. □

3.5 Nauwkeurigheid benadering

De nauwkeurigheid van bovengenoemd algoritme kan worden uitgedrukt in een constante B. De nauwkeurigheid van de n-de benadering van een getal x wordt gegeven door $M_n(x)$ en is per definitie:

Definitie 21.

$$M_n(x) = \frac{1}{|q_n| \cdot |q_n \cdot x - p_n|}$$

Definitie 22.

$$B = \inf\left(M_n(x)\right)$$

Hoe groter B is, hoe beter een algemene benadering is. Ik heb `Magma`-functies geschreven om een schatting van deze B te geven.

Hoofdstuk 4

Hexagonaalrooster

Alle functies die ik in `Magma` geschreven heb, hebben betrekking op het hexagonaalrooster. In dit hoofdstuk zal ik hierover meer zeggen. Ik begin dit hoofdstuk met een bewering over eindig voorgebrachte roosters.

4.1 Eindig voortgebrachte roosters

Definitie 23. *Ik noem een niet-verschoven rooster* eindig voorgebracht *als ring als alle roosterpunten elementen zijn van een ring van de vorm* $\mathbb{Z}\cdot v_1+\mathbb{Z}\cdot v_2+\ldots+\mathbb{Z}\cdot v_n$.

Merk op dat het niet nodig is dat alle elementen van $\mathbb{Z}\cdot v_1+\mathbb{Z}\cdot v_2+\ldots+\mathbb{Z}\cdot v_n$ roosterpunten zijn. Het ligt voor de hand dat $v_1,\ldots,v_n$ elementen zijn uit een lichaam. In elk geval moet vermenigvuldiging gedefinieerd zijn. In dit hoofdstuk beperk ik mij tot voortbrengers uit de complexe getallen.

Als een verschuivingsrooster niet eindig wordt voorgebracht, dan bestaan er r_1 en r_2 zodat $r_1 \cdot r_2$ geen roosterpunt is. Ik definieer nu een verzameling:

$$K := \{k \in \mathbb{N}^* | r_1 \cdot (r_2 \cdot k) \text{ geen roosterpunt}\}$$

Ik zoek nu een $k \in K$ en een r_0 uit het rooster zodat $\frac{r_0}{r_2 \cdot k}$ (opgevat in $\mathbb{C}$) in de cel rond r_1 ligt. Als ik zo'n k en r_0 vind, dan is $r_0 - r_1 \cdot (r_2 \cdot k)$ geen verschuivingspunt. Voorbeeld 24 is hier een voorbeeld van.

Voorbeeld 24. *Ik beschouw een rooster in* $\mathbb{R}$ *voortgebracht door* $\frac{1}{2}$. *Nu is* $\frac{1}{2} \cdot \frac{1}{2}$ *geen roosterpunt, dus* $r_1 = r_2 = \frac{1}{2}$. *Ik begin met* $k = 1$.

$$\begin{array}{ccccc} r_0 & = & \frac{1}{2} \cdot \frac{1}{2} & + & r_3 \\ & & \| & & \cup \\ \Downarrow & & \frac{1}{4} & & \left[0, \frac{1}{2}\right) \\ \left[\frac{1}{4}, \frac{3}{4}\right) & & & & \end{array}$$

Er is maar één verschuivingspunt in dat gebied: $\frac{1}{2}$, *maar die voldoet niet (*$\frac{1}{2} = 1 \cdot \frac{1}{2} + 0$*).*

Ik ga verder met $k = 3$.

$$\begin{array}{ccccc} r_0 & = & \frac{1}{2} \cdot \left(\frac{1}{2} \cdot 3\right) & + & r_3 \\ & & \| & & \cup \\ \Downarrow & & \frac{3}{4} & & \left[0, \frac{1}{2}\right) \\ \left[\frac{3}{4}, \frac{5}{4}\right) & & & & \end{array}$$

Er is weer precies één verschuivingspunt in dit gebied: 1. 1 *voldoet:* $1 = \frac{1}{2} \cdot \left(\frac{1}{2} \cdot 3\right) + \frac{1}{4}$.

In het algemeen geldt ook zoiets. Het is mogelijk te beginnen bij de kleinste $k \in K$ en daarbij de enige mogelijke r_0 te proberen.

