Data transporteren gaat met behulp van een code. Een van de bekendste is de Morse code, waar van alles mee mis kan gaan alleen al door de verschillende ‘lengtes’ van de letters. Geavanceerdere codes zijn juist ‘zelf fout detecterend’ en zelfs ‘zelf fout herstellend’. En ineens bleek reeds bestaande wiskunde een prachtige toepassing te hebben. **Gerard van der Geer** hield deze voordracht op ‘Leve de Wiskunde!’

Codes

Inleiding

Wat zijn codes? Wellicht is het beter het eerst te hebben over de toepassing van codes. Codes worden gebruikt om (digitale) data foutenvrij te transporteren. Dit betreft alle vormen van digitaal datatransport, variërend van satellietcommunicatie, internet en fax tot en met het lezen van een CD. In het algemeen ontstaan bij het transport kleine veranderingen door ruis, en het gaat erom deze fouten te herkennen en zo mogelijk te corrigeren. Het faxapparaat, de CD-speler en de computer doen dat ook daadwerkelijk.
Een analoog fenomeen ervaren we als we de krant lezen. Kranten zijn nooit vrij van drukfouten; desondanks kunnen we de krant zonder moeite lezen, de fouten herkennen en meestal eenvoudig verbeteren. Dat berust op het feit dat we bij wijze van spreken Van Dale in ons hoofd hebben en weten welke woorden toegelaten zijn. Treffen we woorden aan die niet in Van Dale staan, dan is er blijkbaar een fout opgetreden. We proberen het dan te repareren door het niet bestaande woord te vervangen door een woord dat erop lijkt en wel in het woordenboek voorkomt. Erop lijken betekent hier: op zo weinig mogelijk plaatsen verschillen van een woord in het woordenboek.

Zo’n reparatie lukt omdat tekst meestal een grote redundantie bevat. Hiermee komen we bij het eerste principe van de coderingstheorie: efficiënte benutting van een zekere redundantie. Hoeveel redundantie taal bevat, blijkt als we uit een bekend gedicht alle klinkers weglaten. Het blijft leesbaar, en zelfs als we het niet (her)kennen, zouden we het kunnen reconstrueren, zoals archeologen met beschadigde teksten doen.

BR LLN GPFLN ST RH,
BR LLN WPFLN SPRST D KM NN HCH,
D VGLN SCHWGN M WLD,
WRT, WRT, BLD RHST D CH.

Of bijvoorbeeld bij een andere beroemde regel

T B R NT T B, THT S TH QSTN.

Of standaard sms-communicatie: *w8; ikgatzz*.

We versturen informatie in woorden die in een bepaald alfabet geschreven zijn. We kiezen een ‘woordenboek’ van toegelaten woorden en spreken af alleen die woorden te versturen. Als we aan de andere kant van de verbinding een niet toegestaan woord ontvangen, dan signaleren we een fout en vervangen dat woord door het meest nabije woord (of althans een nabij woord) uit het woordenboek. Dat functioneert goed als de ‘afstanden’ tussen de woorden in het woordenboek voldoende groot zijn. Als veel woorden slechts een letter verschillen, functioneert het maar matig.

Om het versturen van informatie in de praktijk uit te voeren, moeten we een alfabet kiezen. Voor digitaal datatransport ligt de keuze voor de hand: we schrijven alle informatie in nullen en enen, dus het alfabet is {0; 1}. Vroeger heette dat kort en lang, namelijk bij de Morsecode die lang bij de telegrafie is gebruikt.

symbool	codewoord	symbool	codewoord
A	·–	N	–·
B	–···	O	–––
C	–·–·	P	·––––·
D	–··	Q	––·–
E	·	R	·–·
F	··–·	S	···
G	––·	T	–
H	····	U	··–
I	··	V	···–
J	·–––	W	·––
K	–·–	X	–··–
L	·–··	Y	–·––
M	––	Z	––··

In tegenstelling tot het gebruik van de Morsecode uit vervlogen tijden, ligt de lengte van codewoorden vast en worden de data in eenheden van vaste lengte verpakt.