Vermoeden 25. *Zo'n k en r_0 is altijd te vinden.*

Uit dit vermoeden volgt dat er altijd een tegenvoorbeeld te vinden is als er geen ring $\mathbb{Z} \cdot v_1 + \ldots + \mathbb{Z} \cdot v_n$ bestaat waarin alle roosterpunten zitten.

4.2 Euclides

4.2.1 Standaard rooster

Indien er een ϕ die daalt bij elke stap van het euclidisch algoritme, dan bewijst dat dat een bepaalde invoer een eindige kettingbreukontwikkeling heeft. Ik beschouw het rooster zoals beschreven in 1.4.6 voortgebracht door 1 en ζ_6. Volgens de constructie moet 1 of -1 een element zijn van de cel rond 0. Als 1 een element is van de cel rond 0, dan is de kettingbreukontwikkeling van 1 oneindig. De kettingbreukontwikkeling van 1 zou eindig moeten zijn, omdat $1 = \frac{3}{3}$, en 3 en 3 niet-verschoven roosterpunten zijn.

Maar dit geldt ook voor -1:

$$\underline{(-\alpha)} = (0) \cdot \underline{(\alpha)} + \underline{(-\alpha)} \tag{4.1}$$

$$\underline{(\alpha)} = (0) \cdot \underline{(-\alpha)} + \underline{(\alpha)} \tag{4.2}$$

$$\vdots \quad \vdots \tag{4.3}$$

Dus voor het rooster volgens de standaardconstructie kan zo'n ϕ niet bestaan, omdat het niet waar is.

4.2.2 Veranderen rand cel

Ik heb niet geprobeerd de constructie aan te passen zodat zowel -1 als 1 niet tot de cel rond 0 behoren. Ik verwacht dat het gevolg van een dergelijke gewijzigde constructie is dat er een ander punt uit de ring is die oneindige cykels maakt.

4.2.3 Verkleinen rooster

De rede dat een kettingbreuk oneindig wordt, komt doordat er blijkbaar teveel elementen in de cel liggen. Een manier om dit te voorkomen is door de cel kleiner te maken. Laat α een element uit $[0, 1)$ zijn. Ik bekijk het rooster zoals beschreven in 1.4.6 (hexagonaalrooster) voorgebracht door α en $\alpha \cdot \zeta_6$. Het tegenvoorbeeld van hierboven werkt nu niet meer.

Echter de kleinste ring die $\mathbb{Z} \cdot 3 \cdot \alpha + \mathbb{Z} \cdot (\alpha + \zeta_6 \cdot \alpha)$ bevat is niet eindig voortgebracht. Dus volgens vermoeden 25 kan een tegenvoorbeeld gevonden worden.

4.2.4 Rotatie rooster

Er is nog een manier om 1 en -1 uit de standaardcel te halen: het hele rooster te draaien. De draaiing moet van de vorm ζ_i zijn (anders is het niet-verschoven rooster niet eindig voorgebracht). Dus dat is het rooster zoals beschreven in 1.4.6 (hexagonaalrooster), voortgebracht door ζ_i en $\zeta_i \cdot \zeta_6$.

Ik heb deze optie niet verder uitgewerkt.

4.3 Norm

Een ϕ-functie die scrict daalt is teveel om te hopen. Dus een dalende ϕ-functie zou al mooi zijn. Ik beschouw weer het hexagonaalrooster zoals beschreven in 1.4.6 voorgebracht door 1 en ζ_6.

Ik neem de volgende norm:

$$\phi\left(a+b\cdot\zeta_6\right)=\min\left(\left\{a^2+b^2,(a+b)^2+a^2,(a+b)^2+b^2\right\}\right)$$

Deze norm heb ik bedacht aan de hand van de kortste afstand in het driehoekrooster om van punt 0 naar een roosterpunt te komen. Hoewel ik deze norm slechts beperkt getest heb, lijkt deze in `Magma` inderdaad dalend te zijn (en constant tijdens cykels).

Hoofdstuk 5

Magma implementatie

De implementatie over deze kettingbreuken zijn te onderscheiden in twee gedeeltes:

1. Kettingbreuken over willekeurige getallen. Doordat de representatie van deze getallen eindig zijn, is de precisie hiervan beperkt.
2. Kettingbreuken over uitbreidingslichamen van $\mathbb{Q}$.