Zulke codes heten *blokcodes*. Een voorbeeld van zo'n code is de ISBN-code, die we in boeken vinden. Het is een blokcode van lengte tien, en het alfabet bestaat uit tien Arabische cijfers. De eerste negen cijfers van de ISBN-code geven ISBN land-uitgever-kengetal-...

$$\text{ISBN } a_1 - a_2a_3a_4\ldots a_8 - a_9 - \ldots$$

zoals bijvoorbeeld

$$\text{ISBN } 3 - 540 - 03525 - \ldots$$

waarna het tiende symbool ervoor zorgt dat:

$$a_{10} + 2a_9 + 3a_8 + \ldots + 9a_2 + 10a_1$$

deelbaar is door 11. In ons voorbeeld is $a_{10} = 7$. (De 'tien' wordt weergegeven met X.) Een ander voorbeeld van een code uit het verleden is door de 'twee uit vijf'-code. De naam zegt voldoende: een woord van lengte 5 geschreven in het alfabet {0; 1} zit in het woordenboek precies dan als het twee keer een 1 bevat.

symbool	codewoord
1	11000
2	10100
3	01100
4	10010
5	01010
6	00110
7	10001
8	01001
9	00101
0	00011

Getalsystemen

We moeten ons eerst bezighouden met wat basiskennis. We kennen allemaal de verzameling van de natuurlijke getallen N:

$$\{1, 2, 3, 4, \ldots\}$$

Die kunnen we optellen en vermenigvuldigen, maar aftrekken lukt niet altijd. Daarom nemen we de negatieve getallen erbij, en krijgen de verzameling van de gehele getallen Z:

$$\{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$$

Aftrekken lukt nu altijd, maar delen vaak niet. Daarom nemen we de breuken er ook bij en krijgen de rationale getallen Q:

$$\{\ldots, -\tfrac{2}{3}, -\tfrac{3}{2}, -4, -3, -\tfrac{1}{3}, -\tfrac{1}{2}, -2, -1, 0, 1, 2, \tfrac{1}{2}, \tfrac{1}{3}, 3, 4, \ldots\}$$

Nu kunnen we optellen, aftrekken, vermenigvuldigen en delen door getallen ongelijk 0. Zo'n systeem heet een *lichaam*. Sommige mensen zijn ook met de rationale getallen nog niet tevreden, en bekijken dan de reële getallen op de getallen-rechte, die ook een lichaam vormen, of een nog groter lichaam, dat van de complexe getallen. Maar het kan allemaal veel simpeler. Zo is er bijvoorbeeld een lichaam met maar twee elementen $F_2 = \{0; 1\}$. De optelling gaat zo: $0 + 0 = 0$, $0 + 1 = 1 + 0 = 1$, $1 + 1 = 0$, en de vermenigvuldiging zo $0 \cdot 0 = 0$, $0 \cdot 1 = 1 \cdot 0 = 0$, $1 \cdot 1 = 1$. Blijkbaar geldt $2 = 0$ en $-1 = 1$. Delen door 1 gaat ook. Als we voor 0 'even' en voor 1 'oneven' lezen, luidt een van de optelregels 'oneven plus oneven is even'. Er zijn oneindig veel van zulke eindige getalsystemen. Zo is er bijvoorbeeld het lichaam F_3. Het bestaat uit drie elementen {0, 1, 2} met de regels voor het optellen:

$$\begin{array}{lll} 0+0=0 & 1+0=1 & 2+0=2 \\ 0+1=1 & 1+1=2 & 2+1=0 \\ 0+2=2 & 1+2=0 & 2+2=1 \end{array}$$

en voor het vermenigvuldigen:

$$\begin{array}{lll} 0\times 0=0 & 1\times 0=0 & 2\times 0=0 \\ 0\times 1=0 & 1\times 1=1 & 2\times 1=2 \\ 0\times 2=0 & 1\times 2=2 & 2\times 2=1 \end{array}$$

In deze context kennen we ook een soort complexe getallen, bijvoorbeeld het lichaam F_9, een getalsysteem met 9 elementen dat een uitbreiding is van F_3:

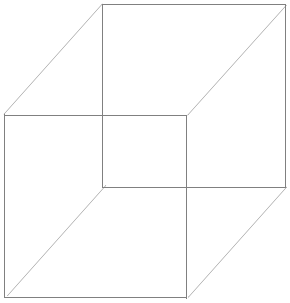
$$\begin{array}{lll} 0 & 1 & 2 \\ i & 1+i & 2+i \\ 2i & 1+2i & 2+2i \end{array}$$

met extra regel: $i \times i = -1$. Dus er geldt $i = \sqrt{-1} = \sqrt{2}$. Een voorbeeld van optellen wordt gegeven door: $(1 + i) + 2i = 1$, en van vermenigvuldigen door: $i \times (1 + i) = 2 + i$. Men ziet, het leven wordt veel eenvoudiger als we met zulke getalsystemen werken.
Deze getalsystemen zijn rond 1830 ontdekt door de beroemde wiskundige Galois, die zeer jong in een duel is gestorven. Maar het heeft lang geduurd voordat deze lichamen ook algemeen ingang in de wiskunde hebben gevonden. De Amerikaanse wiskundige Moore liet in 1893 zien dat er voor iedere macht p^n van een priemgetal p precies één lichaam F_{p^n} is met p^n elementen.