Voor beide gevallen zijn vergelijkbare functies beschikbaar (alleen de naam is anders).

5.1 Kettingbreuken over $\mathbb{C}$ in `Magma`

De volgende functies zijn bedoelt om door een 'gebruiker' gebruikt te worden:

`hexa_cf`

input

x Het getal $x \in \mathbb{C}$ waarvan de kettingbreuk gemaakt moet worden.

output Een rij wijzergetallen.

opmerking Deze functie eindigt alleen als x door een eindige complexe kettingbreuk benadert kan worden.

`hexa_pq`

input

x Het getal $x \in \mathbb{C}$ waarvan de kettingbreuk gemaakt moet worden.

output Een rij met twee elementen:

1. Een rij p's.
2. Een rij q's.

De breuk $\frac{p_i}{q_i}$ is de benadering van de wijzergetallen tot plaats i.

opmerking Deze functie eindigt alleen als x door een eindige complexe kettingbreuk benadert kan worden.

`hexa_m`

invoer

x Het getal $x \in \mathbb{C}$ waarvan de kettingbreuk gemaakt moet worden.

uitvoer Een rijtje r_i met $r_i = |q_i| \cdot |q_i \cdot x - p_i|$. Deze getallen geven aan hoe precies de benadering was.

opmerking Deze functie kan gebruikt worden om de constante B af te schatten.

`hexa_m_random`

invoer

`range` Lijst of verzameling punten die gebruikt wordt op een willekeurig getal te maken.

uitvoer Willekeurig getal. Dit getal is van de vorm $\frac{a+b\cdot z}{c+d\cdot z}$, waarbij z een voortbrenger is van de cyclische groep van orde 6 (in het complexe vlak).

`hexa_m_vals`

invoer

`range` Lijst of verzameling van punten waaruit een willekeurig getal gemaakt wordt.

`nb_of_values` Aantal keer dat de M_n van een willekeurig getal uitgevoerd moet worden.

uitvoer De maximale gevonden $M_n(x)$.

Sommige functies worden gebruikt door andere functies. Van deze functies zal ik slechts een korte omschrijving geven.

`hexa_dist`(x, c) De afstand tussen een punt x en een punt c.

`hexa_abs`(x) De absolute waarde van een punt.

`hexa_roosterpunt`(x) Bepaald het dichtstbijzijnde roosterpunt van een gegeven punt x.

In de implementatie heb ik gekozen voor een zeshoekig rooster. Dit kan worden aangepast door de functie `hexa_roosterpunt` aan te passen. Het hexagonaalrooster komt op meer plaatsen in de implementatie terug, bijvoorbeeld in de notatie van een punt.

5.2 Kettingbreuken in $\mathbb{Q}\left(\sqrt{d}\right)$

Ik heb ook ook functies geïmplementeerd om kettingbreuken te maken uit complexe getallen waarin wortels zitten. De volgende functies heb ik geïmplementeerd:

5.2.1 Data typen

Er zijn verschillende soorten informatie die steeds weer opnieuw terugkomen. Deze leg ik eerst uit zodat de omschrijving van de functies makkelijker wordt.

brongetal Een getal waarop het kettingbreuk algoritme uitgevoerd wordt.

maxlengte De maximale lengte van de uitvoer van het algoritme.

bereik Een verzameling punten die gebruikt kan worden om willekeurige elementen te vinden.

#waarde Het aantal keer dat een algoritme herhaalt wordt.

roostermult Roostermult is een optioneel argument die bij veel functies voorkomt. Het staat toe de voortbrengers van het rooster met een constante te vermenigvuldigen. Normaal wordt het onderliggende rooster voortgebracht door 1 en ζ_6. Als roostermult waarde α heeft, dan wordt er gewerkt met een hexagonaalrooster voortgebracht door α en $\alpha \cdot \zeta_6$.