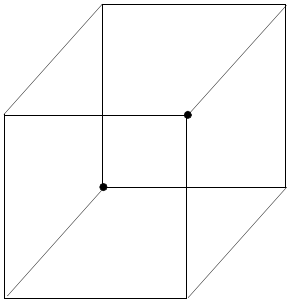
Meetkunde

Maar de wiskunde is niet alleen getallen, er is ook meetkunde. In de meetkunde van de driedimensionale ruimte gebruiken we de coördinaten x, y en z, en de lineaire algebra regelt de boekhouding daarvan. De coördinatenassen zijn zelf weer reële rechten. Als we nu het lichaam van de reële getallen vervangen door het lichaam F_2 wordt alles veel eenvoudiger. De driedimensionale ruimte bestaat dan uit acht punten die we als de hoekpunten

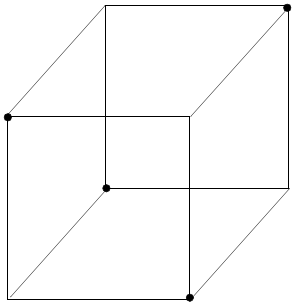
van een kubus kunnen zien. We kunnen ook weer van vlakken en lijnen spreken, en deze met lineaire vergelijkingen beschrijven, bijvoorbeeld het vlak $x + y + z = 0$. Als coëfficiënten treden natuurlijk alleen 0 en 1 op.



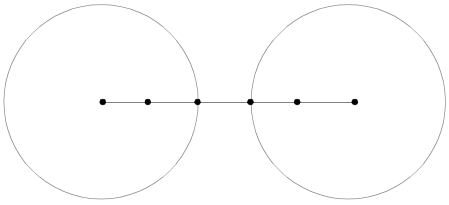
De acht punten uit onze driedimensionale ruimte kunnen we zien als woorden van lengte 3, geschreven in ons alfabet {0, 1}. Een code is dan niets anders dan een deelverzameling in deze driemensionale ruimte. Laten we maar een voorbeeld bekijken. Bij de *herhalingscode* van lengte 3 wordt ieder bit informatie drie keer herhaald. De toegelaten woorden zijn dan 000 en 111. We gebruiken dus twee punten van de kubus, namelijk (0, 0, 0) en (1, 1, 1). Dat is niet bijster spannend.



Een iets betere code bestaat uit vier van de acht mogelijke woorden {000; 011; 101; 110}. Deze punten zijn precies de punten die aan de vergelijking $x + y + z = 0$ voldoen. De code bestaat dus uit de punten in het vlak $x + y + z = 0$. We kunnen de coördinaten x en y gebruiken voor de informatie die we willen versturen, en z is dan het controlesymbool.



Bij de al genoemde twee uit vijf-code gaat het om een deelverzameling van tien punten van de vijfdimensionale ruimte over het lichaam F_2.
Het heeft nu ook zin om van afstand te spreken. De *Hamming-afstand* van twee woorden is het aantal coördinaten waar twee woorden verschillen. Dat is een goede afstandsfunctie. Bij de herhalingscode van lengte 3 is de afstand tussen de woorden 3. Als we bij het verzenden één of twee fouten maken, kunnen we dat direct herkennen. Als we één fout maken, kunnen we – onder de aanname dat er hoogstens één fout gemaakt is – de fout direct herstellen. Het meest nabije woord uit de code (afstand 1) is eenduidig bepaald. De code heet 1-fout corrigerend.
Een formele definitie van een code van lengte n kunnen we nu geven: een deelverzameling van de n-dimensionale ruimte over F_2. Als we een lineaire deelruimte als deelverzameling kiezen, dan heet de code *lineair*.
Om een goede code te krijgen, moeten we ervoor zorgen dat de woorden zo ver mogelijk uit elkaar liggen. Anderzijds willen we dat ons woordenboek niet al te klein is. Deze voorwaarden zijn in zekere zin met elkaar in tegenspraak, en de coderingstheorie probeert een goed compromis te vinden.
Laten we het preciezer bekijken. Als de woorden van een code altijd minstens afstand $2t + 1$ hebben, en we maken bij het versturen van een woord hoogstens t fouten, dan is er precies één codewoord dat het meest nabij is.



De afstand tussen twee codewoorden is 5

Zelfs als we $2t$ fouten maken, gaat een codewoord nog niet in een ander codewoord over, want binnen een bol met straal $2t$ rondom een codewoord c ligt maar één codewoord, namelijk c. De code herkent dus $2t$ fouten en verbetert t fouten. De minimale afstand tussen twee woorden is dan ook een belangrijke invariant van een code. Deze afstand heet *minimumafstand* en we gebruiken daarvoor meestal de letter d. Een lineaire code heeft drie belangrijke invarianten: de woordlengte n, de dimensie k van de code en de minimumafstand d. Deze invarianten voldoen aan bepaalde relaties, bijvoorbeeld geldt:

$$k + d \leq n + 1$$

Vaak kiezen we voor een heel grote woordlengte, met andere woorden: we laten n heel groot worden en zijn dan geïnteresseerd in de verhoudingen $R = k/n$, $\delta = d/n$, waarbij R staat voor de *rate* (efficiëntie) en δ voor de relatieve afstand.

Het begin van de coderingstheorie

Het begin van de coderingstheorie is goed te dateren. De wiskundige Richard W. Hamming kon in 1947 de com-

puter van Bell Telephone Labaratories (Bell Labs) alleen in het weekeinde gebruiken. Deze computer was een mechanische relaiscomputer. Om een indruk te geven: dit Model V bevatte 9000 relais, nam 100 m^2 in beslag en had een gewicht van tien ton. Over de snelheid zullen we het maar niet hebben. Ik citeer Hamming:

> Two weekends in a row I came in and found that all my stuff had been dumped and nothing was done... And so I said: 'Damn it, if the machine can detect an error, why can't it locate the position of the error and correct it?'

Deze computer van Bell Labs gebruikte de twee uit vijfcode die we al eerder tegenkwamen. De verzuchting van Hamming betekende het begin van de coderingstheorie. Hamming construeerde zijn eerste 'Hamming-code' op 27 juli 1947 en verbeterde hem in 1948. Dit was de (7, 4)-Hammingcode, een geraffineerde code. Deze heeft een lengte 7 en dimensie 4, dus het woordenboek bestaat uit een vierdimensionale lineaire deelruimte van de zevendimensionale ruimte over F_2. Er worden per woord vier informatiebits verzonden en drie controle-symbolen (*parity checks*) toegevoegd. Dat is efficiënter dan één keer herhalen van ieder woord. Dus een informatiewoord wordt gecomplementeerd:

$$(x_1, x_2, x_3, x_4) \longmapsto (x_1, x_2, x_3, x_4, x_5, x_6, x_7)$$

met drie extra symbolen x_5, x_6 en x_7 gegeven door:

$$x_5 = x_2 + x_3 + x_4$$
$$x_6 = x_1 + x_3 + x_4$$
$$x_7 = x_2 + x_2 + x_4$$

Dus bijvoorbeeld:

$$(0, 0, 0, 0) \longmapsto (0, 0, 0, 0, 0, 0, 0)$$
$$(1, 0, 0, 0) \longmapsto (1, 0, 0, 0, 0, 1, 1)$$
$$(1, 1, 1, 1) \longmapsto (1, 1, 1, 1, 1, 1, 1)$$

Een equivalente manier om deze code te beschrijven is met vergelijkingen:

$$\begin{array}{cccccccccccc} & & & & x_4 & + & x_5 & + & x_6 & + & x_7 & = & 0 \\ & x_2 & + & x_3 & + & & & & x_6 & + & x_7 & = & 0 \\ x_1 & + & & x_3 & + & & x_5 & + & & & x_7 & = & 0 \end{array}$$

Dit stelsel lineaire vergelijkingen correspondeert met de matrix:

$$H = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

in de zin dat een vector c in de code ligt dan en slechts dan als $H c^t = 0$. Merk op dat de kolommen van H een-een corresponderen met de getallen 1 tot en met 8 in het tweetallig stelsel. Stel, we zenden (1, 1, 1, 1, 1, 1, 1) en ontvangen (1, 1, 1, 1, 0, 1, 1), een fout op plaats 5. Bereken nu:

$$\begin{array}{ccccccccccc} s_1 & = & x_4 & + & x_5 & + & x_6 & + & x_7 & = & 1 \\ s_2 & = & x_2 & + & x_3 & + & x_6 & + & x_7 & = & 0 \\ s_3 & = & x_1 & + & x_3 & + & x_5 & + & x_7 & = & 1 \end{array}$$

Deze vector $\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$ hoort bij een fout op plaats:

$$5 = 1 \cdot 2^0 + 0 \cdot 2^1 + 1 \cdot 2^2.$$

We hebben niet alleen ontdekt dat er een fout is, we kunnen hem – onder de aanname dat er maar één bit veranderd is – ook verbeteren. De vector $\begin{pmatrix} s_1 \\ s_2 \\ s_3 \end{pmatrix}$ geeft de positie van de fout aan.