Sommige komen alleen voor in interne functies:

wortelinfo Wortelinfo is een tuple van:

- Een lichaam $\mathbb{F}$ waaruit de elementen komen.
- Een lijst van worterinfo'. Voor elke wortel is er een element in deze rij.

wortelinfo' Wortelinfo geeft informatie over een specifieke wortel. Het bevat de gegevens nodig om $\frac{p_i}{q_i}$ uit te rekenen, waarbij $\frac{p_i}{q_i}$ een benadering van $\sqrt{d}$ is (voor willekeurige i). d staat ook in het object. Dit object kan gemaakt worden door de functie `hexa_wortel_initinfo`. Deze functie wordt aangeroepen met een argument d. Deze functie heeft dan betrekking of $\sqrt{d}$. Deze functie

wordt ook aangeroepen door `hexa_wortel_findroots`. De inhoud is afhankelijk van de kettingbreuk van $\sqrt{d}$. De pre-periode van deze kettingbreuk noem ik k. en de periode noem ik l. Het object zelf is een Tuple van lengte 7. Het bevat de volgende elementen:

1. Geïtialiseerd als p_{k+1}. In het algemeen bevat het p_i.
2. Geïtialiseerd als q_{k+1}. In het algemeen bevat het q_i.
3. Geïtialiseerd als p_k. In het algemeen bevat het p_{i-1}
4. Geïtialiseerd als q_k. In het algemeen bevat het q_{i-1}
5. De wijzergetallen van de periode van $\sqrt{d}$.
6. De plaats in de periode tot waar $\frac{p_i}{q_i}$ uitgerekend is.
7. d: de wortel waarmee het object begon.

5.2.2 Functies

`hexa_wortel_cf`

invoer

x Een brongetal waarvan de complexe kettingbreuk.

n De maximale lengte van het antwoord.

uitvoer De wijzergetallen van de complexe kettingbreuk van x van lengte ten hoogste n.

`hexa_wortel_pq`

invoer

x Een brongetal waarvan de complexe kettingbreuk.

n De maximale lengte van het antwoord.

uitvoer De convergenten van de benadering van x met maximale lengte n.

`hexa_wortel_create`

invoer

verz Een verzameling van wortels. Deze functie geeft een Magma NumberField terug die wordt voortgebracht door de volgende rij polynomen: `[ x^2+x+1 ] cat [ x^2 - d : d in verz]`.

uitvoer Een Magma NumberField terug van de volgende rij polynomen: `[ x^2+x+1 ] cat [ x^2 - d : d in verz]`.

opmerking Deze functie kan gebruikt worden als basis van een brongetal.

`hexa_wortel_m`

invoer

x Een brongetal waarvan de complexe kettingbreuk.

n De maximale lengte van het antwoord.

uitvoer Een reëel getal die aangeeft hoe goed de benadering van de n-de convergent van x is.

opmerking De convergent wordt berekend met behulp van het complexe kettingbreuk-algoritme.

`hexa_wortel_m_random`

invoer

bereik Het bereik waaruit willekeurige getallen gekozen moeten worden.

n De diepte tot waarop kettingbreuken uitgerekend moeten worden.

wortels Een verzameling wortels waaruit willekeurige brongetallen opgebouwd worden.

uitvoer Deze functie genegeerd een willekeurig brongetal $x = \sum_{k=1}^{\#p} \frac{r_{4\cdot(k-1)}}{r_{4\cdot(k-1)+1}} \cdot p_k + \frac{r_{4\cdot(k-1)+2}}{r_{4\cdot(k-1)+3}} \cdot p_k \cdot y$, waarbij $r_0, \ldots, r_{4\cdot\#p}$ willekeurige getallen, $p_1, \ldots, p_{\#p}$ priemgetallen uit 'wortels', en y nulpunt van $x^2 + x + 1$.
Vervolgens berekend de functie `hexa_wortel_m` van de convergenten van dit getal uit, en geeft dat (een lijst dus) als uitvoer terug. Oftewel: de functie genereert een willekeurig getal, en geeft terug hoe goed de benadering was.

`hexa_wortel_m_vals`
invoer
bereik Het bereik waaruit willekeurige getallen gekozen moeten worden.
#waardes Het aantal keer dat willekeurige brongetallen uitgerekend worden worden en daarvan de nauwkeurigheid wordt bepaald.
maxlengte De maximale lengte van een kettingbreukbenadering.
wortels Een verzameling wortels waaruit willekeurige brongetallen opgebouwd worden.
uitvoer Deze functie voert `hexa_wortel_m_random` #waardes keer uit. Het resultaat is het maximum van de verkregen waardes.

`hexa_wortel_norm` **invoer**
`elem` Element waarvan de norm bepaald moet worden.
uitvoer De norm van `elem`.