Hammings (7, 4)-code werd door Holbrook voorzien van een ontwerp voor een schakeling, en dit geheel werd geregistreerd als Hamming-Holbrook-patent. Dit patent werd een aantal jaren later (1956) vrijgegeven in het kader van de antitrustwetten tegen het Bellmonopolie.
In hetzelfde jaar dat Hamming zijn code ontwierp, verscheen ook het boek van Claude Shannon, *The Mathematical Theory of Communication*, waarin hij bewees dat er goede, dat wil zeggen efficiënte, codes zijn. Het argument van Shannon was statistisch, en gaf niet aan hoe deze codes te vinden waren. Dit werd dan ook de centrale vraag van de coderingstheorie. In de jaren na Hamming werden er zeer veel speciale codes gevonden en vaak opnieuw ontdekt. Naast Hamming heeft ook Golay in het begintijdperk vele goede codes gevonden. Later is daarover helaas een prioriteitsstrijd tussen Hamming en Golay ontbrand.

De ternaire Golay-code en de Toto

In verband met Golay is het aardig terug te gaan naar de tijd dat het Nederlandse voetbaltotoformulier nog bestond uit elf wedstrijden, met natuurlijk drie mogelijke uitkomsten: winst, verlies en gelijkspel. Dus een ingevuld totoformulier is een woord van lengte elf in drie letters, waarvoor we natuurlijk de elementen van het lichaam $F_3 = \{0, 1, 2\}$ mogen nemen. Dat geeft dus 3^{11} mogelijke uitkomsten voor het totoformulier. In de elfdimensionale ruimte over F_3 ligt nu een prachtige code, de *ternaire Golay code*. Deze code bestaat uit $3^6 = 729$ woorden en wordt gedefinieerd door de matrix G:

$$\begin{array}{cccccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 2 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 2 & 2 \\ 0 & 0 & 0 & 1 & 0 & 0 & 2 & 1 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 1 & 0 & 2 & 2 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 2 & 2 & 1 & 0 \end{array}$$

in die zin dat de rijen van G een basis vormen voor de lineaire deelruimte die onze code is. De minimumafstand

tussen de woorden is $d = 5$, de code is dus twee-fouten-verbeterend. Een bol met straal 2 rond een punt bevat:

$$1 + 11 \times 2 + \binom{11}{2} \times 2 \times 2 \;=\; 243 \;=\; 3^5$$

woorden. Er geldt $729 \times 243 \;=\; 3^{11}$, dus de bollen met straal 2 rondom de codewoorden overdekken de gehele ruimte. De code is *perfect*. Je hoefde bij de Nederlandse toto dus maar 729 formulieren in te vullen om er zeker van te zijn dat je op een formulier hoogstens twee fouten had. Daarmee heeft een aantal lieden veel geld verdiend, totdat de leiding van de toto na consultatie van wiskundigen doorhad wat er aan de hand was en de lengte van de woorden heeft aangepast: dertien in plaats van elf.

Goppa Codes

In de zoektocht naar codes zijn zeer veel goede codes gevonden. De Reed-Solomon code is een voorbeeld dat wijde toepassing heeft gevonden bij planeetverkenners, maar ook bij muziek-CD's. We nemen als alfabet het lichaam F_q met $q = p^m$, zeg $q = 2^m$. De structuur van F_q staat toe om een element a te kiezen zodat F_q bestaat uit 0 en de machten a^j met $1 \le j \le q - 1$ van a:

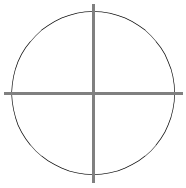
$$F_q \;=\; \{0\} \cup \{a, a^2, a^3, \ldots, a^{q-1} \;=\; 1\}$$

Zo'n element a heet een primitief element.
De Reed-Solomon-code is nu een lineaire deelruimte van F_q^n met $n = q - 1$. De woordlengte van onze code is dus $n = q - 1$ en de dimensie $k = r$ waarbij de r basisvectoren van onze code (een lineaire deelruimte van F_q^n gegeven zijn als:

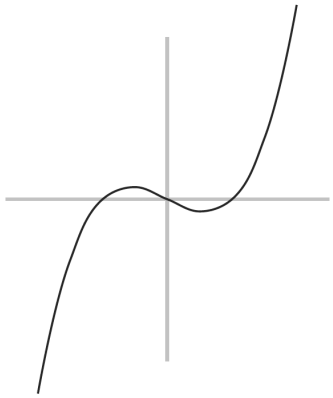
$$(a^j, a^{2j}, a^{3j}, \ldots, a^{(q-1)j}) \qquad j = 0, \ldots, r - 1$$

De minimumafstand is dan $d = n + 1 - r$. We kunnen de woorden van deze code nu ook interpreteren als de waarden van de functie X^j in de punten P_i met coördinaat $X = a^i$ op de lijn over F_q.