De volgende functies zijn bedoelt voor intern gebruik. Ik zal toch een korte omschrijving geven van deze functies.

`hexa_wortel_square_rec` Invoer: x brongetal, wortelinfo, n recursie variabele. Uitvoer: x^2.

`hexa_wortel_abs` Invoer x brongetal, wortelinfo. Uitvoer: $|x|$.

`hexa_wortel_compare_minmax_rec` Invoer: x brongetal, wortelinfo, n recursie variabele. Uitvoer: `[ minumum, maximum ]`, waarbij minimum een onderschatting is en maximum een overschatting van x.

`hexa_wortel_compare_minmax` Roept `hexa_wortel_compare_minmax_rec` aan met de variabelen en met 1 als recursie-constante.

`hexa_wortel_lt` Invoer: x brongetal, y brongetal, wortelinfo. Uitvoer: waar indien $x < y$.

`hexa_wortel_dist` Invoer: x brongetal, y brongetal, wortelinfo. Uitvoer: $||x - y||$.

`hexa_wortel_infoupdate` Invoer: wortelinfo. Berekent de volgende convergenten en stopt deze in wortelinfo.

`hexa_wortel_entier_rec` Invoer: x brongetal, wortelinfo, n recursie variabele. Uitvoer: `[ min, max ]` als $n > 1$; `<succes, resultaat>` als $n = 1$, waarbij succes waar indien een entier gevonden is, en resultaat een mogelijke waarde.

`hexa_wortel_entier` Invoer: x brongetal, wortelinfo. Uitvoer: $\lfloor x \rfloor$.

`f9a` Invoer: a, b en d. Uitvoer: `<pre-periode, periode>`, waarbij pre-periode de pre-periode van $a + b \cdot \sqrt{d}$ is, en periode de periode van $a + b \cdot \sqrt{d}$

`hexa_wortel_initinfo` Invoer: wortel. Uitvoer: wortelinfo'. Berekent een wortelinfo' voor een gegeven wortel.

`hexa_wortel_findroots` Invoer: wortels, Uitvoer: wortelinfo. Maakt een lijst van wortelinfo''s, gebaseerd op de rij wortels.

`hexa_wortel_splits_z3` Invoer: x brongetal. Uitvoer: $< \alpha, \beta >$. Dit splits een getal x in $\alpha + \beta \cdot \zeta_3$.

`hexa_wortel_splits` Invoer: x brongetal. Uitvoer: $< \alpha, \beta >$. Dit splits een getal in constanten maal zijn voortbrengers.

`hexa_wortel_dichtsbij` Invoer: x brongetal, lijst, wortelinfo. Uitvoer: element uit lijst met de kleinste afstand tot x.

`hexa_wortel_is_roosterpunt` Invoer: a geheel getal, b geheel getal. Uitvoer: waar indien $a + b \cdot \zeta_6$ roosterpunt is, onwaar anders.

`hexa_wortel_roosterpunt_var` Invoer: x brongetal, wortelinfo. Uitvoer: het roosterpunt van x.

`hexa_wortel_roosterpunt` Invoer: x brongetal. Uitvoer: roosterpunt van x.

`hexa_wortel_eucstring` Invoer: p, q, r, ro roosterpunt. Uitvoer: string van de vorm: "p = ((ro))*(q) + (r)".

`hexa_wortel_cf_var` Invoer: x brongetal, n maxlengte, wortelinfo. Uitvoer: de wijzergetallen van x met maximale lengte n.

`hexa_wortel_wortel_pq_var` Invoer: x brongetal, n maxlengte, wortelinfo. Uitvoer: de convergenten van x met maximale lengte n.

`hexa_wortel_m_var` Invoer: x brongetal, n maxlengte, wortelinfo. Uitvoer: een getal over de benadering van de kettingbreuk van x met maximale lengte n.

5.3 Maxima

In heb vergelijkbare functies geïmplementeerd in Maxima. Maxima is een Open-Source algebra-pakket die geschreven is in Lisp, en kan derhalve willekeurige variabelen in een vergelijking kan laten staan. Het gevolg is natuurlijk wel dat het langzamer is dan Magma. Het grootste probleem met de functies is dat Maxima niet altijd in staat bleek te controleren of een getal groter is dan een ander getal (de getallen zijn dan natuurlijk lang). Dat is een essenciële stap voor het kettingbreukalgoritme.