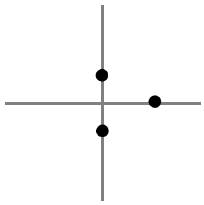
De Russische wiskundige Goppa had rond 1980 het prachtige idee om dit te generaliseren door functies op een algebraïsche kromme te evalueren en zo codes te construeren.
Een algebraïsche kromme in het vlak kan gegeven worden door een veeltermvergelijking $f(x, y) = 0$. Een lineaire vergelijking definieert een lijn, maar bijvoorbeeld $x^2 + y^2 = 1$ geeft de vergelijking van de cirkel:



Een ander voorbeeld wordt gegeven door de kromme door de vergelijking $y = x^3 - x$:



We vatten zulke plaatjes op als de grafiek van de kromme in $R \times R$, met R de reële getallen. Maar wat let ons om zulke vergelijkingen over eindige lichamen te beschouwen? Zo kunnen we over F_3 de kromme gegeven door $y^2 = x^3 + 1$ bekijken. Oplossingen van de vergelijking zijn de paren (x, y) gelijk aan (0, 1), (0, 2), (2, 0). Anders gezegd, de kromme gaat door de punten (0, 1), (0, 2) = (0, – 1) en (2, 0):



Een tegenwerping zou kunnen zijn dat zo'n kromme maar eindig veel punten bezit, wat de term kromme misschien misplaatst maakt. Maar over de reële getallen heeft de vergelijking $x^2 + y^2 = -1$ ook geen oplossingen (want kwadraten van reële getallen zijn niet-negatief) terwijl het niet onredelijk is toch van een algebraïsche kromme te spreken. Tenslotte zijn er wel veel punten als we overgaan op de complexe getallen. Zo is het ook met onze vergelijking $y^2 = x^3 + 1$ over eindige lichamen. Over F_3 zijn er maar drie punten, maar over F_9, een uitbreiding van F_3, zijn er al meer punten, bijvoorbeeld $(1, i)$.
Een kromme heeft een geslacht en dat is een maat voor de complexiteit van de algebraïsche kromme. Het geslacht, meestal aangegeven met de letter g, is een geheel getal ≥ 0. De lijn heeft geslacht 0.
Voor een (gladde) vlakke kromme van graad d in het projectieve vlak is er een klassieke formule van Plücker waarmee we het geslacht kunnen uitrekenen:

$$g \;=\; \frac{(d-1)(d-2)}{2}$$

Als we de kromme over de complexe getallen bekijken, wordt het een oppervlak (een zogenaamd Riemannoppervlak) en dan is g het aantal 'gaten' in het Riemannoppervlak, zoals in de volgende figuur:

Riemannoppervlak met twee gaten

Goppa's idee is nu als volgt: neem een vectorruimte L van functies op een algebraïsche kromme C en neem punten P_1 , ..., P_n van de algebraïsche kromme met coördinaten in F_q. We evalueren de functies uit deze vectorruimte nu in de punten van de algebraïsche kromme:

$$L \ni f \to (f(P_1), f(P_2), \ldots, f(P_n))$$

en dat levert ons een woord van de code, en het geheel van de zo verkregen woorden is een *Goppacode*. Het grote voordeel is nu dat we de theorie van de algebraïsche krommen kunnen gebruiken om iets over deze codes te weten te komen. De algebraïsche meetkunde levert bijvoorbeeld onmiddellijk:

$$k + d \geq n + 1 - g$$

met g het geslacht van de kromme. De ongelijkheid $k + d \geq n + 1 - g$ voor een Goppacode geeft na delen door n de ongelijkheid:

$$R + \delta \geq 1 + (1 - g)/n$$

Als we nu $R + \delta$ zo groot mogelijk willen maken bij vast geslacht g en vaste q, dan moeten we n zo groot mogelijk maken. Dat betekent dat we bij een vast gekozen geslacht en een vast eindig lichaam F_q het aantal rationale punten, dat wil zeggen het aantal punten met coördinaten in F_q, zo groot mogelijk moeten maken.
Het idee van Goppa opende een heel nieuw venster op codes en het duurde niet lang voor de eerste spectaculaire resultaten kwamen. Door gebruik te maken van zogenaamde modulaire krommen die veel punten bezitten, konden Tsfasman, Vladuts en Zink laten zien dat er een rij codes is met asymptotisch heel goede eigenschappen, zo goed, dat deze codes asymptotisch beter zijn dan de zogenaamde Gilbert-Varshamov-grens, een grens waar coderingstheoretici twintig jaar lang tegenaan gehikt hadden en die ze nooit hadden overwonnen. Zo luidde de spectaculaire stelling (Tsfasman, Vladuts, Zink, 1982): Er is een rij Goppacodes C_i met parameters (n_i, k_i, d_i) over F_{p^2} zodat

$$k_i/n_i + d_i/n_i \to 1 - \frac{1}{p-1} \qquad (i \to \infty)$$

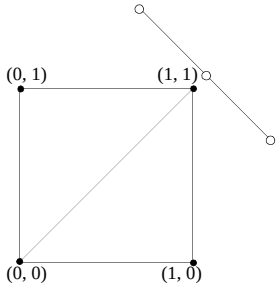
Voor $p \geq 7$ is dit beter dan de Gilbert-Varshamov-grens. Het aardige is dat 'modulaire krommen', een onderwerp uit de binnenlanden van de zuivere wiskunde, zo een volledig onverwachte toepassing vonden.

De jacht op krommen met veel punten

Met het idee van Goppa, en zeker na de eerste resultaten, werd de jacht op krommen over eindige lichamen met veel punten geopend. Al in de jaren veertig had de wiskundige André Weil naam gemaakt door een bovengrens voor het aantal rationale punten op een kromme van geslacht g over een eindig lichaam met q elementen te bewijzen. Deze grens staat nu bekend als de Hasse-Weil-bovengrens:

$$\#C(F_q) \leq q + 1 + 2g\sqrt{q}$$

Alhoewel deze bovengrens scherp is als g klein is ten opzichte van q, liet de Japanse wiskundige Ihara rond 1985 zien dat voor g groot ten opzichte van q dat niet meer het geval is, en dat er dan scherpere bovengrenzen zijn. Dit was het begin van een aantal verbeteringen van de Hasse-Weil-grens. Daarbij rijst dan nog de vraag hoe goed deze nieuwe bovengrens is voor een gegeven paar (g, q). Om daar een antwoord op te vinden, luidt het devies: construeer krommen met veel punten.
Als we werken over het lichaam F_2 met twee elementen, dan gaat de kromme met vergelijking $x^3y + y^3 + x + x^2y^2 + y^2 + x^2 + x^2y + xy^2 = 0$ door alle vier punten van het (x, y)-vlak. Als we werken in het projectieve vlak over F_2, moeten we nog een lijn op oneindig toevoegen, en we vinden dan zeven punten in het projectieve vlak en ook zeven lijnen met op iedere lijn drie punten:



$x^3y + y^3z + z^3x + x^2y^2 + y^2z^2 + z^2x^2 + x^2yz + xy^2z = 0$ is een kromme van geslacht 3 die door alle zeven punten van het projectieve vlak gaat. Omdat de bovengrens in dit geval ook zeven is, zien we dat het maximale aantal punten van een kromme van geslacht 3 over F_2 gelijk is aan 7.
Maar in het algemeen is het vrij hopeloos om te proberen met zulke expliciete vergelijkingen krommen met veel punten over een eindig lichaam te maken, en veel geraffineerdere, geavanceerde technieken zijn nodig om iets te kunnen uitrichten.

De tabel voor p = 2.

g/*q*	2	4	8	16	32	64	128
1	5	9	14	25	44	81	150
2	6	10	18	33	53	97	172
3	7	14	24	38	64	113	192
4	8	15	25	45	71-74	129	215
5	9	17	29-30	49-53	83-82	132-145	227-234
6	10	20	33-35	65	86-96	161	243-258
7	10	21-22	34-38	63-69	98-107	177	258-283
8	11	21-24	34-42	61-75	97-118	169-193	266-302
10	13	27	42-49	81-87	113-139	225	289-345
11	14	26-29	48-53	80-91	120-150	201-236	
12	14-15	29-31	49-57	83-97	129-161	257	321-388
13	15	33	56-61	97-102	129-172	225-268	
14	15-16	32-35	65	97-107	146-183	241-284	353-437
15	17	33-37	57-67	98-113	158-194	258-300	386-455
16	17-18	36-38	56-71	95-118	147-204	267-316	
17	17-18	40	63-74	112-124	154-212		
18	18-19	41-42	65-77	113-129	161-220	281-348	
19	20	37-43	60-80	129-134	172-228	315-364	
20	19-21	40-45	68-83	127-140	177-236	297-380	
21	21	41-47	72-86	129-145	185-244	281-396	
22	21-22	42-48	74-89	129-150		321-412	
23	22-23	45-50	68-92	126-155			
24	21-23	49-52	81-95	129-161	225-267	337-444	513-653
25	24	51-53	86-97	144-166		335-460	
26	24-25	55	82-100	150-171		385-476	
27	24-25	50-56	96-103	145-176	213-290	401-492	
28	25-26	53-58	97-106	145-181	257-298	513	577-745
29	25-27	52-60	97-109	161-187	227-306		
30	25-27	53-61	96-112	162-192	273-313	401-536	609-784
31	27-28	60-63	89-115	165-197		386-547	578-807
32	26-29	57-65	90-118				
33	28-29	65-66	97-121	193-207			
34	27-30	65-68	98-124	183-213		447-582	
35	29-31	64-69	112-127		253-352		
36	30-31	64-71	107-130	185-223		441-604	
37	30-32	66-72	121-132	208-228			
38	30-33	64-74	129-135	193-233	291-375	449-627	
39	33	65-75	120-138	194-239			
40	32-34	75-77	103-141	225-244	293-390	489	650
41	33-35	65-78	118-144	216-249	308-398		
42	33-35	75-80	129-147	209-254	307-405	513-672	
43	33-36	72-81	116-150	226-259	306-413	483-684	
44	33-37	68-83	130-153	226-264	325-420		
45	33-37	80-84	144-156	242-268	313-428		
46	34-38	81-86	129-158	243-273			
47	36-38	73-87	126-161				
48	34-39	80-89	128-164	243-282			
49	36-40	81-90	130-167	213-286			
50	40	91-92	130-170	255-291		561-762	

Algebraïsche krommen met veel punten over eindige lichamen staan tegenwoordig niet alleen in de belangstelling vanwege de toepassingen in de coderingstheorie, maar worden ook toegepast in de cryptografie, waarbij men informatie versleutelt om te verhinderen dat derden mee kunnen lezen. Bankpasjes zijn hiervan een goed voorbeeld.

Om onze vooruitgang op dit gebied in de gaten te houden wordt er door Van der Vlugt en mij een tabel bijgehouden met de wereldrecords voor gegeven geslacht g en aantal elementen van het lichaam q. De tabel (zie www.science.uva.nl/~geer/) geeft voor een paar (g, q) met $1 \leq g \leq 50$ en q een kleine macht van 2 of 3 het maximum aantal punten op een kromme van geslacht g over het lichaam F_q. Is dit aantal niet bekend, dan wordt een interval gegeven waarbinnen dit aantal moet liggen, en als dit interval groot is, wordt dit zelfs weggelaten. Hiernaast staat de tabel voor q een macht van 2. De vele witte plekken laten zien dat onze kennis nog beperkt is, ondanks dat zeer veel wiskundigen hun krachten aan dit probleem hebben gewijd.

Ten slotte

De toepassing van algebraïsche krommen op de coderingstheorie kwam als een volslagen verrassing. Maar naast deze toepassing heeft de theorie van algebraïsche krommen, een onderwerp dat teruggaat tot de negentiende eeuw, recent nog een andere spectaculaire toepassing gevonden, namelijk in de snarentheorie (*string theory*), een veelbelovende ontwikkeling in de mathematische fysica. Beide toepassingen hebben massa's nieuwe vragen opgeleverd en ook een nieuwe intuïtie geschapen over algebraïsche krommen.

Achteraf lijkt het misschien minder opmerkelijk dat krommen – in zekere zin de eenvoudigste niet-lineaire meetkundige objecten – zo toepasbaar bleken, maar het blijft toch verbluffend om te zien hoe de Stelling van Riemann-Roch, een hoogtepunt uit de algebraïsche meetkunde van de negentiende eeuw, perfect toepasbaar blijkt op problemen van digitaal datatransport bijna anderhalve eeuw later.

Als er een les te trekken valt, dan is het wel die van de eenheid van de wiskunde. Maar wat ook opvalt is dat hoewel alle ingrediënten, zoals eindige lichamen en algebraïsche krommen, al in de negentiende eeuw beschikbaar waren, het toch tot het einde van de twintigste eeuw geduurd heeft voordat er echt grote belangstelling kwam voor krommen over eindige lichamen.

Gerard van der Geer
Faculteit Wiskunde en Informatica,
Universiteit van Amsterdam